Privileged Password Sharing: "root" of All Evil

Sponsored by Quest Software

Privileged Password Sharing: "root" of All Evil

February 2012

A SANS Whitepaper

Written by: J. Michael Butler

Privilege and Risk *PAGE 2*

Advice and Workarounds *PAGE 6*

Conclusion: The Value of Automation *PAGE 11*

Introduction

Privileged accounts are difficult to manage in any enterprise running multiple distributed operating systems and versions of those systems. The more disparate the systems, the larger the problem. Take, for example, an environment that has HP UX, Red Hat Linux, IBM AIX, mainframes, Active Directory, Windows 2003 Server, Windows 2008 Server, and a few other odds and ends. How can one administrator provision and keep track of every privileged user on every system? For that matter, how can a team of administrators control who is doing what, on which server, and to what end?

To keep things manageable in such environments, administrators usually resort to using generic privileged accounts with a single password shared by all administrators to those systems. This way, every Linux and UNIX administrator knows the root password for all Linux and UNIX servers and can log into any of the servers to perform any task. Although the ability for anyone to step up and do any task sounds great on the Star Trek Bridge, it creates unacceptable risk and compliance violations in the real world.

In that real world, all applications and systems have vulnerabilities. Applications and systems with privileged access provide a serious threat vector that could allow a malicious user to have privileged access—greatly increasing the negative impact of an exploited vulnerability. Most application IDs and passwords can also be used by any individual to gain privileged access to resources that should be off limits to that person.

According to the 2011 Verizon Data Breach Report, trusted insiders, including those with elevated privileges, usually steal larger quantities and more valuable forms of information. The report also notes that the actual number of insider incidents almost doubled in 2010.[1] In January 2011, Vodafone dismissed employees in Australia for either misusing a privileged password or providing it to criminals, causing a breach in the customer information database.[2] T-Mobile also had a breach that reports indicate was based on common knowledge of administrative passwords.[3] Although these examples come from the mobile telephone industry, the issue exists for any organization in which privileged user passwords are shared. It is especially important to note that a breach initiated from a shared password does not have to originate inside the organization!

This paper addresses the issue of managing privileged accounts and offers advice on how to move toward better centralization of privileged account management.

1 www.verizonbusiness.com/resources/reports/rp_data-breach-investigations-report-2011_en_xg.pdf

2 www.wcjb.co.uk/vodafone-terminates-staff-over-data-breach-49340

3 http://threatpost.com/en_us/blogs/hacking-group-teamp0ison-claims-breach-t-mobile-011612

Privilege and Risk

Root, administrator, super user and domain admin are all privileged accounts that have unlimited access. Such users can perform any task, often with impunity, because multiple users know the same account ID and password. Their roles give them elevated privileges that essentially give users full control over the systems they are managing. These elevated privileges increase the risk associated with the accounts, according to the Department of Health and Human Services, which cites the risk of substantial damage these accounts pose and how these accounts require additional safeguards.[4]

Just what are those threats? Privileged accounts threaten the enterprise because these accounts can breach personal data, complete unauthorized transactions, cause denial-of-service attacks and hide activity by deleting audit data, according to an article in TechTarget.[5]

As to controls, NIST SP 800-53 covers the establishing, activating, modifying, reviewing, disabling and removing of accounts under its suggested control points.[6] In the real world of distributed system management, this is more easily said than done due to the quantity and versions of systems being administered, and the lack of support these systems have for such controls. Implementation of adequate controls is further compounded by the silo mentality of groups managing these systems, root or administrator passwords hard-coded into the systems for ease of use, and sharing of privileged passwords between administrators.

Difficult to Automate and Centralize

Every OS has some native method of managing privileged accounts; however, each method is typically proprietary and does not translate between disparate operating systems or even between servers with the same operating system. Worst-case scenario: a super user must be responsible for provisioning every user manually, providing them the exact administrative permissions required for each one to do his or her job. The provisioning steps of a single server must be repeated for every other server to be accessed by the same user. Of course, provisioning the same user on different systems requires different steps, further complicating the process and raising costs of administration. Finally, the manual process requires that the provisioning super user enter every command flawlessly—leaving potential for error in the hands of human operators.

All of the provisioning processes can be scripted or coded to eliminate the risks associated with manually entering each user. Although the benefits of automation—savings of time and reduction of errors—are obvious, the up-front costs for development and testing and the continuing costs of maintaining a custom-programmed system tend to make this option impractical, especially for large enterprises. The more computers in the enterprise and the more diverse the operating systems, the more complex writing and maintaining custom provisioning scripts becomes.

4 www.hhs.gov/ocio/policy/hhs-ocio-2010-0002.001s_hhs_rules_of_behavior.html

5 http://searchsecurity.techtarget.com/magazineContent/Privileged-account-management-critical-to-data-security

6 http://infohost.nmt.edu/~sfs/Regs/sp800-53.pdf Page 40

Most enterprises of any size have multiple operating systems, including Windows and Linux and/or UNIX systems. In addition, as the enterprise adds more functionality and applications, multiple versions of each of these operating systems are often in use. It is not uncommon to see operating systems that are no longer supported, such as Windows 2000, still in use alongside Windows 2003 and 2008 servers. These Wintel systems are often operating alongside multiple versions of Red Hat Linux, HP UX, AIX, and Solaris.

As a result of these hierarchies of systems, Windows teams, Linux/UNIX teams and other administrator groups all work in their own silos. Windows administrators are not comfortable with allowing UNIX administrators to have administrative access to Active Directory domains. Likewise, UNIX administrators do not want domain admins accessing UNIX systems with the power of the root ID. So, group ownership of root and administrator accounts complicates provisioning even further. Setting up automated scripts and applications is also complicated by politics and requires exceptional teamwork.

Shared Passwords

These challenges lead to the simplest and most dangerous solution in many environments—generic administrative IDs and password sharing.

In spite of decades of warnings from every conceivable security and audit-related organization—including the National Institute of Standards and Technology (NIST) and ISACA—organizations persist in allowing UNIX administrators to share the root password for administrative tasks. Unless the organization has only one operating system and one administrator who performs all administrative tasks, the practice of sharing the root password among two or more admins is unacceptable. Once more than one person knows the root password, plausible deniability becomes a factor: who performed the task and when that task was performed does not mean much when 15 administrators have the same password and level of access.

Using Hard Hard-Coded Passwords

Here is a scenario one might see. Code being promoted to production does not function correctly, even though it did in the test and Quality Assurance (QA) environments. The application, as it turns out, has a hard-coded privileged ID and password that work fine in testing. If this is determined to be a permissions issue, the developers expect operations to give the application administrative or root access to enable the app to work in production as it did in testing.

This scenario has at least two significant issues. First, it uses a hard-coded password that cannot be changed regularly without rewriting the code; and second, it also defeats the principle of least privilege by assigning an application more rights than it needs to function. Usually this lapse in security is due to time constraints and pressures to complete the project—pressure from management, sales and the customer.

The first rule regarding hard-coded passwords is not to use them. There are two ways around this: writing the code in such a way as to make them unnecessary and using a tool to manage the passwords so they can be changed automatically behind the scenes. Using custom code to accomplish this is cumbersome at best. Imagine the mindboggling exercise of determining how to encrypt the password that allows access to the current password used by the application (which no human should know) and automatically changing that password without affecting the application or requiring any code changes. Organizations can do it, but at least two questions arise: Who will continue to support this internally written software? What kind of QA testing has been done to ensure that the method is really secure?

These risks go away if we can automatically assign regularly changing passwords or a temporary password to a specific function for only as long as the function is required and then disable the privileged access once the function is complete. Throw in really complex automatically-generated passwords, and the risk becomes even lower.

Audit Deficiencies

In the United States GLBA,[7] HIPAA,[8] PCI,[9] SOX[10] and FFIEC[11] are a few of the compliance organizations mandating that we must be able to prove we have control over our privileged users, know who holds these master passwords and track everything those users do. Use of all privileged accounts must be audited on a regular basis, and the audit logs must reside on a separate computer from the one being audited so the privileged user does not have rights to change the stored audit logs.

Unfortunately, many audit logs typically reside on the system being audited with no centrally located repository to protect those logs. Other issues include pulling and pushing the logs securely, and synchronization of all log events to establish chronological order. Without these capabilities, it's difficult to tell what the privileged user did while he or she was in the system.

7 http://business.ftc.gov/documents/bus67-how-comply-privacy-consumer-financial-information-rule-gramm-leach-bliley-act

8 www.hhs.gov/ocr/privacy

9 www.pcisecuritystandards.org/security_standards

10 http://uscode.house.gov/download/pls/15C98.txt

11 www.ffiec.gov

Sudo Difficulties

Relying on the root ID causes UNIX and Linux to natively have an all-or-nothing approach to privileged permissions. The ability to use "sudo" capability to delegate and manage privileged accounts has been available for years. The implementation of an enterprise-wide privileged access control manager using sudo, however, has always been challenging due to the limitations of native sudo capabilities. One limitation of native sudo is the inability to apply and manage a single policy sudoers file across all servers and networks. This, then, requires inefficient and potentially inconsistent system-by-system management. Tasks like provisioning and de-provisioning a user mean touching every single server in one's environment. Miss one, and a potential security vulnerability exists.

Another limitation is the lack of reporting on access rights for compliance. Searching through several different sudoers files, or even just one sudoers file with thousands of lines of code to determine who has access to what is highly inefficient and error prone. In addition, native sudo lacks any auditing capabilities. Providing compliance reports to auditors is daunting if not impossible. This makes the native sudo capabilities of UNIX and Linux systems difficult to implement and maintain when more than just a few users and systems are being managed.

A very small environment may be able to use sudo and local sudoers files to keep systems in sync; but once an enterprise gets more than just a few servers, use of sudo and sudoers files becomes impractical. In such a situation, a vendor solution may be preferable. The open source version 1.8 of sudo offers API plug-in capabilities.[12] Even with that feature, however, native implementation is still a challenge, especially if every system has its own sudoers file. Using the capability to keep a single sudoers file to specify elevated privileges for appropriate users on specific systems across the enterprise may reduce the amount of administration required for maintenance of the privileged access system.

Although it is possible to set up a custom system, the better solution may be to look for a vendor that meets your organization's requirements and can centralize privileged account management for the organization. One of the main advantages of a vendor-supplied system is the support the vendor can give to keep the privileged access management system running with little to no impact on the organization administrators.

12 www.gratisoft.us/sudo/man/1.8.3/sudo_plugin.man.html

Advice and Workarounds

With current technology, it is no longer necessary for any user to know the root password. Only a minimal number of tasks, such as the installation of a new OS on a new system, require root or administrator level access—and they require access only for the period of time it takes to complete the task. It makes no sense to allow multiple persons to know the root password to conduct their specific tasks during set timeframes, just as it makes no sense for multiple users to know the Active Directory domain account used to set up the forest in a Windows environment.

To limit the scope of damage done by a super user or an attack that gets to administrator accounts, organizations need to enact the principle of least privilege. According to NIST SP800-123, least privilege means that for each task or process, the administrator is granted the minimum rights required to perform the current task.[13]

Least privilege applies not only to users, but also to applications. In an article published more than a decade ago by NIST, there is a discussion of operating systems that "provide mechanisms by which one can assign programs only the specific privileges that they need."[14]

Good security is good business. Every information security policy should require the use of the principle of least privilege, particularly as it applies to privileged accounts. That policy must then be translated into the organization's practices and procedures. It is always better to have several administrators with limited access rather than one person with unlimited access, as noted in NIST SP800-27.[15]

Again, politics can complicate matters. Users are not fond of losing rights, even if the truth is that they do not need them. Administrators often take such changes in policy as personal affronts, even when the change just makes good sense from a security standpoint. Consider these feelings when upgrading rules, roles and management of privileged users and their applications.

To present appropriate steps to address these issues, we suggest adopting the following privileged password management controls:

1. Automate Management

Privileged account IDs should be provisioned or de-provisioned by adding or removing the ID from a central location. If admins have to touch multiple systems to add or decommission a user, the task is no longer simple and is costly and error prone. Plan for a single system to be the central console from which all add/change/delete administration of users can be done.

13 http://csrc.nist.gov/publications/nistpubs/800-123/SP800-123.pdf Page 2-4

14 http://csrc.nist.gov/publications/nistbul/html-archive/dec-99.html

15 http://csrc.nist.gov/publications/nistpubs/800-27A/SP800-27-RevA.pdf Page 18

How do we accomplish automated central control of privileged users? One option is to create a proprietary system for this purpose. A second option is using open source tools, such as the sudo project. Unfortunately, sudo offers no support for Windows—only UNIX type systems.[16] Being an open source application, ongoing support could be problematic. There are other open source options and detailed instructions regarding setting up one or two operating systems with Lightweight Directory Access Protocol (LDAP), for example. But the open source options are not robust enough for most environments without the administrator investing a lot of time in modifications. One can, however, turn to vendor solutions.

A growing number of management products are available; but all are not equally comprehensive, nor do they all fit into every environment. When considering a commercial solution, calculate the cost of trying to manage systems manually versus developing and maintaining your own scripts and middleware. How much would comprehensive automation save the organization? How does this compare to the cost of acquiring, implementing and maintaining licenses for vendor products?

2. Vault Passwords

One of the requirements for any solution should be to provide an automated password vault for FireIDs (used in emergency production support) so no human actually manages the FireIDs.

An approved user should be able to check out a FireID, use it and check it back in with a one-time use password that will change once the ID is checked back in. Further, the system should log everything the user does while logged in under the privileged FireID. This protection is to provide accountability: an administrator will be hesitant to abuse the privileged ID knowing that his or her actions are being tracked under a unique password for the privileged ID, as shown in Figure 1.

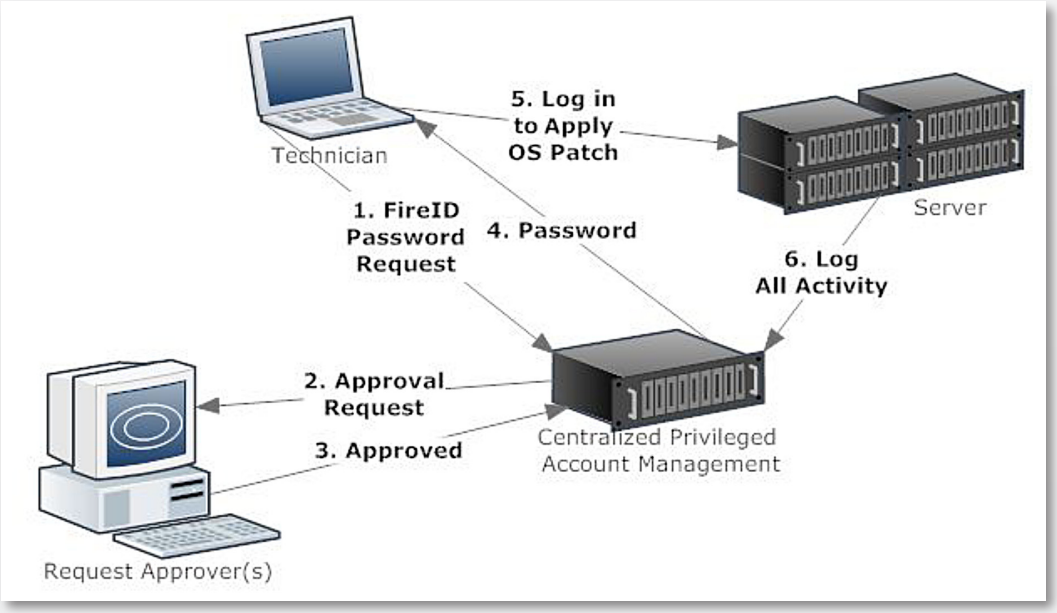

Figure 1. Scenario of a Technician Requesting a FireID

16 www.gratisoft.us/sudo

3. Control Access to Access

A robust system should be able to provide access to a user based on need, without the user having to actually be privileged for all functions. That is, when the user performs an assigned task requiring privileged access, the system gives access without the user actually being privileged for any other actions. So, the system should allow for certain users to be assigned administrative tasks without being an administrator. This way, the principle of least privilege can be supported. Various administrative tasks should be distributed to different qualified users, in keeping with a policy that requires duties to be segregated for best security. These controls should also be time based, so that once the task is completed, that level of access is removed.

Privileged user management must also consider external support. When an outside vendor configures and maintains new systems, it becomes possible that a back door might be left open.

Say, for example, the ACME Voicemail System vendor sets up a redundant server configuration on the organization's network that is to be supported by the vendor's technicians. Once the system is set up, multiple vendor users could know privileged password(s) for the system and possibly for the domain. Without controls in place, these IDs and passwords could continue to exist indefinitely.

Ideally, the centralized account management system should provision an ID and password for the outside support vendor that is limited in scope and time. Like other users with privilege, the IDs should be revoked once the support vendor completes its task until the next time it needs access under a newly provisioned or reinstated password at that time. Similarly, contractors should be treated as employees following rules that are at least as stringent as those governing the employees.

Policies must include an approval workflow that is implemented before users are provisioned, changed or deleted. The data owner should have a say about who has access to the data, IT should advise regarding appropriate access and management should sign off on all access requests. No access should be granted without a properly approved request. The automated central system for controlling privileged access should automatically send out approval requests to all the appropriate parties.

This approval practice should also carry over into the checkout of super user passwords. To protect the administrator/root/super user ID, approval by more than one user should be required to release the password for use. With this security measure in place, if admin, Harry, needs access to install an OS patch that requires the root ID and password, Harry must have George and Sue both key in their approval before the access management system releases the password to Harry. Neither George nor Sue knows the root password. They just have to approve its release before the system provides it to Harry.

4. Change or Hide Passwords

It is also important that the privileged password be changed after temporary use, such as in the preceding scenario. The access management system can be configured so that when Harry checks the ID back in, the system immediately changes the password to something no one knows. Alternatively, the password can expire after a predetermined time, thus ending the privileged access without Harry having to check the ID back in.

When an application requires temporary use of a privileged password, a centralized system should be capable of providing the elevated privilege to the application as needed. Because the password is actually controlled by the access management system, it can be complex and can change within seconds or minutes without having a negative impact upon performance. This complexity and speed make it very difficult—or theoretically impossible—for malicious users to acquire and use the password.

Applications should not be required to know a privileged ID and password. Instead, the application should call a routine that provides the needed access credentials. In the same way, nonprivileged users can be provided privileged access to their job function without having to know the password for a privileged ID. The privileged ID and password, then, can actually be hidden from both applications and users (see Figure 2).

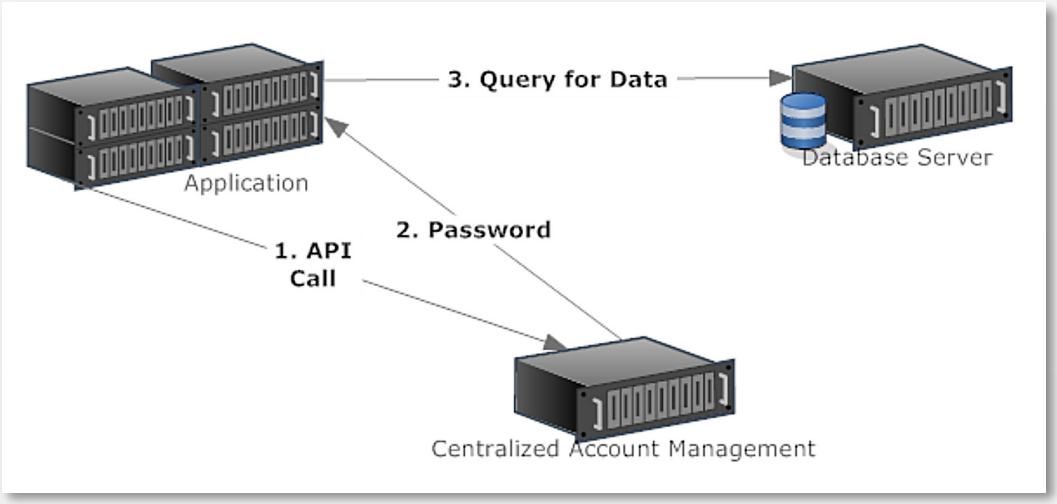

Figure 2. Application Using API to Access Data

5. Use One-Time Passwords

Contractually and through regulatory guidance, it is clear that certain entities expect admins to log into systems using multifactor authentication. A perfect example is the PCI DSS requirement that multifactor authentication be used for any privileged access to a system containing Non-Public Personal Information (NPI). One of the most reliable ways to accomplish multifactor authentication is through the use of one-time passwords. Another very important requirement of any centralized account management system is the functionality that will allow it to leverage multifactor authentication solutions.

6. Centralize Credentials and Rights

Any centralized privileged account management system must integrate with organizations' user directories, including Windows Active Directory or LDAP. A centralized user management system should be able coordinate with the user ID store to provide or deny appropriate privileged access to any system in the enterprise. In addition, a robust privileged access manager should be able to achieve the same level of controls in Windows as well as non-Windows operating systems. In this case the AD or LDAP store could be used to elevate privileges of an authorized user, as needed to accomplish a specific task that would normally require a local admin account, such as installing an application or changing file permissions.

7. Monitor and Audit Use

Audit logs for troubleshooting and reviewing inappropriate activities are important for tracking down what happened in a system. But real-time monitoring is even more important because it can help stop or prevent malicious activities while they are happening. So, from both the reporting and preventive perspectives, monitoring and auditing of privileged use are critical.

In the SANS 20 Critical Security Controls guidelines, Control 12, which pertains to privileged access, reminds us to focus auditing on privileged functions and to look for anomalous behavior, such as system reconfigurations during the night shift.[17] Some of the other actions that should be audited include adding a new privileged account, log in and log out times, source IP and/or MAC address of the privileged user and commands executed while logged in. The guidelines recommend that notice be e-mailed to security administrators when new privileged accounts are added and that repeat notices should go out every 24 hours to ensure appropriate action is taken.

To take auditing a step further, it is advisable to record entire sessions so they can be played back to determine precisely what a privileged user did and how he or she did it. From a forensic analysis standpoint, such keystroke logs can be most helpful for proving a case regarding any accusations of malicious behavior, whether intentional or accidental.

Finally, audit logs must be reviewed—or they might as well not exist. Critical Control 14 states, "...security personnel and/or system administrators should run biweekly reports that identify anomalies in logs. They should then actively review the anomalies, documenting their findings."[18]

Privileged account auditing should include reports for alerting appropriate personnel as noted previously, forensics research pertaining to incidents, and compliance with legal and regulatory requirements. Multiple privacy-related agencies actively seek all the relevant information they can acquire pertaining to how well an organization protects nonpublic, personal data, health-related data, credit card data and any other relevant confidential data. Reports provided to these agencies must prove the organization's access policies and show who had access to what resources.

Each organization must consider its own legal, regulatory and internal obligations to determine which reports it needs to produce. The internal audit department or the third party who conducts the organization's annual SSEA16, PCI DSS, or similar audits can help with the determination.

17 www.sans.org/critical-security-controls/control.php?id=12

18 www.sans.org/critical-security-controls/control.php?id=14

Conclusion: The Value of Automation

Due to human nature, privileged user malfeasance will always be problematic, and violations will continue to occur. According to privacyrights.org, on August 3, 2011, contractor personnel for the Department of Veterans Affairs, Washington, D.C., had been sharing accounts and passwords that allowed them access to VA networks, the Veterans Health Information System, and Technology Architecture systems.[19] In 2010, the now well-known HSBC case was reported, in which an internal privileged user stole account information for more than 24,000 accounts over a three-year period.[20]

Historically, managing privileged users in heterogeneous enterprises has been so difficult that administrative groups simply resort to sharing a single administrator password among them. Even embedded solutions like sudo (for LINUX and UNIX) cannot be scaled without extensive development of connectors and scripts.

In this age of risk, organizations can no longer afford to manage their privileged user accounts on disparate systems manually. Not only are they more vulnerable to catastrophic attacks, they are also out of regulatory compliance. In the case of an incident, it's impossible to assign attribution in a shared password group, let alone monitor the actual generic password user for events and compliance.

Then there's the cost. Leaving privileged account management in the hands of just a few humans checking individual system logs can be expensive, time consuming and error prone. A better way would be to automate as many of these tasks as possible, which is one of the key philosophies behind the SANS 20 Critical Controls.[21] Automating privileged user account management should include the capability to vault passwords; control a user's access to privileged access; change passwords automatically and hide passwords used by applications and scripts; use OTP (one time password) interfaces; and centralize all credentials, rights, and reporting capabilities regarding monitoring and auditing of privileged access.

Such a system will help bring about compliance, reduce error, protect the organization's good reputation and lower costs.

19 www.privacyrights.org/data-breach

20 www.bcs.org/content/conWebDoc/34736

21 www.sans.org/critical-security-controls/guidelines.php

About the Author

J. Michael Butler, GSEC, CISA, GCFA, EnCE, is an information security consultant with a leading provider of technical services to the mortgage industry. Butler's responsibilities have included computer forensics, information security policies (aligned to ISO and addressing federal and state disclosure laws), enterprise security incident management planning, internal auditing of information systems and infrastructure, service delivery, and distributed systems support. He has also been involved in authoring SANS security training courseware, position papers, articles and blogs. Butler has more than 28 years of experience in the computer industry.

SANS would like to thank its sponsor:

Last Updated: October 21st, 2017

Upcoming SANS Training

Click Here for a full list of all Upcoming SANS Events by Location

SANS San Diego 2017	San Diego, CAUS	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Seattle 2017	Seattle, WAUS	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Gulf Region 2017	Dubai, AE	Nov 04, 2017 - Nov 16, 2017	Live Event
SANS Milan November 2017	Milan, IT	Nov 06, 2017 - Nov 11, 2017	Live Event
SANS Amsterdam 2017	Amsterdam, NL	Nov 06, 2017 - Nov 11, 2017	Live Event
SANS Miami 2017	Miami, FLUS	Nov 06, 2017 - Nov 11, 2017	Live Event
SANS Paris November 2017	Paris, FR	Nov 13, 2017 - Nov 18, 2017	Live Event
Pen Test Hackfest Summit & Training 2017	Bethesda, MDUS	Nov 13, 2017 - Nov 20, 2017	Live Event
SANS Sydney 2017	Sydney, AU	Nov 13, 2017 - Nov 25, 2017	Live Event
GridEx IV 2017	Online,	Nov 15, 2017 - Nov 16, 2017	Live Event
SANS San Francisco Winter 2017	San Francisco, CAUS	Nov 27, 2017 - Dec 02, 2017	Live Event
SANS London November 2017	London, GB	Nov 27, 2017 - Dec 02, 2017	Live Event
SIEM & Tactical Analytics Summit & Training	Scottsdale, AZUS	Nov 28, 2017 - Dec 05, 2017	Live Event
SANS Khobar 2017	Khobar, SA	Dec 02, 2017 - Dec 07, 2017	Live Event
SANS Austin Winter 2017	Austin, TXUS	Dec 04, 2017 - Dec 09, 2017	Live Event
SANS Munich December 2017	Munich, DE	Dec 04, 2017 - Dec 09, 2017	Live Event
European Security Awareness Summit & Training 2017	London, GB	Dec 04, 2017 - Dec 07, 2017	Live Event
SANS Bangalore 2017	Bangalore, IN	Dec 11, 2017 - Dec 16, 2017	Live Event
SANS Frankfurt 2017	Frankfurt, DE	Dec 11, 2017 - Dec 16, 2017	Live Event
SANS Cyber Defense Initiative 2017	Washington, DCUS	Dec 12, 2017 - Dec 19, 2017	Live Event
SANS Security East 2018	New Orleans, LAUS	Jan 08, 2018 - Jan 13, 2018	Live Event
SANS SEC460: Enterprise Threat Beta	San Diego, CAUS	Jan 08, 2018 - Jan 13, 2018	Live Event
SANS Amsterdam January 2018	Amsterdam, NL	Jan 15, 2018 - Jan 20, 2018	Live Event
Northern VA Winter - Reston 2018	Reston, VAUS	Jan 15, 2018 - Jan 20, 2018	Live Event
SEC599: Defeat Advanced Adversaries	San Francisco, CAUS	Jan 15, 2018 - Jan 20, 2018	Live Event
SANS Berlin 2017	OnlineDE	Oct 23, 2017 - Oct 28, 2017	Live Event
SANS OnDemand	Books & MP3s OnlyUS	Anytime	Self Paced