huntr

Open Redirect in unshiftio/url-parse

0

✓ Valid   Reported on Jul 6th 2021

✍️ Description

url-parse mishandles certain uses of backslash such as https:/\ and interprets the URI as a relative path. Browsers accept backslashes after the protocol, and treat it as a normal slash, while url-parse sees it as a relative path.
Similar attacks: https://nvd.nist.gov/vuln/detail/CVE-2021-27515
https://hackerone.com/reports/384029

🕵️ Proof of Concept

Create the following PoC file:

```
// poc.js
var URI = require('url-parse')
var url = new URI("https:/\github.com/foo/bar")
console.log(url)
```

Execute the following commands in another terminal:

```
npm i url-parse # Install affected module
node poc.js #  Run the PoC
```

Check the Output:

```
URI {
  _string: '',
  _parts: {
    protocol: 'https',
    username: null,
    password: null,
    hostname: null,
    urn: null,
    port: null,
    path: '/github.com/foo/bar',
    query: null,
    fragment: null,
    preventInvalidHostname: false,
    duplicateQueryParameters: false,
    escapeQuerySpace: true
  },
  _deferred_build: true
}
```

💥 Impact

Depending on library usage and attacker intent, impacts may include allow/block list bypasses, SSRF attacks, open redirects, or other undesired behavior.

Occurrences

JS index.js L5

References

- Similar to CVE-2021-27515

CVE
CVE-2021-3664
(Published)

Vulnerability Type
CWE-601: Open Redirect

Severity
Medium (5.3)

Affected Version
*

Visibility
Public

Chat with us

Status
Fixed

Found by

ready-research
@ready-research
pro

Fixed by

Luigi Pinca
@lpinca
maintainer

This report was seen 911 times.

ready-research a year ago Researcher

Similar to CVE-2021-27515

ready-research a year ago Researcher

Another example:
var URI = require('url-parse')
var url = new URI("http:/\www.google.com")
console.log(url)

Returns pathname as : pathname: "/www.google.com"

ready-research a year ago Researcher

Using backslash in the protocol is valid in the browser, while url-parse thinks it's a relative path. An application that validates a url using url-parse might pass a malicious link.
https://github.com/unshiftio/url-parse/blob/master/SECURITY.md#history

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

ready-research a year ago Researcher

@maintainer There is another scenario using the latest git clone(seeing so many commits in master)

```
var URI = require('./url-parse/index')
var url = new URI("https://expected-example.com\@observed-example.com")
console.log(url)
```

Will return

```
{
  slashes: true,
  protocol: 'https:',
  hash: '',
  query: '',
  pathname: '/',
  auth: 'expected-example.com',
  host: 'observed-example.com',
  port: '',
  hostname: 'observed-example.com',
  password: '',
  username: 'expected-example.com',
  origin: 'https://observed-example.com',
  href: 'https://expected-example.com@observed-example.com/'
}
```

If url-parse is used to determine a URL's hostname, the hostname can be spoofed by using a backslash () character followed by an at (@) character. If the hostname is used in security decisions, the decision may be incorrect. Depending on library usage and attacker intent, impacts may include allow/block list bypasses, SSRF attacks, open redirects, or other undesired behavior.
Example URL: https://expected-example.com@observed-example.com
It incorrectly returns `observed-example.com` as the hostname instead of `expected-example.com` . I think it should be fixed.

ready-research a year ago Researcher

@admin can you please give access to https://github.com/lpinca one of the maintainers.
https://github.com/unshiftio/url-parse/issues/206#issuecomment-884969958

ready-research a year ago Researcher

@zidingz ^^

Jamie Slome a year ago Admin

@ready-research - I have reached out to Zi who will help further with this.

Z-Old a year ago Admin

Hey ready-research, lpinca should have access to this advisory page now if he's logged via his Github.

Luigi Pinca a year ago Maintainer

@zidingz can you please give access to 3rd-Eden?

Luigi Pinca a year ago Maintainer

It seems to me that

```
var parse = require('url-parse');
console.log(parse('https://expected-example.com\@observed-example.com'));
```

is working correctly and as expected.

```
{
  slashes: true,
  protocol: 'https:',
  hash: '',
  query: '',
  pathname: '/',
  auth: 'expected-example.com',
  host: 'observed-example.com',
  port: '',
  hostname: 'observed-example.com',
  password: '',
  username: 'expected-example.com',
  origin: 'https://observed-example.com',
  href: 'https://expected-example.com@observed-example.com/'
}
```

The same output is given by the WHATWG URL parser.

```
console.log(new URL("https://expected-example.com\@observed-example.com"));
```

```
URL {
  href: 'https://expected-example.com@observed-example.com/',
  origin: 'https://observed-example.com',
  protocol: 'https:',
  username: 'expected-example.com',
  password: '',
  host: 'observed-example.com',
  hostname: 'observed-example.com',
  port: '',
  pathname: '/',
  search: '',
  searchParams: URLSearchParams {},
  hash: ''
}
```

Luigi Pinca a year ago Maintainer

FWIW `'\@' === '@'` so it should eventually be `'\\@'` but it does not seem to change the result.

ready-research a year ago Researcher

https://nvd.nist.gov/vuln/detail/CVE-2020-26291 explains more about the issue.

Arnout Kazemier a year ago Maintainer

Can confirm that the original reported `https:/\` protocol attack is indeed working.

ready-research a year ago Researcher

We provided `expected-example.com` as the hostname here but it is returning `observed-example.com` as the hostname.

Generally, it should convert `'\@' as '/@'` . Which will return the accurate result.

Luigi Pinca a year ago Maintainer

The POC in the original description instead uses `'https:/github.com/foo/bar'` as input because `'https:/\github.com/foo/bar' === 'https:/github.com/foo/bar'` . If an actual backslash is used it works as expected and correctly:

```
var parse = require('url-parse');
console.log(parse('https:/\\github.com/foo/bar'));
```

```
{
  slashes: true,
  protocol: 'https:',
  hash: '',
  query: '',
  pathname: '/foo/bar',
  auth: '',
  host: 'github.com',
  port: '',
  hostname: 'github.com',
  password: '',
  username: '',
  origin: 'https://github.com',
  href: 'https://github.com/foo/bar'
}
```

This is a known bug that is being discussed/addressed in:

https://github.com/unshiftio/url-parse/issues/203

https://github.com/unshiftio/url-parse/pull/204

https://github.com/unshiftio/url-parse/issues/205

I'm not actually sure if it is also a security issue.

Luigi Pinca a year ago Maintainer

We provided expected-example.com as the hostname here but it is returning observed-example.com as the hostname.

Generally, it should convert'@' as '/@'. Which will return the accurate result.

It does it an actual backslash is used (`'\\@'`). `\@` is just `@` .

Luigi Pinca a year ago Maintainer

```
$ node
Welcome to Node.js v16.5.0.
Type ".help" for more information.
> '\@'.length
1
> '@'
'@'
> '\@' === '@'
true
```

ready-research a year ago Researcher

Using backslash issue

```
var parse = require('url-parse');
console.log(parse('https:\github.com/foo/bar'));  //pathname: 'github.com/foo/bar'
```

Using forward slash issue:

```
var parse = require('url-parse');
console.log(parse('https:/github.com/foo/bar'));  //pathname: 'github.com/foo/bar'
```

It should validate both the cases and return `pathname: '/foo/bar'`

ready-research a year ago Researcher

NODE is returning correctly.

```
> new URL("https:\github.com/foo/bar")
URL {
  href: 'https://github.com/foo/bar',
  origin: 'https://github.com',
  protocol: 'https:',
  username: '',
  password: '',
  host: 'github.com',
  hostname: 'github.com',
  port: '',
  pathname: '/foo/bar',
  search: '',
  searchParams: URLSearchParams {},
  hash: ''
}
```

```
> new URL("https:/github.com/foo/bar")
URL {
  href: 'https://github.com/foo/bar',
  origin: 'https://github.com',
  protocol: 'https:',
  username: '',
  password: '',
  host: 'github.com',
  hostname: 'github.com',
  port: '',
  pathname: '/foo/bar',
  search: '',
  searchParams: URLSearchParams {},
  hash: ''
}
>
```

ready-research a year ago Researcher

Based on above `url-parse` should also return
URL {
href: 'https://github.com/foo/bar',
origin: 'https://github.com',
protocol: 'https:',
username: '',
password: '',
host: 'github.com',
hostname: 'github.com',
port: '',
pathname: '/foo/bar',
search: '',
searchParams: URLSearchParams {},
hash: ''
}

In both the cases

ready-research a year ago Researcher

I will open a new issue for `\@`. It is confusing here if we discuss both the topics. Thanks & cheers.

Luigi Pinca a year ago Maintainer

The first snippet does not actually uses a backslash. The input in that case is `'https:github.com/foo/bar'` but yes that should also work as you say.

```
new URL('https:github.com/foo/bar')
URL {
  href: 'https://github.com/foo/bar',
  origin: 'https://github.com',
  protocol: 'https:',
  username: '',
  password: '',
  host: 'github.com',
  hostname: 'github.com',
  port: '',
  pathname: '/foo/bar',
  search: '',
  searchParams: URLSearchParams {},
  hash: ''
}
```

Basically all special schemes (https://url.spec.whatwg.org/#url-miscellaneous) should work like that. But again I'm not sure this is a security issue, for example:

```
new URL('sip:/github.com/foo/bar')
URL {
  href: 'sip:/github.com/foo/bar',
  origin: 'null',
  protocol: 'sip:',
  username: '',
  password: '',
  host: '',
  hostname: '',
  port: '',
  pathname: '/github.com/foo/bar',
  search: '',
  searchParams: URLSearchParams {},
  hash: ''
}
```

ready-research a year ago Researcher

Based on the further usage of the pathname in the application it depends. For the same issue, we have the above CVE's raised(with the same result, but the only diff is they used backslashes). Anyway, we can still reproduce the same with single backslash. So I think we can consider this a security issue. And should fix the issue.

ready-research a year ago Researcher

If you agree, can you mark this as a valid issue on the top.

Luigi Pinca a year ago Maintainer

I'm not sure I agree. Why is this a security issue only for some schemes/protocols? See my previous comment with the `https:` and `sip:` schemes using the Node.js WHATWG URL parser. I think this is more a follow every bit of the WHATWG URL specification issue.

Luigi Pinca a year ago Maintainer

Should we fix this? Yes we should follow the spec and make the behavior consistent with the Node.js URL parser (and the browser URL interface).

Is this a security issue? I'm not sure. If it is, why isn't it also a security issue for non special schemes (`sip:` , `ldap:` , etc.)?

Luigi Pinca a year ago Maintainer

Maybe the answer is that a browser can only make requests to URLs with special schemes?

ready-research a year ago Researcher

Node.js using slashedProtocol : https://github.com/nodejs/node/blob/master/lib/url.js#L99-L114 like `https,https,ftp......` (AND There is no browser to redirect to.)

I think we should at least follow the spec for these as this refer to a module that targets browsers. And using above the target destination can be controlled by the end-user, which will concern security.

ready-research a year ago Researcher

`Maybe the answer is that a browser can only make requests to URLs with special schemes?` --YES

Luigi Pinca a year ago Maintainer

FWIW https://github.com/nodejs/node/blob/master/lib/url.js is the legacy url which works exactly like url-parse :)

ready-research a year ago Researcher

So can we consider this as a valid issue? With respect to these special schemas(browser can only make requests to URLs with special schemas)

Luigi Pinca a year ago Maintainer

This is similar to https://advisory.checkmarx.net/advisory/CX-2021-4306 so yes, I think we should.

Anyway according to https://datatracker.ietf.org/doc/html/rfc3986#section-3 `https:github.com/foo/bar` , `https:/github.com/foo/bar` , `https:\github.com/foo/bar` are not valid URLs because they have an authority component and the authority component must be preceded by a double slash (`//`).

```
$ php -a
Interactive shell

php > var_dump(parse_url('https:/github.com/foo/bar'));
array(2) {
  ["scheme"]=>
  string(5) "https"
  ["path"]=>
  string(19) "/github.com/foo/bar"
}
php > var_dump(parse_url('https:\\github.com/foo/bar'));
array(2) {
  ["scheme"]=>
  string(5) "https"
  ["path"]=>
  string(19) "\github.com/foo/bar"
}
```

```
$ python
Python 3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit (AMD64)]
Type "help", "copyright", "credits" or "license" for more information.
>>> from urllib.parse import urlparse
>>> o = urlparse('https:/github.com/foo/bar')
>>> o
ParseResult(scheme='https', netloc='', path='/github.com/foo/bar', params='', query='
>>> o = urlparse('https:\\github.com/foo/bar')
>>> o
ParseResult(scheme='https', netloc='', path='\\github.com/foo/bar', params='', query=
>>>
```

It is the WHATWG URL standard that defines a special behavior when dealing with invalid special URLs. In particular

https://url.spec.whatwg.org/#scheme-state -> 2.7

https://url.spec.whatwg.org/#special-authority-slashes-state -> 2

https://url.spec.whatwg.org/#special-authority-ignore-slashes-state

...

Arnout Kazemier a year ago Maintainer

Anyway, we can still reproduce the same with single backslash. So I think we can consider this a security issue. And should fix the issue.

I have a working patch for this specific issue to bring it inline with how the WHATWG URL parse works in the browser when it comes to handling a single slash (ignoring it, adding a double slash in it's place).

```
new URL('https:/github.com/foo/bar')
URL {origin: "https://github.com", protocol: "https:", username: "", password: "", ho
hash: ""
host: "github.com"
hostname: "github.com"
href: "https://github.com/foo/bar"
origin: "https://github.com"
password: ""
pathname: "/foo/bar"
port: ""
protocol: "https:"
search: ""
searchParams: URLSearchParams {}
username: ""
__proto__: URL
```

And url-parse with my patch applied:

```
{
  slashes: true,
  protocol: 'https:',
  hash: '',
  query: '',
  pathname: '/foo/bar',
  auth: '',
  host: 'github.com',
  port: '',
  hostname: 'github.com',
  password: '',
  username: '',
  origin: 'https://github.com',
  href: 'https://github.com/foo/bar'
}
```

ready-research a year ago Researcher

@3rd-eden @lpinca Thank you for the validation of the issue.

Can you please hit the validate button and also confirm the fix using the action buttons on the advisory page?

Arnout Kazemier validated this vulnerability a year ago

ready-research has been awarded the disclosure bounty ✓

The fix bounty is now up for grabs

Arnout Kazemier a year ago Maintainer

I've confirmed the issue, will confirm the fix once the PR is landed.

ready-research a year ago Researcher

@3rd-eden Thank you for the confirmation.

Luigi Pinca marked this as fixed with commit 81ab96 a year ago

Luigi Pinca has been awarded the fix bounty ✓

This vulnerability will not receive a CVE ✗

ready-research a year ago Researcher

@lpinca Thanks for the quick fix. I am not able to reproduce the vulnerability and the above patch fixing this issue and working fine with both `/` and `\`.

Jamie Slome a year ago Admin

Nice work all! 🎉

We will have a CVE assigned and ready to publish today.

Arnout Kazemier a year ago Maintainer

Released 1.5.2 with fix and updated SECURITY.md with attribution.

Jamie Slome a year ago Admin

CVE published!

https://github.com/CVEProject/cvelist/pull/2353

Sign in to join this conversation