Securing Account Credentials to Protect Your Organization

blog.netwrix.com/2023/04/21/secure-credentials-and-protection

Compromising the credentials of Active Directory accounts remains a primary way for adversaries to gain a foothold in an organization's IT ecosystem. They use a range of tactics, including credential stuffing, password spraying, phishing and brute-force attacks

This blog post details key best practices for effective user credential management. Then it dives into how software can help enforce those best practices and further secure user credentials.

Handpicked related content:

Remote Access Security Best Practices

Best Practices for Users

Here are the top ways that individuals can protect themselves against credential theft.

Use Multifactor Authentication (MFA)

Defends against: **Most credential-related attacks, including credential stuffing, password spraying, phishing, keyloggers, brute-force and local discovery.**

If you only do one thing to protect your credentials, it should be this: Use multifactor authentication whenever possible. In fact, Microsoft reports that MFA could prevent 99.9% of account compromises.

Applications and services that offer MFA usually enable you to active it in your account settings (usually under Privacy or Security). We've found MFA for accounts that never prompted us to use them, just by poking around in the settings on the website. The most common MFA option is a verification code sent by SMS, but there are also hardware-based MFA options available, such as Google's Titan Security Key.

Even if you're using MFA, it's still important to change your password if it's stolen or breached.

Avoid Reusing Passwords

Defends against: **Credential stuffing attacks**

In a credential stuffing attack, adversaries attempt to use breached credentials to log on to various services, hoping that users used the same username/password combination for multiple sites.

The defense is simple: Never reuse passwords. Of course, remembering dozens or hundreds of passwords is difficult, so consider using a password manager. That way, you need to remember only one master password — which should be long and complex. The password manager will automatically generate strong passwords for each of your accounts and store them securely, making it easy to create a unique password for each account.

Keep in mind that password manager software vendors can be breached, so stay vigilant about security incident announcements and response measures.

Avoid Using Common Passwords

Defends against: **Password spraying attacks**

In a password spraying attack, threat actors programmatically apply a large dictionary of well-known passwords against one or more services. For example, weak passwords like “123456”, “password” and “qwerty” continue to see frequent use.

The best way to defend against this attack is to use strong, unique passwords, which is also aided by the use of a password manager.

Avoid Using Simple Passwords

Defends against: **Brute-force attacks**

In a brute-force attack, adversaries keep guessing passwords for an account until they gain access (or get locked out). Typically, the attack starts with short, simple passwords and expands in complexity if those attempts fail. Previously it was recommended to choose longer passwords containing numbers and special symbols — a complex password that is at least 11 characters long could take years to brute force, even with a top-of-the-line cracking rig. However, with the growth of computing tools and techniques available to attackers, NIST has ceased to insist on password complexity requirements, and at the moment the use of passphrases is considered to be the best practices.

Brute-force attackers can target specific individuals, so it’s also important to avoid using passwords that contain using personal and context-specific information can reduce the number of passwords they need to guess. Personal information is any public information that can be tied in passwords, such as your birthday or names of family members and pets. Context-specific information includes the name of the website or service; for instance, don’t use “google” in your Google password.

Best Practices for Software Vendors

While users should take responsibility for securing their credentials, data privacy laws like the GDPR and the CCPA require websites, web applications and other software to implement protections against credential attacks and can impose stiff fines if failure to comply results in a breach.

Here are some ways websites, web applications and other software can protect against user credential theft.

Enable Multi-Factor Authentication (MFA) using Authenticator Apps

As stated earlier, using multi-factor authentication is the most important thing users can do to secure their credentials, so any software or service requiring login to an account should provide MFA for users.

However, many MFA implementations rely on verification codes delivered via SMS, which is generally an insecure option for MFA for two reasons:

- **SIM swaps** — With enough of a user's personal information, including their phone number, an attacker can trick a user's phone service provider into transferring their phone number to the attacker's SIM. This allows the attacker to receive all SMS messages, including MFA verification codes, intended for the victim.
- **Intercepted SMS messages**— Attackers can intercept SMS messages by exploiting vulnerabilities in the Signaling System No 7 (SS7) protocol. This is not a problem domestically in the US, but SS7 is used to change networks and operators when a smartphone is used in some other countries. Attackers can abuse known vulnerabilities in SS7 by using only Linux and an SS7 software development kit.

Accordingly, SMS should be avoided for sending MFA verification codes to users. Instead, vendors should use authentication apps, such as Google Authenticator or Okta Verify. These apps were built specifically for MFA, and attackers won't be able to intercept the verification codes using the SMS vulnerabilities described above.

Enable Single Sign-in (SSO)

Single sign-on enables users to access multiple applications and services with a single set of login credentials, which is more convenient for them. Moreover, it reduces risk for vendors because an identity provider (IdP), not the service provider, is responsible for verifying credentials.

Before implementing SSO for your software or service, you'll need to pick an SSO standard. Some popular options are:

- **SAML** — The most mature of the standards on this list, Security Assertion Markup Language (SAML) is an authentication (AuthN) and authorization (AuthZ) protocol that enables identity providers to send authorization credentials to service providers.
- **OAuth 2.0** — The successor to OAuth 1.0, OAuth 2.0 is an authentication framework that enables applications to obtain limited access to user accounts managed by an identity provider.
- **OpenID Connect** — The successor to OpenID 1.0 and 2.0, OpenID Connect is an authentication protocol that relies on OAuth 2.0 to allow users to grant service providers access to their identity using JSON web tokens (JWTs).

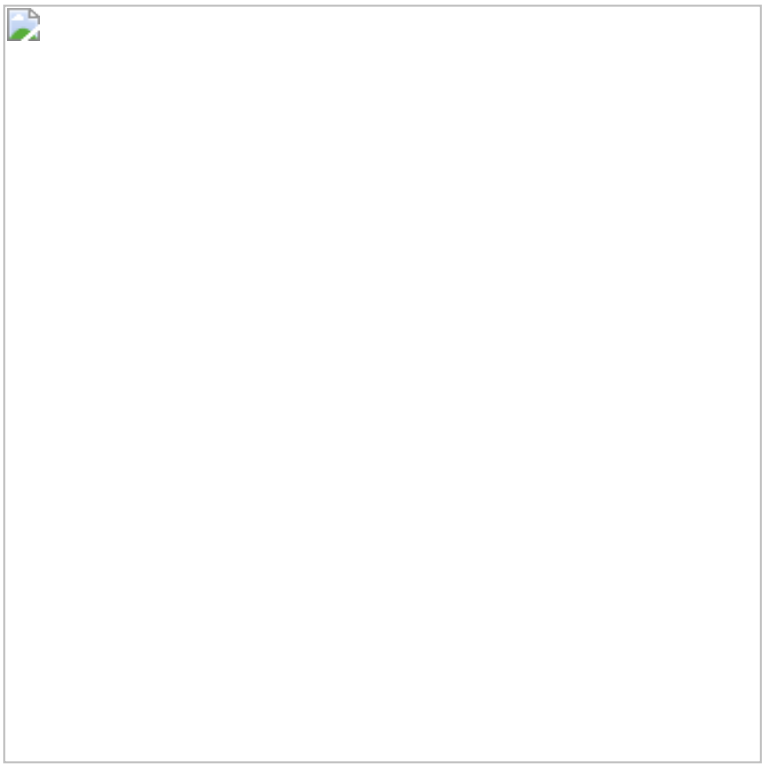
It should be noted that OAuth 2.0 is more aimed at limiting access scope than SAML and is far more popular than SAML for web and mobile applications.

To use these options for SSO, you also need to pick an identity provider that supports your chosen protocol. For SSO for individuals, options include Google, Facebook and Microsoft. For a corporate environment, popular identity providers include Azure Active Directory, G Suite and PingFederate.

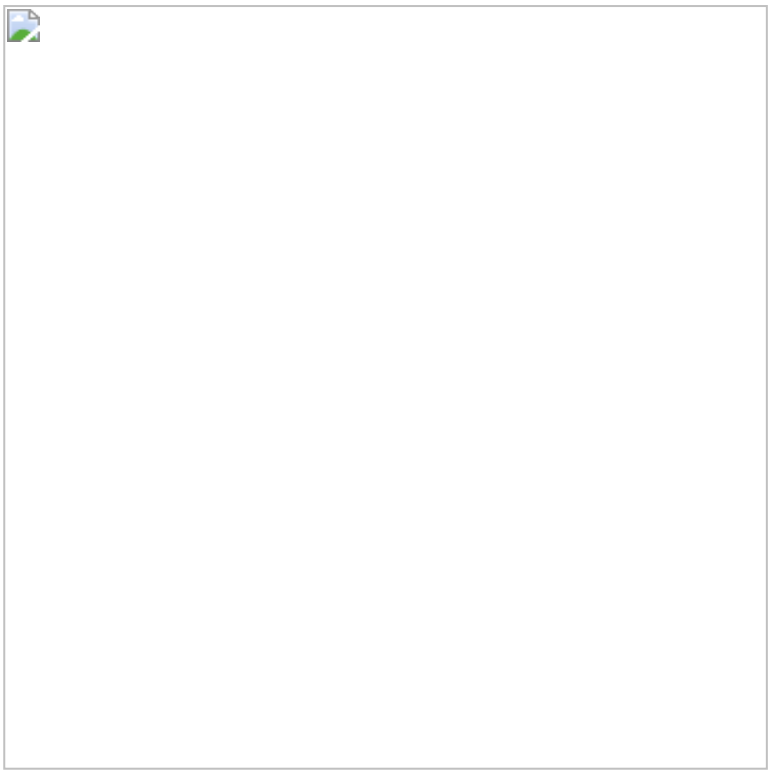
Secure Error Messages during Login

This tip is easy to overlook; however, the error messages returned from failed logins can give attackers performing reconnaissance plenty of information they can use to in credential stuffing, brute-force and phishing attacks on a specific victim.

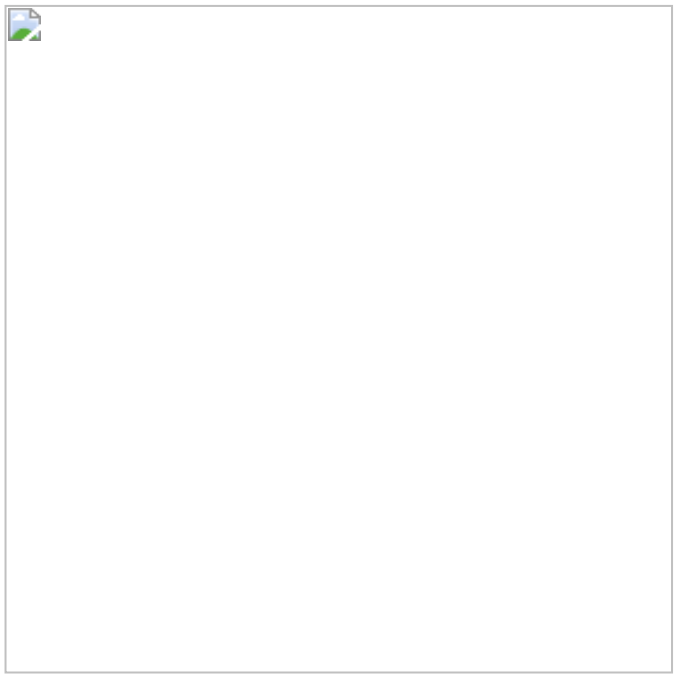
For example, an attacker who attempts to log in to Facebook might see an error stating that the email or phone number they entered doesn't match a Facebook account:



But if the email or phone number does match a valid account, they get a message stating the password is incorrect password, which helps them gather a list of valid account names, emails, phone numbers:



To give you an idea of what a good generic login error message looks like, look at this error from GitHub, which is the same for both a non-existent account and an existing account with an incorrect password:



Store Only Salted & Hashed User Passwords

Hashing is at the core of any password storage workflow, but it's worth mentioning because there are still websites out there that store passwords in plaintext.

Any time you need to store a user's password, it should be salted and hashed using modern cryptographic techniques such as PBKDF2 and Bcrypt. These algorithms are deliberately slow to deter programmatic brute-force attacks, compared to fast algorithms like SHA-256 that facilitate programmatic attacks.

In simple terms, salting is the process of adding random data to a user's password before it's sent to the hashing algorithm, which adds complexity to the resulting hash and makes pre-computed rainbow table attacks and brute-force dictionary attacks more difficult.

Check Candidate Passwords against Databases of Breached and Well-known Passwords

This is a perfect example of helping users help themselves. When users type in a new password, whether when creating a new account or changing their existing password, it should be compared against a list of well-known and breached passwords, such as the Have I Been Pwned database from Troy Hunt, which are ripe for credential stuffing attacks.

If the candidate password is found in the database, the user should be required to choose a different password. The error message should explain to the user why their candidate password was rejected, to prevent them from becoming confused or frustrated by the password selection experience.

Use HTTPS Rather than HTTP

HTTP doesn't encrypt communications between a client (web browser) and a server, which means everything — including credentials — is in plaintext. For better security, your website, application or service should allow the installation of a TLS/SSL certificate to enable encrypted HTTPS traffic for all communications.

This may not be news to developers today, but there should also be safeguards in place that prevent users from accidentally accessing an HTTP version of your website. Primarily this is done by redirecting any HTTP requests to the HTTPS version of the requested page; however, HTTP Strict Transport Security (HSTS) can also be used to mitigate man-in-the-middle and protocol downgrade attacks.

HSTS is a directive from a website or service that is included in the response header and informs user agents (browsers) that only HTTPS can be used for access. This has the added benefit of rejecting any JavaScript calls to load resources via HTTP, which could be the result of a cross-site scripting (XSS) attack, and also disallows manual acceptance of insecure, invalid or expired TLS/SSL certificates.

Directly Alert Users to Unusual & Security-related Events

Any time there's suspicious activity on an account, the user should be notified. An example would be a login attempt for their account from a region of the world they have never logged in from before. Users should also be informed via email any time any of

their personal information or password is changed. These alerts enable users to promptly revert unwanted changes and reset their password if their credentials might have been compromised.

It can also be useful to periodically inform users about the security features your service offers that the they may not be aware of, such as MFA.

Adhere to NIST Password Guidelines

The National Institute of Standards and Technology provides password guidelines that are regularly updated to reflect evidence-based best practices. These guidelines provide a solid foundation for password policy for websites, applications and services.

Additional Security Measures

In addition to helping protect user credentials, developers can take additional measures to secure their websites and web applications. For example, implementing a web application firewall (WAF) helps protect against malicious file execution, SQL injection, XSS, and denial-of-service (DoS) and distributed denial-of-service (DDoS) attacks by inspecting all HTTP/S requests before they're served.

Another example is bot detection software, which can inspect incoming requests to filter out bots before that traffic communicates with your website or application. This is a more advanced technique, but it can help defend against credential stuffing, password spraying and brute-force attacks.

How Netwrix Helps with Credential Security

80% of breaches involve weak or compromised passwords, and the top 10 most common passwords still including "123456", "password", and "qwerty". Netwrix StealthINTERCEPT Enterprise Password Enforcer helps safeguard your organization from credential-based attacks by using a dictionary of more than half a million known compromised passwords, along with complexity, character substitution and testing tools. Netwrix solutions can identify weak and compromised passwords and prevent them from being used, and even provide users with guidance on how to choose a stronger passwords.

On top of enforcement, the Netwrix Active Directory Security Solution can help you assess weak passwords, remove excessive rights, detect advanced attacks on AD credentials, and replace risky standing privileges with just-in-time ephemeral accounts.

FAQ

What are secure credentials?

Secure credentials are authentication information, such as usernames and passwords, that are protected against unauthorized access. This means that the credentials are encrypted and stored in a secure manner, and access to them is restricted to only those

who are authorized to use them, often with additional security measures such as multifactor authentication.

What is credential protection?

Credential protection is the process of securing usernames, passwords and other credential data against unauthorized access and misuse. This includes implementing security measures such as encryption, multifactor authentication, secure storage and monitoring for any suspicious activity.

Why is credential security important?

Credential security is important to protect sensitive information and prevent unauthorized access to accounts, systems and data. This helps prevent identity theft, financial loss, and damage to personal and professional reputations.

Joe Dibley
Security Researcher at Netwrix and member of the Netwrix Security Research Team. Joe is an expert in Active Directory, Windows, and a wide variety of enterprise software platforms and technologies, Joe researches new security risks, complex attack techniques, and associated mitigations and detections.