# CWE Detail – CWE-338

## Description

The product uses a Pseudo-Random Number Generator (PRNG) in a security context, but the PRNG's algorithm is not cryptographically strong.

## Extended Description

When a non-cryptographic PRNG is used in a cryptographic context, it can expose the cryptography to certain types of attacks. Often a pseudo-random number generator (PRNG) is not designed for cryptography. Sometimes a mediocre source of randomness is sufficient or preferable for algorithms that use random numbers. Weak generators generally take less processing power and/or do not use the precious, finite, entropy sources on a system. While such PRNGs might have very useful features, these same features could be used to break the cryptography.

## Threat-Mapped Scoring

Score: 1.8

Priority: P4 - Informational (Low)

## Observed Examples (CVEs)

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

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

## Modes of Introduction

**•** Architecture and Design: N/A

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

## Common Consequences

**•** Impact: Bypass Protection Mechanism — Notes: If a PRNG is used for authentication and authorization, such as a session ID or a seed for generating a cryptographic key, then an attacker may be able to easily guess the ID or cryptographic key and gain access to restricted functionality.

## Potential Mitigations

**•** Implementation: Use functions or hardware which use a hardware-based random number generation for all crypto. This is the recommended solution. Use CyptGenRandom on Windows, or hw\_rand() on Linux. (Effectiveness: N/A)

## Applicable Platforms

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

## Demonstrative Examples

**•** The random number functions used in these examples, rand() and Random.nextInt(), are not considered cryptographically strong. An attacker may be able to predict the random numbers generated by these functions. Note that these example also exhibit CWE-337 (Predictable Seed in PRNG).

## Notes

**•** 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.