# CWE Detail – CWE-1389

## Description

The product parses numeric input assuming base 10 (decimal) values, but it does not account for inputs that use a different base number (radix).

## Extended Description

Frequently, a numeric input that begins with "0" is treated as octal, or "0x" causes it to be treated as hexadecimal, e.g. by the inet\_addr() function. For example, "023" (octal) is 35 decimal, or "0x31" is 49 decimal. Other bases may be used as well. If the developer assumes decimal-only inputs, the code could produce incorrect numbers when the inputs are parsed using a different base. This can result in unexpected and/or dangerous behavior. For example, a "0127.0.0.1" IP address is parsed as octal due to the leading "0", whose numeric value would be the same as 87.0.0.1 (decimal), where the developer likely expected to use 127.0.0.1. The consequences vary depending on the surrounding code in which this weakness occurs, but they can include bypassing network-based access control using unexpected IP addresses or netmasks, or causing apparently-symbolic identifiers to be processed as if they are numbers. In web applications, this can enable bypassing of SSRF restrictions.

## Threat-Mapped Scoring

Score: 0.0

Priority: Unclassified

## Observed Examples (CVEs)

**•** CVE-2021-29662: Chain: Use of zero-prepended IP addresses in Perl-based IP validation module can lead to an access control bypass.

**•** CVE-2021-28918: Chain: Use of zero-prepended IP addresses in a product that manages IP blocks can lead to an SSRF.

**•** CVE-2021-29921: Chain: Use of zero-prepended IP addresses in a Python standard library package can lead to an SSRF.

**•** CVE-2021-29923: Chain: Use of zero-prepended IP addresses in the net Golang library can lead to an access control bypass.

**•** CVE-2021-29424: Chain: Use of zero-prepended IP addresses in Perl netmask module allows bypass of IP-based access control.

**•** CVE-2016-4029: Chain: incorrect validation of intended decimal-based IP address format (CWE-1286) enables parsing of octal or hexadecimal formats (CWE-1389), allowing bypass of an SSRF protection mechanism (CWE-918).

**•** CVE-2020-13776: Mishandling of hex-valued usernames leads to unexpected decimal conversion and privilege escalation in the systemd Linux suite.

## Modes of Introduction

**•** Implementation: Input validation used may assume decimal bases during conditional checks, when it may not always be the case.

**•** Implementation: The application may rely on a service that supports different numerical bases.

## Common Consequences

**•** Impact: Read Application Data — Notes: An attacker may use an unexpected numerical base to access private application resources.

**•** Impact: Bypass Protection Mechanism, Alter Execution Logic — Notes: An attacker may use an unexpected numerical base to bypass or manipulate access control mechanisms.

## Potential Mitigations

**•** Implementation: If only decimal-based values are expected in the application, conditional checks should be created in a way that prevent octal or hexadecimal strings from being checked. This can be achieved by converting any numerical string to an explicit base-10 integer prior to the conditional check, to prevent octal or hex values from ever being checked against the condition. (Effectiveness: N/A)

**•** Implementation: If various numerical bases do need to be supported, check for leading values indicating the non-decimal base you wish to support (such as 0x for hex) and convert the numeric strings to integers of the respective base. Reject any other alternative-base string that is not intentionally supported by the application. (Effectiveness: N/A)

**•** Implementation: If regular expressions are used to validate IP addresses, ensure that they are bounded using ^ and $ to prevent base-prepended IP addresses from being matched. (Effectiveness: N/A)

## Applicable Platforms

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

## Demonstrative Examples

**•** If run\_ping() were to be called with one or more zero-prepended octets, validate\_ip() will succeed as zero-prepended numerical strings can be interpreted as decimal by a cast ("012" would cast to 12). However, as the original IP with the prepended zeroes is returned rather than the casted IP, it will be used in the call to the ping command. Ping DOES check and support octal-based IP octets, so the IP reached via ping may be different than the IP assumed by the validator. For example, ping would considered "0127.0.0.1" the same as "87.0.0.1".

**•** Since the regular expression does not have anchors (CWE-777), i.e. is unbounded without ^ or $ characters, then prepending a 0 or 0x to the beginning of the IP address will still result in a matched regex pattern. Since the ping command supports octal and hex prepended IP addresses, it will use the unexpectedly valid IP address (CWE-1389). For example, "0x63.63.63.63" would be considered equivalent to "99.63.63.63". As a result, the attacker could potentially ping systems that the attacker cannot reach directly.

**•** The entry for night.example.com has a typo "010" in the first octet. When using ping to ensure the servers are up, the leading 0 causes the IP address to be converted using octal. So when Kelly's script attempts to access night.example.com, it inadvertently scans 8.1.0.8 instead of 10.1.0.8 (since "010" in octal is 8 in decimal), and Night is free to send new Tweets without being immediately detected.