SQL injection attempts killls Etherpad lite #3502

Closed fpoulain opened this issue on Oct 23, 2018 · 13 comments · Fixed by #3797

Labels security Serious Bug

Milestone 1.8.3

fpoulain commented on Oct 23, 2018

Hi,

On our server we were getting some Etherpad outage. We relied it to a nasty query:

```
https://pad.bling.org/javascripts/lib/ep_etherpad-lite/static/js/pad.js?
callback=require.define&vLtF%3D6904%20AND%201%3D1%20UNION%20ALL%20SELECT%201%2CNULL%2C%27%3Cscript%3Ealert%28%22XSS%22%29%3C%2Fscript%3E%27%2Ctable_name%20FROM%20information_schema.ta
-%2F%2A%2A%2F%3B%20EXEC%20xp_cmdshell%28%27cat%20..%2F..%2F..%2Fetc%2Fpasswd%27%29%23
```

A "minimal" query example:

```
https://pad.bling.org/javascripts/lib/ep_etherpad-lite/static/js/pad.js?
callback=require.define&vLtF%3D6904%20AND%201%3D1%20UNION%20ALL%20SELECT%201%2CNULL%2C%27%3Cscript%3Ealert(%22XSS%22)%3C%2Fscript%3E%27
```

This provoke an immediate crash:

```
oct. 23 18:17:19 pad.bling.org run.sh[8976]: [2018-10-23 18:17:19.994] [ERROR] console - Error: ENAMETOOLONG: name too long, open '/var/www/etherpad-
lite/var/minified_L2phdmFzY3JpcHRzL2xpYi9lcF9ldGhlcn
oct. 23 18:17:19 pad.bling.org run.sh[8976]:     at Error (native)
oct. 23 18:17:19 pad.bling.org run.sh[8976]: [2018-10-23 18:17:19.995] [INFO] console - graceful shutdown...
oct. 23 18:17:20 pad.bling.org run.sh[8976]: [2018-10-23 18:17:20.091] [INFO] console - db sucessfully closed.
```

We are running the 1.7.0 flavor on Debian Stretch with node v6.14.4 and no specific customization.
We reproduced the behavior on two independents Etherpad installation.

muxator commented on Oct 23, 2018 Contributor

That's true, indeed.
Thanks for the precious info, **@fpoulain**, much appreciated!

muxator added the security label on Oct 23, 2018

muxator added this to the **1.8** milestone on Oct 23, 2018

JohnMcLear commented on Jan 19, 2019 Member

Hey guys, just a reminder about responsible disclosure. Posting publicly without giving us chance to pick can be quite dangerous.

fpoulain commented on Jan 21, 2019 Author

> Hey guys, just a reminder about responsible disclosure. Posting publicly without giving us chance to pick can be quite dangerous.

Hum ... actually it has been already disclosed because it is used by some nasty guys.

Also, I don't see how to report it in a non public way. The project description don't mention anyway private feedback loop for security issues. How do you think would I had reported it?

1

JohnMcLear commented on Jan 21, 2019 Member

Afaik we have a responsible disclosure policy. I thought it was on etherpad.org and the github readme.
...

fpoulain commented on Jan 21, 2019 • edited Author

It could be a good idea to add some invitation "found a security issue? tell us about it via ...". Before opening this issue I spent few minutes on github's readme and on etherpad.org without finding such an invitation. Also, as a non native english reader, I could have missed the good terms to seek for.

JohnMcLear commented on Jan 21, 2019 Member

Noted. Tnx
...

marksteward commented on Jan 21, 2019

I couldn't find one either, but I did find you requesting a PR to add one #2499.

Perhaps consider just creating a https://securitytxt.org/ as a quick fix?

muxator modified the milestones: **1.7.5**, **1.8** on Feb 5, 2019

muxator modified the milestones: **1.8.0**, **1.8.1** on Dec 7, 2019

JohnMcLear added the Serious Bug label on Mar 29, 2020

JohnMcLear commented on Mar 29, 2020 Member

Offending code is somewhere in here: https://github.com/ether/yajsml/blob/master/server.js#L98

JohnMcLear commented on Mar 29, 2020 Member

Disclaimer: I'm not clever enough to resolve this. I need someone with expertise to help!

tomnomnom commented on Mar 30, 2020 • edited Contributor

Hey! Doing some looking into this now.

First off: to fend off any fears about SQLi: this has nothing to do with SQL injection, as evidenced by this alternative payload that also causes the same crash:

```
/javascripts/lib/ep_etherpad-lite/static/js/pad.js?
callback=require.define&footlefootlefootlefootlefootlefootlefootlefootlefootlefootlefootlefootlefootlefootlefootlefootlefootlefootlefootlefootle
```

This is a filename length issue in the caching middleware.

Currently the cache key is generated as follows:

```
var cacheKey = Buffer.from(path).toString('base64').replace(/[/+=]/g, '');
```

This causes the cache key length to be controlled by the length of the path.

This is a problem when the cache key is used as a filename on disk, as many filesystems limit filename length to 255 characters. This is why there's an `ENAMETOOLONG` error in the log.

To remedy this, I suggest making the cache key a hash of the path instead of the base64 encoded version. This will ensure a fixed-length filename.

I'll submit a PR that does this shortly.

tomnomnom added a commit to tomnomnom/etherpad-lite that referenced this issue on Mar 30, 2020

`Switches cacheKey from base64 of path to sha256 of path; fixes ether#…` ... `751237e`

tomnomnom mentioned this issue on Mar 30, 2020

Switches cacheKey from base64 of path to sha256 of path; fixes #3502 #3794

Merged

muxator commented on Mar 30, 2020 Contributor

Hi, **@tomnomnom**, this is clever. It completely makes sense that cache keys are of fixed length. In this way they stay dependent on the content, with practically non existent risk of collisions.

I'll have a look tonight.
Well done!

1

muxator closed this as completed in `fc754c9` on Mar 30, 2020

muxator commented on Mar 30, 2020 Contributor

Merged the PR, many thanks!

A further note about this: from the nodejs docs **we may have to consider the possibility of the `crypto` module being unavailable**:

> It is possible for Node.js to be built without including support for the crypto module. In such cases, calling require('crypto') will result in an error being thrown

Maybe embedded platforms may not have it (e.g.: ARM, MIPS, Raspberry PIs & similar)? I do not know. A description of such scenarios can be found here:

> - running non-standard node in a resource- or security-constrained environment (see node#5611 for their explicit support of this scenario)
> - running in emulated environment (browserify, webpack etc.)
> - building node from source and omitting openssl/crypto for random reason (see StackOverflow question or another nodejs post)

Anyway, the TypeScript guys dealt with this same issue in microsoft/TypeScript#19100, and they resolved it in an elegant way in microsoft/TypeScript@ `9677b06` .

If the importing crypto fails at runtime, they replace the hash algorithm the djb2 algorithm, which is way weaker, but works for their case.

An example adapted for our case may be:

```
function djb2Hash(data) {
        const chars = data.split("").map(str => str.charCodeAt(0));
        return `${chars.reduce((prev, curr) => ((prev << 5) + prev) + curr, 5381)}`;
}

console.log(Buffer.from(djb2Hash('This is a 😎 test of the djb2 hash function')).toString('hex'));
// prints 36373536373437333033
```

I am not asking you to do this, but the djb2 story is fun: see here, and the original mailing list post by Daniel Bernstein from 1991. **He was 20 at the time**.

muxator mentioned this issue on Mar 30, 2020

caching_midleware: also run when nodejs does not have crypto module #3797

Merged

muxator commented on Mar 30, 2020 Contributor

Follow up which uses `djb2` when there is no crypto support: #3797.

Assignees

No one assigned

Labels

security Serious Bug

Projects

None yet

Milestone

1.8.3

Development

Successfully merging a pull request may close this issue.

caching_midleware: also run when nodejs does not have crypto module
muxator/etherpad-lite

5 participants