# CWE Detail – CWE-311

## Description

The product does not encrypt sensitive or critical information before storage or transmission.

## Extended Description

N/A

## Threat-Mapped Scoring

Score: 0.0

Priority: Unclassified

## Observed Examples (CVEs)

**•** CVE-2009-2272: password and username stored in cleartext in a cookie

**•** CVE-2009-1466: password stored in cleartext in a file with insecure permissions

**•** CVE-2009-0152: chat program disables SSL in some circumstances even when the user says to use SSL.

**•** CVE-2009-1603: Chain: product uses an incorrect public exponent when generating an RSA key, which effectively disables the encryption

**•** CVE-2009-0964: storage of unencrypted passwords in a database

**•** CVE-2008-6157: storage of unencrypted passwords in a database

**•** CVE-2008-6828: product stores a password in cleartext in memory

**•** CVE-2008-1567: storage of a secret key in cleartext in a temporary file

**•** CVE-2008-0174: SCADA product uses HTTP Basic Authentication, which is not encrypted

**•** CVE-2007-5778: login credentials stored unencrypted in a registry key

**•** CVE-2002-1949: Passwords transmitted in cleartext.

**•** CVE-2008-4122: Chain: Use of HTTPS cookie without "secure" flag causes it to be transmitted across unencrypted HTTP.

**•** CVE-2008-3289: Product sends password hash in cleartext in violation of intended policy.

**•** CVE-2008-4390: Remote management feature sends sensitive information including passwords in cleartext.

**•** CVE-2007-5626: Backup routine sends password in cleartext in email.

**•** CVE-2004-1852: Product transmits Blowfish encryption key in cleartext.

**•** CVE-2008-0374: Printer sends configuration information, including administrative password, in cleartext.

**•** CVE-2007-4961: Chain: cleartext transmission of the MD5 hash of password enables attacks against a server that is susceptible to replay (CWE-294).

**•** CVE-2007-4786: Product sends passwords in cleartext to a log server.

**•** CVE-2005-3140: Product sends file with cleartext passwords in e-mail message intended for diagnostic purposes.

## Related Attack Patterns (CAPEC)

* CAPEC-157
* CAPEC-158
* CAPEC-204
* CAPEC-31
* CAPEC-37
* CAPEC-383
* CAPEC-384
* CAPEC-385
* CAPEC-386
* CAPEC-387
* CAPEC-388
* CAPEC-477
* CAPEC-609
* CAPEC-65

## Attack TTPs

**•** T1539: Steal Web Session Cookie (Tactics: credential-access)

**•** T1040: Network Sniffing (Tactics: credential-access, discovery)

**•** T1005: Data from Local System (Tactics: collection)

**•** T1552.004: Private Keys (Tactics: credential-access)

**•** T1111: Multi-Factor Authentication Interception (Tactics: credential-access)

**•** T1056.004: Credential API Hooking (Tactics: collection, credential-access)

## Modes of Introduction

**•** Architecture and Design: OMISSION: This weakness is caused by missing a security tactic during the architecture and design phase.

**•** Operation: N/A

## Common Consequences

**•** Impact: Read Application Data — Notes: If the application does not use a secure channel, such as SSL, to exchange sensitive information, it is possible for an attacker with access to the network traffic to sniff packets from the connection and uncover the data. This attack is not technically difficult, but does require physical access to some portion of the network over which the sensitive data travels. This access is usually somewhere near where the user is connected to the network (such as a colleague on the company network) but can be anywhere along the path from the user to the end server.

**•** Impact: Modify Application Data — Notes: Omitting the use of encryption in any program which transfers data over a network of any kind should be considered on par with delivering the data sent to each user on the local networks of both the sender and receiver. Worse, this omission allows for the injection of data into a stream of communication between two parties -- with no means for the victims to separate valid data from invalid. In this day of widespread network attacks and password collection sniffers, it is an unnecessary risk to omit encryption from the design of any system which might benefit from it.

## Potential Mitigations

**•** Requirements: Clearly specify which data or resources are valuable enough that they should be protected by encryption. Require that any transmission or storage of this data/resource should use well-vetted encryption algorithms. (Effectiveness: N/A)

**•** Architecture and Design: Ensure that encryption is properly integrated into the system design, including but not necessarily limited to: Encryption that is needed to store or transmit private data of the users of the system Encryption that is needed to protect the system itself from unauthorized disclosure or tampering Identify the separate needs and contexts for encryption: One-way (i.e., only the user or recipient needs to have the key). This can be achieved using public key cryptography, or other techniques in which the encrypting party (i.e., the product) does not need to have access to a private key. Two-way (i.e., the encryption can be automatically performed on behalf of a user, but the key must be available so that the plaintext can be automatically recoverable by that user). This requires storage of the private key in a format that is recoverable only by the user (or perhaps by the operating system) in a way that cannot be recovered by others. Using threat modeling or other techniques, assume that data can be compromised through a separate vulnerability or weakness, and determine where encryption will be most effective. Ensure that data that should be private is not being inadvertently exposed using weaknesses such as insecure permissions (CWE-732). [REF-7] (Effectiveness: N/A)

**•** Architecture and Design: When there is a need to store or transmit sensitive data, use strong, up-to-date cryptographic algorithms to encrypt that data. Select a well-vetted algorithm that is currently considered to be strong by experts in the field, and use well-tested implementations. As with all cryptographic mechanisms, the source code should be available for analysis. For example, US government systems require FIPS 140-2 certification. Do not develop custom or private cryptographic algorithms. They will likely be exposed to attacks that are well-understood by cryptographers. Reverse engineering techniques are mature. If the algorithm can be compromised if attackers find out how it works, then it is especially weak. Periodically ensure that the cryptography has not become obsolete. Some older algorithms, once thought to require a billion years of computing time, can now be broken in days or hours. This includes MD4, MD5, SHA1, DES, and other algorithms that were once regarded as strong. [REF-267] (Effectiveness: N/A)

**•** Architecture and Design: Compartmentalize the system to have "safe" areas where trust boundaries can be unambiguously drawn. Do not allow sensitive data to go outside of the trust boundary and always be careful when interfacing with a compartment outside of the safe area. Ensure that appropriate compartmentalization is built into the system design, and the compartmentalization allows for and reinforces privilege separation functionality. Architects and designers should rely on the principle of least privilege to decide the appropriate time to use privileges and the time to drop privileges. (Effectiveness: N/A)

**•** Implementation: When using industry-approved techniques, use them correctly. Don't cut corners by skipping resource-intensive steps (CWE-325). These steps are often essential for preventing common attacks. (Effectiveness: N/A)

**•** Implementation: Use naming conventions and strong types to make it easier to spot when sensitive data is being used. When creating structures, objects, or other complex entities, separate the sensitive and non-sensitive data as much as possible. (Effectiveness: Defense in Depth)

## Applicable Platforms

**•** None (Class: Not Language-Specific, Prevalence: Undetermined)

## Demonstrative Examples

**•** The code stores the user's username and password in plaintext in a cookie on the user's machine. This exposes the user's login information if their computer is compromised by an attacker. Even if the user's machine is not compromised, this weakness combined with cross-site scripting (CWE-79) could allow an attacker to remotely copy the cookie.

**•** While successful, the program does not encrypt the data before writing it to a buffer, possibly exposing it to unauthorized actors.

**•** Though a connection is successfully made, the connection is unencrypted and it is possible that all sensitive data sent to or received from the server will be read by unintended actors.

## Notes

**•** Relationship: There is an overlapping relationship between insecure storage of sensitive information (CWE-922) and missing encryption of sensitive information (CWE-311). Encryption is often used to prevent an attacker from reading the sensitive data. However, encryption does not prevent the attacker from erasing or overwriting the data.