Dev server redirects to arbitrary url when path starts with double slash // #1639

Closed   **raminfp** opened this issue on Dec 5, 2015 · 11 comments

raminfp commented on Dec 5, 2015

```
from flask import Flask
from flask import request

app = Flask(__name__)

@app.route('/')
def index():
    return  'hello'

app.run(debug=True, port=8000, host='0.0.0.0')
```

If I try to navigate to http://127.0.0.1:8000//google.com (2 slashes), I get redirected to google.com. I correctly get a 404 with http://127.0.0.1:8000/google.com (1 slash) and http://127.0.0.1:8000///google.com (3 slashses). This is a vulnerability, I shouldn't be redirected to arbitrary urls. Flask should prevent arbitrary redirects from urls.

wlhlm commented on Dec 5, 2015

As the documentation states the integrated server is intended **for development environments only**:

> Depending on what you have available there are multiple ways to run Flask applications. You can use the builtin server during development, but you should use a full deployment option for production applications. (Do not use the builtin development server in production.) Several options are available and documented here.

This doesn't seem like an issue to me, or does this happen with other servers as well (gunicorn, uwsgi, etc.)?

raminfp commented on Dec 5, 2015   Author

I didn't test with other servers but I ran the Flask server and the issue happened. Use the code above, run the server, and check the requests and responses.

```
GET //google.com HTTP/1.0
Host: 127.0.0.1:8000
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8
Cookie: csrftoken=v9WXXZW8uDWz7XrarUBqmZdH1drdWErD
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.71 Safari/537.36

HTTP/1.0 301 MOVED PERMANENTLY
Content-Length: 243
Content-Type: text/html; charset=utf-8
Date: Sat, 05 Dec 2015 19:17:30 GMT
Server: Werkzeug/0.11.2 Python/2.7.10
Location: http://google.com/

GET / HTTP/1.1
Host: google.com
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8
Cookie: PREF=ID=1111111111111111:FF=0:LD=en:TM=1446740391:LM=1446740590:GM=1:V=1:S=bvrh0iJSlDPfDRcm; HSID=A-RWO0xT0s-nF2oZZ; APISID=vdJAmppxhi_CMLCt/Af_PAw9M7_IGr6RNZ;
SID=DQAAACABAAAvvyatOpvJoggRTBvHH5GR3Y_H8Wt6nwYv6ofWOlggT5ZHjeOPn_0sAUzzt5N5Pl4JjB-JorMQWb-
DCcc0zW0dJ8nk5H6i2UPDZqreIPFmbCzlOflvGprvnqs1lWsRM_G80yqPllj_9rwoDwSM1hnZOdyBWVKl35xZ9uzSOWbYWTxskIL_eMIl_2QNVx19cjQvG3_pDgITPG8BUCtH0ssh157iq0eZ2vrufgKBBvXI6V2XD4INK0-
dUmmBve736aaknl2F0YiXFDyseCDpt2UIic3Sr9sz0AP7Ii5vjwkMYxDvzojydebuxMOiAcdTrD1lfO4PrpSQEZiFhOKr2Gjko5OUO5T1rW4kVDEZA9fSHBNNVUcg92iDEfEXNlg9ZrM;
NID=74=BDFEAdVHJvHDMRIRd9sFoMfDkdTmzmqYazWGe2d1_JUmenhx-TBsrMTOQ0VMpnXsO3rofrG1U2PwyQhNX_bVimqc_bNptfFKL0H8zFoCq4aLSco4Y3843bc09cqwNoYyQE15BSoC81VQVEsjgqol0Qn-rkiFiyiys-
LKbW0jqb9OMlPPgRawh3wqec0cJlHE8iiRxa54YW7QD_t3sPYApMIqPJKcVU6otl5YPUitkPJhuzO-ZDjkClsJXVaUKAbDwVWmawr1FAjsAsS3m8beMGZw
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.71 Safari/537.36

HTTP/1.1 301 Moved Permanently
Cache-Control: public, max-age=2592000
Content-Length: 219
Content-Type: text/html; charset=UTF-8
Date: Sat, 05 Dec 2015 19:18:00 GMT
Expires: Mon, 04 Jan 2016 19:18:00 GMT
Location: http://www.google.com/
Server: gws
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1; mode=block
```

davidism closed this as completed on Dec 5, 2015

davidism reopened this on Dec 6, 2015

raminfp commented on Dec 6, 2015   Author

I tested in Firefox, Chrome, and Opera and redirected to domains such as google.com, bing.com, yahoo.com, and msn.com.

> Running the app with gunicorn prevents this as well.

This bug exists in Flask on `//` redirect to other site. If the user runs Flask on a VPS without using another web server, this issue can be used in a phishing attack.

Why do `///` and `/everything` display "Not Found"? I think this issue should be fixed.

ThiefMaster commented on Dec 6, 2015 Member

You are not supposed to do that. Open redirects are mostly an issue on
high-profile/trusted sites and those are even less likely to use the dev
server in production.

@davidism: 301 redirect maybe? That'd explain why it happens for you with
no server running.

raminfp commented on Dec 6, 2015 Author

> those are even less likely to use the dev server in production.

If I run the service on Linux with bash and access the Flask app, this is an issue. Example: http://148.251.25.244:9090//google.com

ThiefMaster commented on Dec 6, 2015 Member

The dev server is not acceptable for any production setup, period. If someone uses it anyway, it's his fault if his box gets pwned or abused.

I just looked into it and the problem happens due to an incorrect `SERVER_NAME` being passed to the werkzeug from the dev server (it's set to `google.com` when using the two slashes)

RonnyPfannschm... commented on Dec 6, 2015 Contributor

Can someone still report a bug to werkzeug perhaps so it can decide on how to handle this?

ThiefMaster commented on Dec 6, 2015 Member

OK, this is not a flask bug, possibly not even a werkzeug bug. `BaseHTTPRequestHandler` (python stdlib) doesn't normalize consecutive slashes into a single one so it passes a path of `//google.com` to the dev server which runs this code:

```
request_url = url_parse(self.path)
```

The double slash results in the `netloc` being set instead of the `path` so eventually it does end up in `HTTP_HOST`. As a workaround we could normalize slashes before using `self.path` in `WSGIRequestHandler`.

ThiefMaster mentioned this issue on Dec 6, 2015

dev server sets wrong HTTP_HOST when path starts with a double slash pallets/werkzeug#822

Closed

raminfp commented on Dec 6, 2015 Author

You mean the issue is here?

```
class WSGIRequestHandler(BaseHTTPRequestHandler, object):
    """A request handler that implements WSGI dispatching."""

    @property
    def server_version(self):
        return 'Werkzeug/' + werkzeug.__version__

    def make_environ(self):
        request_url = url_parse(self.path) ### issue here?
```

https://patreon.thecthulhu.com/extracted/patreon.tar.gz/venv/lib/python3.4/site-packages/werkzeug/serving.py

ThiefMaster commented on Dec 6, 2015 Member

Yes. `self.path` is set in the Python stdlib and contains `//google.com` in your case.
Later it's used to populate HTTP_HOST: https://github.com/mitsuhiko/werkzeug/blob/master/werkzeug/serving.py#L129-L130 (please use the proper repo when linking to code, not some leak that just happens to contain a copy of a virtualenv containing werkzug)

davidism commented on Dec 6, 2015 Member

Ultimately, the answer is still the same: don't use the dev server in production, it is not designed to be secure. At this point, it's clear that this is not a Flask bug, and it has been reported in pallets/werkzeug#822. Closing.

davidism closed this as completed on Dec 6, 2015

davidism changed the title ~~Open Redirect Vulnerability~~ **Dev server redirects to arbitrary url when path starts with double slash //** on Dec 6, 2015

github-actions bot locked as **resolved** and limited conversation to collaborators on Nov 14, 2020

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

5 participants