Rotten code, aging standards, & pwning IPv4 parsing

Kelly Kaoudis, @kaoudis
Institute for Defense Analyses (IDA), 14 Jul 2022
originally presented at DEF CON 29 with Sick Codes

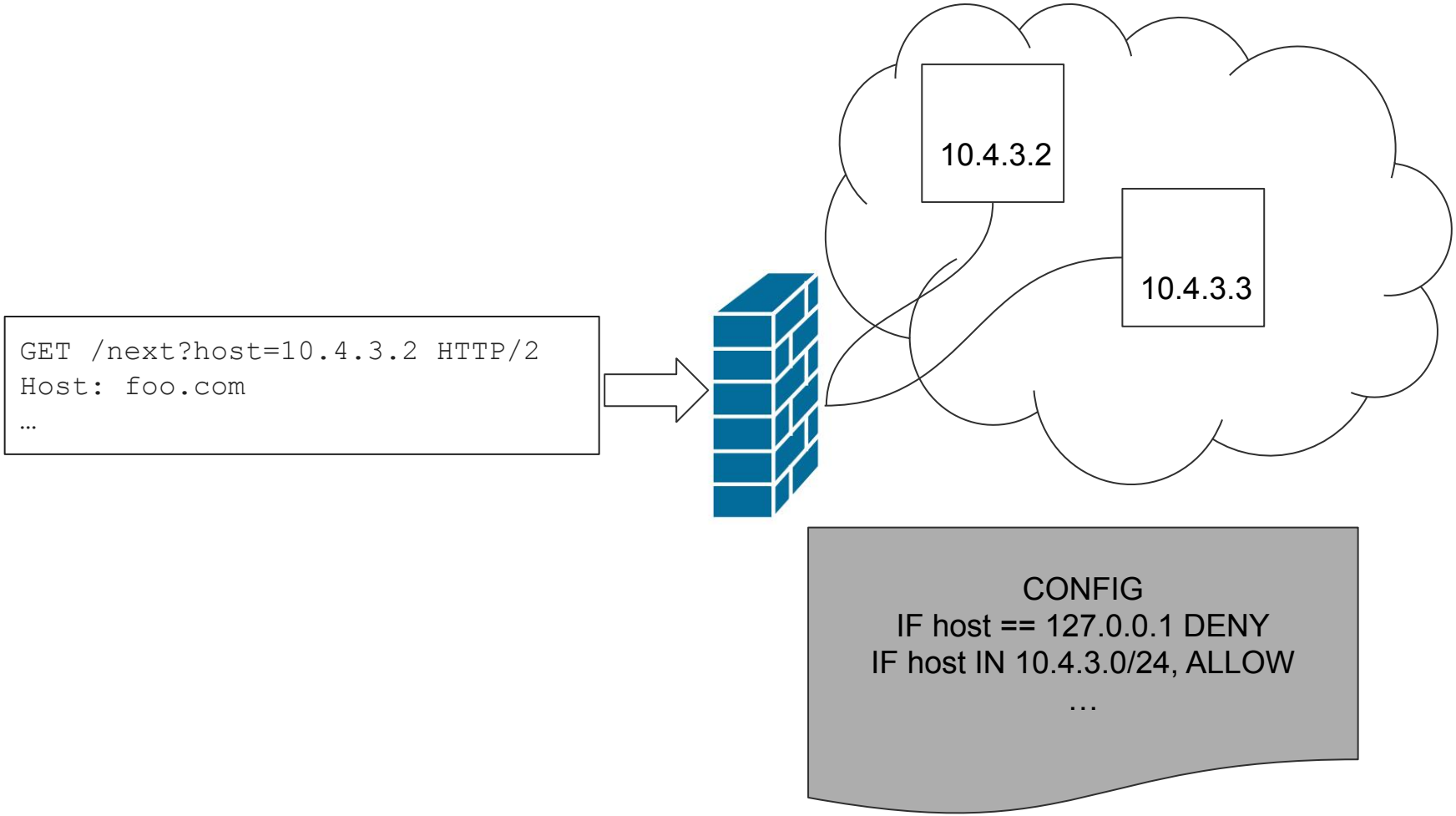
agenda

- the problem
- what makes this interesting?
- examples
- (brief) history
- potential defenses
- credits

***problem*: are two IP addresses equivalent?**

example: ***127.0.0.1***

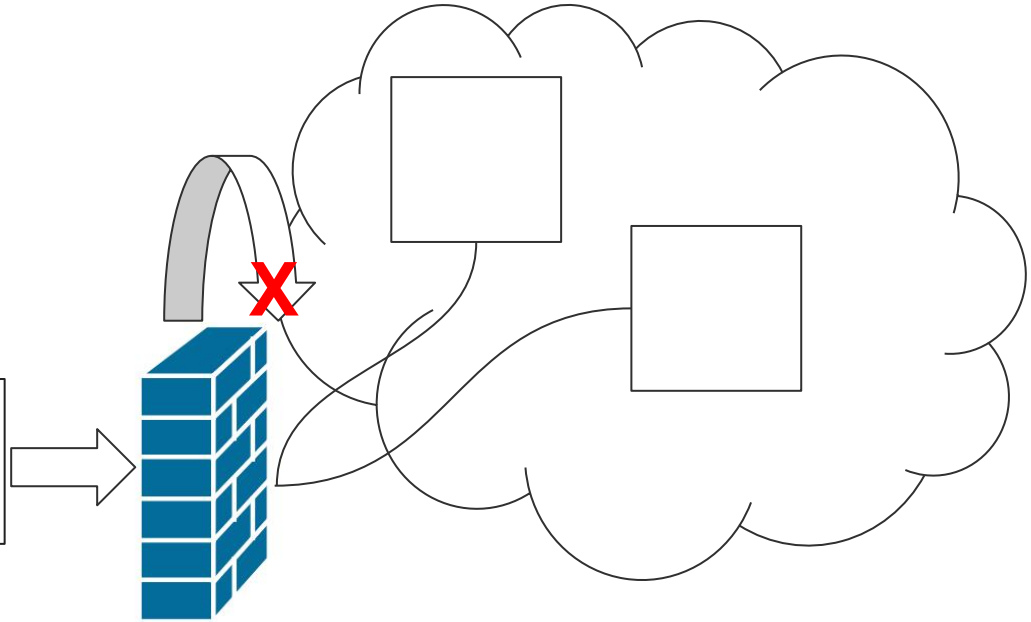
```
NetRange:       127.0.0.0 - 127.255.255.255
CIDR:           127.0.0.0/8
NetName:        SPECIAL-IPV4-LOOPBACK-IANA-RESERVED
NetHandle:      NET-127-0-0-0-1
Parent:          ()
NetType:        IANA Special Use
OriginAS:
Organization:   Internet Assigned Numbers Authority (IANA)
RegDate:
Updated:        2013-08-30
Comment:        Addresses starting with "127." are used when one program needs to talk to another program running on the same machine using the Internet
Comment:        Protocol.  127.0.0.1 is the most commonly used address and is called the "loopback" address.
Comment:
```


GET /next?host=10.4.3.2 HTTP/2
Host: foo.com
…
10.4.3.2
10.4.3.3
CONFIG
IF host == 127.0.0.1 DENY
IF host IN 10.4.3.0/24, ALLOW
…

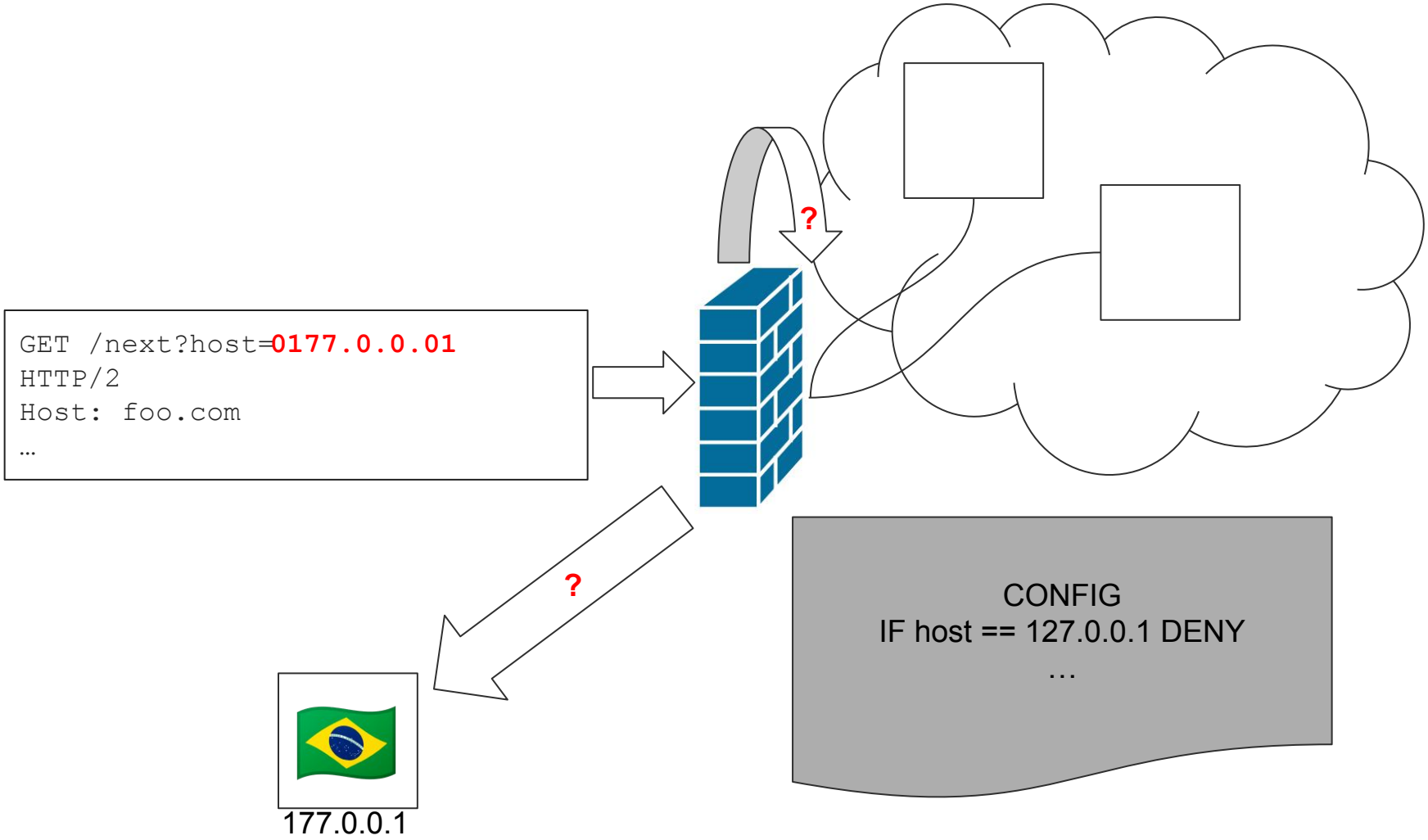
issues to exploit

- different parsers treating exact same literal differently
- the same parser treating semantically equivalent literals differently
- design decisions (library interface, fail-open)

```
GET /next?host=127.0.0.1 HTTP/2
Host: foo.com
…
```



CONFIG
IF host == 127.0.0.1 DENY
IF host IN 10.4.3.0/24, ALLOW
…


GET /next?host=0177.0.0.01
HTTP/2
Host: foo.com
…
?
?
CONFIG
IF host == 127.0.0.1 DENY
…
177.0.0.1

Sable Moth illustration from The Naturalist's Miscellany (1789-1813) by George Shaw (1751-1813), CC0.

127.0.0.1 ?= 0177.0.0.01

CWE-707: Improper Neutralization

Weakness ID: 707
Abstraction: Pillar
Structure: Simple

Presentation Filter: Complete

Description

The product does not ensure or incorrectly ensures that structured messages or data are well-formed and that certain security properties are met before being read from an upstream component or sent to a downstream component.

Extended Description

If a message is malformed, it may cause the message to be incorrectly interpreted.

Neutralization is an abstract term for any technique that ensures that input (and output) conforms with expectations and is "safe." This can be done by:

- checking that the input/output is already "safe" (e.g. validation)
- transformation of the input/output to be "safe" using techniques such as filtering, encoding/decoding, escaping/unescaping, quoting/unquoting, or canonicalization
- preventing the input/output from being directly provided by an attacker (e.g. "indirect selection" that maps externally-provided values to internally-controlled values)
- preventing the input/output from being processed at all

https://cwe.mitre.org/data/definitions/707.html

hypothesis:

(semantically but not literally equivalent) identifier* comparison is difficult

*https://www.rfc-editor.org/rfc/rfc6943

***law of least surprise*:**
surprise leads to sadness :’(

responsibly disclosed & patched* (thank you, triagers and patchers!)

README.md

Netmask

The Netmask class parses and understands IPv4 CIDR blocks so they can be explored and compared. This module is highly inspired by Perl Net::Netmask module.

Synopsis

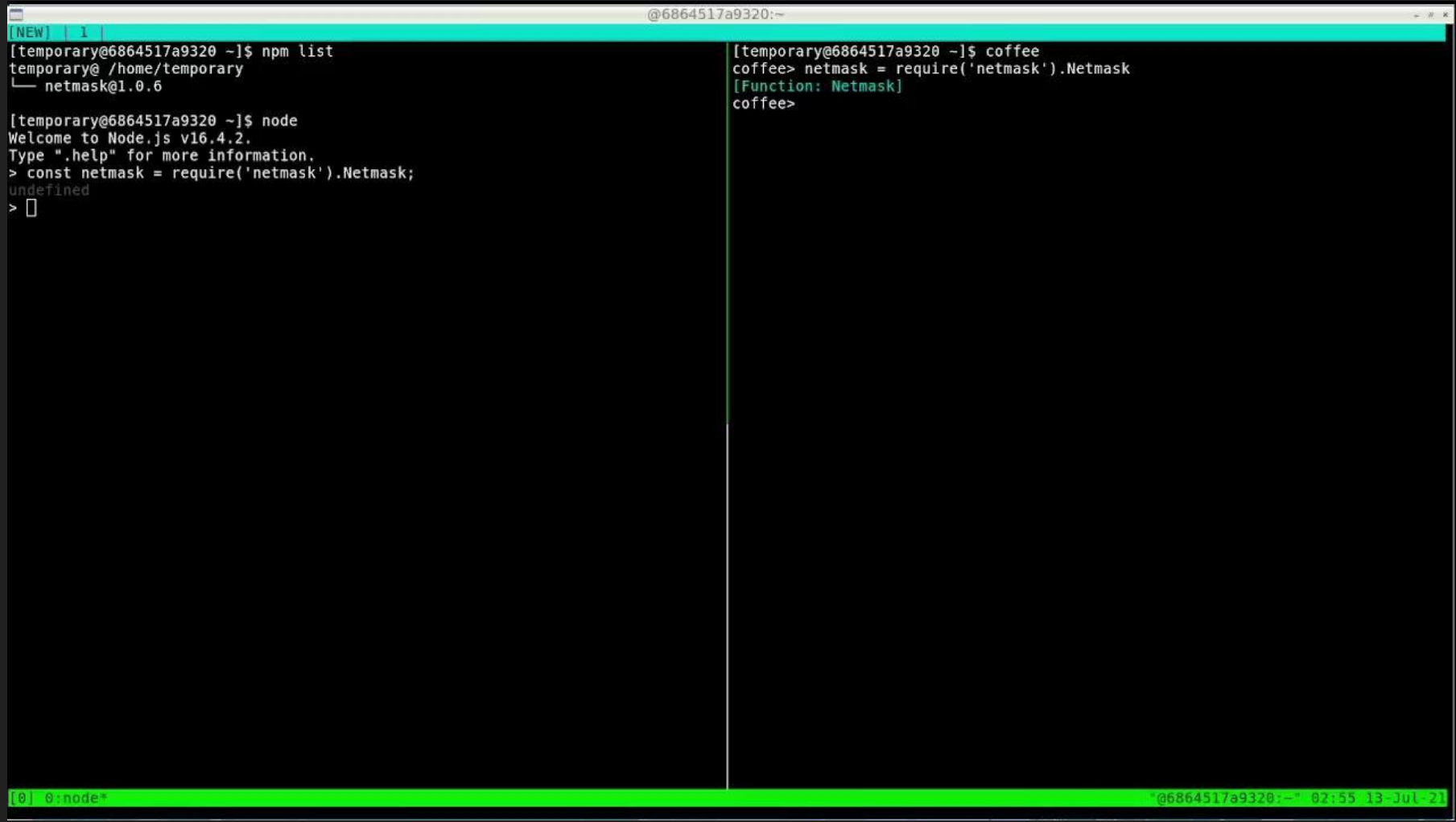
```
var Netmask = require('netmask').Netmask

var block = new Netmask('10.0.0.0/12');
block.base;                    // 10.0.0.0
block.mask;                    // 255.240.0.0
block.bitmask;                 // 12
block.hostmask;                // 0.15.255.255
block.broadcast;               // 10.15.255.255
block.size;                    // 1048576
block.first;                   // 10.0.0.1
block.last;                    // 10.15.255.254

block.contains('10.0.8.10');    // true
block.contains('10.8.0.10');    // true
block.contains('192.168.1.20'); // false
```

CVE-ID	
CVE-2021-28918	Learn more at National Vulnerability Database (NVD) • CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information

Description
Improper input validation of octal strings in netmask npm package v1.0.6 and below allows unauthenticated remote attackers to perform indeterminate SSRF, RFI, and LFI attacks on many of the dependent packages. A remote unauthenticated attacker can bypass packages relying on netmask to filter IPs and reach critical VPN or LAN hosts.


@6864517a9320:~
[NEW] | 1 |
[temporary@6864517a9320 ~]$ npm list
temporary@ /home/temporary
└── netmask@1.0.6
[temporary@6864517a9320 ~]$ node
Welcome to Node.js v16.4.2.
Type ".help" for more information.
> const netmask = require('netmask').Netmask;
undefined
>
[temporary@6864517a9320 ~]$ coffee
coffee> netmask = require('netmask').Netmask
[Function: Netmask]
coffee>
[0] 0:node*
"@6864517a9320:~" 02:55 13-Jul-21

CVE-2020-28360

- ***private-ip*** (JS, NPM package)
- incorrect categorization into private or public IP ranges
- ***private-ip*** incorrectly parses non-dotted-decimal input
- “is it private”? “is it public”?

CVE-2021-28918

- ***netmask*** (JS, NPM package)
- ***netmask*** incorrectly parses non-dotted-decimal input
- library interface gives range, so application logic using ***netmask*** must categorize input

private-ip

- 43,576 weekly NPM downloads
- 22 public NPM dependent packages
- 2,168 GitHub repositories downstream

netmask

- **5,687,274** weekly NPM DLs
- **211** public NPM dependents
- **365,501** GitHub downstream repos (~278k as of March 2021)

netmask

- **5,687,274** weekly NPM DLs
- **211** public NPM dependents
- **365,501** GitHub downstream repos (~278k as of March 2021)

SIGN IN

The Register

{* SECURITY *}

Sitting comfortably? Then it's probably time to patch, as critical flaw uncovered in npm's netmask package

Are you local? Catastrophically local?

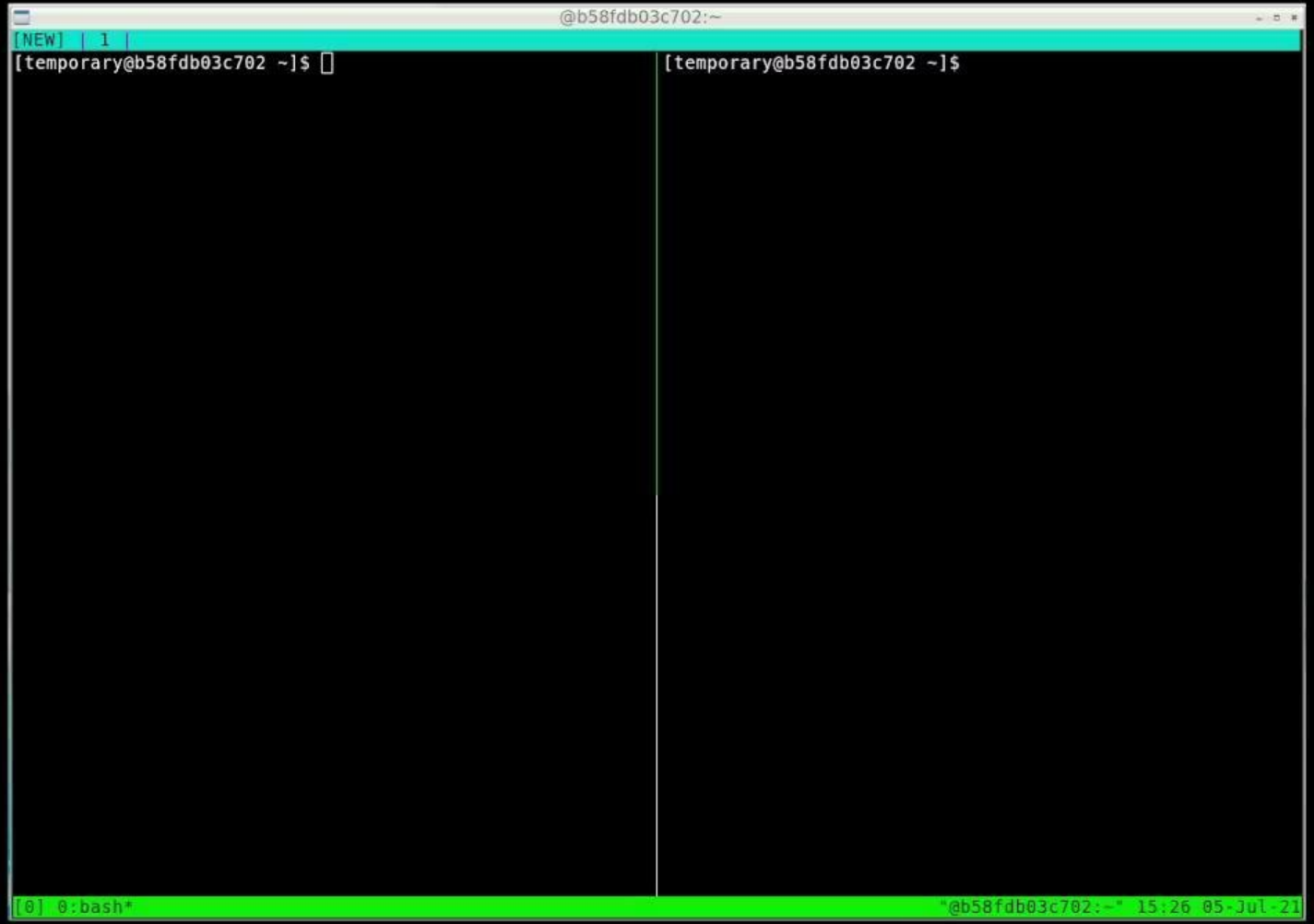
Richard Speed

Mon 29 Mar 2021 // 18:27 UTC

37

The widely used npm library netmask has a networking vulnerability arising from how it parses IP addresses with a leading zero, leaving an estimated 278,000 projects at risk.

Researchers Victor Viale, Sick Codes, Kelly Kaoudis, John Jackson, and Nick Sahler have disclosed a digital nasty, tracked as CVE-2021-28918, in the hugely widespread netmask npm package.


@b58fdb03c702:~
[NEW] | 1 |
[temporary@b58fdb03c702 ~]$
[temporary@b58fdb03c702 ~]$
[0] 0:bash*
"@b58fdb03c702:~" 15:26 05-Jul-21

so what?

javascript quirks

- `0`-prefix base-8 literals pre-ES5 (example **`0254.021.011.02`**)

- **`08`** and **`09`** do not result in parser error unless **`'use strict'`** (and then octal literals start with `'0o'`)

what makes this useful?

- access internal IPs (***SSRF***)
- access localhost / 127.0.0.1 (***SSRF, LFI***)
- reference exploit code from outside the network, by host IP (***SSRF, RFI***)

Description
Improper input validation of octal strings in netmask npm package v1.0.6 and below allows unauthenticated remote attackers to perform indeterminate SSRF, RFI, and LFI attacks on many of the dependent packages. A remote unauthenticated attacker can bypass packages relying on netmask to filter IPs and reach critical VPN or LAN hosts.

Daniel Stenberg
@bagder · Sep 27, 2018

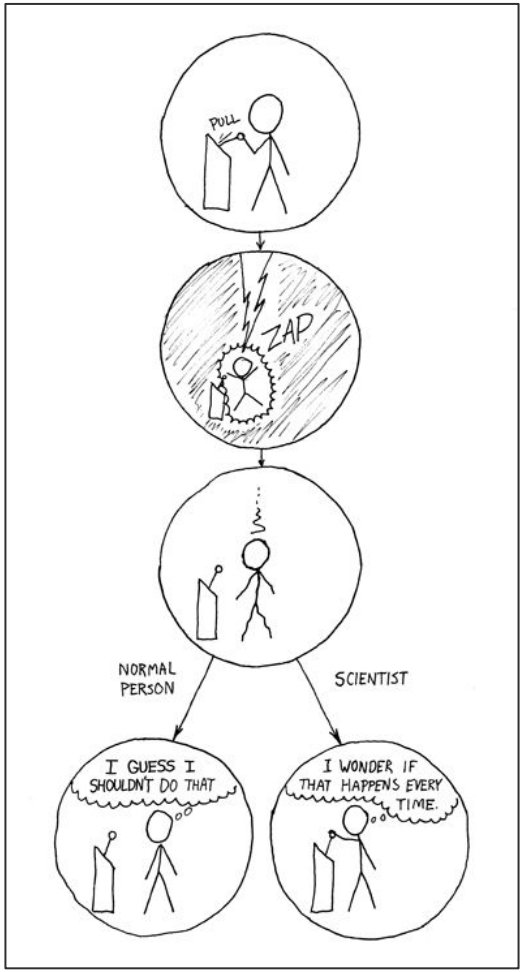
I learned that getaddrinfo() converts IPv4 addresses given as octal to decimal. See "ping 0177.0.0.1" or even "curl 0177.0.0.1" ... (the latter usually gets a 400 due to the funny Host:)

Nicolas Grégoire
@Agarri_FR

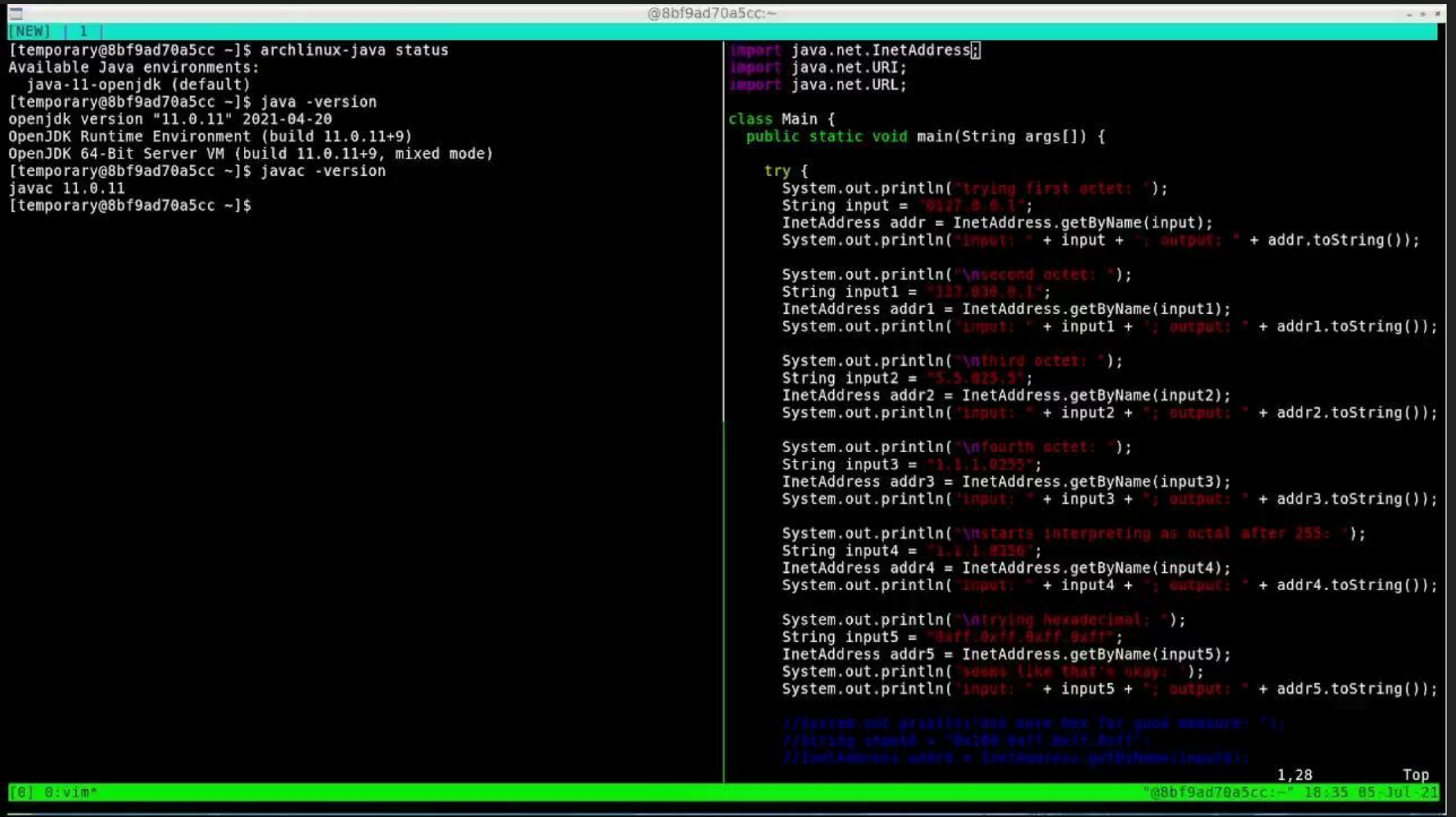
Replying to @bagder

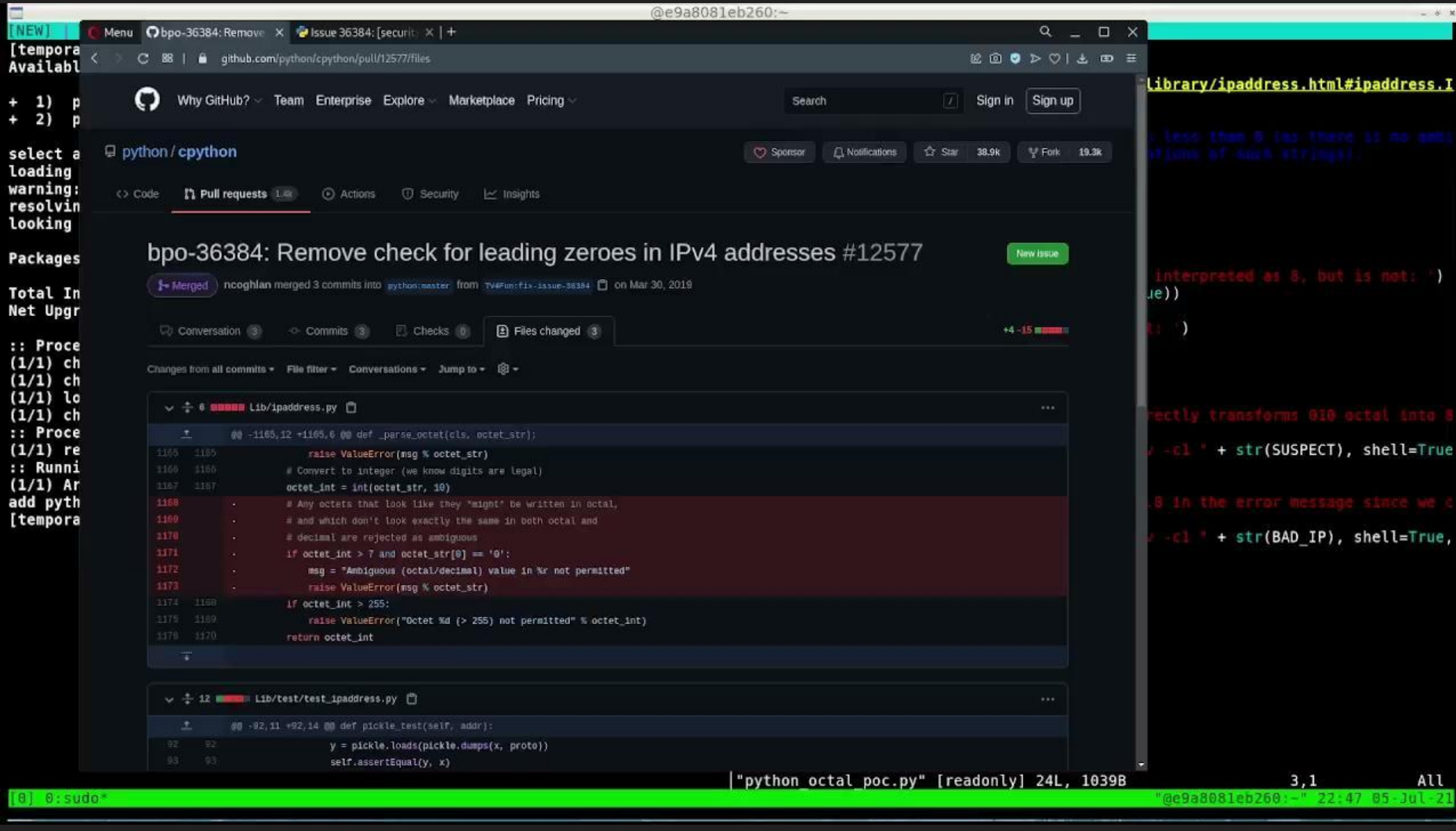
I often use these conversions in order to bypass anti-SSRF blacklists. Cf pages 24 to 28 for additional formats agarri.fr/docs/AppSecEU1...

10:51 AM · Sep 27, 2018



https://xkcd.com/242/


@8bf9ad70a5cc:~
[NEW] | 1 |
[temporary@8bf9ad70a5cc ~]$ archlinux-java status
Available Java environments:
  java-11-openjdk (default)
[temporary@8bf9ad70a5cc ~]$ java -version
openjdk version "11.0.11" 2021-04-20
OpenJDK Runtime Environment (build 11.0.11+9)
OpenJDK 64-Bit Server VM (build 11.0.11+9, mixed mode)
[temporary@8bf9ad70a5cc ~]$ javac -version
javac 11.0.11
[temporary@8bf9ad70a5cc ~]$
import java.net.InetAddress;
import java.net.URI;
import java.net.URL;
class Main {
public static void main(String args[]) {
try {
System.out.println("trying first octet: ");
String input = "0127.0.0.1";
InetAddress addr = InetAddress.getByName(input);
System.out.println("input: " + input + "; output: " + addr.toString());
System.out.println("\nsecond octet: ");
String input1 = "127.030.0.1";
InetAddress addr1 = InetAddress.getByName(input1);
System.out.println("input: " + input1 + "; output: " + addr1.toString());
System.out.println("\nthird octet: ");
String input2 = "5.5.025.5";
InetAddress addr2 = InetAddress.getByName(input2);
System.out.println("input: " + input2 + "; output: " + addr2.toString());
System.out.println("\nfourth octet: ");
String input3 = "1.1.1.0255";
InetAddress addr3 = InetAddress.getByName(input3);
System.out.println("input: " + input3 + "; output: " + addr3.toString());
System.out.println("\nstarts interpreting as octal after 255: ");
String input4 = "1.1.1.0256";
InetAddress addr4 = InetAddress.getByName(input4);
System.out.println("input: " + input4 + "; output: " + addr4.toString());
System.out.println("\ntrying hexadecimal: ");
String input5 = "0xff.0xff.0xff.0xff";
InetAddress addr5 = InetAddress.getByName(input5);
System.out.println("seems like that's okay: ");
System.out.println("input: " + input5 + "; output: " + addr5.toString());
//System.out.println("one more hex for good measure: ");
//String input6 = "0x100.0xff.0xff.0xff";
//InetAddress addr6 = InetAddress.getByName(input6);
1,28 Top
[0] 0:vim*
"@8bf9ad70a5cc:~" 18:35 05-Jul-21


@e9a8081eb260:~
Menu
bpo-36384: Remove
Issue 36384: [securit
github.com/python/cpython/pull/12577/files
Why GitHub?
Team
Enterprise
Explore
Marketplace
Pricing
Search
Sign in
Sign up
python / cpython
Sponsor
Notifications
Star 38.9k
Fork 19.3k
Code
Pull requests 1.4k
Actions
Security
Insights
bpo-36384: Remove check for leading zeroes in IPv4 addresses #12577
New issue
Merged
ncoghlan merged 3 commits into python:master from TV4Fun:fix-issue-36384 on Mar 30, 2019
Conversation 3
Commits 3
Checks 0
Files changed 3
+4 -15
Changes from all commits
File filter
Conversations
Jump to
6 Lib/ipaddress.py
@@ -1165,12 +1165,6 @@ def _parse_octet(cls, octet_str):
raise ValueError(msg % octet_str)
# Convert to integer (we know digits are legal)
octet_int = int(octet_str, 10)
# Any octets that look like they *might* be written in octal,
# and which don't look exactly the same in both octal and
# decimal are rejected as ambiguous
if octet_int > 7 and octet_str[0] == '0':
msg = "Ambiguous (octal/decimal) value in %r not permitted"
raise ValueError(msg % octet_str)
if octet_int > 255:
raise ValueError("Octet %d (> 255) not permitted" % octet_int)
return octet_int
12 Lib/test/test_ipaddress.py
@@ -92,11 +92,14 @@ def pickle_test(self, addr):
y = pickle.loads(pickle.dumps(x, proto))
self.assertEqual(y, x)
"python_octal_poc.py" [readonly] 24L, 1039B
3,1
All
[0] 0:sudo*
"@e9a8081eb260:~" 22:47 05-Jul-21

what did the NIST analysts think?

CVE-2020-28360	Javascript “private-ip”	***critical, 9.8***
CVE-2021-28918	Javascript “netmask”	***critical, 9.1***
CVE-2021-29418	Javascript “netmask” (found by ryotak)	***medium, 5.3***
CVE-2021-29662	Perl “Data::Validate::IP”	***high, 7.5***
CVE-2021-29424	Perl “Net::Netmask” (found by Joelle Maslak)	***high, 7.5***
CVE-2021-29921	Python stdlib “ipaddress”	***critical, 9.8***
CVE-2021-29923	Golang standard library “net”	***high, 7.5***
CVE-2021-29922	Rust standard library “net”	***critical, 9.1***
CVE-2021-33318	.NET C# “IpMatcher”	***critical, 9.8***

language	***prefix***
Javascript	`'0o'` when `'use strict'`, `'0'` otherwise
Perl	`'0o'` or `'o'` (optional)
Python	`'0o'`
Golang	`'0'`
Rust	`'0o'` prefix (also std::fmt::Octal)
Ruby	`'0o'` or `'0'` (optional)
Java	`'0'`
Scala (after 2.10), Kotlin	no octal literal prefix! yolo
C#	`'0'`
F#	`'0o'`
C, C++	`'0'`

archaeology

```
If the af argument of inet_pton() is AF_INET, the src string shall
be in the standard IPv4 dotted-decimal form:

      ddd.ddd.ddd.ddd

where "ddd" is a one to three digit decimal number between 0 and
255.  The inet_pton() function does not accept other formats (such
as the octal numbers, hexadecimal numbers, and fewer than four
numbers that inet_addr() accepts).
```

RFC 6943, Issues in Identifier Comparison for Security Purposes

source: https://www.mckusick.com/beastie/index.html

manpages!

STANDARDS

The inet_ntop() and inet_pton() functions conform to X/Open Networking Services Issue 5.2 (``XNS5.2''). Note that inet_pton() does not accept 1-, 2-, or 3-part dotted addresses; all four parts must be specified and are interpreted only as decimal values. This is a narrower input set than that accepted by inet_aton().

HISTORY

These functions appeared in 4.2BSD.

https://www.unix.com/man-page/FreeBSD/3/inet_pton/

HISTORY

The **inet_addr**(), **inet_network**(), **inet_makeaddr**(), **inet_lnaof**() and **inet_netof**() functions appeared in 4.2BSD. They were changed to use *in_addr_t* in place of *unsigned long* in NetBSD 2.0. The **inet_aton**() and **inet_ntoa**() functions appeared in 4.3BSD. The **inet_pton**() and **inet_ntop**() functions appeared in BIND 4.9.4 and thence NetBSD 1.3; they were also in X/Open Networking Services Issue 5.2 (``XNS5.2'').

https://man.netbsd.org/NetBSD-8.0/inet_pton.3

as builders, what can we do better?

- validate dependencies work as expected
- happy-path *and* negative end to end tests
- also, static analysis and fuzzing can help
- learn from failures and regressions
- follow existing convention (where possible)
- *bonus*: pay attention to vuln releases (props to Joelle Maslak)

References

- **Original talk (DEF CON 29)**
- CVE-2020-28360: Javascript “private-ip” NPM package
- CVE-2021-28918: Javascript “netmask” NPM package
- CVE-2021-29921: Python stdlib “ipaddress”
- CVE-2021-29922: Rust standard library “net”
- CVE-2021-29923: Golang standard library “net”
- CVE-2021-29662: Perl module “Data::Validate::IP”
- Oracle S1446698 (fix ongoing, also in DEF CON 29 talk): Java standard library “java.net.InetAddress”
- CVE-2021-33318: .NET “IpMatcher”, “Watson Web Server” NuGet packages

whoami

- senior software engineer
- application security TL at $company
- good-faith hacker
- kaoudis on most social media, hi@kellykaoud.is

credit also to

- Sick.Codes
- John Jackson (johnjhacking)
- Victor Viale (koroeskohr)
- tensor_bodega
- Harold Hunt (huntharo)
- Cheng Xu (xu-cheng)

Note: none of this research was paid for, all followed coordinated vendor disclosure

thank you!