# CWE Detail – CWE-209

## Description

The product generates an error message that includes sensitive information about its environment, users, or associated data.

## Extended Description

The sensitive information may be valuable information on its own (such as a password), or it may be useful for launching other, more serious attacks. The error message may be created in different ways: self-generated: the source code explicitly constructs the error message and delivers it externally-generated: the external environment, such as a language interpreter, handles the error and constructs its own message, whose contents are not under direct control by the programmer An attacker may use the contents of error messages to help launch another, more focused attack. For example, an attempt to exploit a path traversal weakness (CWE-22) might yield the full pathname of the installed application. In turn, this could be used to select the proper number of ".." sequences to navigate to the targeted file. An attack using SQL injection (CWE-89) might not initially succeed, but an error message could reveal the malformed query, which would expose query logic and possibly even passwords or other sensitive information used within the query.

## Threat-Mapped Scoring

Score: 3.25

Priority: P2 - Serious (High)

## Observed Examples (CVEs)

**•** CVE-2008-2049: POP3 server reveals a password in an error message after multiple APOP commands are sent. Might be resultant from another weakness.

**•** CVE-2007-5172: Program reveals password in error message if attacker can trigger certain database errors.

**•** CVE-2008-4638: Composite: application running with high privileges (CWE-250) allows user to specify a restricted file to process, which generates a parsing error that leaks the contents of the file (CWE-209).

**•** CVE-2008-1579: Existence of user names can be determined by requesting a nonexistent blog and reading the error message.

**•** CVE-2007-1409: Direct request to library file in web application triggers pathname leak in error message.

**•** CVE-2008-3060: Malformed input to login page causes leak of full path when IMAP call fails.

**•** CVE-2005-0603: Malformed regexp syntax leads to information exposure in error message.

**•** CVE-2017-9615: verbose logging stores admin credentials in a world-readablelog file

**•** CVE-2018-1999036: SSH password for private key stored in build log

## Related Attack Patterns (CAPEC)

* CAPEC-215
* CAPEC-463
* CAPEC-54
* CAPEC-7

## Modes of Introduction

**•** Architecture and Design: N/A

**•** Implementation: REALIZATION: This weakness is caused during implementation of an architectural security tactic.

**•** System Configuration: N/A

**•** Operation: N/A

## Common Consequences

**•** Impact: Read Application Data — Notes: Often this will either reveal sensitive information which may be used for a later attack or private information stored in the server.

## Potential Mitigations

**•** Implementation: Ensure that error messages only contain minimal details that are useful to the intended audience and no one else. The messages need to strike the balance between being too cryptic (which can confuse users) or being too detailed (which may reveal more than intended). The messages should not reveal the methods that were used to determine the error. Attackers can use detailed information to refine or optimize their original attack, thereby increasing their chances of success. If errors must be captured in some detail, record them in log messages, but consider what could occur if the log messages can be viewed by attackers. Highly sensitive information such as passwords should never be saved to log files. Avoid inconsistent messaging that might accidentally tip off an attacker about internal state, such as whether a user account exists or not. (Effectiveness: N/A)

**•** Implementation: Handle exceptions internally and do not display errors containing potentially sensitive information to a user. (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)

**•** Implementation: Debugging information should not make its way into a production release. (Effectiveness: N/A)

**•** Implementation: Debugging information should not make its way into a production release. (Effectiveness: N/A)

**•** System Configuration: Where available, configure the environment to use less verbose error messages. For example, in PHP, disable the display\_errors setting during configuration, or at runtime using the error\_reporting() function. (Effectiveness: N/A)

**•** System Configuration: Create default error pages or messages that do not leak any information. (Effectiveness: N/A)

## Applicable Platforms

**•** PHP (Class: None, Prevalence: Often)

**•** Java (Class: None, Prevalence: Often)

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

## Demonstrative Examples

**•** If an exception related to SQL is handled by the catch, then the output might contain sensitive information such as SQL query structure or private information. If this output is redirected to a web user, this may represent a security problem.

**•** If an exception occurs, the printed message exposes the location of the configuration file the script is using. An attacker can use this information to target the configuration file (perhaps exploiting a Path Traversal weakness). If the file can be read, the attacker could gain credentials for accessing the database. The attacker may also be able to replace the file with a malicious one, causing the application to use an arbitrary database.

**•** If this code is running on a server, such as a web application, then the person making the request should not know what the full pathname of the configuration directory is. By submitting a username that does not produce a $file that exists, an attacker could get this pathname. It could then be used to exploit path traversal or symbolic link following problems that may exist elsewhere in the application.

**•** The error message that is created includes information about the database query that may contain sensitive information about the database or query logic. In this case, the error message will expose the table name and column names used in the database. This data could be used to simplify other attacks, such as SQL injection (CWE-89) to directly access the database.