

Guide to Controlling Delegation of Privileged Root Accounts

Protecting Against Security Breaches and
Ensure Regulatory Compliance in Complex
Unix/Linux Environments

A FoxT White Paper

● Containment of Privileged Accounts

Imagine if banks, to enable maintenance, were to issue general access cards to their employees and contractors, allowing them to withdraw any amount of money from any account on any ATM machine at any time, leaving no other trace than a message from their shared identity – something like, “Kilroy was here”. Your life-savings are gone, “Kilroy was here”. Even if corporate policies do not allow misuse, uncontrolled privileges would obviously be an unpleasant scenario. Yet, in many data centers with business-critical assets, corporations are facing a similar challenge: the superuser is called “root” not “Kilroy,” and the actions taken by the superuser may not even be recorded.

Admittedly, the comparison may be somewhat exaggerated, but controlling superuser and other administrative privileges are critical to most organizations. While in the past, controlling privileged accounts may have made good business sense, today, it is mandated by regulations such as Sarbanes-Oxley (SOX) Section 404. For a SOX audit, it is no longer sufficient to say you trust your administrators; you must have controls in place to convince your auditors that no administrator, trustworthy or not, is able to abuse the authority granted.

● 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, generate one set of challenges. On Unix and Linux systems, the unrestricted “root” account poses a different and particularly troublesome situation. This document will primarily focus on the Unix and Linux specific issues, although similar problems also arise on systems with a different security model. It may also be worth noting that the “superuser problem” isn’t limited to the root account. Other privileged accounts, such as database administrator principals, pose a similar risk.

The business problem grows with the number of people who need powerful administrative access for various job functions. Examples of roles and tasks that may require privileged accounts include:

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

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

Many organizations are finding that the total number of users with excessive access privileges is rapidly increasing. Furthermore, granted permissions are rarely reviewed and revoked, which means users unintentionally accumulate more and more privileges over time as their job functions change and new access rights are granted.

Here are four typical cases where users and administrators require privileged access to systems:

1. System administrators who permanently need unlimited access to individual machines or groups of machines for general system maintenance.
2. Users, who for a limited time, may need unlimited access to individual machines or groups of machines in order to carry out specific job functions.
3. Users who permanently or temporarily may need partial access to the system with administrative privileges.
4. Users who permanently or temporarily need access to a defined set of commands or programs, which in turn must be executed in a user context with superuser privileges, although the user launching the process may not need the same privileges.
5. Batch processes requiring administrative privileges to run.

● Poor Support for Delegation of Privileges

Native Unix and Linux systems usually have very poor support for delegation of privileges to handle the use cases defined above in a controlled fashion. The easy way out is to share root passwords with more or less rigorous administrative routines:

- Password written on a piece of paper
- Multiple user accounts with different passwords sharing the root UID 0
- Emergency superuser accounts protected with “dual controls” by splitting passwords in half so two people need to communicate prior to use
- Software-based solutions generating random passwords for emergency superuser accounts

Delegation through “suid” or “sgid” to allow a process to run in the context of the program file’s owner or group is also a common practice to enable users to get their job done. However, it is also a much appreciated door-opener for hackers, since such processes can be exploited. In this case the sudo utility is a better choice.

Yet, all of these approaches fail to solve the core problem. Even elaborate process for password check-out and subsequent password resets remains dependant on someone with unlimited privileges. And there are other considerations:

1. 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 to 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 far from aligned with your business objectives.
2. Auditing – Even if you trust your process for password sharing, you still won’t be able to satisfy your auditor. When the audit log shows that “root” did something, your auditor will want to know the real name behind “root”.
3. Capturing root’s actions - Once root privileges have been acquired, the omnipotent user can do just about anything, including deleting log files and then reconfiguring the system for future private use.

● The Components of a Complete Solution

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

Components for Robust Privileged Account Control



1. Centralized management of user accounts on all protected servers to ensure you can monitor and audit which user has which type of access on which machine. You also need centralized management for fast and efficient disabling of user accounts across the security domain.
2. Enforcement of policies for delegation of privileges. Instead of allowing functional accounts such as “root” or “sysdba” to login, you need to enforce access rules mandating that individual and auditable user accounts be used. A switch to a privileged functional account for specific job functions or tasks can thereafter be tied to the named user. Controlled switching to a privileged account should not require password sharing, which in turn requires centralized management of fine-grained access rules that can be enforced throughout the security domain.
3. Secure communications 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. Furthermore, it is not sufficient to delegate the task to establish encrypted connections to individual users, for instance by 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.
4. Centralized audit logging with detailed records of user activities. It is also important that these logs are kept on a separate security domain so they can be trusted.
5. For sensitive sessions, you must also have the ability to enforce full keystroke logging so administrator activities can be tracked in detail.
6. Security domains for segmentation of realms within which access rules apply. 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.
7. Furthermore, the solution must provide enabling and secure containment of batch processes requiring a privileged user context on an individual machine or – which may seem to contradict item 2 above – across the network.

“Control Objective:

- Senior management should implement a division of roles and responsibilities which should exclude the possibility for a single individual to subvert a critical process. Management should also make sure that personnel are performing only those duties stipulated for their respective jobs and positions. In particular, a segregation of duties should be maintained between the following functions:

- information systems use
- data entry
- computer operation
- network management
- system administration
- systems development and maintenance
- change management
- security administration
- security audit.”

COBIT Auditing Guidelines,
PO4.11: Segregation of Duties

Now, if the above requirements were met by your infrastructure, you would technically be prepared for the challenge of controlling privileged accounts. However, technology alone will not convince your auditors that you are in compliance.

And that is because IT governance is much more than technology. Policies, standards, procedures and guidelines need to be in place to ensure you have people, processes and technology efficiently aligned. You will need to involve many people in the process of planning and documenting controls and the corresponding risk assessments to ensure that you have an effective controls infrastructure in place.

● What to Do?

Organizations struggling to resolve these issues often end up evaluating three different alternatives:

1. Create home-grown solutions based on Operating System capabilities, available utilities such as “sudo”, clever password management procedures, and lots of scripts. Except for in very small organizations, these attempts will either become extremely costly with system administrators programming instead of doing their jobs. This approach is often found insufficient from an auditor’s perspective. Even if home-grown solutions achieve an acceptable level with regard to password management, they fail to provide corresponding auditing capabilities.
2. Combine various commercial or open source point-solutions, to create an operating environment that takes many of these requirements into account. This typically involves using one solution for user provisioning, another for centrally managed secure communications (SSH), a third for password management or other types of root account management, a fourth for keystroke logging, and yet another tool for audit log consolidation. This could actually amount to something quite powerful in the end, yet one important aspect by necessity is lost: centralized management on one security system. Combining multiple technical solutions into one leaves conceptual gaps which in turn leads to security flaws and inefficient management. All things considered, this is not a cost-efficient approach although, ironically, cost-awareness may well be the primary driver for organizations exploring this option.
3. Invest in an Enterprise Access Management solution (EAM). These solutions are everything but lightweight and in reality there are

“Gartner advocates use of an enterprise access management product for large and complex organizations that can derive benefit from having an external access control system for multiple Unix targets. Although these products do address the control of superuser privileges, they do much more besides and, consequently, are more expensive and more complex to install than the Unix-focused tools.”

Controlling Unix Superuser Privileges Is Critical, Gartner Research G00130427, August 31, 2005

only a couple of vendors offering full-blown EAM solutions. This third alternative is actually what analysts Jay Heiser and Ant Allan have recommended for larger organizations. “Gartner advocates use of an enterprise access management product for large and complex organizations that can derive benefit from having an external access control system for multiple Unix targets. Although these products do address the control of superuser privileges, they do much more besides and, consequently, are more expensive and more complex to install than the Unix-focused tools.” (Controlling Unix Superuser Privileges Is Critical, Gartner Research G00130427, August 31, 2005).

It is not difficult to see the rationale for the Gartner recommendation: Alternative 1 and 2 do not provide a complete solution. Yet, apart from the fact that EAM packages are “more expensive and more complex to install” and to operate, they may also fail to deliver all necessary components. Individual requirements, such as keystroke logging and centrally managed and mandated secure communications, may not be included or not integrated with their centralized management.

This is why many FoxT customers have concluded that something between the second and third option would be ideal: EAM-like centralized management of all components needed, delivered in one single and fully integrated software package, but without the overhead of a full-blown EAM solution. And fortunately, this is exactly what FoxT has to offer.

● FoxT Enterprise Access Controls Management and Corporate Controls Combined

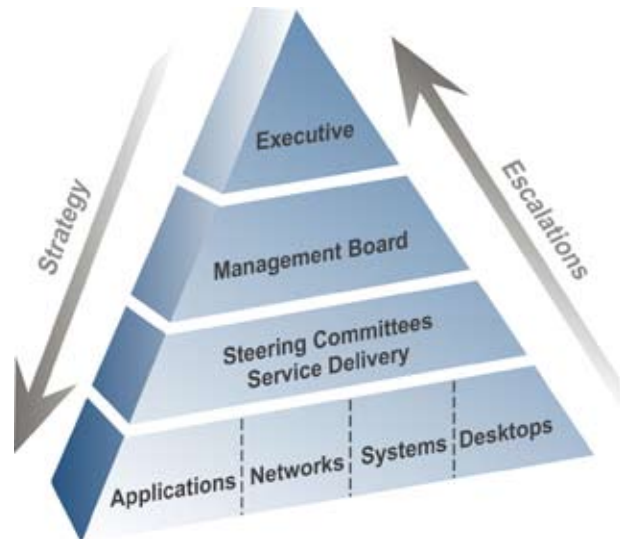
BoKS Access Control products within the FoxT Enterprise Access Controls Management family have been in use by a large customer base for more than a decade including many of the top US banks. Their requirements and understanding of the underlying business problem have, over the years, greatly contributed to the refinement of what today is the most complete solution available on the market with regard to containment of privileged accounts.

The FoxT Corporate Controls solution complements FoxT Enterprise Access Controls Management with the addition of centralized compliance and risk management. Through a collaborative interface, users can plan, document, and manage the testing and reporting of all necessary controls, which greatly facilitates interactions with auditors.

“Governance is the set of responsibilities and practices exercised by the board and executive management with the goal of providing strategic direction, ensuring that objectives are achieved, ascertaining that risks are managed appropriately and verifying that the enterprise’s resources are used responsibly.”

IT Governance Institute

Combined, these products help customers establish a rich controls environment for IT governance which allows management to efficiently communicate policies to an organization that is equipped with the tools needed to translate business rules into enforced access controls for the existing IT infrastructure.



FoxT products help customers establish a controls environment for IT governance which allows management to efficiently communicate policies to an organization that is equipped with the tools needed to translate business rules into enforced access controls for the existing IT infrastructure

● Basic FoxT Concepts

At the heart of FoxT Enterprise Access Controls Management is a multi-service security server, BoKS Manager, which provides centralized management of all of the features on the above wish-list, and a few more. The following FoxT concepts are important foundations for the solution:

1. Host Groups – Realms of centrally managed access rules that enable virtual network segmentation.
2. Access Routes – The atoms of FoxT access rules. In a FoxT protected network, everything is locked down except as explicitly defined by positive access routes with rules defining the allowed access method; from where and to where a connection may be made; and specifying at what times the connection is permitted.
3. Authentication Methods – apart from access routes, a user also

must also be authenticated as mandated by centrally managed rules on the security server. This way, you can enforce different types of authentication strengths (passwords, tokens, X.509 certificates, etc.) depending on who the user is and/or along which access route the user tries to connect.

4. User Classes – enable role based access controls. Permissions can be granted to individual users or, preferably, user roles, so called User Classes. A user can belong to multiple User Classes and thus inherit the privileges of multiple roles.

On the FoxT management console, the above techniques provide security administrators with the necessary parameters to enable efficient translation of business rules into precise and fine-grained access rules, which are adapted to the existing infrastructure.

● Managed SSH and Keystroke Logging

In addition to the above fundamental building blocks, the FoxT multi-service security server fully integrates centrally managed SSH and keystroke logging/auditing features. As a result, access routes can be configured in such a way that a user can access a specific host only via specifically allowed SSH methods. Furthermore, access via a specific access route may enforce key stroke logging for the entire session. Since the server also comes with its own built-in certificate authority, it can handle certificate-based authentications from protected clients.

● Developer Joe's User-Experience

The following use-case illustrates the strength of the BoKS Access Controls methodology. Application developer Joe needs occasional access to a specific host in the data center on which he needs to alter system configurations, change data in a database via SQL commands, and compile new versions of a software package. Due to the nature of his tasks, security administrators have decided he should be permitted to “su to root” on this machine. However, he is not allowed to login to the machine from any other point than his own workstation, and only via SSH using his Smart card for X.509 certificate-based authentication. Although the target host itself is not in

production, it does contain copies of sensitive data to which Joe should have access for testing purposes only. Auditors demand that system administrators can monitor exactly what Joe is doing while using his superuser privileges.

This is Joe's user experience using BoKS Access Controls:

1. Joe inserts his smart card and enters its PIN code on his workstation.
2. He then launches the built-in SSH client on his FoxT Windows Desktop and clicks the preconfigured connection to the server.
3. Joe's experience is single sign-on. Behind the scenes the SSH daemon on the host successfully challenges Joe's credentials on his smart card and then establishes the connection, after verifying with the security server that Joe is allowed to create an interactive shell over SSH on this machine.
4. Joe types "su" in his terminal window and becomes root. No questions asked! Again, Joe's experience is single sign-on. Behind the scenes, the security client on the host has verified with the rules engine on the FoxT central security server that user Joe is allowed to do this switch without knowing the root password. The certificate-based authentication is considered to have established his identity securely.
5. Joe gets a normal Bourne shell in the user context "root". Every key stroke is thereafter being recorded in a keystroke audit log which in turn is sent to the FoxT central security server. Keystroke logging is terminated when Joe exits his superuser session.

An important achievement here is that superuser containment and radically improved security is achieved while users concentrate on their jobs rather than on their passwords. Security, usability, and job efficiency, all achieved by a single solution.

If desired, auditors and security administrators can follow each of these steps and examine in detail what Joe did on the server. If keystroke logging has been configured with full I/O, the log file will even include the command output which Joe saw on his screen.

Maybe Less is More for Joe?

This may seem like pretty good security. Yet, security administrators feel

uncomfortable and decide to restrict Joe's access a bit more. The sum total of commands that Joe needs to execute in a superuser context is fairly limited. Thus, instead of granting access via "su", access routes are reconfigured using the FoxT built-in command "suexec". This can be used to grant execution rights to specific named commands or programs only, "sudo"-like. Joe remains Joe, but the commands are executed with superuser privileges.

Joe's experience remains exactly the same as above except for in step 4 where instead he types "suexec" followed by the various commands. His privileges have been reduced to an absolute minimum but his job-efficiency remains unaltered.

Since security administrators now know exactly what Joe can and cannot do, there may no longer be a need for keystroke logging which in turn reduces the amount of data needed for auditing

Joe's Batch Jobs

One more thing about Joe: Every Monday night he populates his test database with live data from various other production systems. This is a tricky task for many reasons. First, large reports must be launched on a number of production databases, which impacts their performance. Second, output with sensitive data needs to be copied over the network to his test server where the various reports finally are merged and loaded as input in his test database.

Third, permissions for all of these actions require the privileges of a database administrator for which a functional user account "oracle" is used. Fourth, connections must be encrypted and safe, thus SSH is a requirement. And fifth, the task is carried out by batch jobs running at night to minimize performance losses.

The problem: how do we ensure that the functional "oracle" account can be authenticated with a secure SSH connection to numerous sensitive production hosts without risking that these permissions are misused by others? If credentials for logon were to be included in scripts, anyone with read-access to these files will know how to impersonate "oracle". If Joe is allowed to "su" to oracle, Joe can abuse this privilege himself.

With FoxT you could handle this in various ways. One interesting option is SSH host based authentication. Host based authentication simply means that

“Information security governance consists of the leadership, organisational structures and processes that safeguard information. Critical to the success of these structures and processes is effective communications amongst all parties based on constructive relationships, a common language and shared commitment to addressing the issues.”

IT Governance Institute
- Information security
Governance: Guidance for
Boards of Directors and
Executive Management,
2nd Edition

the server to which you connect trusts that the identity of the user has been properly established by the host from which the connection is made. This enables single sign-on and “oracle” doesn’t have to know a password.

Now, security administrators are typically very hesitant to allow SSH host based authentication in a conventional peer-to-peer SSH environment. And this for a good reason: it is vulnerable to man-in-the-middle attacks and the trust between hosts is not ideal. If the first server was exposed to some kind of fraud, the next server will be fooled as well.

In a FoxT managed environment things are radically different. All host keys are managed by the security server. Host based authentication means the target machine asks the central security server to securely verify the identity of the host from which the authentication is made. Furthermore, since FoxT is locking down the entire security domain, it is possible to ensure that no living user can ever be authenticated as “oracle”. The only way to ever launch a process in the user context of “oracle” is by means of explicitly granted “suexec” rights, permissions only given to user Joe. Thus, for the connections made by Joe’s batch jobs, it is safe to allow SSH host based. We can trust the origin: only batch process “oracle” can use this type of access route and “oracle” in turn can only be launched by Joe, although Joe himself is unable to ever become “oracle”.

If anyone remains hesitant, FoxT can restrict the corresponding access route time-of-day and day-of-week parameters to the hours on Monday nights when the job is scheduled to run.

● Nobody Needs to Know the Root Password - Ever!

FoxT Enterprise Access Controls Management can actually be deployed with consistent access routes in such a way that nobody ever knows the root password. Users granted the right to switch to a root context based on access route permissions can be authenticated using the credentials of their own user accounts.

Naturally, one may need to prepare for an emergency situation if a host must be accessed through native Unix or Linux authentication. This could easily be achieved with a FoxT configuration allowing the security server to set random passwords on managed privileged accounts. These passwords can then be checked out from a safe credential store on the security server in case of an emergency.

● FoxT Corporate Controls for Security Planning

The FoxT Enterprise Access Controls Management solution provides an organization with extremely flexible tools for controls enforcement. However, in order to use these tools, structured planning, documentation, and decision making, based on the organizations actual risk appetite will be required.

Security plans cannot be left to IT security and system administrators to figure out by themselves. Processes to enable proper security planning and to establish an environment for sound IT governance require a collaborative approach. Intelligence from many parts of the enterprise must be brought together in a structured process for decision making.

FoxT Corporate Controls compliance and risk management tools can be used to collectively determine an organization's access management strategy and establish an environment for efficient IT governance. The solution guides an organization through the controls definition and risk assessment process. For example....

- Should user Joe at all be granted access to a copy of live data? Or do we first need to verify that customer names and credit card numbers have been scrambled? What needs to be considered to estimate the risks?
- Is Joe an employee or a contractor? Does it matter? Is Joe just one in a group of 100 with similar rights? How does this impact our risks?
- What would it cost to implement automatic scrambling of data in databases from which data is being retrieved? Do we have foreign key constraints that make this task very difficult or expensive? If so, are the risks motivating such costs?
- What legal obligations do we have to protect the privacy of individuals about whom we keep records in these data-bases?

Organizations will be able to determine answers to all of these questions with a collaborative approach. Business unit managers, IT Directors and internal auditors will want to have their say.

This is where FoxT Risk Manager and FoxT Compliance Manager complement the FoxT Enterprise Access Controls Management solution. Combined, FoxT enables an organization to manage the assessment, planning, and enforcement aspects of access management and compliance.

● Summary

The need to control privileged accounts goes beyond the requirement of complying with Sarbanes-Oxley and other regulations: organizations need to ensure stakeholders that they are doing all they can to protect sensitive data assets. FoxT Enterprise Access Controls Management and Corporate Controls come together to enhance existing identity and access management infrastructures with collaborative risk assessment and compliance management; centralized user administration; centralized policy management; centralized audit logging; and encrypted communications. Now organizations can effectively contain privileged accounts and ensure that auditors trust administrators.

● About FoxT

FoxT offers a comprehensive and effective controls solution for corporate governance, risk, compliance and information security management. FoxT combines corporate controls, application controls and secure IT controls software into a single, integrated platform that provides continuous visibility into business processes and performance. Through smart integration and automation, FoxT solutions reduce total cost of ownership, deployment risk, and time to market, making compliance sustainable and transforming it into a source of competitive advantage. Headquartered in Mountain, California, FoxT serves Global 1000 customers in 32 countries. For more information, contact Elliott Zember at +1 650 687 6248 or visit www.foxt.com.



FoxT
883 North Shoreline Blvd.
Building D, Suite 210
Mountain View, California 94043
www.foxt.com
650.687.6300

Copyright © 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.