

New issue

[Jump to bottom](#)

Hardcoded Flask Secrets Key - "Privilege Escalation" #292

🔒 Closed

Securitybits-io opened this issue on Feb 16 · 1 comment

Assignees



Labels

bug_Normal

Projects

📁 1.9.9

Securitybits-io commented on Feb 16

In the sourcecode there are 3 relevant places that the Flask Secrets Key are hardcoded. Flask signs all their client sessions with this secret key, usually defined in an *Environment Variable*. In this case though there's these three places that these are hardcoded into.

Code4

Commits0

Issues1

Discussions0

Packages0

Languages

Python4

[Advanced search](#)
[Cheat sheet](#)

4 code results in [FreeTAKTeam/FreeTakServer](#) or view all results on [GitHub](#)

FreeTAKServer/controllers/services/DataPackageServer.py

```

38 app.config['SQLALCHEMY_DATABASE_URI'] = DatabaseConfiguration().DataBaseConnectionString
39 app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
40 app.config['SECRET_KEY'] = 'vnkdjnfjknfl1232#'

```

Python Showing the top match Last indexed on Dec 4, 2021

FreeTAKServer/controllers/services/legacy_api/DataPackageServer.py

```

39 app.config['SECRET_KEY'] = 'vnkdjnfjknfl1232#'
40 cors = CORS(app, resources={r'/*': {'origins': '*'}})
41 app.config['CORS_HEADERS'] = 'Content-Type'

```

Python Showing the top match Last indexed on May 19, 2021

FreeTAKServer/controllers/services/API/RestAPI.py

```

56 APIPipe = None
57 CommandPipe = None
58 app.config['SECRET_KEY'] = 'vnkdjnfjknfl1232#'
59
60
61 @app.errorhandler(404)
62 def page_not_found(e):

```

Python Showing the top match Last indexed on May 18, 2021

FreeTAKServer/controllers/services/RestAPI.py

```

70 CommandPipe = None
71 app.config['SECRET_KEY'] = 'vnkdjnfjknfl1232#'
72 eventDict = {}
73
74 @app.errorhandler(404)
75 def page_not_found(e):

```

Python Showing the top match Last indexed on Dec 22, 2021

This gives a malicious user the ability to sign their own cookies (using for example: [Flask-Unsign](#)), and internally change the UID of the current user and assume any other user, for example UID 1 which is the Admin. (Privilege Escalation)

Another interesting issue that you run into aswell is that having two Flask servers with the same *secret key* makes it possible for a user to reuse a UID 1 cookie from Server A, and apply that cookie to Server B logging in to the same UID 1. (Lateral movement/Authentication bypass).

  **brothercorvo** assigned [naman108](#), [jonaugustine](#) and **brothercorvo** on Feb 16

  **brothercorvo** added the `bug_Normal` label on Feb 16

  **brothercorvo** added this to **To do** in **1.9.9** via `automation` on Feb 16



naman108 added a commit that referenced this issue on Feb 16



addressed issues [#293](#) and [#292](#)

✗ 33180eb



naman108 mentioned this issue on Feb 16

Syntax updates, xml reception, fixed security vulnerabilities #294

Merged

brothercorvo commented on Mar 12

Collaborator

fixed in 1.9.8.5



brothercorvo closed this as completed on Mar 12



1.9.9 [automation](#) moved this from To do to Done on Mar 12

Assignees



naman108



brothercorvo



jonaugustine

Labels

[bug_Normal](#)

Projects

No open projects

1 closed project ▾

Milestone

No milestone

Development

No branches or pull requests

4 participants

