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

1

Valid Reported on Feb 14th 2022

Description

Incorrect conversion of @ in protocol in the href leads to improper validation of hostname.

Proof of Concept

Url-parse is not able to verify broken protocol. This will allow to bypass hostname validation.

```
parse = require('url-parse')

console.log(parse("http:@/127.0.0.1"))
```

Now imagine if there is blacklist check for domain 127.0.0.1

Result:

```
{
  slashes: true,
  protocol: 'http:',
  hash: '',
  query: '',
  pathname: '/127.0.0.1',
  auth: '',
  host: '',
  port: '',
  hostname: '',
  password: '',
  username: '',
  origin: 'null',
```

```
    href: 'http:///127.0.0.1'
  }
```

Here the hostname check equals null which will clearly bypass 127.0.0.1 blacklist check (interpreted as relative path instead). Now if you use href (http:///127.0.0.1) to fetch URL then it will fetch 127.0.0.1.

Impact

Bypass hostname check. It leads to an authorization bypass according to https://www.huntr.dev/bounties/6d1bc51f-1876-4f5b-a2c2-734e09e8e05b/. (A similar report)

Recommended Fix

Correct the href attribute.

Occurrences

JS index.js L17-L540

References

- A similar report

CVE
CVE-2022-0639
(Published)

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

Severity
Medium (6.5)

Visibility
Public

Status
Fixed

Found by
haxatron

@haxatron

pro

Fixed by

Luigi Pinca

@lpinca

maintainer

This report was seen 1,226 times.

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

haxatron modified the report 9 months ago

haxatron modified the report 9 months ago

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

haxatron modified the report 9 months ago

Luigi Pinca 9 months ago Maintainer

The protocol is correctly handled. For example

```
$ node
Welcome to Node.js v17.5.0.
Type ".help" for more information.
> var parse = require('.')
undefined
> parse('http:a/127.0.0.1')
{
  slashes: true,
  protocol: 'http:',
  hash: '',
  query: '',
  pathname: '/127.0.0.1',
  auth: '',
  host: 'a',
```

```
  host: 'a',
  port: '',
  hostname: 'a',
  password: '',
  username: '',
  origin: 'http://a',
  href: 'http://a/127.0.0.1'
}
```

is parsed as expected. The problem here is that the library returns an invalid hostname (`''`) because it is not validated, but this is documented. It is user responsibility to validate the hostname before doing any check against it.

In https://www.huntr.dev/bounties/6d1bc51f-1876-4f5b-a2c2-734e09e8e05b/ the issue was worse because the returned hostname was actually valid.

```
$ node
Welcome to Node.js v17.5.0.
Type ".help" for more information.
> var parse = require('.')
undefined
> parse('http://evil.com:\ba@example.com')
{
  slashes: true,
  protocol: 'http:',
  hash: '',
  query: '',
  pathname: '/',
  auth: 'evil.com:\ba',
  host: 'example.com',
  port: '',
  hostname: 'example.com',
  password: '\ba',
  username: 'evil.com',
  origin: 'http://example.com',
  href: 'http://evil.com:\ba@example.com/'
}
```

haxatron 9 months ago Researcher

Not if theres @ in the protocol, the href will be converted to a parseable URL

haxatron 9 months ago Researcher

I modified the report a few moments ago to reflect the incorrect conversion of the href. I believe https://www.huntr.dev/bounties/6d1bc51f-1876-4f5b-a2c2-734e09e8e05b/ also mentions that if the href is used to fetch the url then this will result in security consequence

Luigi Pinca 9 months ago Maintainer

See my previous comment. If the returned hostname is invalid the user should consider the URL invalid.

haxatron 9 months ago Researcher

Additionally a null hostname is still valid as it would be interpreted as a relative path.

haxatron 9 months ago Researcher

Developers can use the fact that it is a relative path (no hostname) as a form of whitelist. The user can thus bypass this by doing the above.

Luigi Pinca 9 months ago Maintainer

A relative URL does not have a protocol.

haxatron 9 months ago Researcher

It is possible that developers validate a hostname using a blacklist check of characters and when passing in an empty hostname, the blacklist will be bypassed.

I note that in https://www.huntr.dev/bounties/1625557993985-unshiftio/url-parse/, the PoC also demonstrated the same thing, https protocol but was treated as a relative path, and also was fixed.

haxatron 9 months ago Researcher

https://advisory.checkmarx.net/advisory/CX-2021-4306 too

Luigi Pinca 9 months ago Maintainer

If the protocol is special and there is no hostname, then the URL is invalid. The result is the same with the legacy Node.js URL parser:

```
$ node
Welcome to Node.js v17.5.0.
Type ".help" for more information.
> url.parse('http://@/127.0.0.1')
Url {
  protocol: 'http:',
  slashes: true,
  auth: '',
  host: '',
  port: null,
  hostname: '',
  hash: null,
  search: null,
  query: null,
  pathname: '/127.0.0.1',
  path: '/127.0.0.1',
  href: 'http:///127.0.0.1'
}
```

The linked issues are similar but not the same. I think there is no fix in this case. The Node.js WHATWG url parser throws an error. `url-parse` does not throw errors by design.

The only thing I can think of is using replacing the empty hostname with something else but I'm not sure it is any better.

haxatron 9 months ago Researcher

The fix would to detect if are any @ characters after the : in protocol pre-parsing, and then remove it before feeding it to the parser. As it stands, there are no other characters which give this special behaviour

Luigi Pinca 9 months ago Maintainer

That would break valid cases where `@` is used as the first character of the use
example, `http:@@example.com` which is valid.

haxatron 9 months ago Researcher

detect the first instance of : and match any @ that come after.

haxatron 9 months ago Researcher

ensure that the / comes after the @ before removing

Luigi Pinca 9 months ago Maintainer

Consider something like this `http://:@/@/:@/@/127.0.0.1`. We could do

```
url.replace(/^(:?@\/)+/, '/')
```

after extracting the protocol but that is not correct. It would result in a URL with `127.0.0.1` as hostname which is not not correct.

haxatron 9 months ago Researcher

This works:

```
clean = function (url) { return url.replace(/(?<=:)(@*?)(?=\/)/,"") }

// clean
console.log(clean("http://:@/@/:@/@/127.0.0.1"))

// unclean
console.log(clean("http:@//127.0.0.1"))
console.log(clean("http:@@@@@@//127.0.0.1"))
```

Result:

```
http://:@/@/:@/@/127.0.0.1
http://127.0.0.1
http://127.0.0.1
```

```
http://127.0.0.1
```

Let me test above even further

haxatron 9 months ago Researcher

```
clean = function (url) { return url.replace(/(?<=:)(@*?)(?=(\/|\\))/,"") }

// clean
console.log(clean("http://:@//@/:@/@/127.0.0.1"))

// unclean
console.log(clean("http:@//127.0.0.1"))
console.log(clean("http:@\\127.0.0.1"))
console.log(clean("http:@@@@@@//127.0.0.1"))
```

Luigi Pinca 9 months ago Maintainer

Fails with `http://:@/127.0.0.1` and again, doing this changes the URL in a way that is not correct. In a URL like this the host is `''`, not `127.0.0.1`.

haxatron 9 months ago Researcher

It does not fail with http://:@/127.0.0.1...

```
clean = function (url) { return url.replace(/(?<=:)(@*?)(?=(\/|\\))/,"") }

console.log(clean("http://:@/127.0.0.1"))
```

Result:

```
http://:@/127.0.0.1
```

Luigi Pinca 9 months ago Maintainer

Parse that.

haxatron 9 months ago Researcher

ok, yes I did not notice that that payload works too...

In the interest of time, are you willing to mark this as valid if I am able to propose a fix to this?

Luigi Pinca 9 months ago Maintainer

I'm not sure this is valid. I understand the risk but again, it is documented that `url-parse` might return invalid hostnames.

In this case the hostname is correctly parsed as `''` and a special protocol is also correctly parsed (`http:`) so the user should proceed no further because the URL is invalid. A URL with a special protocol cannot have an empty hostname.

haxatron 9 months ago Researcher

It's no differrent than https://advisory.checkmarx.net/advisory/CX-2021-4306 or https://www.huntr.dev/bounties/1625557993985-unshiftio/url-parse/. I'd say this carries additional risk because the href is completely useable.

Luigi Pinca 9 months ago Maintainer

It is different. In those two cases, there is a parsing error. In this case, there is no parsing error.

haxatron 9 months ago Researcher

A security issue is valid when some feature / behaviour, intended or unintended, carries a risk to the user when using a specified component. Whether it is a parsing error or not, shouldn't dictate whether a security issue is valid.

You can go ahead and mark this invalid, if you still believe it is not something to fix

haxatron 9 months ago Researcher

FWIW parse_url in PHP

```
<?php

var_dump(parse_url("http://:@/"));
```

will return false

Luigi Pinca 9 months ago Maintainer

I understand. The problem is that I think there is no proper fix for this. An error should be thrown. I think it is time to deprecate `url-parse` and tell users to use the browser `URL` interface now that it is widely available.

Luigi Pinca 9 months ago Maintainer

Let's keep this open for a while. I think I have a potential fix.

haxatron 9 months ago Researcher

After re-examining this issue as well as the internal workings of the library, I better understand what you mean by it being hard to fix (in reference to the http://@/google.com case, I was toofocused on http:@/google.com), and it being a parsing error.

Luigi Pinca 9 months ago Maintainer

Yes, and the fix I was thinking about is not good enough. I can only think about two solutions:

Remove empty auth after extracting the protocol. `url.replace(/^(:?@\/)+/, '/')` .

Replace the empty hostname with something else (still invalid like `^`).

Both solutions are not satisfactory as they change the meaning of the original URL.

The parser behavior is correct and it is the same when parsing `http://` or `htt`
difference is that in this case there is also a `pathname` .

As said above the documentation already mentions that the returned hostname might be invalid. Maybe it should be extra explicit about the fact that it is not sufficient to rely on the `host` / `hostname` properties.

haxatron 9 months ago Researcher

If you were to ask me, after looking at the library internals, the simplest code-level fix (to automatically protect users). The simplest fix I can think of is check if the auth+@ component equals to full address string, if it is then that means that the hostname+port is empty, so equate hostname to the auth+@, then discard the auth, so for instance:

if url,auth+@===address, then do: url.hostname = address, url.host = address, url.auth = ''

The hostname is equated to url.address this time.

When the href is built this time it is now populated because the hostname is non-empty.
This mitigates the scenario as now the href is equal to the input address string and would be considered invalid

But ultimately it is up to you where you want to go with this.

haxatron 9 months ago Researcher

check should be performed in between https://github.com/unshiftio/url-parse/blob/master/index.js#L315L316, before L316, just before the address is sliced.

Luigi Pinca 9 months ago Maintainer

If I understand correctly that is similar to what I had in mind before but it is not correct. `http://@` would be parsed with `@` as `host` / `hostname`, or worse `http://foo:bar@` with `foo:bar@` as `host` / `hostname`. This is a no go.

Give me some time and let me think a bit more about this.

haxatron 9 months ago Researcher

http://@ and http://foo:bar@ are invalid URLs so it won't matter, when @ and foo:bar@ are parsed as a hostname, the href would include both the @ and foo:bar@ instead of discarding the @ when the auth is empty, this would correct the href to a non-valid href, r
one (http:///127.0.0.1).

Alternatively, url-parse can match the hostname against a valid hostname regex and error out

return boolean false if it encounters one (it may break some users as they may not be handling the error in url-parse or may be relying on a property of the url object, if that occurs then maybe

returning a url object and then zeroing out all the properties (converting all of them to '') present would work?)

If not then I think improving the documentation would be the way to go.

Luigi Pinca 9 months ago Maintainer

See https://github.com/unshiftio/url-parse/pull/226. The Go language URL parser has the same behavior https://go.dev/play/p/QneLlIgcaRK.

There is a side effect if the user does something like this

```
$ node
Welcome to Node.js v17.5.0.
Type ".help" for more information.
> var Url = require('.')
undefined
> var url = new Url('http://')
undefined
> url.toString()
'http:///'
> url.pathname = '/pathname'
'/pathname'
> url.toString()
'http://@/pathname'
>
```

but I think this is not an issue.

haxatron 9 months ago Researcher

Can confirm the fix in https://github.com/unshiftio/url-parse/pull/226 works.

Luigi Pinca validated this vulnerability 9 months ago

haxatron has been awarded the disclosure bounty ✔

The fix bounty is now up for grabs

Luigi Pinca 9 months ago Maintainer

It might take a long time before I can confirm the fix because I cannot publish a new version to npm. I have to ask other maintainers to do that.

Luigi Pinca marked this as fixed in 1.5.7 with commit ef45a1 9 months ago

Luigi Pinca has been awarded the fix bounty ✔

This vulnerability will not receive a CVE ✖

index.js#L17-L540 has been validated ✔

Luigi Pinca 9 months ago Maintainer

For posterity's sake, the second part of my first comment is not correct.

`http://evil.com:\ba@example.com/` works as expected when parsed with a parser that follows the WHATWG URL Standard, even if the backspace (`\b`) is not encoded to `%08` .

```
$ node
Welcome to Node.js v17.6.0.
Type ".help" for more information.
> new URL('http://evil.com:\ba@example.com/')
URL {
  href: 'http://evil.com:%08a@example.com/',
  origin: 'http://example.com',
  protocol: 'http:',
  username: 'evil.com',
  password: '%08a',
  host: 'example.com',
  hostname: 'example.com',
  port: '',
  pathname: '/',
  search: '',
  searchParams: URLSearchParams {},
  hash: ''
}
```


See also https://huntr.dev/bounties/800caf4a-4be3-49e4-b764-506ef369fcb1.

Sign in to join this conversation

Chat with us