House Absolute(ly Pointless)

- Home
- Archives
- About
-

Security Issues in Perl IP Address distros

2021-03-29

Edit on 2021-03-29 21:40(ish) UTC: Added Net-Subnet (appears unaffected) and reordered the details to match the list at the top of the post.

Edit on 2021-03-30 14:50(ish) UTC: Added Net-Works (appears unaffected).

Edit on 2021-03-30 15:40(ish) UTC: Added Net-CIDR (some functions are affected).

Edit on 2021-03-31 01:05(ish) UTC: Added Net-IPv4Addr (affected).

Edit on 2021-04-05 01:21(ish) UTC: Net-CIDR-Lite 0.22 contains a remediation.

Edit on 2021-04-05 19:30(ish) UTC: Net-IPAddress-Util 5.000 contains a remediation.

warning

TLDR: Some Perl modules for working with IP addresses and netmasks have bugs with potential security applications. See below for more details on the bug and which modules are affected.

- Net-IPv4Addr: Affected.
- Net-CIDR-Lite: Vulnerable before the 0.22 release. Upgrade now.
- Net-Netmask: Vulnerable before the 2.00000 release. Upgrade now.
- Net-IPAddress-Util: Vulnerable before the 5.000 release. Upgrade now.
- Data-Validate-IP: Depends on exactly how it's used. See below for details.
- Net-CIDR: Depends on exactly how it's used. See below for details.
- Socket: Appears unaffected.
- Net-DNS: Appears unaffected.
- NetAddr-IP: Appears unaffected.
- Net-Works: Appears unaffected.
- Net-Subnet: Appears unaffected.
- Net-Patricia: Appears unaffected.

Yesterday, a security issue with the NPM package `netmask` was published[1]

One of the most ridiculous URL paths I've ever seen.

.

The issue itself is pretty straightforward. Your OS will allow an IP address like `010.0.0.1` or a netmask like `010.0.0.0/8`. That `010` is treated **as an octal number, not a base-10 number with a leading zero!** That means that `010.0.0.1` is actually `8.0.0.1`. We can confirm this with `ping`:

```
1 ping 010.0.0.1
2 PING 010.0.0.1 (8.0.0.1) 56(84) bytes of data.
```

But the NPM `netmask` package would treat this as `10.0.0.1`. This confusion means that an application could be tricked into thinking a public IP - `8.0.0.1` - was part of a private subnet - `10.0.0.0/8`. And conversely, you could trick it into thinking a private IP - `10.0.0.1` written as `012.0.0.1` - was part of a public subnet - `12.0.0.1`.

This has security implications for any application that is trying to distinguish between public and private IP addresses or networks for access control, firewalling, etc.

As I was reading about this I checked out the Git repo for the `netmask` package. Its README says "This module is highly inspired by Perl Net::Netmask module."

And at that point I realized that it was quite possible that this affected Perl code as well! So I started digging into this by looking at various CPAN modules for working with IP addresses, networks, and netmasks.

Here's the current state of CPAN modules, ordered roughly by their position in The River of CPAN (which basically means how many modules depend on them).

Net-IPv4Addr

warning
This distribution is affected by this issue. In addition, this module is almost certainly no longer being maintained. Emails to the author bounce.

This distribution has 4 direct dependents and 12 total dependents.

```
1 perl -MNet::IPv4Addr=:all -E 'say $_ for ipv4_network("010.0.0.1")'
2 10.0.0.0
3 8
```

Net-CIDR-Lite

info
This distribution was vulnerable prior to its 0.22 release made on 2021-04-04. Thanks to Stig Palmquist for taking this distro over and releasing a fix!

This distribution has 24 direct dependents and 36 total dependents.

```
1 perl -MNet::CIDR::Lite -E 'my $c = Net::CIDR::Lite->new; $c->add("010.0.0.0/8"); say $_ for $c->list_range'
2 Can't determine ip format at /home/autarch/.perlbrew/libs/perl-5.30.1@dev/lib/perl5/Net/CIDR/Lite.pm line 38.
3         Net::CIDR::Lite::add(Net::CIDR::Lite=HASH(0x55fe55ade740), "010.0.0.0/8") called at -e line 1
```

Net-Netmask

info
This distribution was vulnerable prior to its 2.0000 release earlier today. Great job on the quick response, Joelle Maslak!

This distribution has 22 direct dependents and 30 total dependents.

So for versions before 2.0000 we see this:

```
perl -MNet::Netmask -E 'say defined Net::Netmask->new2(q{010.0.0.0/8}) ? 1 : 0'
0
```

Note the use of the `new2` constructor. The old `new` constructor cannot be changed to return `undef` for backwards compatibility reasons. Fortunately, it's probably not vulnerable in any exploitable way, as it returns a 0-length subnet:

```
perl -MNet::Netmask -E 'say Net::Netmask->new(q{010.0.0.0/8})'
0.0.0.0/0
```

Net-IPAddress-Util

info
This distribution was vulnerable prior to its 5.000 release made on 2021-04-04. Thanks to Paul W Bennett for the fix!

This distribution has no dependents.

```
perl -MNet::IPAddress::Util=IP -E 'say IP(q{010.0.0.1})'
8.0.0.1
```

Data-Validate-IP

info
This distribution doesn't misparse octal numbers, but you could be affected depending on exactly how your code uses this distro. See below for details.

This distribution has 21 direct dependents and 60 total dependents.

This distribution returns false for any `is_*_ipv4` method that includes an octal number. So both `is_private_ipv4('010.0.0.1')` and `is_public_ipv4('010.0.0.1')` return false. **Depending on how you're using this module, it's *possible* that this could lead to bugs, including bugs with security implications.**

I updated the documentation to explicitly recommend that you **always call `is_ipv4()` in addition to calling a method like `is_private_ipv4()`**. The `is_ipv4()` method will always return false for IP addresses with octal numbers.

While this isn't strictly POSIX-correct, this seems like the safest behavior for a module like this. It's better to be too strict if this eliminates a potential footgun.

If you are using this distribution, I highly encourage you to audit your use of it in a security context!

Net-CIDR

info
This distribution is affected, but it has a function to validate CIDR strings that you should use before calling any other functions.

This distribution has 17 direct dependents and 25 total dependents.

The distribution provides a number of functions for working with networks and IP addresses. Most of these are not affected. However, two are:

```
perl -MNet::CIDR -E 'say for Net::CIDR::addr2cidr("010.0.0.1")'
010.0.0.1/32
010.0.0.0/31
...
010.0.0.0/8
10.0.0.0/7
8.0.0.0/6
...

perl -MNet::CIDR -E 'say Net::CIDR::cidrlookup("10.0.0.1", "010.0.0.0/8")'
1
```

However, this distribution also contains a `cidrvalidate` function that will return false for any CIDR string with a leading 0 in an octet. The documentation explicitly tells you to use this before passing the data to other functions.

If you are using this distribution, I highly encourage you to audit your use of it in a security context!

Socket

note
This distribution appears to be unaffected by this issue.

This distribution has 275 direct dependents and 9,936 total dependents.

```
perl -MSocket -E 'say inet_ntoa(inet_aton(q{010.0.0.1}))'
8.0.0.1

perl -MSocket=inet_pton,inet_ntop,AF_INET -E 'say inet_ntop(AF_INET, inet_pton(AF_INET, q{010.0.0.1}))'
Bad address length for Socket::inet_ntop on AF_INET; got 0, should be 4 at -e line 1.
```

The `inet_pton()` function is just returning `undef` for this octal-formatted address.

Net-DNS

note
This distribution appears to be unaffected by this issue.

This distribution has 104 direct dependents and 561 total dependents.

If you try to resolve an IP address, it turns this into a reverse lookup, but it treats the IP as text:

```
perl ./demo/perldig 010.0.0.1
;; Response received from 127.0.0.53 (40 octets)
;; HEADER SECTION
;;      id = 6342
;;      qr = 1   aa = 0   tc = 0   rd = 1   opcode = QUERY
;;      ra = 1   z  = 0   ad = 0   cd = 0   rcode  = NXDOMAIN
;;      qdcount = 1      ancount = 0      nscount = 0      arcount = 0
;;      do = 0

;; QUESTION SECTION (1 record)
;; 1.0.0.010.in-addr.arpa.      IN      A
```

So it's not a useful answer, but it's not looking up the *wrong* address.

NetAddr-IP

note
This distribution appears to be unaffected by this issue.

This distribution has 36 direct dependents and 110 total dependents.

```
perl -MNetAddr::IP -E 'say NetAddr::IP->new(q{010.0.0.024})'
8.0.0.20/32
```

Net-Works

note
This distribution appears to be unaffected by this issue.

This distribution has 3 direct dependents and 7 total dependents.

```
perl -MNet::Works::Network -E 'say Net::Works::Network->new_from_string(string => q{010.0.0.1/8})'
010.0.0.1/8 is not a valid IP network at /home/autarch/.perlbrew/libs/perl-5.30.1@dev/lib/perl5/Net/Works/Network.pm line 120.
        Net::Works::Network::new_from_string("Net::Works::Network", "string", "010.0.0.1/8") called at -e line 1

perl -MNet::Works::Address -E 'say Net::Works::Address->new_from_string(string => q{010.0.0.1})'
010.0.0.1 is not a valid IPv6 address at /home/autarch/.perlbrew/libs/perl-5.30.1@dev/lib/perl5/Net/Works/Util.pm line 70.
        Net::Works::Util::_validate_ip_string("010.0.0.1", 6) called at /home/autarch/.perlbrew/libs/perl-5.30.1@dev/lib/perl5/Net/Works/Address.pm line 74
        Net::Works::Address::new_from_string("Net::Works::Address", "string", "010.0.0.1") called at -e line 1
```

Thanks to Stig Palmquist for checking this one and letting me know.

Net-Subnet

note
This distribution appears to be unaffected by this issue.

This distribution has 3 direct dependents and 7 total dependents.

```
perl -MNet::Subnet -E 'my $m = subnet_matcher(q{10.0.0.0/8}); say $m->(q{012.0.0.1}) ? 1 : 0'
1

perl -MNet::Subnet -E 'my $m = subnet_matcher(q{012.0.0.0/8}); say $m->(q{10.0.0.1}) ? 1 : 0'
1
```

Net-Patricia

note
This distribution appears to be unaffected by this issue.

This distribution has 1 direct dependent and 1 total dependent.

```
perl -MNet::Patricia -E 'my $p = Net::Patricia->new; $p->add_string("010.0.0.0/8"); say $p->match_string("8.0.0.1") ? 1 : 0'
1

perl -MNet::Patricia -E 'my $p = Net::Patricia->new; $p->add_string("8.0.0.0/8"); say $p->match_string("010.0.0.1") ? 1 : 0'
1
```

---

1. One of the most ridiculous URL paths I've ever seen. ↩

‹ Down the Golang nil Rabbit Hole Next API Design and the Recent IP Address Module Issues Prev ›
