Best Practices for

Securing Privileged Accounts
## Contents

1 **Introduction** .................................................. 1

2 **Risk management** .............................................. 2  
   2.1 Baseline risks .................................................. 2  
   2.2 Securing privileged accounts .................................. 3  
   2.3 New risks once automation manages access to privileged accounts 4  
   2.4 Protecting the privileged access management system .......... 4  

3 **Server infrastructure** ........................................... 6  
   3.1 Server numbers and placement .................................. 6  
   3.2 Database type and placement .................................... 7  
   3.3 Use of firewalls .................................................. 8  
   3.4 Virtual vs. physical servers ..................................... 10  
   3.5 Impact on network and storage – randomizing passwords .......... 10  
   3.6 Impact on network and storage – session recording .......... 11  
   3.7 Server sizing ..................................................... 12  
   3.8 Load balancing and replication ................................... 12  

4 **When and how to randomize privileged passwords** ............... 15  

5 **Securing access disclosure** ..................................... 16  
   5.1 Identification, authentication and authorization .................. 16  
   5.2 Access disclosure mechanisms .................................... 18  
   5.3 Concurrent disclosure (checkin/checkout) ......................... 19  

6 **Recording login sessions** ........................................ 21  

7 **Reporting on access disclosure** .................................. 22  

8 **System monitoring and maintenance** ............................. 23  
   8.1 Allocating staff to monitor and maintain the system .......... 23  
   8.2 Monitoring system health ......................................... 24  

9 **Configuring target system integrations** .......................... 26
Best Practices for Securing Privileged Accounts

10 Production migration 28

11 API Considerations 29

12 Summary 32

APPENDICES 33

A Hitachi ID Privileged Access Manager Overview 34
1 Introduction

This document describes the business problems which privileged access management systems are intended to address. It goes on to describe best practices for defining and enforcing policies regarding discovering systems on which to secure access to sensitive accounts, updating and storing privileged passwords and enabling access to privileged accounts.

Privileged accounts include administrator accounts, embedded accounts used by one system to connect to another and accounts used to run service programs.

*Hitachi ID Privileged Access Manager* is a security product that enables organizations to define and enforce policies regarding access to privileged accounts on a variety of systems. It enables organizations to control what users and programs have access to what privileged accounts, to control when such access is allowed and how access is activated and deactivated. Access rights may be permanent or temporary. All access is logged and subject to audit reports.

The bulk of this document is general in nature – applying equally to any privileged access management system. A few sections are specific to *Privileged Access Manager* and are identified as such.

Look for the marks throughout this document to find best practices.
2  Risk management

2.1  Baseline risks

Organizations face significant security exposure in the course of routine IT operations. For example, dozens of system administrators may share passwords for privileged accounts on thousands of devices. When system administrators move on, the passwords they used during their work often remain unchanged, leaving organizations vulnerable to attack by former employees and contractors.

Other security problems related to administrator accounts include:

- When many people need to know a single password, it is difficult to coordinate changes to the password in a manner which does not - at least temporarily - make the password unavailable to some of the people who need it.

- Lockout of an administrator account can be catastrophic. Consequently, administrator passwords are frequently not subject to a policy requiring regular updates or intruder lockouts. This can make administrator passwords less secure than user passwords.

- When multiple administrators share access to a single privileged account, it is impossible to associate administrative changes with the people who initiated them. This lack of accountability may violate internal control requirements.

Some organizations manage the most secure passwords by periodically changing them, writing them down and literally storing them in a safe. This approach sounds secure but it creates its own set of business risks:

- If an administrator is called upon to repair a system outside of normal office hours, he has no easy access to the privileged password. This may significantly impair the administrator's ability to quickly and effectively fix the system in question.

- A physical disaster at one site, such as an earthquake, hurricane, fire or flood may make a privileged password database unavailable at other sites, amplifying the scale of damage.

In addition to privileged accounts used by IT system administrators, there are also privileged accounts used by one application to connect to another. For example, many web applications use a login ID and password to connect to databases, directories or web services. These accounts may have their own security risks:

- Embedded passwords are often stored in unencrypted text files. This means that an intruder who compromises the security of the operating system where an application is installed will also compromise the integrity of the network services which the application routinely connects to.

- Applications are often replicated across multiple servers, to provide higher throughput and fault tolerance. This means that multiple application servers may house their own copies of the same plaintext login IDs and passwords. This makes it very difficult to change passwords, since every change must be coordinated between a back-end system and multiple instances of a front-end system.
Finally, unattended processes on Windows systems also run with a login ID and password. This includes service accounts, scheduled tasks, anonymous access to web content and more. Many applications only work when these services have elevated privileges. This also creates business risk:

- Where the service account is a local account on the system where a service executes, any change to the service password must be coordinated with every service which uses the account. This is difficult, in part because of the variety of Windows components which may be affected – service control manager, Windows scheduler, IIS and third party software.

- Where a service program runs on a domain controller or runs using Active Directory credentials, a change to a single password in AD may trigger a need to notify many Windows components, on many computers that participate in the AD domain of the new password.

In each of the above cases, the risks can be summarized as:

- Weak password management means that the most sensitive passwords are often the least well defended.

- The need to coordinate password updates among multiple people and programs makes changing the most sensitive, privileged passwords technically difficult.

- Inability to secure sensitive passwords exposes organizations to a variety of security exploits.

- Strong, manual controls over access to privileged accounts may sometimes create unanticipated risks, such as impaired service in IT operations and escalation of physical disasters from one site to an entire organization.

- Inability to associate administrative actions with the people who initiated them may violate internal control requirements.

### 2.2 Securing privileged accounts

A privileged access management system works to mitigate the baseline risks identified in the previous section:

- The passwords associated with privileged accounts are periodically randomized. This means that they cannot be shared or stored in plaintext files.

- A combination of persistent access control rules and one-time workflow request processes support disclosure of access to privileged accounts.

- IT staff are personally authenticated prior to requesting, approving or gaining access to a privileged account. This ensures accountability for changes they may make using that account.

- Access disclosure can take a variety of forms. It does not necessarily (or normally) mean that passwords are displayed to users.

- Password changes are coordinated between back-end systems and the programs (front-end systems) that need to use them.
• Login sessions to privileged accounts are recorded, at least at the level of meta data – who connected to which account on which system at what time from what device – and in some cases in detail – screen capture, key logging, etc.

### 2.3 New risks once automation manages access to privileged accounts

Once a privileged access management system is deployed, baseline risks are addressed but new risks must be considered:

• Disclosure of privileged passwords would be catastrophic, as it would enable an intruder to impersonate any privileged user.

• Damage to the credential vault or loss of access to this database would create an operational disaster across the entire organization, since administrators would be locked out of every system.

### 2.4 Protecting the privileged access management system

• The privileged access management system's credential vault must be protected against disclosure:
  
  – The system itself must be installed on a secure platform – for example, with a minimal set of services running, in a physically secure facility, with a minimal number of people able to gain administrative access to the console, etc.
  
  – All sensitive data, and in particular all stored passwords, must be encrypted.
  
  – Encryption keys must themselves be protected in a manner that is difficult to defeat.
  
  – Authorization policies must identify the users who can sign into the system at all and control which users can gain access to what privileged accounts.
  
  – Workflow rules must control unusual requests for access, to ensure that all such requests are authenticated, validated and authorized before access is granted.

• The privileged access management system as a whole must be designed to ensure high availability:
  
  – There can be no single point of failure:
    
    • The system’s credential vault must be replicated between at least two servers.
    
    • Multiple privileged access management servers must be deployed at physically distinct sites, so that a physical disaster at one location does not escalate into a situation where access to privileged accounts is impossible from other locations.

  – In the event that a single instance of the password management system or credential vault is off-line, other instances should continue to be able to, at least, disclose access to authorized users.

    • Password changes should be suspended at other locations if new password values cannot be replicated, since storing new password values at just a single site would create a (new) single point of failure.
• The password change process should be protected against race conditions.
  – Failure to update a password on a target system should not result in the new password being stored in the credential vault.
  – In (rare) cases where it is impossible to determine whether a password update succeeded or failed, both the old and new password should be stored in the credential vault.

• The password change process should support recovery of managed systems from backup media.
  – Very old password should remain accessible in a history table, in the event that they are needed to activate a backup copy of a system.
3 Server infrastructure

This section describes how a privileged access management system should be configured, to support the business objectives of high scalability and high availability.

3.1 Server numbers and placement

Fundamentally, a privileged access management system should always be deployed with at least two servers, preferably located in two different physical sites. This arrangement prevents system failure due to component failure:

- Hardware failure on a single server.
- Network connectivity lost between users and a single server.
- A complete site is knocked offline due to a physical, network or power outage.

Multiple servers should carry on near-real-time data replication. As the number of servers grows, so too will the volume of replication traffic between them. Since servers should be at different physical sites, the database replication traffic will be carried over a wide area network. The net result of all this is that while having at least two servers is essential, having too many servers will significantly reduce overall system performance. Consequently, it is recommended that organizations deploy no more than three replicated privileged access management servers.

The next question is where the servers should be placed on the network. While it is impossible to answer this question in the general sense – no two network topologies are alike – it is possible to offer some general guidance:

- Connectivity between multiple, replicated servers should be fast, to minimize the performance penalty due to replication.
  - This means that password management servers should be placed in sites with excellent wide area network bandwidth.

- Connectivity between password management servers and devices on which they manage passwords should be fast, since the native password update protocols of some systems do not perform well if bandwidth is low or latency is high.
  - This means that password management servers should be installed in major data centers, alongside as many managed systems as possible.

- Connectivity between end users and the password management servers is typically over HTTPS, which is designed to tolerate low bandwidth and high network latency.
  - This means that password management servers do not have to be co-located with users – it is more important to place them near target systems.
3.2 Database type and placement

Whereas the preceding guidance applies to any privileged access management system, the following guidance is specific to Hitachi ID Privileged Access Manager:

- **Privileged Access Manager** supports two types of databases:
  - Microsoft SQL Server (2005, 2008) and Oracle (10g, 11g).
  - “Standard,” “enterprise” and free, “express” editions of these database servers are technically supported, but express editions should only be used in proof of concept or demonstration systems.
  - The choice of database – Microsoft or Oracle – should be based on organizational norms. Organizations that typically deploy SQL Server on Windows servers should use SQL Server, while organizations that prefer Oracle databases for mission-critical applications should use that database.

- Organizations may opt to:
  - Install the database server software on the same server as the **Privileged Access Manager** application itself (shown in Figure 2 on Page 8); or
  - Install the database server software on a dedicated server, physically and logically near each **Privileged Access Manager** application server (shown in Figure 1 on Page 7); or
  - Leverage an existing, replicated, “enterprise-scale” database server and configure each **Privileged Access Manager** server to connect to this existing infrastructure (shown in Figure 3 on Page 8).

Figure 1: Two PAM servers, each also hosting its own database

- While any of the above three configurations will technically work, Hitachi ID Systems recommends the first option:
  - Configuring each **Privileged Access Manager** server with its own, physically distinct database instance increases reliability by eliminating single points of failure (i.e., the enterprise database infrastructure and/or connectivity to that infrastructure would constitute a single point of failure).
Best Practices for Securing Privileged Accounts

Where physical (rather than virtual) servers are used to host *Privileged Access Manager*, sharing hardware between the *Privileged Access Manager* application and the database reduces hardware cost, while only minimally impacting performance.

- If an organization chooses to deploy the *Privileged Access Manager* database on separate server from the *Privileged Access Manager* application, the two should at least be on the same network segment.
  - This reduces communication delays between the application and database and consequently improves runtime performance.

3.3 Use of firewalls

A privileged access management system contains and controls disclosure of very sensitive information. This naturally raises the question of whether and how firewalls could be leveraged to increase the security of the system itself.

It is reasonable to place firewalls between end users and the privileged access management system. Assuming that the user interface runs on HTTPS on port TCP/IP 443, it is straightforward to limit inbound connections from users to just port 443.

Moreover, an application-level firewall (configured as a reverse web proxy) could:
• Optionally, terminate SSL connections from users.

• Check incoming traffic and block unreasonable HTML form payloads, cookies, etc.

A firewall between a privileged access management system and devices on which passwords are being managed can also be used. Since every type of target system may use a different protocol, the configuration of a firewall in this location could allow any connection initiated by the privileged access management system but block any connection initiated in the other direction.

Finally, a firewall may be considered between the privileged access management system and its internal database. This is actually not recommended:

• There is little or no security benefit, since the privileged access management system will need largely unlimited access to the database – to store and retrieve passwords, change configuration settings, etc.

• There is a possibility of introducing performance and reliability problems. For example, a firewall might (mistakenly) terminate an inactive database connection. A subsequent burst of activity on the privileged access management system would have to handle this error condition. This sort of problem can be quite difficult to diagnose.

An optimal firewall configuration to protect multiple PAM servers is illustrated in Figure 4.

![Figure 4: Protecting PAM servers with firewalls](image_url)
3.4 Virtual vs. physical servers

The following guidance is specific to *Hitachi ID Privileged Access Manager*:

- *Privileged Access Manager* is tested on and supports running on virtual machines, such as VMWare ESX servers.
- Virtual servers typically (but not in every configuration) exhibit 2x to 3x slower I/O performance as compared to physical servers.
- Placing the *Privileged Access Manager* application on a virtual server is therefore acceptable, subject to the ability of the virtual infrastructure to handle the required workload of running a database, auto-discovery, password changes and access disclosure requests.
- Whether the database server (Microsoft SQL Server or Oracle Database) will exhibit adequate performance on the virtual server is also an important consideration, but this is a topic best addressed by Microsoft or Oracle, rather than Hitachi ID Systems. Hitachi ID Systems's only comment is that “it works when we test it, but performs less well than hardware.”
- Putting the above into practical perspective, a system where 5,000 passwords are randomized daily can reasonably be configured to run with *Privileged Access Manager* and SQL Server on two replicated dual-core virtual servers.

3.5 Impact on network and storage – randomizing passwords

Whenever *Hitachi ID Privileged Access Manager* randomizes a password, the resulting database records (current password value, password history, log events, etc.) consume 5.1 kbytes each on Microsoft SQL Server and 3.1 kbytes each on Oracle Database.

Using the round number of 6 kbytes/password change, this means that an organization wishing to secure 5,000 privileged accounts; change passwords daily and retain archival password data for 3 years will require a database with:

- 6 kbytes x 5000 passwords/day x 365 days/year x 3 years
- = 32,850,000 kbytes of database disk space
- = 32,850 Mbytes of disk space
- = 33 Gbytes of disk space

Assuming some overhead for workflow requests, system configuration, etc. (i.e., double the space requirement to be safe), it is reasonable to configure the above system with about 60 Gbytes of disk, regardless of which database type is used.

It should be noted that the Express Editions of Microsoft SQL Server and Oracle Database are both limited to 2 GBbyte databases, which underscores the need to deploy with a standard or enterprise edition of either database in a production deployment, rather than the Express Edition (which is only suitable for test, QA, etc.).
Similarly, the amount of traffic that *Privileged Access Manager* transmits between servers to replicate the storage of changed passwords is just under 1 kByte/password. Using the same example organization, we can estimate about 5 MByte/day of replication traffic associated with password updates - though probably more than that will be needed to replicate workflow requests, login audit records, access disclosure, etc. – adding up to about 100 MByte/day total replication traffic.

The above bandwidth is strictly to handle replication between servers. Additional bandwidth is consumed between *Privileged Access Manager* and target systems (to randomize passwords) and between end users and *Privileged Access Manager* (to request and acquire privileged access):

- To reset a single password on AD or Unix: about 10 kbytes.
- To list all users in AD, so they can sign into the privileged access management system: 500 kbytes for every 1000 users.
- To list all computers in AD, so they can be automatically managed: 500 kbytes for every 1000 computers.
- To list local service and administrator accounts on a single Windows computer: under 100 kbytes (assuming just a few accounts are found).
- To request access to a privileged account from the privileged access management system web UI: under 500 kbytes.

Total network impact can be estimated based on the rough metrics above, multiplied by the workload projected for the system.

### 3.6 Impact on network and storage – session recording

When *Hitachi ID Privileged Access Manager* is used to record login sessions while users are connected to privileged accounts, there are essentially two data streams – “low intensity” data such as keyboard events, copy buffer data, window titles and so on and “high intensity” data principally consisting of screen captures and (where enabled) webcam pictures.

Of these, the “high intensity” data stream is far larger so will be considered exclusively. Typical data streams are as follows:

1. 10kbytes/sec per monitor per active user session.
2. The data is stored on the filesystem, not the database.

Assuming that 100 sessions will be recorded concurrently for 8 hours/day, 220 days/year with data retained for 7 years, this amounts to:

1. Network bandwidth per user PC: 100kbits/sec.
2. Total bandwidth to all *Privileged Access Manager* servers: 10Mbits/sec.
3. Network storage across all Privileged Access Manager servers:

(a) 30 Gbytes/day
(b) 6.4 Tbytes/year
(c) 45 Tbytes in archive

3.7 Server sizing

The preceding discussion is helpful when deciding where to place Hitachi ID Privileged Access Manager servers and databases. This leaves open the question of how to configure the hardware for each server. Following is a reasonable configuration, which attempts to balance performance with cost given component costs as of January, 2010:

- Quad core CPU.
- 4GB RAM.
- 2 600GB 10K RPM serial-attached SCSI disks in a mirrored configuration.
- Gigabit NIC.
- Redundant power supply.

A single server configured like this can reasonably change (randomize) at least 100,000 passwords on target systems every 24 hours while concurrently servicing at least 100 concurrent, interactive user sessions.

In any deployment, at least two such servers should be deployed, with each server housing a complete database replica and the Privileged Access Manager software. The servers should be installed at different sites.

A virtual machine configuration should be configured with similar disk, I/O, CPU and memory capacity.

3.8 Load balancing and replication

Given that at least two privileged access management systems are deployed and assuming that both servers are active at all times, the next question is how to configure load balancing so that users can access both servers.

Load balancing can be accomplished using a variety of mechanisms, including:

- Associating multiple IP addresses (i.e., one per server) with a single DNS name, as illustrated in Figure 5 on Page 13.
  - For example, when a user attempts to connect to the URL https://pam.acme.com/ the DNS address pam.acme.com may resolve to one of two different IP addresses, chosen at random.

- Using a reverse web proxy in the same manner as a load balancing device (i.e., same strategy as above but using the TCP rather than IP level protocol), as illustrated in Figure 6 on Page 13.
• Directing all connection attempts to a single IP address, but using a load balancing network device at that address to forward connections to multiple servers through some load distribution algorithm, as illustrated in Figure 7 on Page 13.

![Diagram 5: Load balancing using multiple IPs on a single DNS name](image)

Figure 5: Load balancing using multiple IPs on a single DNS name

![Diagram 6: Load balancing with a reverse web proxy](image)

Figure 6: Load balancing with a reverse web proxy

![Diagram 7: Load balancing by routing to different IP addresses](image)

Figure 7: Load balancing by routing to different IP addresses

Each of these techniques will work. DNS may be preferable since it requires no special infrastructure and – depending on the type of DNS server software used – may be configurable so that the server IP address returned from each DNS query is chosen to be the server closest to the requesting user.

This is best illustrated with an example:

• Consider an organization with two data centers – one in New York and another in London.

• Assume that a privileged access management server is deployed to each data center, with the following DNS names and IP addresses:
- pam-nyc.acme.com at 10.10.10.1
- pam-lon.acme.com at 10.20.20.1

- The DNS server is configured to resolve the DNS names of both servers to the appropriate IP addresses.

- The DNS server is configured to return a PTR record to either pam-nyc.acme.com or pam-lon.acme.com when requests are made for pam.acme.com.
  
  - The choice of PTR records to return is optimized based on the IP address of the DNS client (i.e., London or NYC).
  
  - In the event that either PAM server is offline, NYC users can still request pam-lon and London users can still explicitly request pam-nyc.
  
  - In the event that an entire site is offline, the default load balancing algorithm will resolve the still-accessible, local PAM server.

Regardless of the load balancing technology used, sessions from a given client to a given PAM server should be “sticky” in the sense that the same PAM server will be used throughout the session. This is important as it eliminates the need for multiple PAM servers to replicate session state data, so significantly lowers the need for bandwidth.
4 When and how to randomize privileged passwords

The entire premise of a privileged access management system is to secure privileged accounts by periodically scrambling their passwords, so that current password values are not known to users or programs until and unless they are actually needed and that disclosure is authorized and logged.

This premise raises an obvious question: when should passwords be randomized and how should the random passwords be composed?

- Since both changing passwords and strong, random password values are inexpensive in terms of disk space and network bandwidth, it seems reasonable to change privileged passwords often – for example, every 24 hours.

- A given password should not be changed if either:
  - It is in use – by users or programs.
  - It is impossible to replicate storage of the new password value to (e.g., network is down, etc.) since the new password, stored in a single database, would then constitute a single point of failure.

- Passwords should also be changed after each administrator session is finished. For example, if an administrator “checks out” a password for an hour and “checks in” the the same password 30 minutes later, then the privileged access management system should immediately randomize the password. This ensures that any subsequent changes made to that system were not made by the administrator in question – reducing the time window for which the administrator is held accountable.

- Random passwords should be long and complex enough to be impractical to guess (by a human being) or crack (by a machine). Passwords should not be arbitrarily long, however – they should be short enough for a human being to be able to quickly write down and type, in cases where actual password disclosure is required, such as when login is required to the console of a server which is offline.
  - For example, a 16-character random password constructed from lowercase letters, uppercase letters and digits may have any of $4.76 \times 10^{28}$ combinations – in other words, impervious to guessing attacks.
  - As it happens, most systems (but not older IBM mainframes, unfortunately) support passwords in this format.
5 Securing access disclosure

5.1 Identification, authentication and authorization

A privileged access management system’s job is not only to randomize and store privileged passwords but also to connect users and programs to privileged accounts. Otherwise, the privileged accounts whose passwords were randomized would become inaccessible.

Access disclosure must be controlled:

- Users and programs which would like to gain access to a privileged account (“clients”) must first identify themselves.
- Once identified, clients must authenticate themselves – i.e., prove that they really are who they claim to be.
- The privileged access management system must determine which privileged accounts an authenticated client is authorized to access.
- The privileged access management system must record all activity – identification, authentication, authorization and access disclosure, to create a trail of accountability.

Following are some best practices for identification, authentication, authorization and audit:

- Introducing a new set of user identifiers or passwords would just add complexity and is strongly discouraged.
- Assuming that organizations which intend to deploy a privileged access management system already have a corporate user directory, such as Active Directory, it makes sense to leverage unique user identifiers from this directory to identify users of the privileged access management system.
  - In the case of a single AD domain, the `samAccountName` or `userPrincipalName` attributes can be used.
  - In the case of multiple AD domains or other directories, a domain-qualified identifier such as a fully qualified e-mail address may be more appropriate.
- Once a (human) user has been identified, he must also be authenticated:
  - The same directory which was used to identify users does not have to be used to also authenticate them, so long as login accounts on different systems can be reliably correlated.
  - In organizations with a low security threshold, Integrated Windows Authentication (IWA) can be used, for example using Kerberos tokens. It should be noted that since the user may have authenticated hours earlier and may have walked away from his workstation, this is only as secure as the physical security of the user’s location plus the time duration of inactivity before the user’s screen saver is activated.
  - In organizations with a medium requirement for security, the user’s Active Directory password may be re-entered (i.e., HTML forms based authentication) and validated against Active Directory
or another system or application (e.g., LDAP, Lotus Notes, RACF, Oracle database, etc.). This is “fresh” authentication so is presumably a bit more secure than IWA.

– In organizations with a need for stronger security, a hardware token with a one time password technology (e.g., RSA SecurID) or a smart card may be used. These two-factor authentication technologies are less convenient to use, have difficult boundary conditions (e.g., lost or stolen token or smart card) but are quite difficult to impersonate.

• There are two basic strategies for authorizing access disclosure:

  – Users may be assigned permanent access rights. For example, a set of users may be defined who are allowed to access local administrator accounts on a set of Windows servers whenever they need to.

  – Users may request temporary access rights. For example, a programmer may request access to a production application server for a four hour interval, to perform a version upgrade or assist with troubleshooting a production problem.

• Permanent access rights are best accomplished using access control lists (ACLs):

• Individual users are placed into user groups.

  – This can be offloaded to an existing system, such as Active Directory.

• Individual managed systems where passwords to privileged accounts are managed are attached to managed system policies.

  – This can be automated using expressions based on data such as server name, IP address, OS type and patchlevel, etc.

• Designated groups of users can be assigned specific rights to systems attached to designated policies.

  – For example, users whose AD account is a member of IT_WINDOWS_OPS are able to retrieve passwords for the local Administrator account on servers belonging to the NYC_WINDOWS_SERVERS policy.

• Temporary access rights are best accomplished using a request workflow:

  – A request has at least three participants (one person may take on more than one of these roles):

    • A single requester, who fills in the request to disclose access.

    • A single recipient, who will get temporary access if the request is approved.

    • One or more authorizers, who are chosen based on the identity of the requester, the identity of the recipient and the managed system(s) identified in the request.

  – Requests may require entry of supporting information, such as a ticket number, time interval during which access is required, etc.

  – Depending on whether the list of available servers is considered to be a secret or public knowledge within an organization, the system may be configured either to allow any user to act as a requester, or only some users.
– The choice of authorizers should, in general, be based on the identities of the recipient and the managed system and privileged account being requested.

– It is best to invite more authorizers than are strictly required to approve a request. This means that if some are busy or unavailable, others may still respond.
  * For example, each managed system policy may have three authorizers assigned, but any one of them has authority to approve access disclosure.

– It is best to invite every relevant authorizer at the same time, rather than waiting for one to approve before inviting the next. This improves response time and reduces system complexity.
  * For example, if the recipient’s manager plus one of three system or policy owners must approve a request, it is best to invite them all at the same time.

– It should be assumed that sometimes authorizers will be unavailable (time off, in a meeting, etc.). The system should cope with these situations:
  * Check authorizers’ out-of-office status on the e-mail system before inviting them to act. If they are out, invite someone else.
  * Send periodic reminders to non-responsive authorizers.
  * Invite alternate authorizers if authorizers continue to be non-responsive.

– It follows that some mechanism is needed to identify alternate authorizers if the original ones are unresponsive or known to be unavailable.
  * A common technique is to invite an authorizer’s manager to act if the authorizer is unavailable.
  * Another approach is to designate a security team to which all ignored requests are escalated.

5.2 Access disclosure mechanisms

A privileged access management system is normally configured to control access by people and programs to privileged accounts. The previous section covered authentication and authorization of users who wish to gain this access. This still leaves open the question of how disclosure is actually accomplished.

There are several approaches to disclosing access:

• Where access to a privileged account is disclosed to a human system administrator:
  – The simplest approach is to display the current (most recently scrambled) value of the password for the privileged account.
    * This is appropriate when the privileged account is on a system that is not reachable over the network, so the administrator will have to physically type the password at the system’s console.
    * When displaying a password, it makes sense to automatically remove it from the administrator’s screen after a short while, to minimize the risk of “shoulder surfing.”
– Where there is connectivity to the target system, it makes sense to avoid displaying privileged passwords entirely:

  • The main method for doing this is to automatically launch a login session (RDP, SSH, vSphere, SQL Studio, etc.) from the privileged access management system’s user interface, so that the administrator never sees the password. This approach offers administrators a single sign-on process, thereby reducing the burden of having to sign into privileged accounts through the privileged access management system and not directly.

  • Another good mechanism is for the privileged access management system to create a temporary trust relationship between the privileged account and the authorized user’s regular account. This can be done by manipulating SSH .ssh/authorized_keys files or Windows group memberships. This approach is advantageous because logs on the managed system, not only on the privileged access management system, indicate who signed on to perform administrative actions. It is also advantageous in that it offer single sign-on to Unix administrators from their Unix workstations (no ActiveX).

  • Another approach, where the above two are not available, is to place a copy of the privileged password in the administrator’s copy buffer, so that he can paste it into a login prompt and not see it at all.

  • In general, the options offered for access disclosure should be policy driven. Most users should not have an option to display passwords, for example.

• Where a service account's password is changed:

  – The privileged access management system should update the Service Control Manager, Windows Scheduler, IIS or other Windows component with the new password value, so that the next time the account in question is used to start a process, the correct password is available.

  – This mechanism needs to be extensible, since third party programs may also need to be able to start processes, and starting processes on Windows always requires a login ID and current password.

• Where an embedded application account's password is changed:

  – An API should be exposed, allowing the client application to fetch the new password when it next needs to connect to the server where a password was changed.

  – Alternately, a process on the privileged access management system should be able to “push” the new password to the application, for example by rewriting a configuration file where the password is stored.

### 5.3 Concurrent disclosure (checkin/checkout)

Since a privileged access management system is able to control disclosure of access to privileged accounts, it is also in a position to control how many people can gain access to the same privileged account at the same time. This is useful for two reasons:

• It helps avoid confusion that may arise when administrators do not coordinate their changes.
• It improves accountability by limiting the number of people who could have been responsible for a change made to a system at a given time.

With this in mind, it's reasonable to promote some best practices:

• On most systems, the limit should be one administrator at a time. This maximizes accountability and eliminates the possibility of poor coordination causing configuration errors.

• On very large systems, where there are constant administrative changes, the limit should be set higher – for example 2 or 3 administrators at a time. In this case, when the second administrator starts a session, he should be informed of the first administrator's session in the privileged access management system's user interface, while the earlier administrator should be sent an e-mail or SMS message to notify him of the new session.

With concurrency controls in place, a risk arises that one administrator will check out access to a privileged account, leave the session active and stop working (go home, leave for lunch, etc.). If another administrator needs access to the same system during the time interval when the first administrator's session is still active but the first administrator has left, then the system will be inaccessible. To mitigate this risk, it is important to set time limits on administrative sessions – for example, a 1 hour default and a 4 hour maximum. This reduces the time window during which a system is unmanageable because of an unused but still open session.

A second consideration that relates to concurrency controls is how to enforce them in the event that a password was actually displayed to the user who gained access to a privileged account? The administrator in question will still have the password, even after the password checkout time interval has elapsed. To reliably end the administrator's session, it is important to, if possible:

• Randomize the privileged password as soon as possible:
  – When the administrator pro-actively "checks in" the privileged access, or
  – When the permitted time interval has elapsed.

• If technically possible (it may not be) and acceptable to the administrators in question, terminate still-open connections between the administrator and the system in question (e.g., SSH, RDP, etc.).
  – This may only work if the privileged access management system can itself connect to the system being managed and remotely (a) enumerate and (b) terminate sessions.
  – This may not be desirable, since administrators may be working for a longer time than initially anticipated and terminating their sessions may be disruptive to their work.
6 Recording login sessions

Where a privileged access management system is used to sign users into privileged accounts, it can also be used to record their actions. In principle, session recording could use a variety of techniques – packet sniffing, desktop instrumentation, server instrumentation or a proxy connection.

In the interests of supporting as many types of protocols and client/server administrative protocols as possible, it makes sense to instrument the client rather than the server or an intermediate proxy system. Client workstations can be monitored using ActiveX technology, so no software need be installed on either user PCs or servers.

ActiveX instrumentation is only available on Internet Explorer on Windows. Where IT staff need to connect to systems from other types of endpoint devices, it makes sense to implement Citrix Presentation Manager or Windows Terminal Server systems, so that users can continue to launch connections from their accustomed endpoint devices but session recording will still work.

Given that instrumentation is used, it makes sense to do the following:

1. Limit instrumentation to connections to sensitive systems. This will reduce network and storage requirements.
2. Capture video and keyboard data, to create an audit trail of user actions.
3. Reconstruct keystroke data into text, to the extent possible, so that it can be searched.
4. Capture the copy buffer, in the event that a user pastes rather than types commands.
5. Capture window titles, process names and UI text labels where possible, to simplify search.
6. Capture webcam data periodically, as evidence that the user at the endpoint is indeed the same person assigned the ID that was authenticated into the privileged access management system.
7. Apply strict controls over access to recordings – for example with one approval to perform a search and a second approval to retrieve a given session. This reduces the risk of inadvertent data leakage.
8. Disable webcam recordings if a session is established from outside physically secure offices – e.g., from a public space or a user’s home, as this might capture inappropriate, private images.
9. Disable webcam recordings if a session is established from a non-corporate endpoint device, as there is no reason to assume that this is legal.
10. Consult legal counsel regarding what can be recorded, what sort of notification users should be given that their logins are recorded, where data may be stored, how long it can or must be retained, etc.
7 Reporting on access disclosure

A fundamental capability of a privileged access management system is to create accountability for administrators who used shared, privileged accounts. This is done by (a) logging all access disclosure and (b) reporting on this disclosure.

Reports on privileged sessions should be run in two ways:

- Periodically, using a random sample, to verify that administrators are only accessing appropriate systems and accounts.
  - This is similar in principle to spot-checking tickets on a train: one only has to check a few, random passengers in order to get good compliance for ticket purchases by all passengers.
  - The sample frequency may depend on the sensitivity of the managed systems in question. For example, frequent samples make sense for very sensitive systems and infrequent samples for lower risk servers.
- In response to security incidents or configuration problems:
  - As soon as possible, after the fact.
  - To see who could have made changes that contributed to the event in question.

An important question is who should be allowed to run reports on access disclosure? Since the reports only indicate who had access to what, but not what they did, it seems reasonable to have a default policy that allows any IT user to report on the activity of any other. This “transparent” model encourages good behaviour since administrative sessions are “public knowledge” among IT staff. A transparent policy also supports troubleshooting, since if one administrator sees a configuration problem on a system, he can quickly determine who may have made the change in question and ask them why.

The only real exception to the transparent approach to reporting is if a small team of administrators needs to make changes that are so sensitive that other administrators should not know about them. For example, if mass layoffs are being planned, including layoffs of other administrators, it makes sense to keep this secret. Since this sort of scenario is quite rare, it still seems reasonable for most organizations to have transparent administration practices by default, and only change the policy under very unusual circumstances.

Another consideration is how long to retain records of access requests, privileged access sessions, reports that were run, etc. Since disk space is relatively inexpensive, it seems reasonable to archive at least several years’ worth of data on-line.

Finally, in addition to allowing IT users to run reports (and see one-anothers’ activity), IT security auditors and corporate risk officers should be empowered to run the same reports – they should be able to see what IT staff are doing, without being able to gain access to systems themselves. In other words, the right to run reports should not be connected to the right to gain access to privileged accounts.
8 System monitoring and maintenance

8.1 Allocating staff to monitor and maintain the system

Between 1/4 and 1 full time equivalent position is required to effectively manage a production privileged access management system. The responsibilities of ongoing system management can be roughly broken down into two roles – a project coordinator and a technical system administrator.

The responsibilities of the long-term privileged access management system project coordinator include:

- Advocacy and “evangelism” to maximize the use of the system.
- Answering questions from stakeholders and users about the system’s capabilities and integration points.
- Coordinating the addition of new integrations.
- Ensuring that IT users get adequate training.
- Providing IT security and audit groups with access rights reports and/or training so they can generate their own reports.
- Measuring the impact of the system, in particular in relation to improved security and audit capabilities.
- Representing the system in IT architecture planning meetings.
- Coordinating with the software vendor to learn about new versions, to raise support incidents, etc.

The project coordinator’s skills are basically competent IT project management.

The responsibilities of the privileged access management system’s technical administrator include:

- Monitoring server health.
  - For example, CPU consumption, disk usage, network bandwidth consumption, etc.
- Monitoring event logs.
  - For example, failed updates to target systems, rejected requests to disclose passwords, problems with data replication, etc.
- Applying user interface customizations.
- Planning for and performing software upgrades.
- Adding new integrations.
- Periodic database maintenance (backup, restore, etc.)

The technical system administrator’s skills may include any of:
Best Practices for Securing Privileged Accounts

• Design:
  – Security policy.
  – Network and data architecture.
  – IT support infrastructure and processes.

• Installation, ongoing administration:
  – Windows / Active Directory administration.
  – Web server configuration and management.
  – Web applications.

• Initial integration and ongoing updates and extensions:
  – Expertise with each type of managed system (Windows, Unix/Linux, routers, etc.).
  – IT support infrastructure and processes.
  – E-mail infrastructure.

• Development of business logic:
  – Programming or scripting (e.g., Perl, VB, Java, etc.).
  – Familiarity with data sources: LDAP, RDBMS, etc.
  – Familiarity with web applications, including HTML and optionally (to support a more interactive UI) JavaScript.

8.2 Monitoring system health

A production privileged access management system should be monitored, to ensure that it is operating correctly at all times.

This includes:

• Platform monitoring:
  – Disk usage (high usage may cause error conditions).
  – Memory usage (high usage means lots of page swapping and poor performance).
  – Number of processes running (a spike may mean that some processes are not terminating correctly).
  – Number of open network connections (a spike may mean connections to target systems are not closing in a normal fashion).

• Application monitoring:
Best Practices for Securing Privileged Accounts

- Login failures by users of the application.
- Problems with the auto-discovery of computers.
- Problems with the auto-discovery of users on the corporate directory or on managed devices.
- Problems with password resets on target systems.
- Target systems which have not been successfully contacted in a long time.

• Security monitoring:
  - Users who make an unusually high number of access requests.
  - Users who make an unusually high number of login attempts (successful or failed).
  - All rejected requests for access.

Platform monitoring is most effectively handled using a standard IT infrastructure monitoring system, such as HP OpenView or Microsoft Operations Manager.

Application monitoring is most effectively handled by configuring the privileged access management system itself to send e-mails or open support incidents when events of interest happen.

Security monitoring is most effectively handled by periodically running reports against log data and sending those reports to security officers.
9 Configuring target system integrations

A privileged access management system’s value increases as the number of integrations grows. The security benefit is clearly greater if privileged passwords are secured on 1000 systems, as compared to 100.

As the number of integrated systems grows, the cost of adding, maintaining and removing integrations manually will also grow. Automation is needed to scale the system up to more than a few hundred integrations.

Automating the maintenance of integrations means automating several, distinct tasks:

- Automatically discovering target systems.
- On each discovered target system, automatically discovering:
  - Administrator accounts (this is general to all platforms).
  - Service accounts (this is a Windows-specific requirement).
- Batch-loading new integrations and their associated IDs into the privileged access management system.
- Automatically identifying defunct integrations (e.g., no response for N days) and removing them from the active password update process.

In any medium-to-large organization, workstations and servers are activated and retired daily. It therefore seems reasonable to run any auto-discovery process every 24 hours.

There are several technological approaches to discovering servers. Choice of the appropriate method depends on available infrastructure:

- For Windows systems in particular, a list is often maintained in Active Directory.
- For Unix/Linux systems in particular, a list is often maintained in DNS or in a master /etc/hosts file.
- In environments where the above techniques are either not available or not sufficiently inclusive, a TCP/IP port scan of one or more network segments is appropriate.
  - nmap is an excellent and free tool that can be used for this purpose.\(^1\)
- In environments where an inventory tracking system is in place and where it contains complete, accurate, detailed and up-to-date information about IT assets, data about devices that should be integrated can be imported from this system.

Once a system has been discovered using one of the mechanisms described above, the next step is to – initially and periodically – get a list of login accounts from that system and determine which of them qualify as “privileged” – because they are members of administrator-level groups, have a given numeric ID, are used to run services or scheduled tasks, etc.

\(^1\)Download nmap from http://nmap.org/. 

© 2015 Hitachi ID Systems, Inc. All rights reserved.
The mechanism for enumerating IDs and qualifying them as privileged varies from system to system. For example, an SSH script that checks whether a given user has a UID of 0 or belongs to groups such as wheel, root or admin can be used on Unix or Linux systems, while a program that connects over RPC and checks group membership, Windows Service Manager configuration, Scheduler configuration and IIS configuration is appropriate for Windows systems.

The frequency of enumerating privileged IDs on discovered target systems should be high – in order to detect IDs that were created in an unauthorized fashion. On the other hand, it should be low – in order to reduce the run-time of the auto-discovery process and to minimize network impact. In practice, a frequency of between once-daily and once-weekly is a reasonable compromise between these conflicting objectives.

Another question that arises when auto-discovering target systems is what credentials can/should be used when first connecting to each system. Reasonable options include:

- For Windows servers that are domain members, use a domain-level account that is able to sign into each system.
  - The privileged access management system should create its own ID locally on each system and use that on a go-forward basis. This reduces the need for each system to continue to trust a domain-administrator account with local privileges after the initial setup.

- For other types of systems, ask system administrators to create an account specifically for the privileged access management system to use, with a predictable or fixed initial password.
  - The privileged access management system should scramble this password as well as any other ones it is responsible for when it first connects to each system.

Since a privileged access management system attempts to connect to integrated systems often – to change passwords – it can be used as a coarse-grained infrastructure monitoring facility, to raise alarms in the event that a target system is unreachable. Alerts can take the form of e-mails to administrators, tickets in a help desk system, etc.

If a target system is persistently unavailable, it should be automatically removed from the regular password rotation process. This helps keep the database clean as systems are moved or retired. It should be noted that historical password data for every system should be retained – in the event that a system was offline for an extended period due to hardware problems or that it is later restored from backup media.

In any case, a report should be run regularly to identify non-responsive target systems. This will allow system administrators to match the list of non-responsive systems against a list of known-retired systems and to identify anomalies where the lists don’t match.
10 Production migration

A privileged access management system is a very sensitive part of an organization’s infrastructure. As such, it should first be deployed to a test environment and its configuration validated, before moving to production.

Once in production, it makes sense to phase use of the system in, to minimize risk due to configuration problems, software defects, etc.:

• Start in test mode – create test accounts on the first set of integrated systems, scramble their passwords very often (e.g., every few minutes) and verify that the system works as predicted. Only switch to managing passwords in production after this has been shown to work reliably.

• Test failure conditions before going live:
  – Disconnect the network adapter from one of the privileged access management servers and verify that there is no data loss.
  – Disconnect power from one of the privileged access management servers and verify that passwords can still be recovered from the other server.
  – Test database recovery procedures before the system goes into production use – for example, after turning the power back on, on the system tested in the previous step.

• Leave one administrative account on each system outside the scope of the privileged access management system. Use this as a backup in case the system malfunctions. This backup account can be retired after a few months of stable production operation.

• Start with relatively infrequent password changes (e.g., weekly) and once the system has been shown to work correctly, increase the password change frequency to daily.

• Start with access disclosure based on persistent access control rules (e.g., users in group X can gain administrative access to systems in group Y) and only introduce workflow after this basic functionality is well established.

• Start with simple integrations – e.g., Windows and Linux – and gradually add other types of systems.
11 API Considerations

An API allows a privileged access management system to secure passwords that authenticate one application when it connects to another. For example, an e-commerce application may have to sign into a database server to read inventory data and post transactions. Such connections are normally authenticated using a login ID and password.

The security problem when one application uses a password to authenticate to another is that the password may be:

- Stored in a plaintext configuration file.
- Replicated across multiple servers, each of which runs an instance of the application.
- Static (never changes), because of the difficulty of coordinating updates among multiple web and database servers.

To eliminate this problem, a privileged access management system may be used to periodically scramble the embedded account’s password. An API then allows each instance of the e-commerce application to fetch a current password value, with which it can connect to the database, from the privileged access management system. This eliminates static passwords and passwords in plaintext files but creates new challenges to overcome:

- On what platforms and for what runtime environments is the API available?
- How does an instance of the e-commerce application authenticate itself to the privileged access management system?
- How often can the e-commerce application connect to the privileged access management system before a performance bottleneck is created (at either endpoint)? If this is less often than the frequency with which the password is required, how can the password be securely cached between updates?
- What happens if the e-commerce application is unable to connect to the privileged access management system? In other words, does the privileged access management system create a new point of failure?
- When is it safe for the privileged access management system to scramble the password in question?
- Should there be just one password for an entire cluster of application servers, or should each server get its own password?
- The “e-commerce application” in our example is just that – an example. How does this infrastructure scale up to hundreds of applications and how are changes to application source code (to call the API) coordinated?

There are no “one size fits all” answers to these questions. Every organization will have its own priorities and every application will have its own constraints, leading to somewhat different answers. Following are some reasonable approaches to each of the above questions, presented with the understanding that they may or may not suit the needs of a given organization and application.
• **Platform/runtime support:**
  
  – Ideally, the API should be available to any client program, on any platform. The best way to achieve this is to use a platform-neutral API format, such as SOAP (simple object access protocol) – i.e., XML over a web service.
  
  – SOAP is accessible from every modern programming language and run-time environment. In some cases, calling SOAP is a bit complex, so “helper” libraries which wrap around the SOAP transport can reduce the overall development effort.

• **API authentication:**

  – If the client application authenticates itself to the privileged access management system using just its own login ID and password, then there is really not much benefit to the arrangement – it just replaces one (static?) password with another.
  
  – An alternative is to generate a new, random password for the client application periodically. For example, the privileged access management system may generate a new, random string that the client application must use to authenticate, whenever the client successfully authenticates. This is a software-version of one-time password (OTP) technology.
  
  – Another, complementary, approach is for the privileged access management system to check the IP address of the client application at login time. Presumably, a given application will only be running on a given server, so its IP address should be predictable and can act as an authentication factor.

• **Password access frequency and caching:**

  – Fetching a new password for every database transaction may be excessive, especially for applications that make thousands of database connections per second.
  
  – A reasonable alternative is to fetch a “fresh” password every hour or so and cache this password in the interim.
  
  – Caching raises the question of how to protect cached passwords. One option is to keep them in memory only and use an obscuring password (embedded in the application) to make them hard to extract from a core dump.
  
  – Another approach is to store cached passwords on disk. This helps with the availability scenario (below), since in the event that the privileged access management system is offline, a cached password can still be used by the application.
  
  – One suggestion for how to protect cached passwords on disk is to calculate a checksum of the executable program or script that needs a password and use that checksum to encrypt/decrypt cached passwords. Clearly what is proposed here is an effective mechanism to obscure passwords (an intruder with total control over the server in question and ample time will still be able to compromise this). It therefore makes sense to use a secret algorithm to produce the checksum.

• **High availability:**

  – If a password from the privileged access management system’s vault is required to complete transactions, then both the privileged access management system itself and ability to connect to its API become essential to the smooth operation of the application. Either these must be extremely highly available, or compensating measures must be taken to continue operations when the privileged access management system is unreachable.
As mentioned above, it makes sense to cache passwords acquired from the privileged access management system’s API and to encrypt those using a hard-to-find key, such as a checksum of the running executable.

• **Scheduling password changes:**

  – Whenever a password change is in process, there is a risk that a race condition will leave the client application holding a no-longer-valid password.

  – To avoid this problem, it makes sense to track which client applications have “checked out” a given password and for how long they are allowed to keep it. The privileged access management system should not scramble the password while it is in use and should not allow new password check-out events in the interval during which a password is being changed.

  – Moreover, password changes should be scheduled (in the time-of-day and day-of-week sense) to happen during periods when few transactions are normally processed (e.g., 3AM on Sunday morning, etc.).

• **Shared vs. per-client passwords:**

  – Given the challenges in coordinating a safe time to change passwords, plus the appeal of using an application server’s IP address as an authentication factor, it makes sense to grant each instance of an application its own credentials to connect to the privileged access management system’s API.

  – Similarly, it makes sense for the privileged access management system to scramble and securely store passwords for multiple accounts on each system. This way, each client application can connect with its own PAM credentials and fetch its own back-end (database, etc.) credentials without impairing the performance of other application instances.

• **Source code changes:**

  – The simplest approach to eliminating static passwords is to modify the source code for the application which needs those credentials, to call the privileged access management system’s API (directly or through a convenience/wrapper library).

  – If this is infeasible – for example, if the application was written by a third party who is unwilling to modify it – a wrapper program may modify the file or other location where the original application keeps its (plaintext?) password. This is better than nothing, but obviously not preferable.

  – Another approach is to modify an application’s binary, replacing a template string with the current password value, before starting the application. This reduces in-line source code control but only works if applications keep a password in their binary or configuration files. This is plausible but probably indicative of more serious security problems with the application in question.

  – In general, use of a source code control system to track changes makes sense – it will help with version upgrades, for example.

  – It also makes sense to prioritize applications and replace plaintext passwords with API calls in the most sensitive applications first.
12 Summary

A privileged access management system enables organizations to replace well-known, static and insecure passwords with frequent password changes, strong and personal authentication, fine-grained authorization logic and extensive audit logs.

Deploying this sort of system can be invasive – failure of the system, in terms of confidentiality, integrity or availability, would be catastrophic. Consequently, great care must be taken to deploy the system in a manner that is robust, fault-tolerant and secure.

This document outlined an exhaustive set of best practices intended to ensure that a privileged access management system is highly available, secure, scalable and efficient to manage.
APPENDICES
A Privileged Access Manager Overview

Overview:

*Hitachi ID Privileged Access Manager* is a system for securing access to privileged accounts. It works by regularly randomizing privileged passwords on workstations, servers, network devices and applications. Random passwords are encrypted and stored on at least two replicated credential vaults. Access to privileged accounts may be disclosed:

- To IT staff, after they have authenticated and their requests have been authorized.
- To applications, replacing embedded passwords.
- To Windows workstations and servers, which need them to start services.

Password changes and access disclosure are closely controlled and audited, to satisfy policy and regulatory requirements.

Features:

*Privileged Access Manager* secures privileged accounts with:

- **Random passwords:**
  *Privileged Access Manager* is designed to change as many as 2,000,000 passwords per day to new, random values. This minimizes the window of opportunity that hackers and former users have to compromise systems and applications.

- **Encrypted, replicated vault:**
  *Privileged Access Manager* stores randomized passwords in an encrypted and replicated vault. This protects against unauthorized access to passwords and against loss of access to data because of a hardware failure or physical disaster.

- **Many built-in connectors:**
  *Privileged Access Manager* ships with built-in integrations for over 120 systems and applications. That means that it can secure access to sensitive accounts on most servers, directories, network devices, databases and applications without customization.

- **Laptop support with a local service:**
  *Privileged Access Manager* also ships with software that can be installed on laptops running Windows or Linux. This allows it to secure access to computers that are sometimes turned off, unplugged from the network, change IP addresses or physically removed from the premises.

- **Access control policy engine:**
  Security officers set policy on *Privileged Access Manager* to control who can access which accounts. For example, Windows administrators can be granted access to local Administrator accounts, Unix
administrators can be allowed to login as root, etc. The policy engine is very flexible, as it connects
groups of administrators to named accounts on groups of systems.

• **Workflow for one-time access requests:**

  *Privileged Access Manager* includes a powerful workflow engine that allows users to request one-time
  access to privileged accounts. Requests are subject to policy (who can ask, who must approve).

  The workflow engine leverages e-mail to invite authorizers to act and a secure web form for approvals.
  Timely response is assured by inviting multiple authorizers, sending automated reminders, escalating
  requests from non-responsive authorizers to alternates and more.

• **Flexible access disclosure options:**

  Rather than displaying passwords to users, *Privileged Access Manager* can:

  – Launch RDP, SSH, SQL Studio, VMware vSphere and similar sessions, injecting passwords
    without displaying them.
  – Temporarily attach the authorized user’s Active Directory account to a local security group on the
    target Windows server.
  – Temporarily attach the authorized user’s SSH public key to the authorized_users key ring on the
    target Unix or Linux server.

• **Session recording:**

  *Privileged Access Manager* can be configured to record screen, keyboard and other data while users
  are connected to privileged accounts. The recording may be of just the window launched to connect
  a user to a privileged account or of the user’s entire desktop.

  The session recording system is tamper resistant – if users attempt to interrupt recording, their login
  sessions to privileged accounts are disconnected and an alarm is raised.

  Session recordings may be archived indefinitely and may serve a variety of purposes, ranging from
  knowledge sharing and training to forensic audits. Access to recorded sessions is secured through a
  combination of access control policies and workflow approvals, designed to safeguard user privacy.

  The *Privileged Access Manager* session monitoring infrastructure is included at no extra cost. It works
  using ActiveX components and does not require software to be permanently installed on user PCs.
  There is no footprint on managed systems and no proxy servers are used.

  Session monitoring is compatible with all administration programs and protocols, as it instruments the
  administrator’s PC, rather than network traffic. Recordings can be made of SSH, RDP, vSphere, SQL
  Studio and any other administrative sessions launched via *Privileged Access Manager*. Recordings
  can include key-logging, video, webcam, copy buffer and more, based on policy settings and without
  regard to the type of session (protocol, client tool) that was launched.

• **Infrastructure to secure Windows service account passwords:**

  In addition to managing access to administrator accounts, *Privileged Access Manager* can randomize
  passwords used to run services, scheduled jobs and other unattended processes on Windows com-
  puters. It can then notify the Windows Service Control Manager, Scheduler, IIS and other components
  of the new password, so that tasks can be successfully started in the future.

• **An API to replace static, embedded passwords:**

  *Privileged Access Manager* exposes an API that allows one application to securely acquire a password
  that will then be used to connect to another application. This mechanism is used to eliminate plaintext
  passwords in application source code or text files.
Best Practices for Securing Privileged Accounts

• **Auto-discovery:**

  *Privileged Access Manager* includes an advanced infrastructure auto-discovery system, designed to minimize both initial and ongoing configuration. This system can:

  1. Extract a list of systems from AD, LDAP or other sources.
  2. Apply rules to decide whether a given system should be managed.
  3. Apply rules to choose a security policy to apply to each managed system.
  4. Probe systems in a massively parallel fashion, to get a list of accounts, groups and services on each one.
  5. Apply rules to decide which accounts on each system should be managed.

• **Reports:**

  *Privileged Access Manager* includes a variety of built-in reports, that are used to answer questions such as:

  – What computers are on the network?
  – Which computers have been unresponsive during the past 30 days?
  – Which administrators have signed into this computer?
  – Which systems has this administrator managed?
  – Who has made a large number of requests for one-off access?

**Benefits:**

*Privileged Access Manager* improves the security of privileged accounts by:

• Eliminating static, shared, well-known passwords.
• Ensuring that former IT staff cannot access sensitive infrastructure.
• Requiring strong, personal authentication of users prior to accessing privileged accounts.
• Enforcing robust policy over who can access privileged accounts.
• Recording a detailed audit trail of privileged login sessions.

*Privileged Access Manager* reduces the cost of managing passwords on privileged accounts by automating the password change, storage and disclosure process.