No labels

SW – User Authentication

User Authentication is the verification of the identity of a user (by verifying user credentials such as a password and / or token), often as a prerequisite to allow access to resources in an information system. If a user is not forced to provide credentials, but only the user name, this is called identification and is insufficient.

Besides user-related access control, user authentication is also required for monitoring, logging, and auditing user activities

SW-1 – Missing User Authentication

Exploitability	**Group**
High	User Authentication
Question	**Threats and attack examples**

Is it ensured that a successful authentication (not only identification) against the application is required / mandatory before
• any non-public action is executed on behalf of a user,
• any non-public data is presented to a user or to the user's client, or
• access to any non-public subcomponent is granted?
	Unauthorized Access to Resources
	An adversary might be able to bypass user authentication using access paths that are not covered by user authentication mechanisms. The adversary might learn about data they are not authorized to see. Thus, an adversary might have unauthorized access to data or might be able to perform unauthorized actions. Depending on the concrete scenario, the impact might be disastrous.
	Examples:
	• Only the main component of an application enforces authentication. Other components rely on a successful central authentication check done by the main component but do not check if the user is already successfully authenticated. An adversary might be able to access those components directly in order to circumvent the authentication check of the main component. An example is the manipulation of URLs to access parts of an application that usually should not be publicly accessible.
	• Several older router models do not require authentication to modify configuration via special URLs, i.e. there are access paths that circumvent authentication checks. Generally, every access path to any action, data, or subcomponent must check if the user already is successfully authenticated. Examples:
	  ◦ http://myrouter.com/admin/userMngt.asp ↗
	  ◦ http://myrouter.com/configuration.pdf ↗
	  ◦ http://myrouter.com/showConfigParam?id=1234567 ↗
	• An adversary can read sensitive data (e.g. configuration data) of a device and only has to authenticate in case of modifications / changes. The configuration data might be used by an adversary to prepare / configure a manipulated device using this information before they replace the regular one.
	• Authentication is not validated for each access to an object or component because of a missing or insufficient central or systematic approach. An adversary might be able to identify the weak points and abuse them.

Further information	CWE-306: Missing Authentication for Critical Function ↗ CWE-304: Missing Critical Step in Authentication ↗

SW-2 – Client-Side User Authentication

Exploitability	**Group**
High (Web) / Medium (Rich Client)	User Authentication
Question	**Threats and attack examples**
Does your software provide robust and secure server-side (not client-side) authentication mechanisms?	**Unauthorized Access to Resources** An adversary can usually very easily manipulate the client code to circumvent client-side checks. This might be done by e.g. reverse engineering or modifying the code at runtime using a debugger. A client, as a matter of fact, has to be deemed insecure as it is very hard to prevent manipulation of code an adversary has access to. Thus, final authentication decisions must never be taken on the client-side, but always on the server-side.
Further information	CWE-603: Use of Client-Side Authentication ↗ CWE-602: Client-Side Enforcement of Server-Side Security ↗

SW-3 – Weak User Authentication

Exploitability	Group
High	User Authentication
Question	**Threats and attack examples**

Does your software provide robust and secure authentication mechanisms, including at least the following aspects:
• The number of unsuccessful log-on attempts per user is limited, e.g. by (temporary) account locking or by increasing waiting time before another log-on attempt is processed by the server to protect against brute force login attacks.
• A password policy is in place forcing users to enter high quality passwords and to change their passwords regularly. A password policy usually includes:
  ◦ The ability to configure a minimum password length. (e.g. 9 char.)
  ◦ The ability to require a configurable mix of upper and lower case alphabetical characters, numerical char., and special characters.
  ◦ The ability to define customer-configurable expiration periods (e.g. 90 days) after which users are forced to change the password.
  ◦ The ability to prohibit users from reusing passwords that were used in the past, e.g. the last five passwords of the user (reuse restrictions)
  ◦ Offer a password strength indicator when a user sets a new password
Consider enforcing higher password policy requirements (e.g. increased number of characters) for privileged accounts (such as service or administration accounts).
• Failed authentication attempts do not show any specific information (e.g. “password invalid”). Always be as unspecific as possible (e.g. “user name or password invalid”).
• The software uses dedicated components for password fields in user interfaces providing, at least, the following functionality:
  ◦ The actual characters entered by the user are masked to ensure that passwords are not revealed on the screen.
  ◦ Content-caching, auto-completion and copy-to-clipboard are prevented.
Note: Do not implement yourselves; use established mechanisms (see SW-8, SW-9).
	Unauthorized Access to Resources
	An adversary might be able to circumvent or exploit insecure authentication procedures to gain unauthorized access to resources or to escalate their privileges.
	Examples:
	• If an application does not limit the number of unsuccessful log-on attempts, an adversary can easily run a brute-force attack (systematical password guessing based on dictionaries, lists of commonly used weak passwords, and/or generating all passwords up to a given length) to gain access to the system. Please note that there are many dedicated software programs which automate brute-force attacks. (Be aware that account lock out might bear the risk of denial of service attacks!)
	• If there is no password policy in place which enforces strong passwords, brute-force attacks are usually quite efficient.
	• If users are not forces to change passwords regularly, it becomes easier for an adversary to guess passwords and to reuse disclosed passwords
	• An adversary attacking your system might learn from detailed error messages. For example, an adversary might at first try to determine a valid user name by trying different user names until the error message “user unknown” changes to “password invalid”.
	• If no dedicated components for password fields are used, an internal or external adversary might be able to find out a password of a user by looking over the shoulder of the user in order to gain unauthorized access to the system.

Further information	CWE-307: Improper Restriction of Excessive Authentication Attempts ↗

SW-4 – Insecure Initial Passwords

Exploitability	Group
High	User Authentication
Question	**Threats and attack examples**
Does your software provide robust and secure initial passwords or other user credentials including, at least, the following aspects: • A mechanism is implemented that forces the users to change their password immediately upon first use. • Default passwords are never used (installation routine), i.e. the change of default passwords (factory settings) at installation time is enforced. • The software enforces a mechanism to randomly generate initial user authentication secrets fulfilling the defined password quality criteria. • Only well-proven and secure algorithms for automatic password generation are used to avoid predictable sequences. (See e.g. https://bettercrypto.org/ ↗)	**Unauthorized Access to Resources** **An adversary might be able to take advantage of insecure initial user credentials in order to gain unauthorized access to resources or to escalate their privileges.** **Examples:** • If default passwords are used for new users, adversaries can easily use this default password to compromise new user accounts. The adversary might be an internal user knowing the default password, or the default password was made publicly available by a former employee or through publicly available documentation, or an adversary might use social engineering attacks to find out the default password. • As initial passwords are transmitted to a user, there might be other persons apart from the new user who might know the password. Only if a user is forced to change their password immediately upon first use, can it be ensured that only the user themselves know their password. • If a default password is used, e.g. for the initial administrative setup of a system, adversaries can use this knowledge to gain access to any system they can access via a network where the default passwords have not actively been changed to secure new ones. As commissioning engineers might not be security aware they might not change default passwords to speed up the setup process and to ease maintenance later on, i.e., the maintenance personnel only have to remember the default password to gain access to all maintained systems. Default passwords can usually be found easily, e.g. using available documentation, reverse engineering, or debugging. That information might easily be spread in the Internet and, thus, might easily be available to any adversary and damage Siemens’ reputation. • If the initial passwords do not fulfill the defined password quality criteria, an adversary might attack the user accounts of new users taking advantage of weak initial passwords. • If weak algorithms are used for password or cryptographic key generation, the results might be easily predictable.

SW-5 – Hard-coded Passwords

Exploitability	Group
High	User Authentication
Question	**Threats and attack examples**
Does your software implement a mechanism that allows the change of all authentication secrets used by the software, i.e., there are no hard-coded credentials like passwords, certificates or cryptographic keys?	**Unauthorized Access to Resources, Escalation of Privilege** **An adversary might be able to take advantage of hard-coded passwords in order to gain unauthorized access to resources or to escalate their privileges.** **Examples:** If a hard-coded password is used, e.g. for the initial administrative setup of a system or any account, adversaries can use this knowledge to gain access to *any* system they can access via a network. Hard-coded passwords might easily be found out using, e.g. available documentation, reverse engineering, or debugging. This information might easily be spread in the Internet and, thus, might be easily available to any adversary and damage Siemens' reputation.
Further information	CWE-798: Use of Hard-coded Credentials ↗

SW-6 – Insecure Password Change Functionality

Exploitability	Group
High	User Authentication
Question	**Threats and attack examples**
Does your software provide robust and secure functionality to change passwords or other user credentials including, at least, the following aspects: • It is ensured that a user credentials can only be changed by the user themselves or by specially-privileged users (e.g. security administrators). • If a password is changed by a non-privileged user themselves, the application requires re-entering the original password. • If a password is changed by, e.g. a security administrator, the application requires the user to change their password on first log-on. • Questions or other „password hints" (for example the favorite color) are not provided as a user password recovery mechanism.	**Unauthorized Access to Resources, Escalation of Privilege** **An adversary might be able to take advantage of weak password change functionality in order to gain unauthorized access to resources or to escalate their privileges.** **Examples:** • If only specially-privileged users can change the password for a user, the specially-privileged user potentially always knows the new password of the user. • If a user can change the passwords of other users, the authentication mechanism is reduced to absurdity.
Further information	CWE-798: Use of Hard-coded Credentials ↗

SW-7 – Kiosk Mode or Operating System Group Accounts

Exploitability	Group
High – Med (dep. on kiosk mode)	User Authentication
Question	**Threats and attack examples**
Is your software designed and documented in a way that individual operating system accounts can be used? In detail, this includes: • The software does not rely on a generic / group user account on OS level or auto log-on functionality on the OS level. • The security documentation of the software encourages the usage of individual operating system user accounts and discourages the usage of auto log-on functionality. • If group accounts on OS level are required by the use case or customers, a strong kiosk mode to prevent users of the application to directly access the operating system should be used. Note however that most kiosk mode implementation can be bypassed.	**Unauthorized Access to Resources, Privilege Escalation, Non-Traceability of User Activities** **If a client device (e.g. a standard PC) is operated with auto log-on functionality or a group OS account on the OS level, any person with physical access to the device also has access to the generic operating system user account. This allows adversaries to install malware or key loggers which affect all other users.** **Examples:** • If a client device is operated with a generic operating system user account (meaning that users are not logically separated on the OS layer), any person with access to the device and in possession of the user credentials can install a key logger on the generic OS account, which logs all information entered by any user. This might include user names and passwords of other users if the software requires an application-level authentication. An adversary can easily impersonate any other user this way. • Unique Identifiers are prerequisites for monitoring / audit trails / traceability and often required for compliance reasons. Shared / generic accounts introduce various problems such as: • User activities cannot clearly be traced back to their originators. • If an employee leaves the company or changes position, the password of a generic account is typically not changed. • Some security mechanisms are ineffective (e.g. displaying the number of unsuccessful log-on attempts since last successful log-on, etc.) • Therefore, generic / group accounts should only be used in precisely defined exceptional cases. (In other words, accounts / usernames and passwords should not be shared). • A good kiosk mode is hard to implement and has to be designed and implemented very thoroughly. There are e.g. OS keyboard shortcuts and OS file dialogs that can be used to escape a kiosk mode very easily.

SW-8 – Clear-Text Transmission of Passwords

Exploitability	Group
High	User Authentication
Question	**Threats and attack examples**

Does your software protect the user credentials (e.g. password) if it is transmitted? This includes the following aspects:
• The user credentials are protected) while being transmitted during the authentication process. E.g.
  ◦ use TLS-secured protocol alternatives, e.g. HTTPS instead of HTTP
  ◦ use established authentication protocols, e.g. Kerberos, SChannel (supported by Windows AD), SAML.
• Passwords are not transmitted in URL-parameters.
• The user credentials are protected during password distribution (initial password, password reset, transfer of passwords, transfer of user credentials), e.g. via encrypted mails.
In case there is no secure channel to transmit initial passwords, the lifetime of those initial passwords is very limited and change of password is enforced upon first log-on.
	Unauthorized Access to Resources, Privilege Escalation
	• An adversary can obtain an unprotected / weakly protected user credentials (e.g. password) by eavesdropping on the network (e.g. if insecure protocols such as telnet or http with basic authentication are used).
	• An adversary might also obtain unprotected user credentials by eavesdropping on messages (e.g. an e-mail) during initial distribution of a password or during a password reset process. Please note that this threat also applies to physical media (e.g. passwords sent by mail) or passwords provided verbally by phone.
	• If only the hashed password is transmitted and there is no additional protection of the hash value, an adversary might obtain the hash. As a matter of fact, the adversary can use the password hash for authentication and does not need the plaintext password. Therefore, hashing is no appropriate protection for password transmission. Additionally, working with salts is not possible, if only the hash is transmitted.

SW-9 – Insecure Password Storage

Exploitability	Group
High	User Authentication
Question	**Threats and attack examples**

Does your software provide secure password (or other user credentials) storage, including, at least, the following aspects:
• Never store passwords in plaintext, use functions provided by operating systems. Examples are:
  ◦ Windows: Data Protection API, Windows.Security.Credentials API
  ◦ Linux: Password storage from desktop managers, e.g. GNOME Keyring, KDE KWallet
  ◦ Android: Keychain
  ◦ iPhone: Keychain
• User credentials are only stored on the server, not on the client (e.g. file or registry).
• Only when absolutely necessary, implement password storage yourself. In that case:
  ◦ Do not use weak cryptographic algorithms (see SW-28); use proven password hashing algorithms such as PBKDF2 or bcrypt.
  ◦ In addition add salt to passwords before hashing.
  ◦ Computation of hashes of salted passwords is done on the server side.
• Check for leakage of passwords via software components, e.g. logging
• Obfuscate passwords that are stored in RAM, or reduce the time that passwords are stored in RAM buffers and actively overwrite buffers when not needed any more
	Unauthorized Access to Resources, Privilege Escalation
	An adversary might be able to recover the plaintext passwords in order to gain unauthorized access to resources or to escalate their privileges.
	Examples:
	• If an adversary is able to obtain a stored list of all user-names and passwords by compromising the server system (e.g. by malware or unauthorized physical access), the adversary might try to recover the plaintext passwords to be able to impersonate authorized users. Therefore, the adversary might use one of the following techniques:
	• If the passwords are stored in plaintext, the adversary can use them as they are.
	• If the passwords are stored in encrypted form, the adversary might be able to figure out the algorithm and the key used to encrypt the passwords and decrypt them by reverse engineering your code or by using the same method to decrypt passwords, that your code uses, e.g. if this method is part of a DLL.
	• If only password hashes are stored without using a salt, the adversary might use rainbow tables to effectively recover the plaintext passwords.
	• Please note that also any legitimate user with access to the password file could obtain all user passwords and reuse them if the passwords are not properly protected.

Further information	CWE-759: Use of a One-Way Hash without a Salt ↗