



Access Control Excellence

FoxT Extensible RBAC – OS-Level Separation of Privileges

*A Technical Overview of FoxT Extensible Role-
Based Access Control Capabilities*





Introduction

Extensible RBAC – xRBAC – is a new FoxT capability for FoxT ServerControl that provides a centrally managed solution for Operating System RBAC (OS RBAC). OS RBAC is found in most Unix/Linux operating systems as a way to provide delegation and separation of administrative privileged beyond what the operating systems traditionally have provided.

This whitepaper provides an overview of the problem with the traditional superuser model in Unix-like operating systems, how this is addressed with Operating System RBAC and how it is managed through the FoxT ServerControl - xRBAC solution.

The Unix root problem

Traditionally, Unix and Unix-like operating systems have a single, functional system administrator account - root - that can perform all privileged system administration tasks. A single shared root account is an obvious problem with regard to separation of duties as this account can override any access control protection and access and manipulate any resource on a given system. This single root account may be acceptable in many situations if the organization has added a strong access control and audit model around who can access the functional “root” account, but it is clearly a problem in environments where different physical persons must have different, non-overlapping privileges with for example system and application administration.

The term “separation of duties” is key to the discussion, since if use of root can be eliminated, the traditional Unix access control system with permission bits for controlling access to files and other objects is satisfactory for protection of sensitive application data from system administrators.

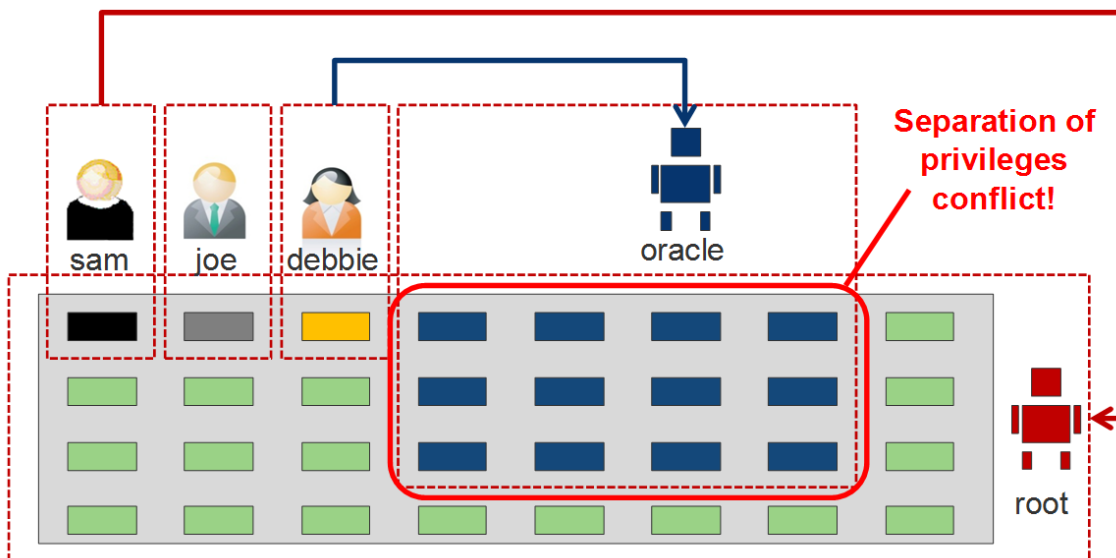


Figure 1

The above picture makes a graphical illustration of the issue with the traditional Unix access control system. Here, 'sam', 'joe' and 'debbie' are physical users on the system where 'oracle' and 'root' are functional IDs. The little colored boxes illustrates various operating system resources such as files, processes etc., and the dotted lines illustrates how access to these resources typically can be set up by use of the traditional Unix access control mechanisms.

The red and blue arrows indicate that 'sam' and 'debbie' have the privileges to elevate their privileges to 'root' and 'oracle' respectively. The elevation is typically granted by the very crude mechanism of simply knowing the target account password to do a successful login or identity transition. In a FoxT protected system, the transition to an elevated privilege is governed by a much more elaborate access control system.

In essence, the problem is summarized by the rounded red rectangle marked by an exclamation mark. This illustrates how 'sam', even though he only should be allowed to maintain the operating system platform, is allowed to access the business data owned by 'oracle' – something only 'debbie' in her role as DBA should be allowed to.

Operating System RBAC

In order to overcome the problems of the all-or-nothing superuser model of the Unix and Unix-like operating systems, the major Unix vendors each implemented their own role-based access control (RBAC) system for dealing with this situation. Although the OS RBAC systems differ in their detailed design, they share the same fundamental philosophy: the ability to move away from the model of system administration by use of an unrestricted root account to the model of granting administrative privileges through roles that can be assumed by physical users.

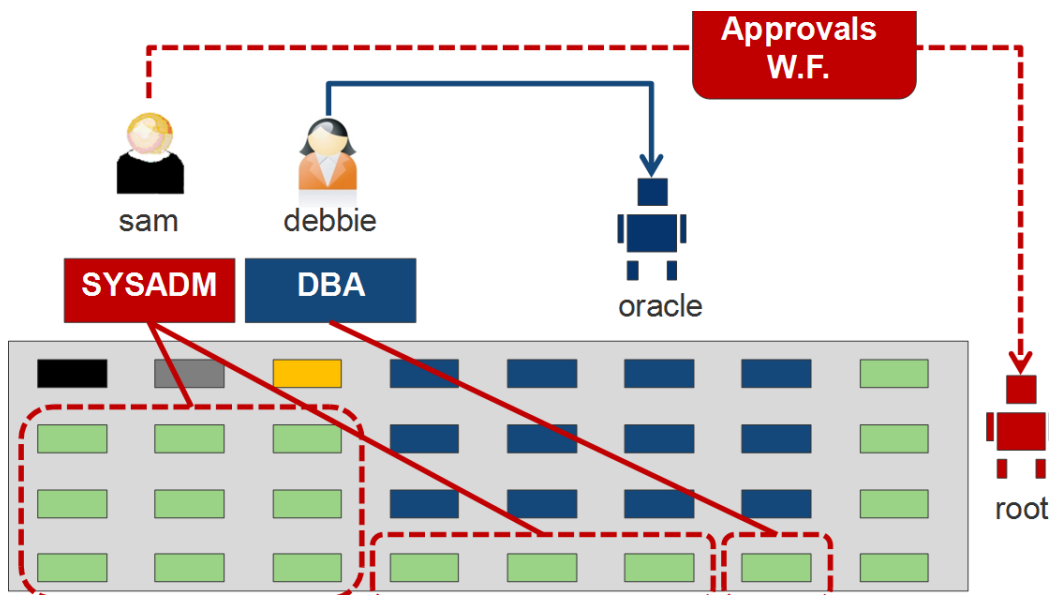


Figure 2



The Operating System RBAC principle is illustrated in the simple picture above. Here we see how 'sam' can assume the role of SYSADM that in turn will grant him access to the root-owned resources such as administrative commands and access to certain files without actually being root.

As normal day-to-day operations can be expected to be handled by use of the commands granted by the RBAC system without the need to share privileged account passwords, it will still be possible to in a controlled manner assume the identity of the root account to use full, unlimited privileges should any unforeseen situations need to be resolved. The ability to actually switch to root can and should be controlled by some kind of approvals workflow and be closely audited as it will very rarely be used with OS RBAC in place. This is illustrated by the dotted line figure 2.

As indicated by the continuous blue line, debbie is still allowed to transition into the oracle account. This is perfectly fine as the traditional Unix access control model gives separation between all accounts except root and this is really what is addressed by OS RBAC. As the picture illustrates, the OS RBAC actually gives the opportunity to grant Debbie even more privileges as some operations typically only allowed for the root user can be delegated to her in a secure fashion.

As an example of how RBAC roles can be created, we can assume we have a server system with a general-purpose configuration that is set to run one or more application workloads. There are several different aspects of administration that apply to this system, as illustrated in the illustration below.

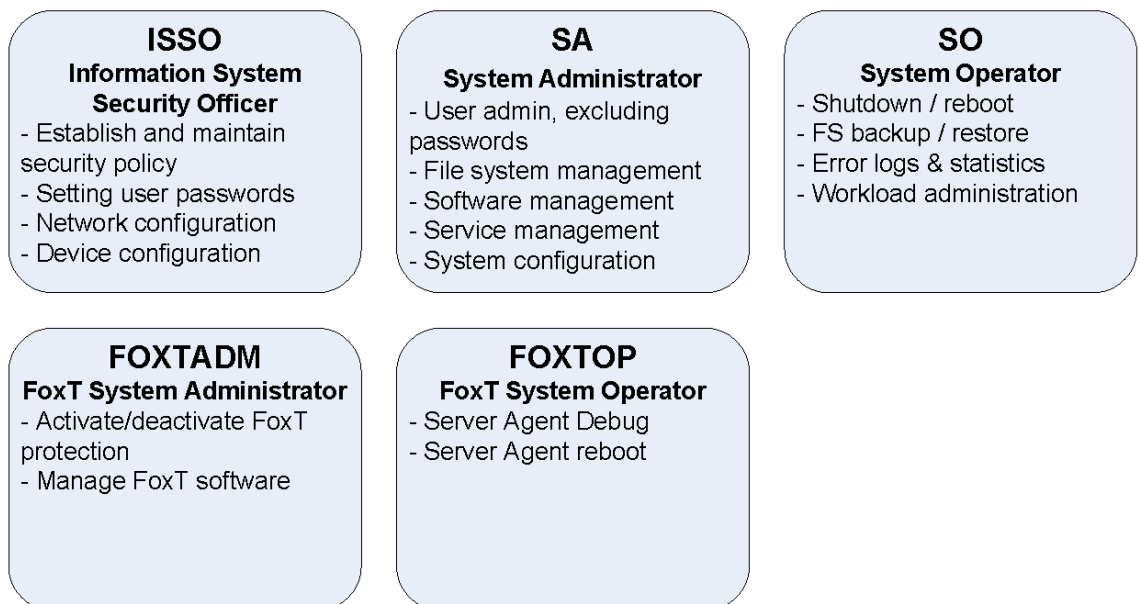


Figure 3

The diagram assumes the existence of the following administrative roles for the operating system itself:

- **ISSO** – Information System Security Officer. This is the role responsible for managing security policies, including RBAC roles. With FoxT xRBAC, this role would have less relevance as much of the security policy management will be centralized in the FoxT system.
- **SA** – Advanced system administration tasks that have to be done from time to time. This includes mounting and un-mounting file systems, managing software and services etc.
- **SO** – More limited, regular system maintenance. This may include backups, harvesting of logs and statistics etc.
- **FoxTADM** – Example of an application specific role, in this case a role allowing for full control of the FoxT software and system on a given Server Agent host without actually being full system administrator.
- **FoxTOP** – Maintenance of the FoxT software that can be delegated to non-security administrators. This could include debugging certain FoxT processes or rebooting the FoxT system on a given Server Agent.

A Typical RBAC System

Although the details may differ, the OS RBAC functions share some common characteristics and a typical system can be illustrated as follows:

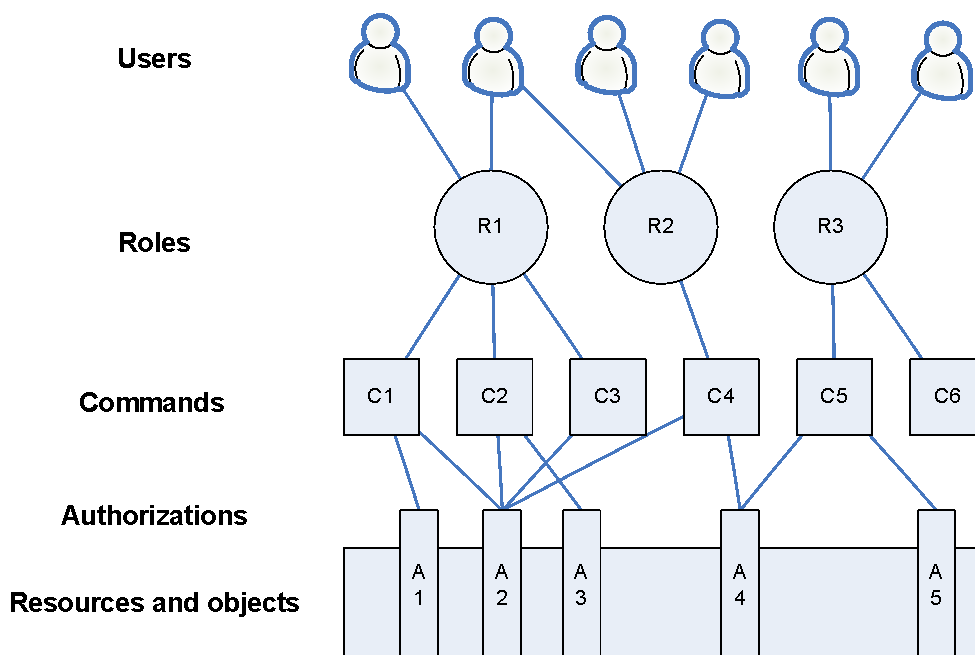


Figure 4



In this picture we can see how:

- Users are granted roles (R1 – R3)
- Each role has been given a set of commands (C1 – C6) that members of it are allowed to run with elevated privileges.
- Some of the commands get their privileges elevated by association of authorizations (C1 – C5) while others may be traditional setuid binaries (C6).
- Authorizations are an abstract representation of certain operations that typically require root privileges to perform, such as mounting file systems.

In addition to the setup in figure 4, there is also typically some way to allow editing of certain (often root-owned) files by overriding the permission bits.

FoxT Extensible RBAC

FoxT Extensible RBAC, or xRBAC for short, tackles the problem of separation of duties for root accounts in Unix environments by leveraging the diverse RBAC systems already provided with operating systems. When coupled to access controls already implemented in a FoxT ServerControl infrastructure, xRBAC capabilities will help organizations more easily map, enforce and audit separation of duties in their Unix environments.

FoxT ServerControl xRBAC has the following characteristics, features and benefits:

- The FoxT Master acts as a central repository for RBAC roles, allowing for easy update and review of OS specific roles across the domain
- Allow roles to be tied to FoxT user accounts and user classes, ensuring that:
 - The roles are automatically provisioned to the managed systems as needed
 - The users are entitled to their roles when accessing a managed system
- Perform real-time access control when a user enters a role
- Provide consistent management of role distribution and associations across heterogeneous systems
- Provides a consolidated audit trail on role usage across the domain
- Provide appropriate reporting on the state of the xRBAC system itself and the RBAC systems it serves
- Offer a strong system of controlled and audited escalation of privileges to unlimited root accounts on the rare occasions when the defined roles are insufficient to resolve a given situation

The below picture illustrates how the xRBAC system fits into the FoxT management infrastructure, both from an administrative perspective as well as how it manages the RBAC related information on the controlled operating systems.

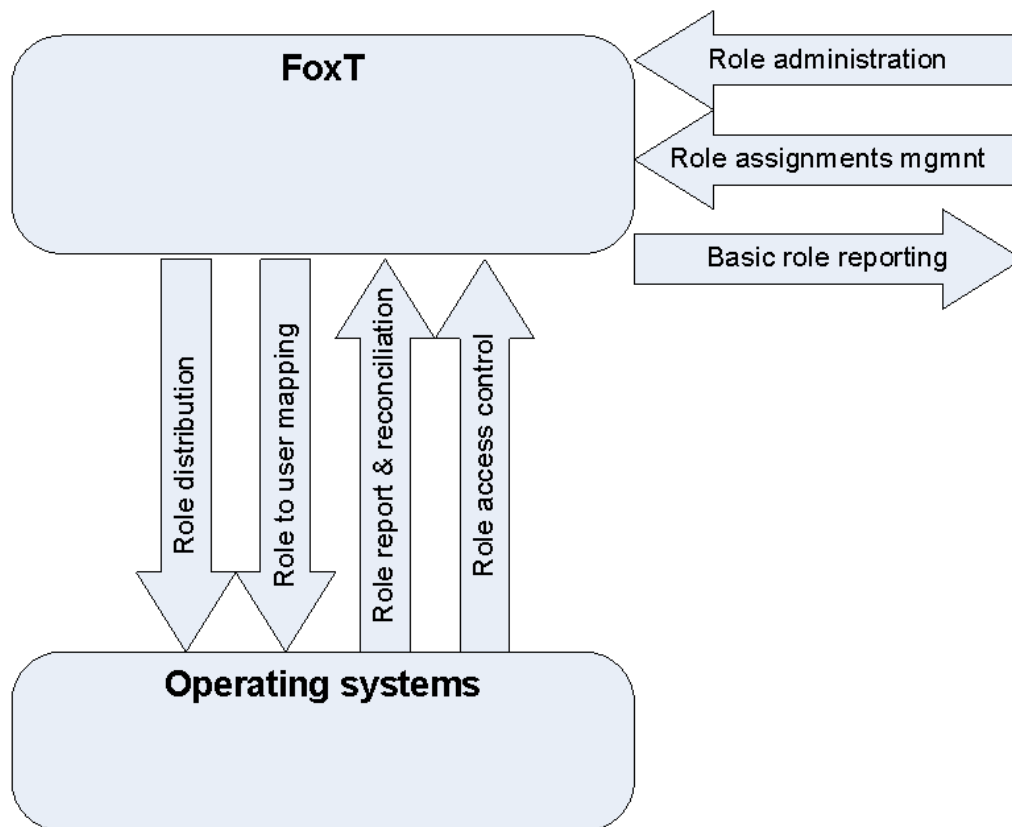


Figure 5

The horizontal arrows describe the centralized administrative interfaces to the system:

- **Role administration** – Management and manipulation of the operating system specific RBAC roles, tied to abstract FoxT xRBAC roles.
- **Role assignments management** – How roles are tied to FoxT user accounts, user classes and mapped to hosts and hostgroups on which the roles should exist.
- **Role reporting** – The functionality to query and list the information and assignments as they are implemented in the FoxT policy store. As the xRBAC system is based on automatic distribution of roles to the local systems, this category also includes reconciliation reports of the “is” and “should” state on the managed systems.



The vertical arrows show the automated actions the FoxT ServerControl infrastructure will perform against the managed systems:

- **Role distribution and role to user mappings** – This is the FoxT ServerControl infrastructure automatically calculating on which hosts a role should exist and to which users it should be granted and then manipulating the local systems databases and configuration files to correctly reflect this. Changes to the state on the local systems is triggered by administrative changes to the FoxT systems, such as adding/removing, roles, adding/removing users to/from user classes, adding/removing hosts to/from hostgroups etc.
- **Role report & reconciliation** – Scheduled or on-demand jobs to report on the “is” compared to the “should” states of the local RBAC systems compared to the state of the FoxT system.
- **Role access control** – Whenever a user enters a role, it will be checked against FoxT access control policies as would use of any other FoxT protected resource. This will also generate an audit trail on role usage.

FoxT xRBAC Operation

FoxT xRBAC is included in the standard FoxT ServerControl packaging but use is governed by separate licensing. The xRBAC functionality is fully integrated with FoxT ServerControl and uses the normal FoxT User Accounts and User Classes to which xRBAC roles can be tied.

OS-specific role definitions will be stored in a central repository in the FoxT ServerControl infrastructure and these roles will automatically be pushed out to the target machines on which they should exist.

Determining which machines the roles should exist on is complex. FoxT xRBAC will not require the role provisioning to simply mirror FoxT ServerControl user account provisioning; two users existing across the same FoxT Hostgroups will be allowed to have different role assignments across these machines and the role provisioning may hence be different.

To illustrate how we go through the entire flow of creating a role and letting a user assume that role on a host, let's assume in the following scenario that we want to grant the user PYRAMID:joe a system administrator role for the IBM AIX 6.1 hosts in the PYRAMID Host Group as illustrated below.

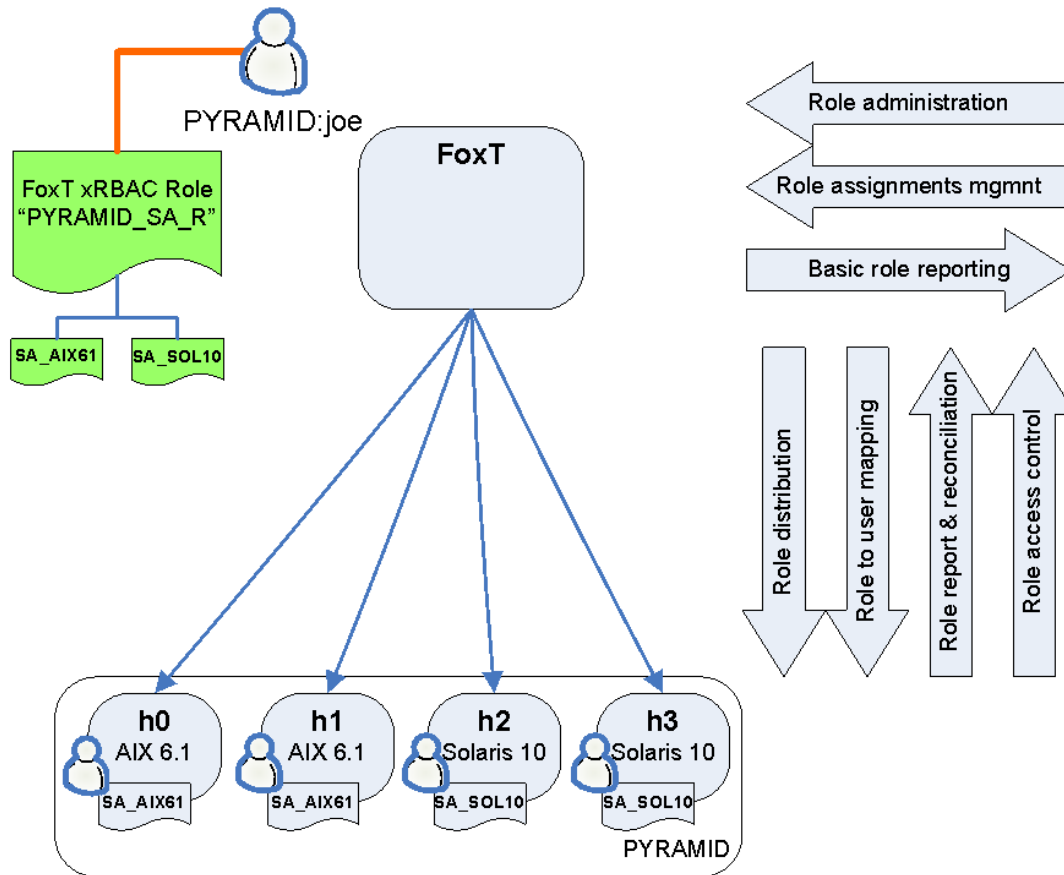


Figure 6

| What | Who | Description |
|----------------------|----------------|---|
| Create / import role | Security admin | The AIX 6.1 role SA_AIX61 is created as a text file and assigned to the abstract FoxT PYRAMID_SA_R role. Note that PYRAMID_SA_R can contain more OS-specific role definitions, such as SA_SOL10. |
| Assign role to users | Security admin | Assuming PYRAMID:joe is already a member of a FoxT user class PYRAMID_SA, we add the role PYRAMID_SA_R to both the user class PYRAMID_SA as well as the group of hosts across which it should exist (PYRAMID). |
| Role provisioning | FoxT infra | The FoxT infrastructure will automatically ensure that the OS-specific roles (in this example SA_AIX61) of the abstract PYRAMID_SA_R are provisioned to all the hosts across which the role has been specified to exist. The FoxT system will automatically ensure that roles are added, updated and removed as needed on the managed systems with future administrative operations. |



| What | Who | Description |
|----------------------------|-----------------------------|---|
| Role binding | FoxT infra | The binding of the user PYRAMID:joe to the AIX role SA_AIX61 will be created by FoxT on the systems on which this role assignment should be in effect. |
| User role access | End user | <p>PYRAMID:joe logs on in the normal, FoxT-controlled fashion to a system in the PYRAMID Host Group on which he has the PYRAMID_SA_R role.</p> <p>Once logged on, he needs to assume the more powerful system admin role and executes (on AIX):</p> <pre>\$ swrole SA_AIX61</pre> |
| Role entry access control | FoxT infra | When PYRAMID:joe attempts to "swrole" into SA_AIX61, this will be checked against authentication and authorization policies and also audit logged to FoxT. |
| System administration | AIX Enhanced RBAC | <p>Once having assumed the role SA_AIX61, 'joe' can then do everything granted to him by this role. This may on AIX enhanced RBAC include managing file systems, installing software, editing certain configuration files etc.</p> <p>Even though he is a system administrator, he will not have the privileges to access the user database, RBAC configuration or FoxT on this system.</p> |
| Ongoing changes | FoxT infra | Any changes made to the Hostgroups, FoxT user definition, role definition and role associations will be handled automatically by the FoxT infrastructure. |
| Reporting / reconciliation | Security admin / FoxT infra | xRBAC includes automatic reporting/reconciliation logic to compare and correct the "should"-state in the FoxT database with the "is"-state of the RBAC systems on the server agents. |

Target RBAC Systems

As FoxT xRBAC is a management system to go on top of the RBAC solutions provided by the OS vendors, only systems with an adequate RBAC system are possible to support. This includes the following operating systems:

- HP-UX 11iv2 & 11iv3 (HP-UX RBAC)
- IBM AIX 6.1 and newer (AIX Extended RBAC)
- Sun Solaris 10 RBAC (Solaris RBAC)
- Red Hat Enterprise Linux 5 (FoxT Linux RBAC)

Worth noting is that the vendors in the Linux space really don't provide adequate RBAC functionality. In the case of Red Hat Linux for example, their RHEL 5 operating system comes with the very powerful SELinux access control system. However, SELinux is only supported in a mode of operation where it is used to harden system services and is not intended to provide administrative separation of privileges.

Due to the increasing importance of Linux as a mission critical enterprise computing platform, FoxT has implemented its own FoxT Linux RBAC system, based on the SELinux functionality to provide similar RBAC functionality as is found on other platforms.

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.

