Password Management Best Practices
15.2 BYOD opportunities

APPENDICES

A Sample Security Questions
A.1 Questions with textual answers
A.2 Questions with numeric answers

B References
1 Introduction

This document describes and justifies password management best practices as applied in medium to large organizations. It offers reasoned guidance to IT decision makers when they set security policy and design network infrastructure that includes passwords.

The guidance in this document is focused on how to best manage user passwords. It is not intended to address the special challenges and techniques that arise when managing privileged passwords, used to sign into administrator, service and embedded accounts.

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

The remainder of this document is organized as follows:

- Why do we still use passwords? – explaining why passwords are unlikely to be entirely replaced in the near future.
- User authentication and passwords: Background information including terminology and an overview of both passwords and other types of credentials.
- Security threats
  An overview of how password security can be compromised.
- The human element
  A reminder that human behaviour has to be considered when designing the security of any system, including one for managing passwords.
- Composing hard-to-guess passwords
  How to estimate password strength and guidance for composing hard-to-compromise passwords.
- Unicode and other non-Latin passwords
  Password composition for users whose first language is not English, and who may normally use multi-byte text input.
- Changing and reusing passwords
  Guidance regarding when to change passwords and whether to allow users to choose the same password twice.
- Keeping passwords secret
  A reminder that passwords are supposed to be secret, and how to help users keep them that way.
- Intruder detection and lockout
  The role that intruder lockouts have in ensuring password security, and guidelines for a balance between keeping out attackers and not bothering legitimate users.
- Encrypting passwords in storage and transit
Password Management Best Practices

The need to encrypt passwords, in motion and at rest.

• **Synchronizing passwords**
  The pros and cons of using the same password value on multiple systems and applications.

• **Single sign-on**
  The pros and cons of replacing multiple login prompts with a single, shared login process.

• **IT support for forgotten and locked out passwords**
  How to assist users who forgot or locked out their password.

• **Mobile devices: challenges and opportunity**
  Enabling access to the password management system from user phones, dealing with passwords cached by various apps on user devices and leveraging phones to resolve login problems and to increase the strength of authentication processes.
Why do we still use passwords?

The end of passwords has been predicted for decades. Biometrics, smart cards, one time password tokens and more have been offered up as alternatives and many of these are gaining market share.

In reality, passwords are likely to remain popular for a long time:

1. Passwords are cheaper to deploy than any alternative, though supporting passwords (forgotten, locked out) can be costly.

2. Some types of credentials can only be used on compatible devices and in certain circumstances:
   
   (a) Smart cards plug into card readers, which are mainly made for PCs (desktops and laptops) – rarely for tablets or smart phones.
   
   (b) One time password tokens only work where there is a network connection – this makes them unsuitable for signing into devices which are sometimes off-line.
   
   (c) Every kind of biometric requires a sensor – finger print reader, retina scanner, camera, microphone, etc. Not every device a user might want to sign into has the requisite sensors.
   
   (d) For every biometric, there are some users who physically cannot enroll – amputees, people whose fingers are too small, people with retinal or iris damage, etc.

3. In many cases, credentials other than passwords are combined with passwords to create stronger authentication. For example, tokens and smart cards are commonly combined with PINs (just numeric passwords).

4. Many solutions marketed as replacements for passwords really just externalize the login process out of an application, to a shared infrastructure, which in all likelihood does use a password. This is true of Kerberos, LDAP authentication, federation with OAuth and SAML, etc.

5. Many legacy applications are simply incompatible with any other credentials – user logins are with an ID and password, and nothing else.

Only when most applications can externalize their login process will organizations be able to seriously contemplate the end of passwords.
3 User authentication and passwords

3.1 Definitions

- **Identification**
  The unique identifier that a user types to sign into a system or application is that user’s login ID on that system.

- **Authentication**
  Authentication is a process by which a user proves his identity to a system – normally when logging in.

- **Authentication factor**
  An authentication factor is something a user presents to a system in order to prove his identity. It may be something he (and hopefully only he) knows, or proof of possession of a physical object, or a measurement of some physical characteristic (biometric) of the living human user. In other words, something the user knows, or something he has, or something he is.

- **Multi-factor authentication**
  Multi-factor authentication means authentication using multiple factors. For example, a user might sign into a system with a combination of two things he knows, or a combination of something he knows and something he has, or perhaps something he knows, something he has and something he is.

  The premise is that adding authentication factors makes it more difficult for a would-be attacker to simulate a legitimate authentication and consequently impersonate a legitimate user.

- **Strong authentication**
  Strong authentication refers to an authentication process which is difficult to simulate. It may be based on use of multiple authentication factors or use of a single but hard-to-spoof authentication factor.

3.2 Types of credentials

There are three types of credentials:

<table>
<thead>
<tr>
<th>Authentication Factor</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Secrets:</strong> Something you know.</td>
<td>Secret information known only the user.</td>
<td>A password or PIN.</td>
</tr>
<tr>
<td><strong>Tokens:</strong> Something you have.</td>
<td>A physical device possessed only be the user.</td>
<td>A one time password (OTP) token, smart card or app on a smart phone.</td>
</tr>
<tr>
<td><strong>Biometrics:</strong> Something you are.</td>
<td>A unique, measurable characteristic of the user.</td>
<td>Voice print verification, fingerprint, vein pattern, retina or iris scan.</td>
</tr>
</tbody>
</table>
Generally, it is more secure but less convenient to combine multiple credentials, preferably of different types, to sign into a system or application. For example, use a password plus a token, or a biometric plus a PIN. This is called multi-factor authentication.

The strengths and weaknesses of different types of credentials can be generally described as follows:

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Secrets</th>
<th>Tokens</th>
<th>Biometrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reliable identification?</td>
<td>Good</td>
<td>Very good</td>
<td>Excellent</td>
</tr>
<tr>
<td>Requires client-side hardware</td>
<td>No</td>
<td>Sometimes</td>
<td>Yes</td>
</tr>
<tr>
<td>Requires client-side software</td>
<td>No</td>
<td>Sometimes</td>
<td>Yes</td>
</tr>
<tr>
<td>Typical deployment cost/user</td>
<td>0</td>
<td>$50</td>
<td>$100</td>
</tr>
<tr>
<td>Works with legacy systems</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Due to cost and compatibility with legacy systems, the most popular type of credential remains the password.

Moreover, as non-password credentials are deployed, they are commonly combined with passwords or PINs. For example, tokens are secure – unless they are stolen, in which case the person who stole the token can impersonate the legitimate user. This is remedied by requiring the user to sign into systems by entering the pseudo-random code displayed on their token and also enter a password or PIN, which a thief would not know.

The remainder of this document discusses how to best manage passwords, to maximize security and minimize cost of ownership. The guidelines here apply primarily to passwords used as a single authentication factor, but can also be applied to passwords used as a second authentication factor, in conjunction with biometrics, tokens or smart cards.
4 Security threats

Passwords are simply secret words or phrases. They can be compromised in many ways:

- User devices may be compromised with a keylogger or similar malware, and disclose user input to an attacker.
- Users may write them down or share them, compromising secrecy.
- Passwords can be guessed, either by a person or a program designed to try many possibilities in rapid succession.
- Passwords may be stored or transmitted over a network either in plaintext or encoded in a way which can be readily converted back to plaintext.

Each of these vulnerabilities create an opportunity for an attacker to acquire password values and consequently impersonate users.

Conversely, if passwords are managed securely by users and if password systems are constructed so as to prevent brute-force attacks and inspection or decryption of passwords in transit and in storage, then passwords can actually be quite secure. This document will describe some of the mechanisms for securing passwords.
5 The human element

Users in a large organization frequently have many passwords, each protecting their account on a different system or application. Users are people, not machines, so their ability to securely manage passwords is intrinsically limited. In particular, it is hard for most people to remember:

- Complicated passwords.
- Many different passwords.
- Passwords that change frequently.
- Passwords for systems that are used infrequently.

Without these constraints, password security would not be a problem and there would be no market for other authentication technologies. For example, if every password was changed every day, was remembered perfectly without being written down and consisted of 100 randomly chosen letters and digits, there would really be no need for authentication technologies other than passwords.

Effective password management is therefore taken to mean password management that is secure, user friendly and supportable within the confines of both technology and human behavior.

When people have trouble remembering their passwords, they usually resort to one or more of the following:

- Write down their passwords – and reduce the security of systems to the security of their physical building, desk or wallet.
- Forget their passwords – and require frequent assistance from an IT help desk.
- Choose very simple, easily compromised passwords.
- Reuse old passwords whenever possible.

Clearly, sound password management practices must take into consideration human limitations, to minimize bad behavior.
6 Composing hard-to-guess passwords

As outlined in Section 4 on Page 6, one of the weaknesses of passwords is that they can be guessed. While a human may give up after ten or a hundred guesses, software is available to automate the guessing process and try millions of possible passwords per second.

Such brute-force attacks are not made against login screens, but rather against copies of the password database, where passwords are normally stored in a hashed (encrypted, but not reversibly) format. The first line of defense is therefore to prevent attackers from getting a copy of a password database.

Attackers who can gain access to a password database can choose from many powerful tools to attempt brute-force attacks, including:

- L0phtCrack: http://www.l0phtcrack.com/ for Windows.

To combat password guessing attack, users should pick hard-to-guess passwords. One way to do this is to ensure that the set of all possible passwords is too large to search thoroughly and then to eliminate probable guesses, which an attacker might try first.

The number of possible password combinations is calculated by taking the number of legal characters in a password and raising that number to the number of characters in the password. The possibilities for some typical combinations are shown below:

<table>
<thead>
<tr>
<th>Character set</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-9</td>
<td>1.00e05</td>
<td>1.00e06</td>
<td>1.00e07</td>
<td>1.00e08</td>
<td>1.00e09</td>
<td>1.00e10</td>
</tr>
<tr>
<td>a-z</td>
<td>1.19e07</td>
<td>3.09e08</td>
<td>8.03e09</td>
<td>2.09e11</td>
<td>5.43e12</td>
<td>1.41e14</td>
</tr>
<tr>
<td>a-z,0-9</td>
<td>6.05e07</td>
<td>2.18e09</td>
<td>7.84e10</td>
<td>2.82e12</td>
<td>1.02e14</td>
<td>3.66e15</td>
</tr>
<tr>
<td>a-z,0-9,3 punct</td>
<td>9.02e07</td>
<td>3.52e09</td>
<td>1.37e11</td>
<td>5.35e12</td>
<td>2.09e14</td>
<td>8.14e15</td>
</tr>
<tr>
<td>a-z,A-Z</td>
<td>3.80e08</td>
<td>1.98e10</td>
<td>1.03e12</td>
<td>5.35e13</td>
<td>2.78e15</td>
<td>1.45e17</td>
</tr>
<tr>
<td>a-z,A-Z,0-9</td>
<td>9.16e08</td>
<td>5.68e10</td>
<td>3.52e12</td>
<td>2.18e14</td>
<td>1.35e16</td>
<td>8.39e17</td>
</tr>
<tr>
<td>a-z,A-Z,0-9,32 punct</td>
<td>7.34e09</td>
<td>6.90e11</td>
<td>6.48e13</td>
<td>6.10e15</td>
<td>5.73e17</td>
<td>5.39e19</td>
</tr>
</tbody>
</table>

Users should choose their passwords from the widest possible set of characters, subject to the constraints of the systems where those passwords reside. For example, most mainframes do not distinguish between uppercase and lowercase and only allow three punctuation marks (fourth row in the table above).

The policy should be based on the smallest acceptable set of possible password values. This, in turn, depends on how fast an attacker can check if a possible password is valid; how much time an attacker has to attack a given password and how log we want the attacker’s probability of success to be.

This is best illustrated with an example:

1. Assumptions:
Encrypted passwords are not available to an attacker and consequently passwords can only be guessed at a rate of 100/second (this number is high / conservative for an over-the-network attack).

An attacker can make an unlimited number of guesses without being detected (this is easy to fix).

Passwords are changed, at most, every 90 days.

2. Objective: the attacker’s probability of success should be no more than 1%.

3. Technical constraints: passwords may only contain lowercase letters, uppercase letters and digits.

4. Calculation:

   (a) An attacker can make $90 \times 24 \times 60 \times 60 \times 100 = 777,600,000$ guesses before a password will expire.

   (b) There should be $77,760,000,000$ possible passwords – about $8 \times 10^{10}$.

   (c) From the above table, a 7 character password is sufficient, so long as users don’t pick easily guessed values (e.g., their name or a dictionary word).

In practice, assuming that encrypted passwords are not available to an attacker and that passwords cannot be captured or decrypted in transit, the following policy is often reasonable:

- To ensure that the search space is sufficiently large:
  - Passwords must be at least seven characters long.
  - Passwords must contain at least one lowercase letter, at least one uppercase letter and at least one digit.
  - If technically possible, passwords must contain at least one punctuation mark, so long as there are many (10 or more) available punctuation marks.

- To eliminate trivial passwords, passwords should not:
  - Contain the user’s name or login ID.
  - Contain a dictionary word, in any language that users can reasonably be expected to know.
  - More than two paired letters (e.g. abbcdde is valid, but abbbccd is not).

Some of these rules merit further discussion, since a naive implementation of the rule can actually reduce password security. Consider the rule “password should not contain a dictionary word.” The letters “a” and “l” are legitimate English words, and if the rule is enforced using a dictionary that contains these, then in effect we have reduced the set of available letters from 26 to 24. To avoid this sort of problem, rules of the form “shall not contain” should be fine-tuned:

- Ignore words with 3 or fewer characters, since including such words in a dictionary check may actually make passwords easier to guess, by eliminating too many legitimate password values.

- Consider dictionary words without regard to case. “Hello12” should not be a valid password, even if the dictionary contains “hello” but not “Hello.”
7 Unicode and other non-Latin passwords

Most password composition rules focus on how passwords composed of Latin characters (i.e., “English language” letters), digits and punctuation marks are composed.

Unfortunately, English is the native language of no more than 400 million people and is spoken by no more than 1.5 billion people – out of a global population of over 7 billion people. In other words, even if other European languages that use a similar character set are included, a focus on Latin characters ignores at least 2/3 of the world’s population.

This not just an academic problem. Many organizations are global in nature and include users who speak a variety of languages. A system or application whose users reside around the world needs to enforce a password policy that makes sense for all of them, not just for those who happen to prefer English.

Password management for multi-lingual users raises some important considerations:

1. Most users, in most countries, have keyboards that include most Latin characters. This means that they are able to type passwords composed of “English” letters, even if they don’t read or write the language.

2. Not every system or application supports non-Latin characters in passwords. When composing an enterprise password policy, it’s important to consider the limitations of in-scope systems and applications. Failing to do this can yield a policy that works on some systems but not others.

3. In countries where Chinese, Japanese or Korean (CJK) is spoken, text input often involves a display component. For example, a Chinese user might type the pinyin representation of a character and the input software will display a list of matching characters. The user will then choose the desired character and start on the next character.

   This pattern is normal for input of text in the CJK languages but is problematic for passwords since password input should not be visible to other people in the physical vicinity.

   Most CJK users avoid this problem by choosing passwords that consist only of Latin characters – something that may surprise native English speakers.

Some best practices follow from this:

1. Encourage or require users to restrict their passwords to Latin characters, both for compatibility and to avoid input methods which display the characters users type.

2. Do use a dictionary lookup to ensure that new passwords are hard to guess, and do include dictionaries of non-English words represented in Latin character sets (e.g., Pinyin, etc.).
Changing and reusing passwords

Over time, passwords may be compromised in many ways, including:

- Users may share them with friends or coworkers.
- Users may write them down and they may subsequently be exposed.
- Passwords may be guessed, either by humans or automation.
- The servers that store passwords may be compromised and their passwords acquired by an attacker (encrypted, hashed or plaintext).
- The networks over which passwords travel, between a user’s endpoint and the system into which the user signs in may be compromised, again leading to compromise of a password.
- Users may be tricked into revealing their passwords (phishing).
- The help desk may be tricked into giving an attacker a valid password (social engineering).

To limit the usefulness of passwords that have been compromised, a best practice is to change them regularly. A common rule in many organizations is to force users to change their passwords when they log in, every 60 or 90 days.

In general, users should be required to change their passwords regularly. The password expiry interval should not be longer than 90 days.

For the same reasons, users should not reuse old passwords, as they may already have been compromised. Many systems support this by recording some representation of old passwords and ensuring that users cannot change their password back to a previously used value.

When password history is enforced, users may figure out the number of passwords stored for this purpose. As this number is usually 10 or fewer, a user who would prefer to keep his password the same may make a series of consecutive password changes where the final password value is the same as the first one.

To defeat such “smart” users, some systems also enforce a password rule that limits the number of password changes that a user can make in any given day. By making the process of defeating password history take several days, users are discouraged from trying to reuse old passwords.

A better approach, though not available on all systems, is to simply maintain an unlimited number of entries in each user’s password history. Since disk storage is inexpensive, this approach is practical on modern systems. With this approach, users can change their passwords as often as they like — and still never be able to reuse old ones.
9 Keeping passwords secret

Passwords are intended to reliably differentiate between the authentic user and impostors. As such, they must be secrets – known only to the user they identify.

Users frequently behave in ways that lead to password disclosure. In particular, users may:

• Choose passwords which are easily guessed – so are not really secret.
• Share their passwords with coworkers, friends or family.
• Write down their passwords and place the written password near their computer or in a supposedly private place like a wallet.

A comprehensive password policy should explicitly forbid these behaviors and specify negative consequences to users who violate the policy.

To help users comply with an effective password policy, user friendly password management tools and processes should be provided. Key among such aids to users are password synchronization and single sign-on.

With either synchronized or automatically filled-in passwords, users have fewer passwords to remember so they are less likely to write down their passwords.
### 10 Intruder detection and lockout

Many systems can detect repeated attempts to sign into an account with incorrect passwords. A pattern of failed logins may be due to:

1. Typing problems. For example, the user may have the Caps Lock or Num Lock key enabled and not realize it.
2. A forgotten password.
3. The wrong password being used. For example, the user may be trying to sign into system A using the password for system B.
4. An attacker trying to sign into an account by guessing the legitimate user’s password.

To deter the last of these, it’s reasonable for a system that detects too many invalid attempts during a short period of time to lock out the user’s account and prevent further attempts, at least for a while. This is called an intruder lockout.

Many systems include an “intruder lockout” feature. If $N$ invalid attempts to login to the same account are made during an interval of $T_{detect}$ minutes, then the account is disabled for $T_{lockout}$ minutes.

While this mechanism is effective against concerted attacks against a single account, it does nothing to prevent an attacker from simultaneously trying to guess the passwords of many users. A more sophisticated mechanism might extend this as follows: If $M$ invalid login attempts are made from network address $A$, to any account, during an interval of $T_{detect}$ minutes, then the address $A$ is disabled for $T_{lockout}$ minutes.

This will deter attackers with access to just a single network device/address from which to carry out their attack.

Intruder lockout also has a down side – it can be used to carry out a denial of service – by systematically locking out many accounts, including administrator IDs.

Because the intruder lockout mechanism cannot differentiate between a password guessing attack, a denial of service attack and a user input problem, the best practice is to combine several intruder lockout-related policies:

- Apply only to regular users, exempting at least one administrator login. This limits the scope of denial of service attacks.
- A compensating control is required here, such as requiring administrator IDs to have very complex passwords that are changed more frequently than user passwords. This way, they cannot be guessed (too hard) or locked out (feature not available).
- Apply a high threshold before triggering an intruder lockout – for example, 10 failed login attempts in 5 minutes. This minimizes lockouts caused by users who have trouble remembering or typing their own passwords.
- Automatically clear lockouts after a short while – for example, after 10 or 15 minutes. This reduces the impact on legitimate users while significantly impairing the ability of attackers to carry out bulk password guessing attacks.
11 Encrypting passwords in storage and transit

Passwords may be stored on user devices and are stored on servers. They must be transmitted, in some form, from the device where it was entered by the user to a server at login time.

Organizations rarely have any influence over how systems and applications store passwords – these are fundamental characteristics of the relevant software and are difficult or impossible to change. Fortunately, most modern applications incorporate strong encryption and reasonably safe protocols. The exception is custom applications, where developers working for the organization should follow best practices, such as storing passwords using a well-known and trusted hashing algorithm such as SHA-256.

Passwords transmitted from a client device to a server are similarly protected only as well as the application in question subject is able. For example, Telnet and TN3270 are plaintext while SSH and SSL are encrypted. It is generally not safe to trust the physical security of the communication path between a user’s device and the systems the user signs into, so protocol level encryption, at least of credentials but preferably of entire sessions, is recommended. Where an application does not support network-level encryption natively, using IPSec to protect all communication is recommended.

Some client devices and applications cache passwords and automatically offer them to servers on behalf of users, as required. Cached passwords should also be cryptographically protected, but in this case encryption rather than hashing is required, since the password normally must be retrievable. Many systems cache plaintext passwords (e.g., Android, iOS) and unfortunately there isn’t much organizations can do about this, other than to promptly change passwords in the event that an endpoint device was lost or stolen.

In some cases, the protocol provided by a vendor may encrypt passwords when they are used to login to a system, but not when a password change is transmitted. This happens with Oracle SQL*Net, for example. In these cases, if password management software is deployed, it is helpful for that software to implement its own encryption for password transmission, beyond that provided natively by the application vendor.
12 Synchronizing passwords

Password synchronization technology helps users to maintain the same password on two or more systems. This, in turn, makes it easier for users to remember their passwords, reducing the need to write down passwords or to call an IT help desk to request a new password, to replace a forgotten one.

There is some debate as to whether password synchronization makes systems more secure or less. The arguments are as follows:

- **Synchronization reduces security:**
  If a single system is very insecure, then compromising that system will give an attacker the passwords for every other system which uses synchronized passwords. This weakness is avoided by minimizing the use of such systems and requiring that users employ unsynchronized passwords on such weak systems.

- **Synchronization improves security:**
  Users with many passwords have trouble remembering them and consequently tend to write them down. System security is reduced to the security of a piece of paper, a note on the user's phone, etc. – i.e., close to zero.

In practice, it is very difficult to get users who have many passwords to stop writing them down or keeping them on insecure media. The two arguments above therefore can be restated as:

- Synchronized passwords: as secure as the least secure system in scope.
- Unsynchronized passwords: as secure as slips of paper.

Most modern systems and applications take reasonable precautions to protect their passwords, so synchronized passwords are normally more secure. To mitigate the risk of a single system compromise being leveraged by an attacker into a network-wide attack, follow these guidelines:

- Exclude insecure systems, such as those that use little or no cryptography to protect passwords, from password synchronization.
- Change synchronized passwords regularly, on the assumption that either some in-scope systems are vulnerable or that passwords may be compromised on the endpoint device or by fooling the user.
- Force users to choose strong (hard to guess) passwords.

A single, hard-to-guess, regularly changing password is more secure than multiple passwords, some of which may be easy to guess, some of which may not be changed regularly and all of which may be written down.
13 Single sign-on

Single sign-on (SSO) is any technology that replaces multiple, independent login prompts with a consolidated authentication process, so that users don’t have to repeatedly sign in.

13.1 Taxonomy of SSO Systems

Many organizations are interested in reducing the burden of repeated login prompts for users. Eliminating some or all of these login prompts is referred to as “single sign-on” though more accurately it is “reduced sign-on.”

There are several classes of reduced sign-on technology:

- **Kerberos / Integrated Windows Authentication:**
  Users sign into a Kerberos domain – typically using a Windows client signing into Active Directory. Their PC is issued a ticket which is passed to other server on the network when the user needs to authenticate to those systems.
  Microsoft's adoption (and extension) of this technology has led many to refer to it as Integrated Windows Authentication (IWA) but the technology was actually originally developed at MIT and is underpinned by public standards.

- **Enterprise Single Sign-on:**
  Client software is deployed to user PCs (typically Windows). This detects login prompts displayed by various applications and automatically populates login IDs and passwords in those dialogs and screens. Application credentials are typically stored in some sort of a secure “wallet” – on the filesystem, in a database or in a directory.
  Enterprise single sign-on (E-SSO) systems are attractive to many organizations because they provide a single mechanism for both legacy applications (client/server, terminal based) and applications with a web user interface.

- **Web Single Sign-on / Access Management:**
  Web single sign-on (WebSSO) systems intercept user attempts to access web applications. A cookie in the user's web browser is checked to see if the user has already been authenticated. If the user has not yet signed on, the web browser is redirected to a login application where the user signs in. The user is then redirected back to the application login page and automatically signed into the web application.
  As the name (and use of cookies) implies, this technology is only applicable to web-based applications.
  Two common architectures of WebSSO systems are available, often in the same product.
  In one approach, a web proxy server is used, which signs users into applications by retrieving application login credentials for every user, on every application, from a secure directory or database. These credentials are injected into the HTTP data stream between the WebSSO system and the application the user wishes to access. In this scenario, nothing is installed directly on the target application (“agentless architecture”); the user is not able to access the application URL directly, without going through the proxy; and a database of credentials must be maintained.
In a second approach, an agent is installed on each web application server. This requires a variety of agents to be developed – one for each type of supported application. Using this architecture, the agent intercepts user access attempts and notifies the application where it is installed of the authenticated user’s identity. In this architecture, there is no database of user credentials.

- **Federated Authentication:**

  Federated access management means that users sign into a web site in one domain – say the on the Intranet of their own corporation – and are able to follow a URL link to a web site on another domain – say a SaaS-hosted application. A trust relationship is defined between the two servers, such that the first server (in the user’s domain of origin) vouches for the identity, authentication status and authorizations of the user to the second server. The most common mechanism for vouching for user data is to issue an assertion, using a standard protocol such as security assertions markup language (SAML).

  Federated access management is also only applicable (in practice) to web applications. Its main advantage is that authentication and authorization information about users can be exchanged between systems under the control of different organizations, eliminating both redundant login prompts and user administration.

### 13.2 Security of SSO Systems

When organizations consider various approaches to single (or reduced) sign-on, they generally need to know:

- **Is it secure?**

  The short answer is usually “yes.”

  The longer answer is more nuanced:

  - **Strength of the initial authentication:**

    SSO works by extending the reach of an initial authentication step. So long as that step is robust, subsequent application logins are also secure. In other words, if users sign into the SSO system with a strong password (as described elsewhere in this document) or using a two-factor technology such as OTP+PIN, Smart card+PIN or biometric+password, then the SSO infrastructure rests on a robust process. On the other hand, if the SSO system depends on a weak password or PIN, on a proximity card, on a single physical authentication factor (which could be stolen) or on an easily recorded biometric, then the entire system is vulnerable.

  - **Strength of stored credentials:**

    Some SSO mechanisms depend on assertions, while others depend on the injection of login IDs and passwords into application login prompts. The latter set includes enterprise SSO and proxy-based web SSO.

    Injection of application passwords depends on reliable, secure storage of passwords. This raises the question of where and how these passwords are stored and the SSO software itself is authenticated before it can retrieve these passwords.

    For example, if a user signs into the SSO system using a biometric or proximity badge, there is no way to generate a consistent encryption key. This means that the SSO system must have direct access to stored passwords, even in cases where the user was not authenticated. On the
other hand, if a user signs into the SSO system with a password, then the user’s SSO password can be used to decrypt other passwords – the SSO system does not have unlimited access to stored credentials.

• **Is it more or less secure than password synchronization?**

  The answer here is also nuanced, and depends on how users sign into the SSO system in the first place.

  – *Users sign into SSO with a password:*

    If password synchronization covers only secure systems, whose password databases have been protected against attack using strong encryption, good password composition rules, regular changes, history, intruder lockout, etc., then the security difference between SSO and synchronization is the difference between compromising a single robust password database (the one used to sign into the SSO system) and compromising any one of several such systems. i.e., infeasible in either case.

  – *Users sign into SSO with a weak authentication factor:*

    If users with into the SSO system using a weak password, proximity badge or single physical authentication factor, then the energy that an attacker must expend to compromise all credentials is lower than what the attacker would have to expend to compromise one of several strong password systems.

  – *Users sign into SSO with a strong authentication factor:*

    If users with into the SSO system using a two-factor technology or strong password, then the energy that an attacker must expend to compromise all credentials is at least as high as what the attacker would have to expend to compromise one of several strong password systems, with the possible exception of injected passwords (see enterprise SSO and proxy-based web SSO above) with a vulnerable encryption key management scheme.
14 IT support for forgotten and locked out passwords

Sometimes users forget their passwords or type the wrong one often enough to trigger an intruder lockout. When this happens, users generally call their IT help desk and ask for the lockout to be cleared or the password to be reset.

The help desk process normally proceeds as follows:

1. User experiences problems trying to log in.
2. User calls the help desk and may wait in a service queue.
3. Support analyst asks the user for his name and a problem description.
4. Support analyst asks the user for information to prove his identity (i.e., authentication of a human by a human).
5. Support analyst compares this information with some on-line resource.
6. If the user is authenticated, the support analyst will generally do one of three things:
   a. Reset the forgotten password.
   b. Clear the intruder lockout.
   c. Forward the support incident to another IT support person, who has the requisite privileges to perform one of the above functions.
7. The user tries to log in with the newly assigned password.
8. The user may be required to change the new password immediately.

14.1 Security challenges

The process described above may be vulnerable to security weaknesses:

- The IT support analyst may forget to authenticate the user at all.
- Information suitable for authenticating the caller may not be available to the IT support analyst.
- Information about caller may be easy for an attacker to guess, making the whole process vulnerable to "social engineering" attacks.
- Many IT staff need to be able to reset passwords and clear lockouts for this process to work effectively. This proliferation of privileged accounts (i.e., those of the IT staff) creates a large "attack surface."
- Password resets on some systems may not be traceable back to an individual support technician. This creates a lack of accountability and allows IT staff to compromise user accounts without consequences.
- The support technician knows the user's new password at the end of the process, so it is no longer truly secret. This is especially problematic when newly reset passwords do not immediately expire.
14.2 Mitigating controls

These problems can be overcome through:

• IT support training, to ensure robust authentication of callers prior to every password reset.

• Ensuring that reliable authentication information is available for every user – including employees, contractors, etc.

• Providing an application proxy that resets passwords on behalf of IT support staff. This way, both IT staff and callers are authenticated before a password can be reset. This reduces the number of privileged accounts and ensures the process is consistent and secure.

• Allowing users to reset their own forgotten passwords and/or clear their own lockouts, after suitable non-password authentication – i.e., self-service password reset. This reduces the number of privileged accounts, enforces a robust authentication process and removes the situation where an IT support analyst knows the user’s new password.

14.3 User authentication

It is important to ensure that users who request a password reset, either by calling the IT support help desk or using self-service, are reliably authenticated. Following are some practical considerations when designing a non-password authentication system:

• One-time-password tokens, smart cards and biometrics are generally more reliable than security questions, because of the vulnerability of security questions to social engineering attacks.

• Today, almost all users carry a mobile phone. Using the phone as a credential – either by deploying an app to it or by sending a PIN to it via SMS or e-mail, makes sense as a first step in the login process.

• Where security questions are used:
  – As many questions as possible should be used.
  – The difficulty of guessing the answer to each question should be estimated. The total difficulty of guessing enough answers to authenticate as a user should be approximately the same as the difficulty of guessing a user’s password.
  – Users should provide answers to both personally-chosen and standardized questions. In the former, the questions are harder to guess but answers are sometimes easy to guess. In the latter, the questions are well known to would-be attackers.
  – A randomly chosen subset of a user’s available security questions should be chosen, to make guessing harder for attackers with a limited set of information about a user.
  – At authentication time, the system should prompt the user to answer standard questions first (attackers will know what these are and may have done some advance research). Ask personally defined questions second, only if the standard ones have been successfully answered.
  – If another credential, such as an e-mailed or SMS-ed PIN is used, deploy that before asking any security questions. Make attackers get past the PIN step before they can even see the user’s security questions.
– Once security questions have been randomly selected from a user’s profile and presented to the user in the login screen, even if the user answers incorrectly, ask the same questions again. Do not ask different questions on each login attempt, until the user has successfully answered one set of questions. This prevents attackers from "shopping" for questions to which they have managed to collect answers.

– Intruder lockout should apply to authentication with security questions just as it does to passwords.

Sample security questions are provided in Section A on Page 25.
15 Mobile devices: challenges and opportunity

Today, almost every user carries at least one mobile device with them at all time. These devices are called 'smart phones' but they are really quite powerful and compact pocket-sized computers, with a decent quality display, touch screen, a rich variety of sensors and often with Internet access. Incidentally, these devices can make phone calls.

Devices owned by the user but accessing corporate information services are often referred to as 'bring your own device' or BYOD for short.

15.1 BYOD challenges

Users expect to be able to interact with a variety of computing service, including corporate password management systems, from these hand-held computers. They already interact with corporate e-mail, calendaring and other applications from these devices. This leads to some challenges:

1. Connectivity:

   Password management systems are typically deployed on-premise, attached to the private corporate network, and firewalled off from the public Internet. That’s a problem for smart phones, since they are attached to the public Internet and usually have no access to services inside the corporate network. Enabling these devices to reach services inside the corporate network perimeter requires either opening inbound connections across firewalls or exchanging messages with an Internet-attached (cloud hosted) proxy system.

2. Cached passwords:

   Text input into smart phones is a pain. Consequently, most phone-based apps try to minimize it, for example by caching passwords to corporate services. For example, users do not have to re-enter their password every time they read their corporate e-mail or check their corporate calendar on their phone. Whenever a user’s password is changed on the network, these cached passwords are invalidated, and indeed the phone could trigger a lockout by offering the cached password too often.

Ideally, organizations deploying a password management system make it available to users via their smart phones. Rather than opening an inbound connection path across their corporate firewall, it is safer to broker communication between phones and an on-premise password management system via a cloud-hosted proxy.

It would be nice if the password management system, perhaps using an app deployed on user phones, could update locally cached passwords after each password change. Unfortunately, this is not technically possible: iOS and Android implement strong isolation between apps, plus many apps use proprietary storage to cache passwords, rather than the provided OS-level key-ring. A second-best solution, which is all that can be done today, is to send reminders from the password management system to the user, after each password change, reminding the user to update phone-cached passwords, along with instructions on how to do that.
15.2 BYOD opportunities

There are contexts where a user needs to reset a password or clear a lockout, but cannot launch a browser on their PC. For example, a user may have forgotten their pre-boot password, used to unlock an encrypted filesystem, or their OS login password, used to sign into their PC. In each of these cases, a smart phone is a viable user interface, as it is most likely not locked out.

There is also a growing demand for users to leverage multi-factor authentication, for a variety of use cases, including authenticating into a self-service password reset system. A smart phone is a device which most users already have, so an app on a phone can be used as a credential (“something you have”) in the login process.
APPENDICES
A Sample Security Questions

A.1 Questions with textual answers

Sample security questions, which may have alpha-numeric questions and so are suitable for a text user interface, include:

- Which bank branch do you live closest too?
- What was the make of your first car?
- What was the model of your first car?
- What is your favorite food?
- Who is your favorite book character?
- What is your favorite game or sport?
- What is your favorite movie?
- What is your favorite pizza topping?
- What is your favorite season of the year?
- What is your favorite sports team?
- In which department in the company did you first work?
- What was your first position in the company?
- Who is the person you admire the most?
- What was the most memorable day in your life?
- Who was your childhood hero?
- What is the nickname of your youngest sibling?
- What is the nickname of your oldest sibling?
- Who was your first boss?
- What award are you proudest of?
- What city were you born in?
- What is the farthest from home you have traveled?
- What is the name of the first school you attended?
- What is the name of the first person you were romantically interested in?
- What is your astronomical sign?
- What is your father’s middle name?
- What is your mother’s middle name?
- Who is your favorite actor or actress?
- What is your favorite musical band?
- What is your favorite beverage?
- What is your favorite board game?
- Who is your favorite book character?
- Who is your favorite author?
- What is your favorite dessert?
- What is your favorite hobby or pastime?
- What is your favorite ice cream topping?
- What is your favorite song?
- What is your favorite television show?
- What is your favorite vacation spot?
- What is your mother’s maiden name?
- What is your place of birth?
- What is your school team’s mascot name?
- What was the breed of your first pet?
A.2 Questions with numeric answers

Sample security questions, that have numeric answers and so are suitable for authentication using a touch-tone phone, include:

• What is your favorite radio station (number on the dial - NNNN)?
• In what year did you start with your company (CCYY)?
• On what date were you hired?
• What is your parents' wedding anniversary date (MMDD)?
• What are the last 4 digits of your SSN or SIN?
• What are the last 4 digits of your childhood home phone number?
• What is a birth date of your youngest child (CCYYMMDD)?
• What is a birth day of your oldest parent (MMDD)?
• What is a birth day of your youngest parent (MMDD)?
• On what date were you married (CCYYMMDD)?
• On what date were you divorced (CCYYMMDD)?
• What is your employee number?
• What is your employee number?
• What is your date of birth (MM/DD/YYYY)?
• The last 4 digits of your passport number:
• The last 4 digits of your driver's license number:
B References

This document was prepared by Hitachi ID Systems and reflects expertise with password management accumulated through many deployments of Hitachi ID Password Manager.

• About Hitachi ID Systems:
  – http://Hitachi-ID.com

• About Password Manager:
  – http://Hitachi-ID.com/Password-Manager

• Password guessing programs:
  – http://sectools.org/search/?q=crackers
  – http://www.l0phtcrack.com/
  – http://www.openwall.com/john/

• NIST / FIPS documents about password management:

• Best practices for managing passwords for privileged users (as opposed to normal users):

• Unicode and code pages (for multi-lingual users):
  – http://unicode.org/

• Password cracking and password security:
  – http://en.wikipedia.org/wiki/Password_cracking

• Choosing secure passwords:

• Disabling LAN Manager passwords on AD:
  – http://support.microsoft.com/kb/299656

• Cryptographic hash functions:

• IPSec:
Password Management Best Practices

- http://www.securityfocus.com/infocus/1519
- http://www.ipsec-howto.org/