# CWE Detail – CWE-200

## Description

The product exposes sensitive information to an actor that is not explicitly authorized to have access to that information.

## Extended Description

There are many different kinds of mistakes that introduce information exposures. The severity of the error can range widely, depending on the context in which the product operates, the type of sensitive information that is revealed, and the benefits it may provide to an attacker. Some kinds of sensitive information include: private, personal information, such as personal messages, financial data, health records, geographic location, or contact details system status and environment, such as the operating system and installed packages business secrets and intellectual property network status and configuration the product's own code or internal state metadata, e.g. logging of connections or message headers indirect information, such as a discrepancy between two internal operations that can be observed by an outsider Information might be sensitive to different parties, each of which may have their own expectations for whether the information should be protected. These parties include: the product's own users people or organizations whose information is created or used by the product, even if they are not direct product users the product's administrators, including the admins of the system(s) and/or networks on which the product operates the developer Information exposures can occur in different ways: the code explicitly inserts sensitive information into resources or messages that are intentionally made accessible to unauthorized actors, but should not contain the information - i.e., the information should have been "scrubbed" or "sanitized" a different weakness or mistake indirectly inserts the sensitive information into resources, such as a web script error revealing the full system path of the program. the code manages resources that intentionally contain sensitive information, but the resources are unintentionally made accessible to unauthorized actors. In this case, the information exposure is resultant - i.e., a different weakness enabled the access to the information in the first place. It is common practice to describe any loss of confidentiality as an "information exposure," but this can lead to overuse of CWE-200 in CWE mapping. From the CWE perspective, loss of confidentiality is a technical impact that can arise from dozens of different weaknesses, such as insecure file permissions or out-of-bounds read. CWE-200 and its lower-level descendants are intended to cover the mistakes that occur in behaviors that explicitly manage, store, transfer, or cleanse sensitive information.

## Threat-Mapped Scoring

Score: 3.25

Priority: P2 - Serious (High)

## Observed Examples (CVEs)

**•** CVE-2022-31162: Rust library leaks Oauth client details in application debug logs

**•** CVE-2021-25476: Digital Rights Management (DRM) capability for mobile platform leaks pointer information, simplifying ASLR bypass

**•** CVE-2001-1483: Enumeration of valid usernames based on inconsistent responses

**•** CVE-2001-1528: Account number enumeration via inconsistent responses.

**•** CVE-2004-2150: User enumeration via discrepancies in error messages.

**•** CVE-2005-1205: Telnet protocol allows servers to obtain sensitive environment information from clients.

**•** CVE-2002-1725: Script calls phpinfo(), revealing system configuration to web user

**•** CVE-2002-0515: Product sets a different TTL when a port is being filtered than when it is not being filtered, which allows remote attackers to identify filtered ports by comparing TTLs.

**•** CVE-2004-0778: Version control system allows remote attackers to determine the existence of arbitrary files and directories via the -X command for an alternate history file, which causes different error messages to be returned.

**•** CVE-2000-1117: Virtual machine allows malicious web site operators to determine the existence of files on the client by measuring delays in the execution of the getSystemResource method.

**•** CVE-2003-0190: Product immediately sends an error message when a user does not exist, which allows remote attackers to determine valid usernames via a timing attack.

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

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

**•** CVE-2004-2268: Password exposed in debug information.

**•** CVE-2003-1078: FTP client with debug option enabled shows password to the screen.

**•** CVE-2022-0708: Collaboration platform does not clear team emails in a response, allowing leak of email addresses

## Related Attack Patterns (CAPEC)

* CAPEC-116
* CAPEC-13
* CAPEC-169
* CAPEC-22
* CAPEC-224
* CAPEC-285
* CAPEC-287
* CAPEC-290
* CAPEC-291
* CAPEC-292
* CAPEC-293
* CAPEC-294
* CAPEC-295
* CAPEC-296
* CAPEC-297
* CAPEC-298
* CAPEC-299
* CAPEC-300
* CAPEC-301
* CAPEC-302
* CAPEC-303
* CAPEC-304
* CAPEC-305
* CAPEC-306
* CAPEC-307
* CAPEC-308
* CAPEC-309
* CAPEC-310
* CAPEC-312
* CAPEC-313
* CAPEC-317
* CAPEC-318
* CAPEC-319
* CAPEC-320
* CAPEC-321
* CAPEC-322
* CAPEC-323
* CAPEC-324
* CAPEC-325
* CAPEC-326
* CAPEC-327
* CAPEC-328
* CAPEC-329
* CAPEC-330
* CAPEC-472
* CAPEC-497
* CAPEC-508
* CAPEC-573
* CAPEC-574
* CAPEC-575
* CAPEC-576
* CAPEC-577
* CAPEC-59
* CAPEC-60
* CAPEC-616
* CAPEC-643
* CAPEC-646
* CAPEC-651
* CAPEC-79

## Attack TTPs

**•** T1033: System Owner/User Discovery (Tactics: discovery)

**•** T1592: Gather Victim Host Information (Tactics: reconnaissance)

**•** T1574.007: Path Interception by PATH Environment Variable (Tactics: persistence, privilege-escalation, defense-evasion)

**•** T1069: Permission Groups Discovery (Tactics: discovery)

**•** T1615: Group Policy Discovery (Tactics: discovery)

**•** T1036.005: Match Legitimate Resource Name or Location (Tactics: defense-evasion)

**•** T1007: System Service Discovery (Tactics: discovery)

**•** T1135: Network Share Discovery (Tactics: discovery)

**•** T1120: Peripheral Device Discovery (Tactics: discovery)

**•** T1082: System Information Discovery (Tactics: discovery)

**•** T1217: Browser Information Discovery (Tactics: discovery)

**•** T1574.006: Dynamic Linker Hijacking (Tactics: persistence, privilege-escalation, defense-evasion)

**•** T1595: Active Scanning (Tactics: reconnaissance)

**•** T1016: System Network Configuration Discovery (Tactics: discovery)

**•** T1087: Account Discovery (Tactics: discovery)

**•** T1083: File and Directory Discovery (Tactics: discovery)

**•** T1049: System Network Connections Discovery (Tactics: discovery)

**•** T1134.001: Token Impersonation/Theft (Tactics: defense-evasion, privilege-escalation)

**•** T1057: Process Discovery (Tactics: discovery)

**•** T1562.003: Impair Command History Logging (Tactics: defense-evasion)

**•** T1590: Gather Victim Network Information (Tactics: reconnaissance)

**•** T1550.004: Web Session Cookie (Tactics: defense-evasion, lateral-movement)

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

**•** T1018: Remote System Discovery (Tactics: discovery)

**•** T1046: Network Service Discovery (Tactics: discovery)

**•** T1124: System Time Discovery (Tactics: discovery)

## Modes of Introduction

**•** Architecture and Design: N/A

**•** Implementation: N/A

## Common Consequences

**•** Impact: Read Application Data — Notes:

## Potential Mitigations

**•** 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)

## Applicable Platforms

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

## Demonstrative Examples

**•** In the above code, there are different messages for when an incorrect username is supplied, versus when the username is correct but the password is wrong. This difference enables a potential attacker to understand the state of the login function, and could allow an attacker to discover a valid username by trying different values until the incorrect password message is returned. In essence, this makes it easier for an attacker to obtain half of the necessary authentication credentials.

**•** 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.

**•** 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.

**•** When the application encounters an exception it will write the user object to the log. Because the user object contains location information, the user's location is also written to the log.

**•** The error clearly exposes the database credentials.

**•** The code displays a user's credit card and social security numbers, even though they aren't absolutely necessary.

**•** The code writes sensitive debug information to the client browser if the "debugEnabled" flag is set to true .

**•** First the application must declare that it requires the ACCESS\_FINE\_LOCATION permission in the application's manifest.xml:

## Notes

**•** Maintenance: As a result of mapping analysis in the 2020 Top 25 and more recent versions, this weakness is under review, since it is frequently misused in mapping to cover many problems that lead to loss of confidentiality. See Mapping Notes, Extended Description, and Alternate Terms.