The 10 Pains of UNIX Security

Learn How Privileged Account Security Solutions are the Right Painkiller
Table of Contents

Introduction: Control Access, Empower Team 3

The 10 Pains of UNIX Security 4

Pain No.1: Protecting the ‘Keys to the Kingdom’ 4

Pain No. 2: Centrally Delegating Root Privileges 5

Pain No. 3: Sudo - An Illusion of Control 5

Pain No. 4: Visible, Hard-Coded Passwords in Applications 7

Pain No. 5: Third Parties Connecting to a Root Session 7

Pain No. 6: Knowing What Actually Happened in the Root Session 8

Pain No. 7: Provisioning New Users When No Identity Management System Exists 8

Pain No. 8: Managing SSH Keys in a Large Unix Environment 9

Pain No. 9: Native Unix Single Sign On 9

Pain No.10: Native Unix Privileged Single Sign On 10

CyberArk’s Privileged Account Security Solution 11
Introduction: Control Access, Empower Team

This whitepaper presents the security pains prevalent in UNIX environments and offers some advice on what requirements to look for when evaluating solutions for these pains.

Back to Basics: ‘Privileged’ Defined

Typical enterprise IT environments are comprised of hundreds or thousands of servers, databases, network devices and applications, all controlled and managed by a variety of privileged and shared administrative identities. Privileged accounts are the most powerful users in any organization, yet these identities are shared, known to many and worst still, their credentials are rarely changed. Privileged accounts include the “root” or “oracle” accounts on UNIX/Linux, Administrator on Windows, Cisco Enable, Oracle database system/sys, MSSQL sa and many more. Put simply, the administrator role gives the user the power to configure virtually every aspect of a system.

Privileged accounts include shared accounts and super user accounts and since both can access your most sensitive data and perform operations whether intentionally, accidentally or by indirect misuse, personalizing access and managing its use becomes paramount.

The Need for ‘Least Privilege’

Privilege isn’t just about access, but what can be done with that access. Mismanaged privileges pose devastating risks to organizations, including financial losses, reputational damage, regulatory penalties and customer loss.

Logging on to a server with a privileged accounts, also carries major operational concerns, as it is all too easy for an administrator to inadvertently change a critical setting or access services for which they have no responsibility. The lack of sufficient auditing can make it difficult to track down problems and ensure that system administrators are only using their privileges for the role they are authorized to perform.

To reduce the risk around shared and privileged accounts there is first and foremost the need for accountability. Knowing who used the privileged account, logging and recording the activities performed and associating it with a reason reduces the risks around ‘insider threats’, meets regulatory and compliance requirements without a tradeoff on workforce productivity.

Restricting access to mission critical servers does not mean that system administrators will not be able to perform their responsibilities. On the contrary, it empowers them to perform commands that are specific to their role and at the same time significantly reduces the security risk posed by the use of privileged accounts with no tradeoff on business continuity, particularly important in times of emergency.
The 10 Pains of UNIX Security

The complexities of Unix environments which include customization and integration challenges using script-based operations make security problems even more challenging. This document tries to shed light on some of the more prominent pains in securing a Unix environment and offers advise on how Privileged Account Security can be the right solution. Capabilities include:

**Pain No.1: Protecting the ‘Keys to the Kingdom’**

The super user, also known as ‘root’ user in UNIX, typically refers to the manufacturer’s default username and password, which is often replaced with account credentials that are then shared by an IT team, developers, DBAs and others. These accounts are in essence the ‘Keys to the Kingdom’ and can access all your mission critical systems. Protecting these keys from accidental or intentional misuse is not an easy task, especially in large organizations. So what are the fundamental requirements to look for when assessing a solution for protecting these keys?

- Personalization of shared and super user account usage – so that you can easily identify who did what, when and why.
- Secure the account credentials – secure passwords in protected storage and ensure they are available wherever and whenever needed independent of network outages. When managing account credentials in Excel files or in secured envelopes, there is always the concern that in times of emergency the password held in these files could have been reset causing prolonged downtime of mission critical systems. Once passwords are centrally managed, versioned and have a full disaster recovery this scenario will never come about.
- Central and flexible policy definitions – organizations may find they are managing tens or even hundreds of thousands of privileged accounts therefore a solution that automatically manages and enforces enterprise policy on local or remote networks across the enterprise with no human intervention is essential. To ease administrative overhead a solution with central management is necessary. Moreover, policies should be flexible enough to suit a variety of business processes such as periodically changing account credentials, implementing access workflows such as dual control, one time password expiration, single sign on without disclosing account credentials and others.
- Monitor and log activity – trace and correlate all actions performed by privileged users and ensure they cannot be tampered with at any stage to meet audit and regulatory requirements.

**Typical Use Cases:**
- Control and monitor access to break-glass root credentials

**Solution:**
- Store root credentials in the Vault for break-glass scenarios
- Root credentials are no longer shared and periodically change
- In break-glass scenarios, authorized users check out the root account and perform their IT work
- Actions taken in the privileged session are recorded and associated with an identifiable personal user for accountability
- Once the account is checked-in the system automatically changes the password
Pain No. 2: Centrally Delegating Root Privileges

Beyond protecting the ‘Keys to the Kingdom’ in a central manner, there is also a need to centrally control root privileges and grant on-demand elevation based on a set of pre-defined permissions. Some solve this pain using a sudo solution, which also has its limitations (as described in pain no. 3) however for those with no sudo solution, look for a solution that enables granular access control for all super users via a central user interface and policy.

Also ensure that the solution provides detailed and tamper proof audit log capturing the activities performed while using the superuser account as well as the output and results of such commands. Because superusers are often able to exploit their super-privileges and cover their tracks, make sure that logging is collected and aggregated on a separate server with tamper proof audit capability.

Pain No. 3: Sudo - An Illusion of Control

Since the emergence of sudo in the 1980s as an open source solution for Privileged Identity Management, it has been adopted by a wide variety of organizations due to its perceived low cost (freeware) for protecting root accounts by personalizing usage, enabling basic logging and accountability.

Sudo (superuser do) allows a system administrator to work using his own account and switch to root or another user identity available on the system only for specific, elevated commands that they need to perform. Sudo was also designed to improve logging, reporting which actions were being performed by specific individuals.

However, sudo creates productivity, compliance and security challenges, particularly in companies experiencing significant IT expansion or large-scale enterprises.

Productivity and TCO challenges:

Since sudo is an open source solution, there is no support center to contact when a problem is encountered. Although there is a large sudo user community that can help in times of trouble, there can be severe direct impact on productivity until a suitable resolution is found.

Sudo also impacts productivity in times of audit. Being a silo solution means that there are separate workflows and the audit logs for super users and shared users when managed in two different systems creating a lot of unnecessary additional work to create correlated reports.

Compliance challenges:

While sudo logs and tracks user activity when running privileged commands, all logs are written to local files and are not secured from being tampered with. Without natively protecting log records from being modified, sudo is perceived by many organizations as not compliant with audit and regulatory standards.

Moreover, the sudo policy is also stored on the local file system. This allows privileged users to modify the access control list and add their own permissions in an uncontrolled and unaudited manner.

Security challenges:

Sudo lacks a local agent, which reduces the level of security, as there is no separation between the end user (running the privileged command) and the actual execution of the command itself. This leaves an opening for someone to potentially interfere or harm the privileged session.

Moreover, the open source nature of sudo means that there are no dedicated QA experts to methodically test the product from the security and feature/function aspects.

A good solution would be one that can identify which elevated commands the user can run, without exposing the root password to the user and creates a separation between the act of invoking the command and running it on a separate terminal.
The 10 Pains of UNIX Security

The command output is then redirected to the user’s terminal where the user can follow the process of the command, creating full isolation of the privileged session. Monitoring these sessions is just as important, therefore look for a solution with keystroke logging so you can know exactly what was done.

All three challenges become even more prominent in large enterprises as sudo was not intended for large-scale deployments. Since sudo is natively a local solution, when it comes to managing hundreds of UNIX servers in a large enterprise environment, a solution with central management capabilities, unified policy, central audit and that has undergone strict testing methodologies becomes paramount and means there is no need for a tradeoff between security, compliance and productivity.

**Typical Use Cases:**

DBA needs to login and run a backup job. Without a sudo-like solution, the user will have to get the root / oracle shared account password and run the backup

**Solution:**

With a sudo-like solution, the user will log-in with his non privileged account and will elevate himself to root in order to run the backup based on pre-defined permissions. Auditors can later review who used the root accounts and what was done with it.
Pain No. 4: Visible, Hard-Coded Passwords in Applications

In today’s complex IT environment, multiple scripts, processes and applications need to access multi-platform resources such as databases, directories and more. The problem is that in order to access these resources, privileged accounts credentials are needed. These credentials are embedded in clear text within the application code, scripts, application server data sources or configuration files and are visible to developers, DBAs, IT personnel and others making the mission-critical applications extremely vulnerable.

Securing, managing and sharing these service accounts on local UNIX servers pose significant challenges and a major overhead to IT departments. When looking to solve this pain, assess solutions that eliminate hard-coded account credentials with central and automatic management for all application credentials and manage Application Server data source credentials. Solutions that prioritize the security aspects such as, credential encryption and a tamper-proof repository for storing audit logs will greatly improve your security posture and enable you to approach compliance with confidence. Moreover, authenticating an application that is requesting credentials based on the application's characteristics e.g. path or signature, will enhance security even further. To guarantee business continuity, ensure that even in network outages passwords are available since many of the applications residing on UNIX servers are mission critical to the organization’s business.

Typical Use Cases:

Shell script connects to a remote system to copy files. It uses a hard-coded root password. Online web application connects to a backend database and uses hard-coded data source credentials

Solution:

Replace hard-coded passwords with a tool that ensures that these passwords will no longer be visible to IT, developers or DBAs and automatically manages them without inflicting downtime on critical Production systems. Moreover, with Application Servers, data source credential management can be done without performing application code changes.

Pain No. 5: Third Parties Connecting to a Root Session

The evolution to a modern datacenter where some servers are managed by third party vendors means that for them to connect to a root session, the organization must disclose the privileged account credentials. Solutions that allow privileged single sign on sessions without revealing the root password is the perfect painkiller. With third parties entering your network from unknown machines, a solution that isolates potentially infected end-points from your internal sensitive systems can prevent the spread of viruses.

Privileged single sign does not stop at third party vendors but can also help you in controlling your internal personnel to avoid disclosing root passwords, especially on sensitive systems. Moreover, by selecting a solution that allows for DVR-like playback you also have visibility into what exactly happened during the session.
Pain No. 6: Knowing What Actually Happened in the Root Session

So you may have solved a few of the pains mentioned earlier such as creating accountability for who is using the privileged account and restricting its use to specific commands for certain individuals, there still exists a need to understand what commands were actually performed in a root session. Both for audit reasons or just as a preventative security measure, monitoring what third party vendors and even internal IT staff are doing in their privileged session means there is a need for a product that provides session monitoring with easy to use playback and search capabilities as well as keystroke logging of input/output.

**Typical Use Cases:**
- Forensic scenarios
  - Someone performed a kill process on the Oracle database
  - Search for relevant root sessions on this server
  - Review personalized video session

Pain No. 7: Provisioning New Users When No Identity Management System Exists

In large UNIX environments, it becomes very challenging to provision new IT users across all machines, especially when looking to consolidate IT resources from separate business units or acquired/merged organizations. Some options that organizations can adopt to reduce the administrative overhead are:

- Use of unmanaged shared accounts - Instead of provisioning personal identities to the UNIX systems, the Unix administrator can create shared (role) accounts that will be used by the IT personnel to perform the required IT tasks. While this option solves the need to manage and provision the IT personal accounts to the UNIX system, it raises an accountability and security problem (as defined in pain #1). Therefore in most cases this is unacceptable option.
- Network Information Service (NIS) - a central directory for defining all UNIX machines such as user accounts, host names etc however many organizations are looking to replace these old, legacy systems as they are not enterprise scale as it requires the duplication of the user repository to the AD/LDAP repository and requires 2 different sets of credentials.
- AD (Active Directory) Bridge solutions enable central management of users who need to be configured in a UNIX environment yet lack the provisioning angle. Although AD Bridge reads from the Active Directory via an agent that is installed on every UNIX machine and can connect to Windows Authentication for Single Sign on, it is a standalone product that has its own costs and requires additional managerial and administrative overhead.
- Similar to AD Bridge products that allow for central management, Privileged Account Security products have the added benefit of managing the entire privileged user lifecycle, which among other things include the definition of role accounts for full accountability, and periodic replacement of account credentials. This is an elegant and easy solution for the user provisioning problem without the need to purchase an AD Bridge product nor to deploy a legacy NIS solution.

**Typical Use Cases:**
- Backup team needs to periodically backup the UNIX systems
- The customer will define backup_user on the UNIXes. This user can be even pre-defined on each Unix server image. This user will be managed by the Privileged Account Security Solution.
- Backup team group will be authorized to use the backup_user credentials.
- When backup team member need to perform an IT task, he can either retrieve its password or transparently connect (privileged SSO) to the system.
- A personalized audit of who used the backup_user will be written.
- If the password was exposed, the system will automatically replace the password (one-time password workflow).
Pain No. 8: Managing SSH Keys in a Large Unix Environment

SSH key pair authentication has been around since the mid nineties and was developed to eliminate the use of shared root passwords. The main usage of SSH key pair authentication is for scripts and unattended applications to eliminate the need to hard-code the root password, but they can also be used by online users. SSH keys however are challenging to periodically replace and since their usage is anonymous, there is no accountability when misused. SSH keys are intended to provide stronger authentication when logging on and are distributed to as many machines in the IT environment as are needed, yet problems arise when trying to update them. Since there is no central management of SSH keys, this solution is not an optimal one in large organizations from an operational perspective. From a security perspective, SSH keys can be stolen or leaked since they are held in local files that can be hacked into.

In a nutshell, SSH keys is a good solution only when you have control of where the keys are distributed and when they are not widespread within the organization, especially when considering the alternative of using hard-coded root passwords.

There are a number of solutions to ease this pain:

- For those who want to continue using SSH keys, implementing a privileged account management solution can centralize control among distributed keys and periodically update them.
- SSH keys can also be secured with solutions that eliminate hard-coded embedded accounts in application scripts and secure them in a central location. This way applications retrieving the SSH keys are tracked and the keys are protected.
- Discard the use of SSH keys, replacing it with a solution for central root account management for both human users and application users where policies can be defined for a variety of workflows e.g. one time use, periodic replacement, dual control etc. Search for a solution that also removes the need for SSH keys in application scripts as these are also prone to hackers.

With a privileged account security solution for shared and application accounts, there is no need for SSH keys as root passwords are secured and routinely changed, audited and monitored and defined by policies for comprehensive access control.

**Typical Use Case:**

A Unix machine that acts as a “Golden Host”, with SSH keys for many machines, that allows jumping without passwords to these machines. The big challenge here is that the SSH keys can be stolen and misused. Moreover, the root access to the target systems is not personalized.

**The Solution:**

Replace SSH keys, with a script that securely fetches the root connections password from a Vault, and transparently connects the users to the target systems. In this way the unsecured SSH keys were replaced by a Vaulted passwords.

Pain No. 9: Native Unix Single Sign On

When having to administer multiple Unix machines while Unix personal accounts are in use (either by IdM or by AD Bridge tools), a solution that automatically connects to multiple Unix machines with one personal account without the need to enter multiple passwords, significantly reduces administrative overhead and minimizes security breaches. Solutions prevalent to solve this pain today are a variety of single sign on products that maintain a central password repository and utilize password injection techniques. Alternatively, AD Bridge products that securely integrate Unix and Linux environments with Active Directory and integrate with Windows authentication, leveraging the Kerberos protocol.
Pain No.10: Native Unix Privileged Single Sign On

How many times have you had to logon to multiple Unix machines to run the same script on all of them but had to remember and enter multiple root passwords? Native Unix Privileged Single Sign On refers to the need to connect to multiple Unix machines without the need to logon with multiple root passwords. Some of the common solutions today include:

1. SSH key authentication can be used instead of logging on with the root password. By connecting with the key and distributing it to multiple machines, a transparent connection is created.

2. A golden host solution is a well-secured machine that has secure access to other Unix machines using SSH key pair authentication. Although it sits in one location and can access multiple Unix machines transparently, from a security perspective, this solution is vulnerable to SSH key leakages which could result in devastating consequences. The security challenges include a lack of audit on root operations making it very hard to trace a user who may have accidently run a script causing damage to all the Unix environment. Moreover, usually such a solution is based on the use of SSH keys where all the challenges referred to in pain no. 8 exist.

Solutions to this pain include:
   a) A single sign on solution with the added functionality of being able to monitor and record the privileged session.
   b) Install a solution on the golden host, which instead of using SSH keys, uses secured password retrieval for applications.
   c) Use a solution that enables privileged on-demand elevation with granular access control and benefit from keystroke logging.

3. A less preferable solution is to write a script with the hard-coded passwords however, the vulnerability of these passwords being visible to too many people is a risk too high to take. Products that manage application accounts and secure them by eliminating hard-coded passwords can be an appropriate solution. By replacing the hard-coded passwords with a call to a command line in order to retrieve the password securely, organizations can maintain a higher security level.

4. A solution that both records a session and enables single sign on is a perfect alternative, especially when wanting to manage privileged users from a holistic perspective e.g. privileged shared account management and control, application accounts, on-demand super user elevation (granular access control) and session monitoring. Although this is not a native Unix solution when implemented standalone, when it comes together with a holistic approach to privileged identity management it significantly reduces administrative and managerial overhead, streamlines reporting and auditing, reduces total cost of ownership and improves your overall security posture.
The 10 Pains of UNIX Security

CyberArk’s Privileged Account Security Solution

The 10 pains illustrated in this whitepaper are consistently creating security, compliance and productivity challenges for large IT environments. By understanding your day-to-day pains, working with hundreds of the largest enterprises in the world, CyberArk is focused on delivering the best-in-class products to help you overcome these exact pains.

CyberArk is the trusted expert in privileged account security. Designed from the ground up with a focus on security, CyberArk has developed a powerful, modular technology platform that provides the industry’s most comprehensive Privileged Account Security Solution. Each product can be managed independently or combined for a cohesive and complete solution for operating systems, databases, applications, hypervisors, network devices, security appliances and more. The solution is designed for on-premise, hybrid cloud and OT/SCADA environments.

CyberArk’s Privileged Account Security Solution includes the following products:

**Enterprise Password Vault**® fully protects privileged credentials based on privileged account security policy and controls for who can access which credentials, and when.

**Privileged Session Manager**® isolates, controls, and monitors privileged user access as well as activities for critical UNIX, Linux, and Windows-based systems, databases, and virtual machines.

**Privileged Threat Analytics™** analyzes and alerts on previously undetectable anomalous privileged user behavior enabling incident response teams to disrupt and quickly respond to an attack.

**Application Identity Manager™** eliminates hard-coded credentials, including passwords and encryption keys from applications, service accounts and scripts with no impact on application performance.

**On-Demand Privileges Manager™** allows for control and continuous monitoring of the commands super-users run based on their role and task.

The CyberArk Privileged Account Security Solution is built on a common, Shared Technology Platform that delivers a single management interface, centralized policy creation and management, a discovery engine for provisioning new accounts, enterprise-class scalability and reliability, and the secure Digital Vault™. The individual products in the CyberArk Privileged Account Security Solution integrate with the Shared Technology Platform, enabling organizations to centralize and streamline management.
To help organizations get started with their privileged account security project, CyberArk offers a free assessment tool, CyberArk DNA™ (Discovery and Audit) that discovers and identifies privileged accounts throughout an enterprise. With a clear record of all service accounts, devices, and applications, CyberArk DNA helps organizations achieve an understanding of the size and magnitude of their privileged account security risk.

CyberArk works with more than 1500 global customers across all verticals, including more than 30 of the Fortune 100. Headquartered in Newton, Mass., CyberArk has offices and authorized partners in North America, Europe and Asia Pacific. For more information, visit www.cyberark.com.