Lua readonly tables (CVE-2022-24736, CVE-2022-24735) #10651

Merged

oranagra merged 4 commits into `redis:unstable` from `oranagra:meir_lua_readonly_tables` on Apr 27

Conversation 4 Commits 4 Checks 13 Files changed 12

oranagra commented on Apr 27 • edited Contributor

Lua readonly tables

The PR adds support for readonly tables on Lua to prevent security vulnerabilities:

- (CVE-2022-24736) An attacker attempting to load a specially crafted Lua script can cause NULL pointer dereference which will result with a crash of the redis-server process. This issue affects all versions of Redis.
- (CVE-2022-24735) By exploiting weaknesses in the Lua script execution environment, an attacker with access to Redis can inject Lua code that will execute with the (potentially higher) privileges of another Redis user.

The PR is spitted into 4 commits.

Change Lua to support readonly tables

This PR modifies the Lua interpreter code to support a new flag on tables. The new flag indicating that the table is readonly and any attempt to perform any writes on such a table will result in an error. The new feature can be turned off and on using the new `lua_enablereadonlytable` Lua API. The new API can be used **only** from C code. Changes to support this feature was taken from https://luau-lang.org/

Change eval script to set user code on Lua registry

Today, Redis wrap the user Lua code with a Lua function. For example, assuming the user code is:

```
return redis.call('ping')
```

The actual code that would have sent to the Lua interpreter was:

```
f_b3a02c833904802db9c34a3cf1292eee3246df3c() return redis.call('ping') end
```

The warped code would have been saved on the global dictionary with the following name: `f_<script sha>` (in our example `f_b3a02c833904802db9c34a3cf1292eee3246df3c`). This approach allows one user to easily override the implementation of another user code, example:

```
f_b3a02c833904802db9c34a3cf1292eee3246df3c = function() return 'hacked' end
```

Running the above code will cause `evalsha b3a02c833904802db9c34a3cf1292eee3246df3c 0` to return `hacked` although it should have returned `pong` . Another disadvantage is that Redis basically runs code on the loading (compiling) phase without been aware of it. User can do code injection like this:

```
return 1 end <run code on compling phase> function() return 1
```

The warped code will look like this and the entire `<run code on compiling phase>` block will run outside of eval or evalsha context:

```
f_<sha>() return 1 end <run code on compling phase> function() return 1 end
```

The commits puts the user code on a special Lua table called the registry. This table is not accessible to the user so it can not be manipulated by him. Also there is no longer a need to warp the user code so there is no risk in code injection which will cause running code in the wrong context.

Use `lua_enablereadonlytable` to protect global tables on eval and function

The commit uses the new `lua_enablereadonlytable` Lua API to protect the global tables of both evals scripts and functions. For eval scripts, the implementation is easy, We simply call `lua_enablereadonlytable` on the global table to turn it into a readonly table.

On functions its more complected, we want to be able to switch globals between load run and function run. To achieve this, we create a new empty table that acts as the globals table for function, we control the actual globals using metatable manipulations. Notice that even if the user gets a pointer to the original tables, all the tables are set to be readonly (using `lua_enablereadonlytable` Lua API) so he can not change them. The following better explains the solution:

```
Global table {} <- global table metatable {.__index = __real_globals__}
```

The `__real_globals__` is depends on the run context (function load or function call).

Why is this solution needed and its not enough to simply switch globals? When we run in the context of function load and create our functions, our function gets the current globals that was set when they were created. Replacing the globals after the creation will not effect them. This is why this trick it mandatory.

Protect the rest of the global API and add an allowed list to the provided API

The allowed list is done by setting a metatable on the global table before initialising any library. The metatable set the `__newindex` field to a function that check the allowed list before adding the field to the table. Fields which is not on the
allowed list are simply ignored.

After initialisation phase is done we protect the global table and each table that might be reachable from the global table. For each table we also protect the table metatable if exists.

Performance

Performance tests was done on a private computer and its only purpose is to show that this fix is not causing any performance regression.

case 1: `return redis.call('ping')`
case 2: `for i=1,10000000 do redis.call('ping') end`

	Unstable eval	**Unstable function**	**lua_readonly_tables eval**	**lua_readonly_tables function**
case1 ops/sec	235904.70	236406.62	232180.16	230574.14
case1 avg latency ms	0.175	0.164	0.178	0.149
case2 total time in seconds	3.373	3.444s	3.268	3.278

Breaking changes

- `print` function was removed from Lua because it can potentially cause the Redis processes to get stuck (if no one reads from stdout). Users should use redis.log. An alternative is to override the `print` implementation and print the message to the log file.

todo:

- [x] Add commit message about where we took the code from
- [x] Remember its not going to be squashed
- [x] check performance
- [x] check debugger

- [x] Remove print from white list

All the work by **@MeirShpilraien**, i'm just publishing it.

MeirShpilraien added 4 commits 7 months ago

`Added support for Lua readonly tables.` … 8b33d81

`Move user eval function to be located on Lua registry.` … 992f9e2

`Protect globals of both evals scripts and functions.` … 3731580

`Protect any table which is reachable from globals and added globals w…` … ✓ efa162b

yossigo approved these changes on Apr 27

View changes

oranagra added **state:major-decision** release-notes labels on Apr 27

oranagra merged commit **89772ed** into `redis:unstable` on Apr 27
13 checks passed

View details

oranagra deleted the `meir_lua_readonly_tables` branch 7 months ago

oranagra mentioned this pull request on Apr 27

Redis 7.0.0 #10652

Merged

tezc mentioned this pull request on Apr 27

Use local function inside Lua scripts RedisLabs/redisraft#335

Merged

bak1an mentioned this pull request on May 3

Fails on redis 6.2.7 and redis 7.0 due to globally reachable lua tables becoming read only
seomoz/qless-core#89

Open

sigmaris mentioned this pull request on May 10

fix: replace usage of global functions in Lua with locals getsentry/sentry#34416

Merged

oranagra mentioned this pull request on Jun 10

[BUG] Redis7 Lua engine and readonly table script #10845

Closed

hrsantiago mentioned this pull request on Jun 10

Dont modify redis read only lua tables ledgetech/lua-resty-qless#14

Merged

hrsantiago commented on Jun 10 • edited

I was running a website using OpenResty and suddenly error messages started to loop due to qless.
It took me a couple hours to figure out what was happening here. This is the fix in my case:
ledgetech/lua-resty-qless#14

This PR seems like a good change, however it might make some devs lose a lot of time because their scripts will break.

Maybe improve the error message? As "Attempt to modify a readonly table" didnt help much.
I didnt even know that this was related to Redis at the start.
Something like: "Attempt to modify a readonly table. Since Redis 7.0.1 lua scripts can't modify global tables. Change your script or use an older version of Redis."

👍 1

MeirShpilraien commented on Jun 11 Contributor

Thanks **@hrsantiago** for the feedback, looking at it now I agree we can improve the error message and indicate that the problem is in the Lua script. **@oranagra** WDYT?

oranagra commented on Jun 11 Contributor Author

The error I see quoted in ledgetech/lua-resty-qless#14 seems already indicating the problem is in a read only table in a Lua script.
And including a redis version number in the error message doesn't seem right. Maybe we can say that "new versions of Redis have a restriction", although... Wasn't it clear since it happened after upgrade?

Anyway, just for the record, looking at the PR you made, I understand that this script used to create a method in the table, and then call that method. But there was actually no need for that method to be part of the table (the change was to make it a stand alone method)..

hrsantiago commented on Jun 11

"Wasn't it clear since it happened after upgrade?"

Not really, I upgraded my archlinux ~2 weeks ago. Yesterday I had to deploy a new feature on my website and then it wasn't working.
To be honest I had no idea that Redis could run lua scripts. So first i thought the problem was inside lua-resty-qless. Luckly there wasn't any updates there, so I didnt waste much time looking at their commits.
I've tried to google "Attempt to modify a read only table script", but that lead me nowhere.

This seems ok enough: ""new versions of Redis have a restriction". It leads people to redis and might guide them on how to fix

tom93 added a commit to tom93/qless-core that referenced this pull request on Jul 5

`Convert method table.extend() to a local function because 'table' is …` … ac2034f

tom93 mentioned this pull request on Jul 5

Convert method table.extend() to a local function because 'table' is … seomoz/qless-core#90

Open

enjoy-binbin mentioned this pull request on Jul 23

[BUG] EVAL "return _G" 0 leads to immediate panic #11030

Closed

Reviewers

yossigo ✓

Assignees

No one assigned

Labels

release-notes state:major-decision

Projects

None yet

Milestone

No milestone

Development

Successfully merging this pull request may close these issues.

None yet

4 participants