Simplifying the Unix/Linux Security Puzzle

BY DEREK A. SMITH

Table Of Contents

INTRODUCTION

Every business is composed of individual users who conduct daily activities, their managers, and the leaders of the organization. The larger the organization, the more layers there are between the individual user and the executive leaders. This concept is also how the distribution of access and credentials work. Some of the employees and leaders might have access to specific data, and others may not. Also, based on what team an employee is assigned to, or their hierarchy in the organization, there could be different levels of access rights.

Based on this scenario, access can become very complicated very fast – especially if your organization is extensively matrixed, and your users are working with highly sensitive data. Solutions such as Privileged Access Management (PAM) are designed to make the management of privileged accounts easier and ensure that only the users you designate can access your organization's most sensitive data. PAM does this through such methods as password vaulting, recording and auditing privileged sessions, and managing access to privileged accounts.

Your privileged accounts are referred to as "the keys to the kingdom" because they are a primary target for cybercriminals who understand the value of these accounts and will do whatever they can to access and control them. Your executive leaders probably have privileged accounts, although I would argue that they shouldn't because they're probably not conducting the activities that need such access. Your system administrators would also have privileged accounts because they will indeed be involved with such activities. As a result, your privileged accounts require the highest levels of security.

THE CHALLENGE

Unix and Linux systems often house those previously mentioned "keys to the kingdom" consisting of an organization's most critical applications and most sensitive data. However, because many of these systems are siloed, many organizations have a tough time trying to effectively and efficiently secure, control, and monitor their privileged accounts and credentials in their Unix/Linux environments. Consequently, the company's sensitive accounts and credentials, which bad actors can use to access sensitive data, change system settings, and delete audit logs, are often left woefully unprotected.

To effectively protect Unix/Linux accounts and credentials, organizations have to implement controls that allow for central management of their privileged accounts throughout the enterprise. These organizations must also secure and switch the SSH keys they use. They also need to proactively secure their privileged user sessions while continuously monitoring privileged access to detect unusual activity.

UNIX/LINUX SECRUITY-RELATED PROBLEMS

Unix/Linux was not created with security in mind. In the early days of their use, perhaps for the first 20 years or so, they were primarily used by computer professionals, and these individuals did not have to worry about computer crime. These professional's main concern was information exchange versus the need to protect the information they were exchanging. Therefore, there was only a need for a basic security system. The idea that Unix/Linux is a "secure operating system" probably was not really due to these systems safety, but more likely because mainstream operating systems developed later were demonstrated to be especially *unsafe*.

On a Unix/Linux system, access to system resources such as your files and memory is granted to processes. The users of your systems initiate these processes, which typically loads a program to perform its tasks.

These tasks can be performed safely if the process has the minimal rights needed to do its jobs. If a process has too many rights, its program may gain access to sensitive system objects that it should not have access to. A hacker can take advantage of this access by using known bugs or backdoors in the program to manipulate sensitive objects, which consequently could cause a loss of data and degradation in performance.

> “To effectively protect Unix/Linux accounts & credentials, organizations have to implement controls that allow for central management of their privileged accounts throughout the enterprise.”

To prevent this, a solution needs to be implemented to limit the rights. Most Unix/Linux systems default to a basic mechanism called file permissions. This methodology gives the decision to access a system object (devices, file, memory, etc.) or not to access these, to the kernel. The kernel makes this decision based on persistent lists of permissions, that are part of the filesystem. The kernel makes its access decisions (grant or deny) based on these lists.

This method only provides limited control. Basically, for files, we have access control lists that can specify access for users, predefined user groups (where everyone has the same rights) and “everyone else.” Your administrators can only define read, write, and execute permissions. If an admin wants to grant one user reading rights, another reading and writing rights, yet another user execution rights, and everyone else no rights at all, then they have a problem. The admin is forced to deal with this type of situation at the application level.

This security model, with its low level of control, often causes Unix/Linux processes to possess more rights than are needed. If more security is required, a coder will have to add it into the program source code. But what if the coder does not have access to the source code? Or what if he made a mistake? This will allow a running process to be manipulated, for example, by using some type of system bug. This situation is compounded by the fact that Unix/Linux permissions are often improperly configured.

There is another issue that is a cause for great concern when it comes to Unix/Linux security, and that is the system user named “root.” Using “root” as far as the kernel is concerned, allows any process to do whatever it wants, including manipulating system data to mask its actions. On a standard Unix/Linux system, an average user can create files and can allow other users to gain access to those files. This is, of course, something that you would want to allow. But let’s say that the user works in the finance department and his files are highly confidential. This user will not likely even be aware of what file permissions are. When they initiate default permissions for the financial files, this action may allow other users in their group to read his files. Regardless, someone with “root access” on a Unix/Linux system will always be able to read and alter their files.

It is ok to allow this user to have access to their files, but the standard Unix/Linux security model assumes that they are also permitted to grant access to files of other users too. And, that may not be what you want to occur.

USING THE SUDO COMMAND TO SECURE UNIX/LINUX PRIVILEGED ACCOUNTS

One option available to us for controlling our privileged accounts is to use the Sudo command. Sudo is a free, open-source access control tool. However, one should know that Sudo requires costly and laborious custom configuration to meet privilege access management and compliance needs. Although it can help with the challenges of privileged accounts, its usefulness is limited.

If your organization is composed of relatively small server infrastructures, Sudo can be useful for controlling your privileged accounts. But if your server infrastructure is medium to large, Sudo is woefully inadequate for managing your account in an environment this size. Also, Sudo falls short when it comes to administration efficiencies, architectural vision, and security-related compliance requirements needed to protect your critical assets adequately.

If you think that Sudo is sufficient to get the job done. You should be aware that there are five primary challenges when using it to control privileged accounts:

1. **Sudo lacks an efficient method for centralized administration.** Your system administrators can spend considerable time trying to use Sudo to build and distribute files across you system. Sudo cannot easily put servers into local categories, classify users by various roles, and define the associated access rules, methods readily available through the latest privileged access management solutions on the market. The point is that it does not integrate well with Identity Management systems.
2. **Sudo becomes yet another security risk because local files control it.** The burden is on your security admins to properly distribute your files. To distribute them properly, each server typically needs to have a unique file, but in most cases, shortcuts are taken by administrators, which results in giving administrative users too much privilege. Also, another significant security risk is that your local admins could easily modify your Sudo configuration.
3. **Using Sudo could become a compliance issue.** The distributed Sudo conf files are not liked by auditors because they utilize "static trust." The Sudo configuration files need to be secured. This could cause your organizations to have problems passing audits.
4. **If you choose to use Sudo, you must have a way to distribute the files.** Regardless of the methodology you select for this distribution, you must be able to maintain it, and this may lead to additional costs.
5. **Sudo is not able to automatically link to multifactor authentication.** Elevating the authentication requirements to elevate privilege provides greater flexibility in the methods used to authenticate any given user, based on the access request parameters.

Although Sudo can provide you with an adequate method to manage access to your privileged accounts if your server infrastructure is small, it is tedious to implement and maintain and has the potential to create even more exposure to insider threat if you are attempting to control access across an extensive, diverse server infrastructure. In addition to requiring high income system administrators to spend a tremendous amount of time building and distributing Sudoers files, Sudo also forces you to rely on those system admins individual expertise to plan and implement Sudo while adhering to the concept of "least privilege," ensuring users only have the rights needed to execute their tasks.

PAM solutions provide you with a highly efficient and effective alternative to Sudo for administering your privileged accounts. With PAM, you will be able to reduce the risk of insider threat, streamline regulatory compliance, and significantly reduce your admin's efforts to administer your server infrastructure. PAM is a much better option to Sudo for controlling privileged account access.

What Do Organizations Really Need to Protect Their Privileged Accounts?

Since Sudo is obviously not the answer, what can organizations do to gain effective and efficient control over their privileged accounts? We recommend that they use a combination of access management capabilities that can be adapted to various situations.

First, we recommend that you employ a method to allow centralized administration of your user accounts across both your "real" and "virtual" Unix/Linux environments. This centralized management will ensure that you can monitor and audit the specific access that a user has and on which machine. You will also be able to monitor what they do while they are there. This is essential for control. Centralized management will also allow for automatic provisioning and rapid disabling of your user accounts.

SEAMLESS, AGNOSTIC INTEGRATION WITH EXISTING CORPORATE DIRECTORIES

There are many types of entities that link to your organizations – web-enabled business processes, external vendors and suppliers, and outsourcing models mean multiple users from a variety of organizations may have access to your organization's data. To deal with this, your privileged access management solution must have the ability to seamlessly integrate with a variety of corporate directories and identity access management (IAM) systems allowing for team and group identities to be automatically associated to the correct systems, business applications, and data.

Contextual Authentication

The initial step when requesting access is authentication. Your system must first authenticate that the user is who he says they are before it allows them access. The trick here is having the ability to adapt the authentication you require to the context of the access requested. One solution is to use multi-factor authentication for all resources and access requests, but that would be more access than is needed for most of your daily tasks and would be very expensive and management-intensive to implement and maintain.

Contextual authentication would enable you to associate strong authentication to higher risk servers and roles. Your chosen privileged access management solution must be flexible enough to work with the authorization rules discussed in the upcoming "granular authorization" section.

By targeting authentication methods that consider the context of the request, you can avoid using strong blanket authentication. This will be less costly and allow for high levels of security.

Granular Authorization

As an alternative to allowing functional accounts such as "root" or "sysdba" to log in, your administrators must have proactive, enforceable authorization rules that require the users to have personal auditable user accounts when conducting tasks. By employing granular access controls, when a user switches to a privileged functional account, specific job functions or tasks will be associated with the identified user.

Your administrator should be able to configure the authorization rules, and the central management policy should be automatically applied based on the identification of the requester, from where they are requesting, which server they wish to access, how they want to access the server (e.g., Secure Shell) and when the access is being requested.

Password sharing should be a thing of the past if you adopt centralized management of adaptive, granular authorization rules. These rules would be enforceable throughout the security domain and allow you to better control switching to a privileged function without the need to share passwords. As a matter of fact, by using a combination of escalating authentication challenges, most of your standard support related functions that use policy-driven access management can be better controlled. Employees may need to provide their password or other authentication mechanisms such as biometrics, smartcards, Kerberos tickets, etc. – again depending on your access policy and where they are on your network.

Access rules that mandate your secure communications

It is not enough to give your users the option to use a secure connection rather than clear-text telnet. You have to actually enforce the use of a secure connection when and where needed. It is not sufficient to delegate the task to establish encrypted connections to these individual users (e.g., allowing them to decide when it is suitable to maintain user and host public SSH keys). This delegation inadvertently creates a new authentication authority, which then undermines your centralized management and your ability to enforce access rules. Adopt a solution that allows you to define what level is needed for a particular access request. The solution should also give you the ability to authenticate and authorize SSH at the sub-service level.

Consolidated audit logging

IT leaders must know what is being accessed on your server network. You need a detailed record of each user's activities when they are accessing your privileged accounts. An effective approach for protecting your privileged accounts is to use centralized audit logging. However, this is a challenging task considering you must consolidate and cross-reference local audit logs from many different servers. To make this task simpler, consider using a solution that can provide you with consolidated audit logs and reports from across your server domains. For safety, it's recommended that these logs be stored on a separate security domain protected against tampering.

Consolidated operational reports

Additionally, IT leaders must have ultimate control of privileged accounts to protect your organization from insider threats. It is also essential for you to provide your senior leaders with the real-time visibility they need (across business units) concerning your employee's behavior and access control status in general.

This "operational data" will enable you to proactively identify access control issues and mediate any problems that may arise. The selected solution should be able to provide both operational and compliance reports. These reports should be customizable because your organization will likely have specific requirements needing specific information in the report.

Secure keystroke logging

Some of your sessions will be very sensitive. For these, it's recommended that you can adaptively enforce full keystroke logging so that every detail of your administrator activities is tracked. Data privacy regulations will likely mandate that your system security staff are not able to access these keystroke logs. You will need to select a solution that provides secure storage of these keystroke session logs, and also limits release only to approved auditors.

Adaptability and maintenance

Every organization is different. Therefore, you need the flexibility to alter your access management solution to meet both your current and future needs. Seek a solution that is easy to configure and a vendor who is very knowledgeable of access management best practices.

TACTICS TO EFFECTIVELY MANAGE UNIX/LINUX PRIVILEGED ACCOUNTS

Managing privileged accounts is an important, yet complicated task. NIST has drafted some guidelines that you can use to outline a system within your organizations to manage privileged accounts. These accounts frequently have little oversight, meaning those who control the accounts, by definition, have broader access and authority than the average user.

To effectively and efficiently control your privileged accounts, you need to be aware of the various ways your users can use root (i.e., superuser) privilege. The following few tactics can provide you with a realistic view of security on your servers.

Know which users have root access to your servers

If your administrator is the only one with "root access, then that's a great starting point. However, to be safe, you should change your root password (s) periodically. Also, you should have a way to store your root access in a safe place, preferably using a PAM tool.

Know how users have root access

Some users may have root access even if they don't know what your root password is. Therefore, don't think you've controlled or eliminated user's access to this privilege just because you changed the root password as previously advised. Periodically examine the /etc/Sudoers file to see if it has been used to grant someone root privileges. Users might have the ability to run specific commands as root or might be able to use the command "Sudo su" to pretend to be root. When this happens, the user will be prompted to enter a password, and they can circumvent security by merely entering their own.

Know if users can log in as root

You lose accountability for users' actions when they can log in directly as root. All you can gather is that someone logged in as root on a specific date at a particular time, and perhaps what they did while logged in. Your legitimate users should only log in as themselves when conducting their routine tasks. They should only use root access when they need to make authorized changes to the system that they cannot do with their own privileges. If you have users that assist with administering your Unix/Linux systems and they perform only a limited task, you should follow the rules of "least privilege" and only provide them the limited privileges they need to complete the task, such as adding or removing accounts or rebooting the servers.

To prevent your employees from logging in directly as root, if you have it (as this may not work on all Unix/Linux systems), set root's shell to /sbin/nologin in the /etc/passwd file. Make sure you verify this works before you log off. You should still have to the ability to su to root.

Note that Root's UID is 0

When examining who has root access on your Unix/Linux system, remember that the root privilege is associated with the UID 0. The UID of 0 has a unique role: it is always the root account. Although the username can be changed on this account and additional accounts can be created with the same UID, neither action is wise from a security point of view. If you have a user set up on your system who also has his UID set to 0, that person will have root privileges when he or she logs in, so be aware.

Know who has access via the Sudoers file

The Sudoers file can grant access in a variety of ways. Sudo can have stringent rules; for example, you can give a user the right to run a single command only on a specific system. Or Sudo can have very loose rules, for example, you can provide a group of users the ability to switch user to root and, then, do anything they want as if they are the superuser. You must make sure you understand how you grant Sudo to your users.

Know when the root's password was last changed

You should be aware of when your root's password is changed at all time. Even if you are the only person who knows the root password, it's good practice to change it periodically in case the hash has been exposed via a system compromise. You don't want a hacker to have this information and use it to exploit your server system.

Know who can access your servers from other systems

Finally, when you are trying to assess the vulnerability of your Unix/Linux system, you should determine who can access your system from other systems. If your server trusts root on other systems, anyone with root access to those servers can assume root on your server. This is something you definitely want to audit and avoid.

To completely have the granular control you need to secure your privileged accounts, it is recommended, and essential that organizations employ a privileged access management (PAM) solution that controls, monitors, logs, and alerts on the use of your organization's privileged accounts.

Implementing the PAM solution will allow you to add a new layer of security between your users and your critical accounts. The next section explains how BeyondTrust's Privilege Management for Unix & Linux can give you the power you need to secure your Unix and Linux systems.

APPENDIX: WHY BEYONDTRUST PRIVILEGE MANAGEMENTFOR UNIX & LINUX

BeyondTrust Privilege Management for Unix & Linux (PMUL) is a least privilege solution that enables IT organizations to eliminate the sharing of credentials by delegating Unix and Linux privileges and elevating rights to run specific Unix, and Linux commands without providing full root access.

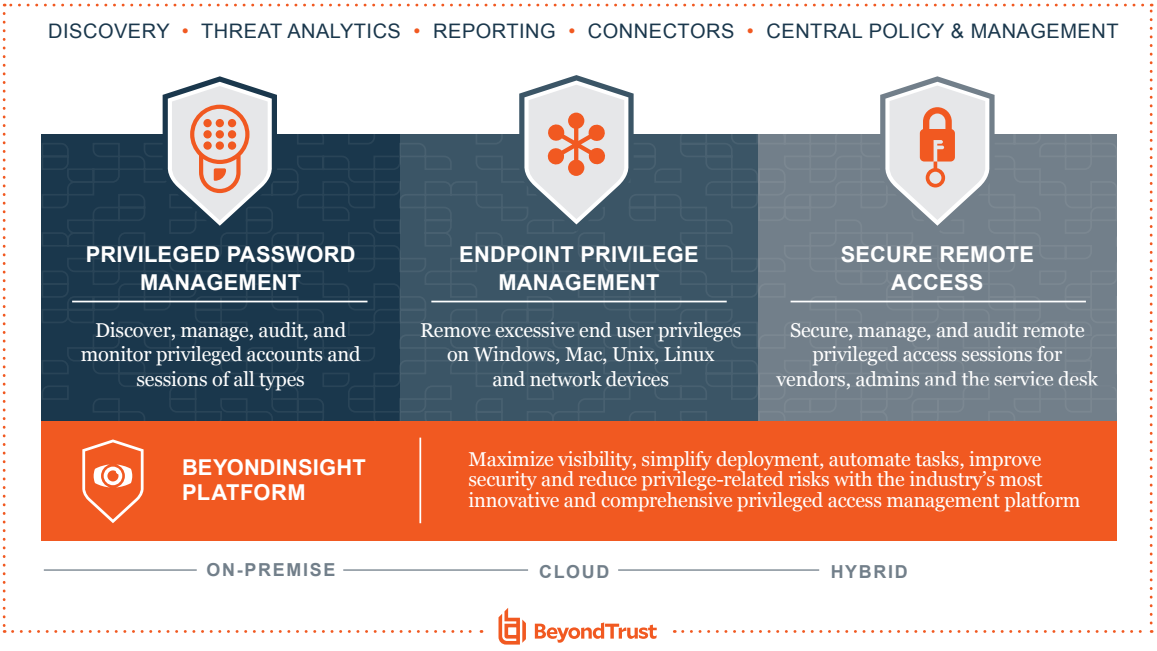
With complete auditing and recording of all user activity – plus a simple graphical user interface for management, and centralized policy management – organizations can easily achieve their security and compliance objectives.

- **Auditing & Governance:** Analyze user behavior by collecting, securely storing, and indexing keystroke logs, session recordings, and other privileged events.

- **Fine-Grained Least Privilege:** Elevate privileges for standard users on Unix and Linux through fine-grained, policy-based controls.
- **Dynamic Access Policy:** Utilize factors such as time, day, location, and application/asset vulnerability status to make privilege elevation decisions.
- **Remote System & Application Control:** Enable users to run specific commands and conduct sessions remotely based on rules—without logging on as admin or root.
- **File & Policy Integrity Monitoring:** Audit and report on changes to critical policy, system, application, and data files.
- **Privileged Threat Analytics:** Correlate user behavior against asset vulnerability data and security intelligence from best-of-breed security solutions.

The BeyondTrust Privileged Access Management Platform

Privilege Management for Unix & Linux and Active Directory Bridge (AD Bridge) are part of BeyondTrust's solution for Endpoint Privilege Management and integrate with other products in the BeyondTrust Privileged Access Management Platform. The platform is an integrated solution to provide control and visibility over all privileged accounts and users.



The Modern Approach to Privilege Management

Most privileged access management solutions just focus on passwords. BeyondTrust is different. Our innovative Universal Privilege Management approach to cyber security secures every user, asset, and session across your enterprise. Deployed as SaaS or on-premises, BeyondTrust's Universal Privilege Management approach simplifies deployments, reduces costs, improves usability, and reduces privilege risks.

Learn more at beyondtrust.com/solutions.

ABOUT DEREK A. SMITH

Derek A. Smith is an expert at cybersecurity, cyber forensics, healthcare IT, SCADA security, physical security, investigations, organizational leadership and training. He is currently an IT Supervisor at the Internal Revenue Service. He is also owner of The Intercessors Investigative and Training Group (www.theintercessorgroup.com/). Formerly, Derek worked for several IT companies including Computer Sciences Corporation and Booz Allen Hamilton. Derek spent 18 years as a special agent for various government agencies and the military.

He is also a cyber security professor at the University of Maryland, University College and Virginia University of Science and Technology and has taught for over 25 years. Derek is retired from the US Army and also served in the US Navy, and Air Force for a total of 24 years. He is completing his Doctorate Degree in Organizational Leadership and has completed an MBA, MS in IT Information Assurance, Masters in IT Project Management, and a BS in Education. Derek has written several books including Cybersense: The Leaders Guide to Protecting Critical Information, and its companion workbook, and he has contributed to several other books as an author and technical adviser.

derekallensmith.net

ABOUT BEYONDTRUST

BeyondTrust is the worldwide leader in Privileged Access Management (PAM), empowering organizations to secure and manage their entire universe of privileges. Our integrated products and platform offer the industry's most advanced PAM solution, enabling organizations to quickly shrink their attack surface across traditional, cloud and hybrid environments.

The BeyondTrust Universal Privilege Management approach secures and protects privileges across passwords, endpoints, and access, giving organizations the visibility and control they need to reduce risk, achieve compliance, and boost operational performance. Our products enable the right level of privileges for just the time needed, creating a frictionless experience for users that enhances productivity.

With a heritage of innovation and a staunch commitment to customers, BeyondTrust solutions are easy to deploy, manage, and scale as businesses evolve. We are trusted by 20,000 customers, including 78 of the Fortune 100, and a global partner network. Learn more at www.beyondtrust.com.