Closed Issue created 2 years ago by **Maxime Besson**

[security:high, CVE-2020-24660] Lack of URL normalization by Nginx may lead to authorization bypass when URL access rules are used

Environment

LemonLDAP::NG version: 2.0.8

Operating system: Debian Stretch, Debian Buster, probably RHEL

Web server: nginx/1.10.3, nginx/1.14.2

Summary

When using Nginx, regexp-based access rules may not be correctly enforced by the handler.

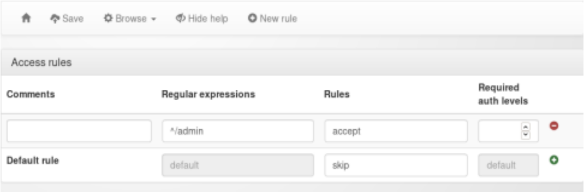
I am doing a CVE request for this bug

Logs

- Content of test vhost:

```
# cat /var/lib/lemonldap-ng/test/admin
SECRET ADMIN FILE
```

- Handler configuration:



Access rules

Comments	Regular expressions	Rules	Required auth levels
	^/admin	accept	
Default rule	default	skip	default

- Proof of exploitation:

```
GET -S  http://test1.example.com/admin/secretfile
GET http://test1.example.com/admin/secretfile
302 Moved Temporarily //AS EXPECTED

$ GET -S  http://test1.example.com/%61dmin/secretfile
GET http://test1.example.com/%61dmin/secretfile
200 OK
SECRET ADMIN FILE  //SHOULD BE PROTECTED

GET -S  http://test1.example.com/x/../admin/secretfile
GET http://test1.example.com/x/../admin/secretfile
200 OK
SECRET ADMIN FILE //SHOULD BE PROTECTED
```

I have also successfully tested this in a reverse proxy configuration, which is a very common, if not the most common use case. I have also tested this without the "skip" keyword, in such a cas, a normal user may be granted access to admin-only resources.

Cause

The problem comes from the fact that the handler tests regexp against the REQUEST_URI variable. Unlike Apache, Nginx does not normalize REQUEST_URI. Because of this, it becomes extremely hard for an admin to write a regexp that correctly catches all of the possible URLs that can be used to target a protected resource (such as /admin).

Solutions

URI::Normalize

Nginx transmits the original URL in a X_ORIGINAL_URL header. We could use this fact to trigger special processing in the handler:

```
    $self->env->{REQUEST_URI} = $self->env->{X_ORIGINAL_URI}
      if ( $self->env->{X_ORIGINAL_URI} );
```

would change to

```
    $self->env->{REQUEST_URI} = normalize_url($self->env->{X_ORIGINAL_URI})
      if ( $self->env->{X_ORIGINAL_URI} );
```

Using `normalize_url` from URI::Normalize which is not in distros but easily embeddable.

Nginx config

We could also make Nginx normalize the URL, with something like this:

```
location / {
    ...
    # Save the normalized URI here
    set $original_uri $uri$is_args$args;
    ...
}

location = /lmauth {
    ...
    fastcgi_param X_ORIGINAL_URI  $original_uri;
    ...
}
```

But that means each webserver we ever want to support will probably have it's own, distinct solution

Edited 2 years ago by Maxime Besson

Drag your designs here or click to upload.

Tasks 0

No tasks are currently assigned. Use tasks to break down this issue into smaller parts.

Linked items 0

Link issues together to show that they're related. Learn more.

Activity

Maxime Besson changed milestone to %2.0.9 2 years ago

Maxime Besson added Bug Handler Security labels 2 years ago

Maxime Besson changed title from **Lack of URL normalization by Nginx may lead to authorization bypass when URL access rules are used** to **[security:high] Lack of URL normalization by Nginx may lead to authorization bypass when URL access rules are used** 2 years ago

Maxime Besson changed the description 2 years ago

Maxime Besson @maxbes · 2 years ago Author Maintainer

1.9 is affected by path tricks (`x/../admin`) but not by URL encoding (`%61dmin`) because it urldecodes the URL before processing (not the righ way to do things, but it can help in most cases)

Edited by Maxime Besson 2 years ago

Clément OUDOT @clement_oudot · 2 years ago Owner

Could you also try this request:

```
GET -S  http://test1.example.com//admin/secretfile
```

Maxime Besson @maxbes · 2 years ago Author Maintainer

We already handle this in ::Common::PSGI::Request:

```
    $self->env->{REQUEST_URI} =~ s|^//+|/|g;
```

Clément OUDOT @clement_oudot · 2 years ago Owner

Ok, so this could work: `GET -S http://test1.example.com/mypath//admin/secretfile`

Even if we protect `^/mypath/admin`

Maxime Besson @maxbes · 2 years ago Author Maintainer

Indeed. We are currently vulnerable to this.

- Apache normalization currently protects us.
- Nginx normalization will protect us if we configure it
- URI::Normalize will not protect us as-is. We would need to improve it.

Edited by Maxime Besson 2 years ago

Clément OUDOT @clement_oudot · 2 years ago Owner

So for %2.0.9 we should maybe just update Nginx configuration files, and add a point in upgrade notes?

Please register or sign in to reply

Clément OUDOT @clement_oudot · 2 years ago Owner

And how should we prevent the "path trick"?

Maxime Besson @maxbes · 2 years ago Author Maintainer

Both URI::Normalize and my Nginx suggestion prevent tricks based on ./ and ../

Please register or sign in to reply

Maxime Besson changed title from **[security:high] Lack of URL normalization by Nginx may lead to authorization bypass when URL access rules are used** to **[security:high, CVE-2020-24660] Lack of URL normalization by Nginx may lead to authorization bypass when URL access rules are used** 2 years ago

Maxime Besson @maxbes · 2 years ago Author Maintainer

CVE-2020-24660

Yadd @guimard · 2 years ago Owner

Hi,

could we provide a fix for 1.9 in the same time ?

Maxime Besson @maxbes · 2 years ago Author Maintainer

If we go for the nginx config solution, it should be pretty much the same fix for 1.9 and 2.0

Please register or sign in to reply

Maxime Besson changed the description 2 years ago

Maxime Besson @maxbes · 2 years ago Author Maintainer

@maudoux have you checked if uwsgi is affected too? Logically it should be. Could you try the nginx patch in the original post (I simplified it, there are now just two lines to change). Not many of our customers use both Nginx and the Handler in production so your input would be appreciated before releasing this.

Christophe Maudoux @maudoux · 2 years ago Maintainer

Hi,

Will be tested and give you some feedback asap...

Edited by Christophe Maudoux 2 years ago

Christophe Maudoux @maudoux · 2 years ago Maintainer

Hi,

I confirm that this issue exists with Nginx/uwsgi. Seems fixed with your Nginx patch. Will be tested on training platform this afternoon and confirm you if OK.

Cheers

Yadd @guimard · 2 years ago Owner

Hi @maxbes, a so small package *(see URI::Normalize source)* could be embedded instead of waiting for dist packages.

I tried another way more generic:

```
use URI;
sub normalizeUrl {
  my($self,$url) = @_;
  return URI->new($url)->canonical->as_string;
}
```

but it fails with double `/`

I suggest to both patch Nginx config and copy URI::Normalize into `LLNG::Common::Somewhere` to be able to update it.

Maxime Besson @maxbes · 2 years ago Author Maintainer

URI->canonical also does not normalize path components: `/./` and `/../` so URI::Normalize is required for a proper LLNG-level fix. We also need to make it normalize multiple slashes because a lot of web applications will merge them

URI::Normalize normalizes also the query string, which Apache and Nginx (with my fix) will not do, so it might cause some slight changes in behavior. Hopefully noone has written manager rules that catch non-normalized query strings explicitely...

Yadd @guimard · 2 years ago Owner

I didn't test URI::Normalize but be careful while normalizing query string: `http://index?arg=http://xx.com` is allowed *(multiple `/`)*

Yadd @guimard · 2 years ago Owner

And we already said that it's not secure to use query arg in a rule

Maxime Besson @maxbes · 2 years ago Author Maintainer

Oh, good point about the query string normalization and URLs

Regarding using query string in a regexp rule: I agree that it's not secure, but I disagree with the solution in the doc.

GET http://test1.example.com/index.php?access=%61dmin currently bypasses the example in doc, even when using Apache

query string normalization is a much harder issue, (sorting, deduplicating) and it interferes with url-decoding (param=http:// and param=http:%2F%2F are different at the URI level but not at the application level). Perhaps we should outright discourage it in docs.

Edited by Maxime Besson 2 years ago

Yadd @guimard · 2 years ago Owner

@maxbes: right, we should outright discourage it in docs

Maxime Besson @maxbes · 2 years ago Author Maintainer

This also affects the portal and probably the manager (but not in the manager's default config):

```
[debug] User rtyler was granted to access to /%63heckuser
[debug] Start routing checkuser
```

Maxime Besson @maxbes · 2 years ago Author Maintainer

The bug in my previous comment also happens on Apache. Perhaps it needs a different issue/cve number in that context.

Clément OUDOT @clement_oudot · 2 years ago Owner

See #2308

Please register or sign in to reply

Clément OUDOT @clement_oudot · 2 years ago Owner

We agreed to only provide updated Nginx configurations and add a specific chapter in 2.0.9 upgrade notes.

Maxime Besson mentioned in commit 917d17f7 2 years ago

Maxime Besson mentioned in commit 91ebcdde 2 years ago

Clément OUDOT made the issue visible to everyone 2 years ago

Clément OUDOT closed 2 years ago

Yadd @guimard · 2 years ago Owner

Hi @maxbes,

is there some other changes that should be imported in Debian stable release *(I saw just `917d17f7` and `91ebcdde` here)*

Maxime Besson @maxbes · 2 years ago Author Maintainer

We should change the default config (`91ebcdde`) and maybe also show an explicit warning on upgrade (NEWS.Debian?). There is no code change yet, maybe in #2308

Edited by Maxime Besson 2 years ago

Yadd @guimard · 2 years ago Owner

Debian package has no default config (test not installed). So a debian/NEWS entry is enough here, thanks!

Yadd @guimard · 2 years ago Owner

This vulnerability is also fixed in node "lemonldap-ng-handler" **0.5.2**. Report sent to npmjs.

Yadd @guimard · 2 years ago Owner

GitHub reference for node handler: https://github.com/LemonLDAPNG/node-lemonldap-ng-handler/security/advisories/GHSA-x44x-r84w-8v67

Yadd @guimard · 2 years ago Owner

Npmjs advisory: https://www.npmjs.com/advisories/1557

Please register or sign in to reply