# CWE Detail – CWE-330

## Description

The product uses insufficiently random numbers or values in a security context that depends on unpredictable numbers.

## Extended Description

When product generates predictable values in a context requiring unpredictability, it may be possible for an attacker to guess the next value that will be generated, and use this guess to impersonate another user or access sensitive information.

## Threat-Mapped Scoring

Score: 3.0

Priority: P2 - Serious (High)

## Observed Examples (CVEs)

**•** CVE-2021-3692: PHP framework uses mt\_rand() function (Marsenne Twister) when generating tokens

**•** CVE-2020-7010: Cloud application on Kubernetes generates passwords using a weak random number generator based on deployment time.

**•** CVE-2009-3278: Crypto product uses rand() library function to generate a recovery key, making it easier to conduct brute force attacks.

**•** CVE-2009-3238: Random number generator can repeatedly generate the same value.

**•** CVE-2009-2367: Web application generates predictable session IDs, allowing session hijacking.

**•** CVE-2009-2158: Password recovery utility generates a relatively small number of random passwords, simplifying brute force attacks.

**•** CVE-2009-0255: Cryptographic key created with a seed based on the system time.

**•** CVE-2008-5162: Kernel function does not have a good entropy source just after boot.

**•** CVE-2008-4905: Blogging software uses a hard-coded salt when calculating a password hash.

**•** CVE-2008-4929: Bulletin board application uses insufficiently random names for uploaded files, allowing other users to access private files.

**•** CVE-2008-3612: Handheld device uses predictable TCP sequence numbers, allowing spoofing or hijacking of TCP connections.

**•** CVE-2008-2433: Web management console generates session IDs based on the login time, making it easier to conduct session hijacking.

**•** CVE-2008-0166: SSL library uses a weak random number generator that only generates 65,536 unique keys.

**•** CVE-2008-2108: Chain: insufficient precision causes extra zero bits to be assigned, reducing entropy for an API function that generates random numbers.

**•** CVE-2008-2108: Chain: insufficient precision (CWE-1339) in  
 random-number generator causes some zero bits to be reliably  
 generated, reducing the amount of entropy (CWE-331)

**•** CVE-2008-2020: CAPTCHA implementation does not produce enough different images, allowing bypass using a database of all possible checksums.

**•** CVE-2008-0087: DNS client uses predictable DNS transaction IDs, allowing DNS spoofing.

**•** CVE-2008-0141: Application generates passwords that are based on the time of day.

## Related Attack Patterns (CAPEC)

* CAPEC-112
* CAPEC-485
* CAPEC-59

## Attack TTPs

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

**•** T1110: Brute Force (Tactics: credential-access)

## Modes of Introduction

**•** Architecture and Design: N/A

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

## Common Consequences

**•** Impact: Other — Notes: When a protection mechanism relies on random values to restrict access to a sensitive resource, such as a session ID or a seed for generating a cryptographic key, then the resource being protected could be accessed by guessing the ID or key.

**•** Impact: Bypass Protection Mechanism, Other — Notes: If product relies on unique, unguessable IDs to identify a resource, an attacker might be able to guess an ID for a resource that is owned by another user. The attacker could then read the resource, or pre-create a resource with the same ID to prevent the legitimate program from properly sending the resource to the intended user. For example, a product might maintain session information in a file whose name is based on a username. An attacker could pre-create this file for a victim user, then set the permissions so that the application cannot generate the session for the victim, preventing the victim from using the application.

**•** Impact: Bypass Protection Mechanism, Gain Privileges or Assume Identity — Notes: When an authorization or authentication mechanism relies on random values to restrict access to restricted functionality, such as a session ID or a seed for generating a cryptographic key, then an attacker may access the restricted functionality by guessing the ID or key.

## Potential Mitigations

**•** Architecture and Design: Use a well-vetted algorithm that is currently considered to be strong by experts in the field, and select well-tested implementations with adequate length seeds. In general, if a pseudo-random number generator is not advertised as being cryptographically secure, then it is probably a statistical PRNG and should not be used in security-sensitive contexts. Pseudo-random number generators can produce predictable numbers if the generator is known and the seed can be guessed. A 256-bit seed is a good starting point for producing a "random enough" number. (Effectiveness: N/A)

**•** Implementation: Consider a PRNG that re-seeds itself as needed from high quality pseudo-random output sources, such as hardware devices. (Effectiveness: N/A)

**•** Testing: Use automated static analysis tools that target this type of weakness. Many modern techniques use data flow analysis to minimize the number of false positives. This is not a perfect solution, since 100% accuracy and coverage are not feasible. (Effectiveness: N/A)

**•** Architecture and Design: Use products or modules that conform to FIPS 140-2 [REF-267] to avoid obvious entropy problems. Consult FIPS 140-2 Annex C ("Approved Random Number Generators"). (Effectiveness: N/A)

**•** Testing: Use tools and techniques that require manual (human) analysis, such as penetration testing, threat modeling, and interactive tools that allow the tester to record and modify an active session. These may be more effective than strictly automated techniques. This is especially the case with weaknesses that are related to design and business rules. (Effectiveness: N/A)

## Applicable Platforms

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

## Demonstrative Examples

**•** Because the seed for the PRNG is always the user's ID, the session ID will always be the same. An attacker could thus predict any user's session ID and potentially hijack the session.

**•** This code uses the Random.nextInt() function to generate "unique" identifiers for the receipt pages it generates. Because Random.nextInt() is a statistical PRNG, it is easy for an attacker to guess the strings it generates. Although the underlying design of the receipt system is also faulty, it would be more secure if it used a random number generator that did not produce predictable receipt identifiers, such as a cryptographic PRNG.

## Notes

**•** Relationship: This can be primary to many other weaknesses such as cryptographic errors, authentication errors, symlink following, information leaks, and others.

**•** Maintenance: As of CWE 4.3, CWE-330 and its descendants are being  
 investigated by the CWE crypto team to identify gaps  
 related to randomness and unpredictability, as well as  
 the relationships between randomness and cryptographic  
 primitives. This "subtree analysis" might  
 result in the addition or deprecation of existing  
 entries; the reorganization of relationships in some  
 views, e.g. the research view (CWE-1000); more consistent  
 use of terminology; and/or significant modifications to  
 related entries.

**•** Maintenance: As of CWE 4.5, terminology related to randomness, entropy, and  
 predictability can vary widely. Within the developer and other  
 communities, "randomness" is used heavily. However, within  
 cryptography, "entropy" is distinct, typically implied as a  
 measurement. There are no commonly-used definitions, even within  
 standards documents and cryptography papers. Future versions of  
 CWE will attempt to define these terms and, if necessary,  
 distinguish between them in ways that are appropriate for  
 different communities but do not reduce the usability of CWE for  
 mapping, understanding, or other scenarios.