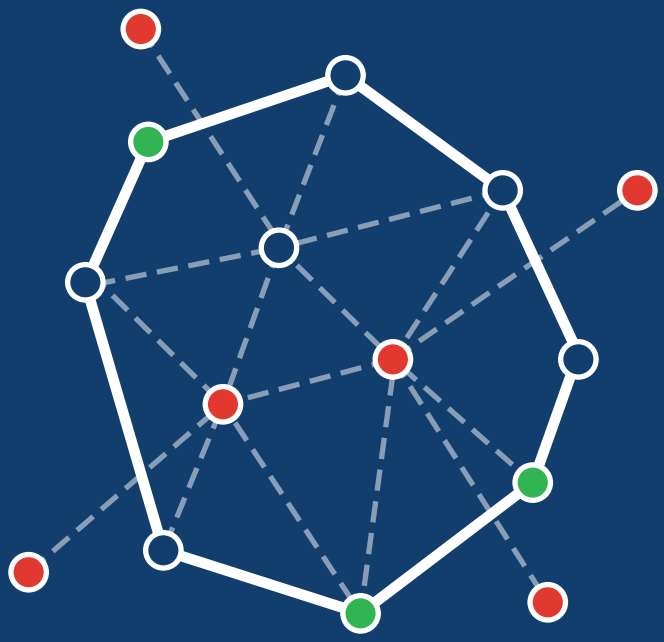
6 CRITICAL CAPABILITIES OF PASSWORD MANAGEMENT



Written By
Nick Cavalancia

BeyondTrust™

VISIBILITY. KNOWLEDGE. ACTION.

INTRODUCTION

If you've been in IT for more than 15 years, you likely can remember a time when there was a single privileged password you needed to remember. My very first was on a Novell NetWare 3.11 box – the password for the Supervisor account was "serious". That was the only one anyone needed to solve every problem on the network.

But today's environment is exponentially more complex. Countless operating systems, directory services, enterprise applications, cloud instances, networking hardware, and even physical security all make up a normal day in the life of an IT organization – each with its' own set of privileged accounts. And it's not only difficult to keep track of all those passwords, but even more so, keeping up with who has access to them and who uses them.

So, if you're like most organizations, you've invested in some kind of a password management solution (which includes everything from an excel spreadsheet, to simple password storage, to advanced password management applications) to house and keep track of privileged passwords.

This ebook is intended for those IT organizations who have already made the step to implement some level of password management, where the assumption is you already have a base set of password management services and functionality in place, including:

Broad System Coverage – Your network is more than just Windows and Active Directory; accounts to manage Unix, routers, databases, cloud instances, and more are all supported.

Centralized, Secure Password Storage – Every privileged password within the IT organization is securely housed and encrypted within the solution.

Check-In/Out Capabilities – Individuals within IT can check passwords out for use and then back in when finished – along with "break glass" access in case of an emergency.

Usage Auditing – Every use of a password is monitored and can be audited later on.

Password Cycling – The passwords housed can be periodically modified, syncing those changes with the appropriate directory service, application, or device.

This ebook focuses on the problems associated with password management, providing thought leadership and guidance in how to enhance the effectiveness and security of password management by looking at 6 critical capabilities:

1. **Password Discovery**
2. **Application Password Management**
3. **Management of SSH Keys**
4. **Privileged Access Control**
5. **Privileged Session Management**
6. **Privileged Threat Analytics**

It's through these 6 capabilities that password management truly takes the concept of securing privileged passwords and puts it into practice.

If you haven't yet taken the step to move to a centralized password management solution, keep reading! Just keep in mind that there is an assumed minimum level of functionality already in place, on top of which you should consider implementing the 6 critical capabilities.

Let's get started!

CHAPTER ONE

PASSWORD DISCOVERY

It's probably safe to say you already know the password to your Administrator account in Active Directory. But there are accounts with privileges everywhere in your network. You no doubt have a few additional domain or enterprise level admin accounts – are those being used? If so, where? How about service accounts? Those are used across countless servers in multiple services per server. How about accounts simply with Log On Locally rights to a server? Do you even know if you have any of those? Or when's the last time they were used?

I've only scratched the surface within the Windows space and, already, it's evident that if you were to be honest, you don't know every account that is being used, where they're used, when they're used, and probably even what the passwords are to some, if not most, of them.

So, the first step in managing privileged passwords, beyond just having a centralized method of storing and making them available, is to perform a discovery of privileged accounts you have in use across your network. Without doing so, you're lulling yourself into a false sense of security – and with only a fraction of the passwords to be managed.

It's more than just finding out where all the passwords are.

So, how do you discover where privileged passwords are being used?

It's more than just finding out where all the passwords are. To properly manage passwords, you need visibility into where every privileged password is and can be used across your network. So having some intelligence around where passwords are being used, why they are used, whether they're actually in use, etc. all will elevate your understanding of how to properly consolidate, secure, and provide access to passwords.

Then, what should you focus on as part of a discovery?

The process of password discovery is made up of three simple steps, all of which need to be repeated over time to ensure that as your environment changes, every password is managed:

1. **Finding what you have** – You can't manage what you don't know exists. So, doing some amount of investigative work around the assets on your network is going to be necessary. Without a third party solution that performs scans of IP addresses, ports, systems, services, applications, etc., it's going to be either a manual exercise, or one that requires some specific use of more than one scripting language. In either case, here's where you should be looking and what you should be looking for:

   Servers – Every server, regardless of OS, should be included. Obvious candidates are Windows and Linux.

   Services / Daemons – These often use privileged accounts. Including them will allow for proper protection of these passwords, since, most often, no one should be performing a traditional logon with the credentials used by a service.

   Applications – This probably would be better thought of as the culmination of a group of services, like SQL Server, or Exchange.

   Networking Hardware – Include firewalls, routers, and any other discoverable device that uses a password to provide elevated access.

   Local Accounts / Directory Accounts – Your goal is to know about the existence of every privileged password that has access to any part of your network, so include any account database, whether local or directory-based.

   Account Details – This should include privilege level, password expiration date, last logged on date, and when the password was last changed.

2. **Understanding what you have** – The goal of discovery isn't to just build a list of accounts and passwords; it's taking all the discovered information and intelligently reviewing it to better understand how and where those passwords are being used. For example, you'd like to know if there are any backdoor admin accounts (those that have never been logged onto, have no password expiration, and have elevated privileges). So, thinking about how to properly report on and present the discovered information is going to be key to the success of your password management efforts.

3. **Managing what you have** – We've already established that you have some centralized password store, so this is the step where you begin to pull those password in under management (which should include a changing of the password both in the directory where the account resides, as well as on the particular system/device/application/service the password is regularly utilized). After all, they are a corporate asset and don't belong to an individual. In addition, you should be thinking about automation for some of the additional tasks around a managed password such as notifying the proper people of the password being taken under management, and grouping passwords based on the discovered assets to automatically provide access.

Take the simple example of a Windows server running Oracle. A discovery of that server would yield the services utilizing a privileged password. The discovery would also show anywhere else that same account is being used (you wouldn't want to not know about a second server using that same account – otherwise you'd cause services to fail once the password was under management and changed). Once pulled in under management, the password would be changed and securely stored, and Oracle admins would be notified of the change, thereby affirming your password management implementation as the central repository to both store, and access, privileged passwords.

Now multiply that example by hundreds of servers, again multiplied by dozens of services and applications on each, and you quickly realize this is going to take quite a lot of work and comes with a decent probability of missing a few uses of privileged passwords. While it is possible to do this manually (following the steps above), automating the process will not only speed it up, but also ensure comprehensive control over the process and result in a more complete set of passwords in use.

USE CASES: WHY DISCOVER PASSWORDS?

There's the obvious need to get passwords under control. That's why you have implemented a password management solution. Beyond that, there are a number of use cases when password discovery is critically important to the password management process:

1. **Knowing where you are vulnerable** – Discovering where passwords are being used, or are simply lying in wait, helps to identify weak spots in security (such as passwords with no expiration), misuse of privileged passwords (such as those backdoor accounts mentioned earlier), and inappropriate use of privileged passwords (such as using the Administrator account across multiple application's service accounts).

2. **Changes in Environment** – New systems and enterprise applications can sprout up daily or weekly, potentially requiring additional privileged accounts. Running periodic discoveries will ensure you have every password identified, secured, and under management.

3. **Applying security to SAM/ITAM** – Scans of systems for hardware and software details do little to make your organization more secure. Password Discovery simply takes the SAM/ITAM thinking a step further to include security.

PASSWORD DISCOVERY: FROM IDENTIFICATION TO ADMINISTRATION

The value of a password management solution comes into play when you have every single password that provides elevated access identified, centralized, stored, and managed. Without performing a discovery of where privileged passwords are being used, your password management solution may as well be shelf ware. Discovery also brings visibility to how and where privileged accounts and passwords are being used, allowing you to rethink the use of accounts, and the access allowed to them.

In the next chapter, we'll take this same concept of discovery and management, and apply it the use of hard-coded passwords within applications and scripts.

CHAPTER TWO

APPLICATION PASSWORD MANAGEMENT

In the last chapter, we covered the need to scour the network for every place a privileged password was being used. Even if accomplished manually, IT has the ability to investigate and identify the accounts and passwords used in systems, services, directories, etc.

But there's one place that's not so obvious, nor easy to access to identify when a privileged password is in use – *hard-coded within applications and scripts*.

There is no manual way to either detect, nor centrally manage passwords stored within applications or scripts.

It's a regular practice of many organizations to embed credentials within a PowerShell, Ruby, or Python script, as well as within an in-house developed application. And those credentials are stored in plain text – whether stored in the script itself, or stored in a file – which is an obvious security faux pas. Some developers equally hard-code passwords right into the compiled code without using a password hash or a library call to an encryption algorithm, creating the possibility for accessing the privileged password by means of reverse engineering of the compiled code.

Even if the password is stored securely, in the light of trying to centrally store and manage password use, you have zero control over the use of these credentials.

So, how do you secure passwords that are sitting in scripts and applications in plain text?

PROTECTING A TICKING TIME BOMB

Unfortunately, there is no manual way to either detect, nor centrally manage passwords stored within applications or scripts. So, in this chapter, we're going to take a slightly different direction in which we're going to describe how third party solutions address

this issue. The goal is to separate the password from the code, so that when it's not being used, it's securely stored away from the user. And when it is needed, the password is only exposed for a short period while being used.

To accomplish this, you need to focus on a few tasks:

1. **Identify where you have hard-coded passwords** – There's no easy answer here. In fact, it's not even feasible to scan reliably for these. The only dependable method is to simply go script by script, app by app.

2. **Replace passwords with password calls** – This is where a third-party solution absolutely must come into play. Your password management solution would need to provide an interface for applications and scripts to call, in order to request the password on an ad hoc basis. Some solutions utilize the IP address from where the call was made as a means of providing access to the password. Keep in mind the passwords should be cycling already, ensuring that after each use, the password is changed and the account is secured.

3. **Separating development from production** – If a production application uses a privileged account with access to data, systems, or administrative capabilities, you may not want that same account available to the dev team during testing of the developed application. Some solutions provide aliasing of accounts so that instead of hard coding "I need the password to the UserX account" and then having to change it just prior to moving to production, instead an alias can be referenced and when moving to production, IT simply modifies the account the alias references.

USE CASES: WHY APPLICATION PASSWORD MANAGEMENT?

It's evident that housing plain text passwords is bad. It's also obvious that hard coding a password into an application doesn't allow IT to secure that password through password expiration cycles. Managing application passwords addresses both those issues. Here are a few other use cases:

2. **Simple, Secure Development** – When you have in-house development, but a need to secure any use of a privileged password, application password management needs to be a part of your strategy. Whether it's a script or an entire application, replacing the password with an API call allows the password to be kept secure, while also simplifying the coding by not requiring updates every time the password changes.

3. **Lower Risk** – By implementing application password management, developers and scripters alike will need to not just call your password management solution, but also make you aware of any new applications or servers running scripts to facilitate access. By knowing when privileged passwords are being used, you'll lower your organization's risk.

BRINGING APPLICATION PASSWORDS INTO THE FOLD

Passwords used in scripts and applications are often overlooked, given how elusive and out of sight they are to IT. But if you're truly serious about bringing every single privileged password under management, these cannot be left out. Unfortunately, there is no means to do this yourself, but third party tools exist to facilitate not just the centralization of those passwords, but also the secure use of them.

In the next chapter, we'll take a look at the use of SSH keys as a means to secure access to Unix servers, providing insight into how they can truly be used for security rather than today's use that largely revolves around convenience.

CHAPTER THREE

SSH KEY MANAGEMENT

In the world of Unix systems, Secure Shell (SSH) keys are heralded as a secure method of establishing sessions. With SSH, a private and public key pair are used, usually with an accompanying pass phrase, to allow only those individuals holding the private key and pass phrase access to the Unix system.

But is it really a secure method of access?

When you think about the key as something you have, and the pass phrase as something you know, SSH keys almost seem like they could be considered a method of two-factor authentication. And then there's all these great security-related terms being thrown around: "key", "passphrase", "private and public key", etc. So, surely it must be an extremely secure method of giving access to your Unix systems, right? Not so fast.

SSH keys are being used more for convenience than for security.

There are a few issues with the use of SSH keys. Let's start with the key itself. In theory, you can generate a single key pair for tens, hundreds, or even thousands of Unix systems, and have each use the same passphrase. And since the private key isn't tied to an individual, anyone who has access to it and the passphrase can access every server using that same key/pass phrase combination. Should a key become compromised that gives access to a server that currently supports multiple keys, there's no way to know which public key is the matching pair for the compromised key. Lastly, there's also no true way to revoke a key, other than to remove it from the Unix server's authorized keys file.

And then there's the pass phrase itself. It has no expiration. The only real way to expire a pass phrase is to remove the associated public key from the Unix server and issue a new one with a new passphrase.

While there is still an account on the back end that needs to be setup to use SSH, IT lulls itself into a false sense of security because you use a "key" and a "passphrase". But when you consider how you're looking to treat passwords as part of an overall password management strategy, you quickly come to realize that this method is a bit more out of control than you'd like and that SSH keys are being used more for convenience than for security.

So, how do you secure SSH key-based access?

There are a few areas to focus on in order to properly secure the use of SSH keys.

1. **Assess the Size of the Issue** – Many of you may live more in the Windows world and don't realize you're using SSH keys at all. It's necessary to identify how many accounts are setup to use SSH keys. It's a bit of manual work without a third party solution, but can be done by searching for the authorized_keys file in the hidden .SSH user folder. Keep in mind, there can be a number of keys in that file and there's no documentation listing who has the private key matching any of the public keys in the file.

2. **Implement Key Rotation** – Like rotating passwords, the rotation of keys eliminates the "one key/1000 systems" problem, but does imply you, then, need to generate 1000 keys – and rotate them frequently! A third party solution would generate unique key pairs for each system, but, if no solution exists, at a minimum establish a rotation frequency and manually change them periodically.

3. **Generate Unique Passphrases** – With a unique pass phrase per key pair, even if the key becomes compromised by someone obtaining the private key in the pair, they cannot gain access. This is, of course, a lot of work. For some organization, too much work. Which simply reinforces the point that SSH keys may be more a matter of convenience than security.

BRINGING APPLICATION PASSWORDS INTO THE FOLD

1. **Secure Access to Unix** – Assuming you're conceptually all in when it comes to pulling all privileged passwords in under management, SSH keys should be seen as just another password but with a key pair tagging along that need to be managed. By using unique passphrases and rotating keys (presumably with new passphrases each time), you'd be providing the same level of security you do to your privileged account passwords.

2. **Secure Script Access** – This use admittedly requires a third-party solution. In the same way we discussed applications and scripts utilizing an API call for any password-related portions of the code, Unix scripts can take similar advantage of centrally calling for a key/passphrase pair against each host the script needs to run on.

3. **Proving Compliance** – Without SSH key management, once someone has the private key and passphrase, they can access a Unix host with all the permissions the associated account affords. Should protected data (as defined by compliance mandates) reside on a Unix host, without SSH key management demonstrating both the security of the keys, as well as the usage of them, it would be nearly impossible to truly prove protected data is secure.

SSH KEYS: A NECESSARY PART OF PASSWORD MANAGEMENT

The parallel is there between privileged accounts and SSH keys. Both provide access, both are largely decentralized and undocumented in their use, and both can be configured to work across multiple systems. It's only in including SSH keys that you will be able to have the same level of security in Unix systems as you do in the Windows world.

CHAPTER FOUR

PRIVILEGED ACCESS CONTROL

Generally speaking, security is divided up into classes of users using defined roles. Take groups within Active directory, for example. Groups are created to identify domain admins, server admins, database admins, etc. and users are added as members of those groups to provide them access. And once this is done, not another thought is given to whether those users should be able to have that privileged access anytime they want, from anywhere they want.

Is that the right approach to allowing access to privileged accounts?

In essence, simply allowing someone to have access all the time lacks context. Organizations concerned about security may want to implement a filtered level of access based on conditions to minimize the abuse of privileges – whether by the employee themselves or via a compromised account and password. For example, in the Target breach, a contractor's account was used. It's doubtful the attackers logged in from the contractor's laptop (the machine IT intended the account to be used from).

In essence, simply allowing someone to have access all the time lacks context.

So, doesn't it make sense that privileged access should also be controlled access?

This is a tough one without a third-party solution, as native Windows and Unix tools provide little in the way of limiting someone's ability to simply log on, and even less exists when you begin looking at restricting access to networking devices. But, given you've already made the move to a password management solution, some or all of the following methods should be available to control the access to privileged passwords, regardless of the platform or device the password provides access to:

1. **Limits of Use** – Ideally, this should include an ability to limit access to check out a privileged password. Additionally, in the case of launching a privileged session (which we'll discuss in the next chapter) rather than checking out a password, you'll see there are some measures you can take to limit this kind of access natively in Windows. In either case, usage limits will vary from account to account, but should be based on:

   When – using time of day, day of the week

   Timeframe – using start and end dates (great for contractor accounts)

   Where – using IP addresses and ranges

   How – whether requesting just the password or by starting a privileged session directly on a server

2. **Usage Approval** – For particularly sensitive passwords, the request to utilize a privileged password can be connected with a request for approval from a peer or a manager, thereby creating a system of checks and balances around the actual use.

3. **Policy-based Grouping** – Multiple accounts may require the same levels of limits and approvals, so having an ability to group similarly restrictive passwords and apply these limits once is important.

USE CASES: WHY PRIVILEGED ACCESS CONTROL?

In most organizations, IT simply chooses to trust a member of the team to do their job properly and allows them 24/7 access via a privileged password. But there are cases where it makes more sense to put limits around when privileged passwords can be used.

1. **Separate User/Admin Accounts** – In those organizations that have even IT use separate accounts, an IP address limits and a peer approval would be a minimally invasive step in the right direction to keep use internal and accountable.

2. **Protecting Against Data Breaches** – According to a recent survey of hackers, the number one target is contractor accounts, with IT administrator accounts in second place. Putting limits on those accounts that match when, where, and how those users can access and utilize privileged passwords will significantly lower the risk of unauthorized access that would otherwise lead to a data breach.

3. **Break Glass Scenarios** – At some point, there will be a need to provide access to someone not normally associated with a given server or application. With access control in place, you retain the ability to provide temporary access and, optionally, require approval for actual use, all giving you the confidence to provide that access to address the latest fire to be put out.

PUTTING CONTROL AROUND PRIVILEGED ACCESS

If you're truly concerned about privileged passwords, there needs to be some context around approved usage that includes limits and even approvals of use. By implementing access control around privileged password use, you'll not only know who has privileged access, but also define when they can use it, from where, how, and how often.

In the next chapter, we'll take the concept of use of privileged passwords to an even deeper level by looking at what actions were taken while in use.

CHAPTER FIVE

PRIVILEGED SESSION MANAGEMENT

In the previous chapters, we've taken a look at how you need to discover, categorize, allow, and secure access to privileged passwords and accounts. By implementing all of this, you certainly have a far better idea of what systems and applications require elevated permissions, and who will need and be able to access them.

But none of these previous capabilities address IT's primary concern: what was done with the password. After all, you're not trying to protect some random set of letters, numbers, and characters; you're trying to protect the applications, systems, and data that powerful string provides access to. Each of the previous capabilities focused on helping to establish what should be protected, who can access, what approvals are necessary, and what methods of access are acceptable. But, even with all of this in place, your IT organization still has no idea of exactly what actions are being taken once a user has access to the password of a privileged account.

Your IT organization still has no idea of exactly what actions are being taken once a user has access to the password of a privileged account.

Think about it – you've put all these precautionary measures in place, only to allow someone with the password to a privileged account to do anything and everything that account allows on any accessible machine? Probably not. That's why all of the previous capabilities in total still fall subject to the need to monitor and manage the privileged session itself.

But, what exactly does it mean to manage a privileged session?

SESSION MANAGEMENT: AN EXTREMELY BRIEF PRIMER

The privileged session is represented by actions performed while logged in using the password-protected privileged account. Management of that session has both to do with where the privileged password can be (and is) used in the first place, knowing what actions are being taken while logged on, and an ability to administratively respond to active sessions should there be an issue.

So, what should you focus on when managing a privileged session?

When you boil it down to the basics, there are three key areas around managing the session you need to focus on:

1. **Limiting The Scope of Aacess** – Which is more a more secure way of managing password usage: giving someone the password to an admin account and letting them have at it, or only allowing that password to be used on a particular server? It's obvious, the most proactive way of knowing what actions are being taken is to limit those actions in the first place. Limiting access to servers through firewall rules (if possible), or through restricting Log on Locally rights (in the case of Windows) will assist in keeping password usage minimized to only approved machines. This would facilitate you still using native tools like MSTSC and PuTTY to gain access to systems. Ideally though, as mentioned, the passwords themselves would kept from the user, by using your password management solution to allow access to a session without giving out the password itself.

2. **Maintaining Session Oversight** – Even if someone is logging into the appropriate server, IT still needs to know about it and, potentially, do something about it. If you're a DIYer, even setting up simple logon auditing and alerts from event log entries in Windows would push you in the right direction so you're aware when a session is established. And if you're strictly using, say, Terminal Services, you could easily reserve the ability to kill a session should you suspect improper use of the privileged password. Depending on the criticality of the account and the server gained access to, these fundamental steps may provide some coverage in establishing oversight, but you may find you need to utilize 3rd party solutions that provide more

automated ways to not just kill sessions, but also to pause sessions (in case a review of why access is being used is necessary without terminating the session) and even remove access in cases of inappropriate access or true breaches of security.

3. **Creating an Audit Trail of Usage** – The challenge with password usage is knowing what they've done with the access. Some applications and services, such as Active Directory, create log entries to record actions performed by a user. But most applications simply do not. So, how are you supposed to maintain an audit trail where none exists? There is no good answer with native OS tools outside of those applications that do provide some level of action information within logs. For everything else, you're going to need to look at some kind of User Activity Monitoring solution that, at a minimum, records the video output of the entire session, tracks specific actions performed, and allows the playback of the session much like a TV show recorded on a DVR.

USE CASES: WHY USE PRIVILEGED SESSION MANAGEMENT?

The overarching goal is to ensure the passwords you've taken such lengthy steps to protect are used properly. Privileged Session Management provides IT with some context around that usage. Here are a few use cases when Privileged Session Management makes sense:

1. **Accountability** – Monitoring and managing privileged sessions provides IT with the ultimate level of accountability. There is little question around what has occurred when IT has the ability to review when a password was used, where, for how long, and what was done while logged on.

2. **Proving Compliance** – Compliance standards around data security, like those found in PCI, HIPAA, and others, require an organization to not just establish an environment that protects data, but to also have the ability to prove that protection is actually established. Privileged Session Management empowers you to prove compliance by demonstrating that only appropriate activity occurred, keeping within compliance standards.

3. **Control** – The degree of control over the generation and usage of sessions is dependent on how you implement Privileged Session Management. Should you use a mix of native solutions, you're going to have minimal control and, therefore, minimal benefit. But, 3rd party solutions do exist that far surpass the basic abilities to control the who, where, and what of a session mentioned in this chapter.

4. **Emergency Access** – Going beyond basic RDP sessions, if you had a third party solution that provided proxied session access without needing to reveal passwords, you'd have an ability to give someone access in a moment of critical need.

MANAGING PRIVILEGED SESSIONS: SOMEWHAT MORE THAN JUST CRITICAL

It can't be stressed enough: with no ability to monitor what is being done with your privileged passwords, the work you've done up to this point to secure them is basically moot. If you're going to do the work to identify where your privileged passwords are used, centrally secure the passwords, and establish rules and workflows to provide access approval, you need to provide this last step in ensuring passwords and their use remains protected.

In the next, and last, chapter, we'll discuss the need to analyze the data collected through session management in an effort to identify and reduce risk around the use of privileged passwords.

CHAPTER SIX

PRIVILEGED THREAT ANALYTICS

The whole point of taking your basic password management and implementing additional capabilities is to heighten the security stance of your IT organization. Why? The obvious answer is to protect against threats, both internal and external. Both are a reality today – and not just for the Home Depot's and the Target's of the world. Organizations of every size are susceptible when they allow access to sensitive or critical company data, applications, and systems by means of privileged passwords.

Conceptually, if you've implemented the previous capabilities, you have created an environment where passwords are made available from a secure repository to only a select group of individuals, allowing very specific types of access, to certain systems. You're able to define how those individuals can access systems, and even facilitate the session. Lastly you're able to monitor those sessions, creating an audit trail of actions performed.

But, even with all that in place, who has time to be on the lookout for threat-related activity?

You do need a way to monitor a user's behavior over time and have a way to look for anomalies that could indicate a threat.

Think about it – with everything up to this point implemented, when it comes to your employees, you need to either decide you trust people, or not. And, generally, this isn't a bad thing. Most employees are trustworthy. But it's the life events (like a divorce, money problems, gambling debt, being passed up on a promotion, etc.) that put external pressure on an individual and increase the possibility they could perform an act harmful to the organization.

External threats come in both opportunistic and

targeted flavors. And you'll never know when they're about to hit.

So, are you supposed to review every action performed with a privileged password?

The answer is a definite no (you don't have the time anyway), but you do need a way to monitor a user's behavior over time and have a way to look for anomalies that could indicate a threat.

Unfortunately, there's no real efficient way to do this manually. But there are some specific parts of privileged password activity that need to be analyzed:

1. **Changes In Use** – The simple act of logging on to a server at 2am should initially raise anyone's eyebrows, even if it ends up that someone was addressing a late night problem. Watching for outliers in the frequency of use, the time of day or day of week, and even duration of the session can indicate an issue. While not super sexy, some of this can be accomplished with Windows servers using logon events within Event Log data.

2. **Changes In Access** – Tying changes in the Operating System configuration, such as an opened port to the Internet, could also be a red flag. This is obviously a tough one, given that you need a rather specific kind of solution in place to not only detect these kinds of changes, but also do so pretty close to real time to be effective.

3. **Changes In Behavior** – This last one is even more impossible without a solution, as it pertains to watching the actions of all privileged password use across multiple users, and determining when an individual deviates from the "norm" established by the pack. Not sure what that means? Think of it this way – let's take just logon times. From a big picture perspective, let's say everyone, for the most part, logs on once a day during business hours, with a logon every other Saturday and one more on a weekday night after 10pm. Not exactly, but more or less, that is the "norm" for all privileged password use. Then someone starts logging in three times a week at night at 7pm. It's an outlier behavior that doesn't fit the norm. It's this kind of analysis of

behavior changes that's necessary to identify a threat.

Threat analytics involves more than just the three factors above, but you do get a basic idea of why it's a necessary part. After all, even though there are a number of possible checks in place to ensure proper use of a privileged password, the very reason all those checks are in place is why you also need analytics.

THE ONLY USE CASE FOR ANALYTICS YOU NEED: PRIVILEGED USE ITSELF

While organizations need to trust IT to keep the business running, there still needs to be someone watching the watchers. With every use of a privileged password that occurs, the company's risk increases. That is why analytics is so important.

Unfortunately, the most you will be able to gather on your own is simply a basic set of information, such as logon times and perhaps a few other factors. The next step of correlating that data set with others elevates the basic information to a level of intelligence where you begin to see trends. It would take some pretty fancy database churning to do that yourself. But it's analytics that takes many more data sets using complicated algorithms, and turns that intelligence into insight, where decisions can be made and actions can be taken.

While an obviously necessary part of ensuring the security and integrity of privileged password use, this is just not feasible outside of using a third party solution.

CONCLUSION

After reading about, in essence, the possibilities of where you can take password management, it sheds some light on how minimal in approach just basic housing of passwords for check-in and out really is. While it does centralize your passwords and keep them documented, it never moves you from controlling the password value to actually managing password use.

The good news is you're already doing the basics. The next step is to determine which of the 6 critical password management capabilities you should be working to implement next. With each added capability, your capacity to control passwords, who can access them, when and how they can be used, and your awareness of what actions are being taken grows.

It's important to note that not every capability applies to your organization, but it's likely there's more than 1 of the 6 that definitely do. So, take a look at your current password management solution, see what's possible to extend its value, and work to create an environment where security is established, compliance is maintained, information elevates to insight, and the focus shifts from control to true management of privileged passwords.