Is there anyway to avoid this security issue in Lua?

Asked 16 years ago Modified 16 years ago Viewed 6k times

7

I was just working on a localizable Lua string solution, when I came up with this hack, problem is I don't know how to avoid getting hacked by it :) So I was wondering if anyone, has done something similar and or knows how to protect from this kind of attack. (in user code)

Since we can do this:

```
=("foo"):upper() -->output: FOO
```

It can be hacked like this:

```
getmetatable("foo").__index.upper = function() print("bye bye
sucker");os.exit() end
=("foo"):upper() -->output: bye bye sucker (application quits)
-- or this way
=string.upper("bar") -->output: bye bye sucker (application quits)
```

Any ideas?

security lua

Share

Improve this question

Follow

edited Nov 28, 2008 at 9:11

asked Nov 28, 2008 at 9:03

Robert Gould

69.7k ● 61 ● 191 ● 275

1 No need to use the string metatable; it is simply the global table string. So using string.upper=function ... end users will still be able to change the function. – jpjacobs Jan 16, 2011 at 22:19

6 Answers

Sorted by: Highest score (default)

11

First and foremost execute untrusted code in sandboxed environment only – as it was said by other posters. Except for loading bytecode chunks, Lua allows all other sandboxing issues to be covered. (And bytecode chunk problems get fixed promptly as discovered.)

See Lua Live Demo for an example of sandboxing. Sources are available here.

Your specific problem with metatables is solved by setting a `__metatable` field:

> If you set a `__metatable` field in the metatable, `getmetatable` will return the value of this field, whereas `setmetatable` will raise an error.
>
> – Roberto Ierusalimschy, Programming in Lua 1st edition, 13.3 - Library-Defined Metamethods

For example:

```
> mt = { __metatable = true }
> t = {}
> setmetatable(t, mt)
> setmetatable(t, mt)
stdin:1: cannot change a protected metatable
stack traceback:
 [C]: in function 'setmetatable'
 stdin:1: in main chunk
 [C]: ?
```

So, all you have to do is:

```
getmetatable("").__metatable = true
```

Share
Improve this answer
Follow

edited Nov 28, 2008 at 19:29

answered Nov 28, 2008 at 17:55
Alexander Gladysh
41.8k ● 33 ● 110 ● 164

__metatable is what I what I was looking for, had forgotten this! – Robert Gould Nov 29, 2008 at 0:04

6

If your hacker has the ability to add code, and you need to allow that code to call things like os.exit, then you're pretty much out of luck anyway.

You can restrict the functions that their code can call, though. It depends on what you still want user code to be able to do. See the doc for setfenv and google for "lua sandbox"

Share Improve this answer Follow

answered Nov 28, 2008 at 9:19
The Archetypal Paul
41.7k ● 20 ● 106 ● 135

I mean of course os module isn't available to users , but I used it for the impact factor in my code. Anyways thanks for the sandbox idea, hadn't thought about googling that.
– Robert Gould Nov 28, 2008 at 9:40

2

I am not sure why you have an issue, since you probably already know about sandboxes: you can remove dangerous functions like io.exit, and you can ensure the overridden functions are only those in the global table of the user, ie. the Lua functions used internally by your application will remain intact.
In any case, if the hacker can call os.exit directly, the fact he can shoot himself in the foot by supercharging an innocent function he will use later is his problem.
Beside, it is a problem only if you run user functions on your server, for example: if the hacker destroys his system, again, that's his problem!
Now, there is also the issue of distributing dangerous code: it is up to you to restrict the power of user scripts. After all, that's what browsers do with JavaScript.

Share Improve this answer Follow

answered Nov 28, 2008 at 9:30
PhiLho
41.1k ● 6 ● 99 ● 136

The problem is overriding string.upper, not directly using os.exit. But I'm not sure if sandboxing will directly protect against the overriding, but now that you mention it I should also hide the get/setmetatable functions from the user environment. – Robert Gould Nov 28, 2008 at 9:45

2

This *security problem* is typically illustrated with this sentence, said by Ford Prefect in the brilliant books *The Hitchhiker's Guide to the Galaxy*: It rather involved being on the other side of this airtight hatchway

My ability to write code cannot be said to be a security vulnerability, and if you can't control your code, *that* is your security problem, not what that code can do.

There are tons and tons of things you can do if you can just get the machine to execute some of your code. The security is to avoid getting the code in there in the first place. Everything after that is just collateral damage.

The way to avoid being hacked by that problem is to avoid getting unknown code into your application.

Share Improve this answer Follow

answered Nov 28, 2008 at 9:41
Lasse V. Karlsen
391k ● 106 ● 646 ● 844

+1 for the reality check, however my goal is to allow some user code to be run. But yes you are right I need to rethink my problem here – Robert Gould Nov 28, 2008 at 9:50

The sandbox approach others are mentioning is unknown to me, so it seems you do have a way to limit this ability in this case. – Lasse V. Karlsen Nov 28, 2008 at 9:58

1

I don't see the possibility to redefine `upper` as the problem. Being able to see `os.exit` is the problem.

As suggested by others, make a sandboxed environment for your scripts. Each script can get a new one; then a person can redefine upper or anything like that, and all they'll screw up is their own thing.

Creating Lua states is so fast and easy, this won't cause any problems.

Another thing you might beware of is eternal loops. Making a 'watchdog' that kills a script after, say, 10000 instructions takes about 10 lines of C code. I can send you sample if you need.

Share Improve this answer Follow

answered Dec 11, 2008 at 11:45
akauppi
18k ● 15 ● 102 ● 124

0

I have no solution (I don't use Lua, I'm just interested in it from afar), but what you're after is called a "sandbox". Google for [Lua sandbox](), I found a few seemingly interesting pages that way. For example: http://lua-users.org/wiki/SandBoxes.

Share
Improve this answer
Follow

edited Nov 28, 2008 at 9:38

answered Nov 28, 2008 at 9:33
bart
7,759 ● 3 ● 35 ● 40