
# USENIX Security '24 Artifact Appendix: Spider-Scents: Grey-box Database-aware Web Scanning for Stored XSS

Eric Olsson
Chalmers University of Technology

Benjamin Eriksson
Chalmers University of Technology

Adam Doupé
Arizona State University

Andrei Sabelfeld
Chalmers University of Technology

## A Artifact Appendix

### A.1 Abstract

This artifact includes source code of the prototype implementation of our database-aware grey-box scanner for stored XSS, as well as Docker Compose setups for a subset of the evaluated web applications.

Running the scanner against a web application will produce a mapping from database columns to unprotected outputs where XSS payloads are executed. These unprotected outputs are code smells that correspond to either dormant stored XSS or stored XSS vulnerabilities.

Manual analysis is required to determine the vulnerability and exploitability of unprotected outputs.

### A.2 Description & Requirements

#### A.2.1 Security, privacy, and ethical concerns

There are no additional risks for reviewers, other than those already encountered when scanning a web application for vulnerabilities; i.e. running an insecure web application.

All vulnerabilities found, including those in the web applications included in this artifact, have been responsibly disclosed to the affected vendors.

#### A.2.2 How to access

Download the artifact from `https://github.com/Spider-Scents/dbfuzz/releases/tag/v0.3`.

#### A.2.3 Hardware dependencies

None.

#### A.2.4 Software dependencies

The scanner is implemented as a Python script, which uses a mysql connection to the database, and Chrome and Chromedriver to interact with the web application.

Python dependencies are included in the Pipfile of the artifact.

Docker and Docker Compose are used to set up the web applications.

mysqldump and mysql commands are used to automatically backup and restore the web application database.

graphviz is used to produce graph visuals.

Chrome/Chromedriver, Docker/Docker Compose, mysqldump/mysql, and graphviz need to be manually installed, while the Pipfile can be used to install required Python dependencies.

Python dependencies include:

- `mysql-connector-python` to connect to the database.
- `requests` to get pages from the web application
- `selenium` to browse the web application.
- `tqdm` to show a progress-bar.
- `graphviz` to produce graph visuals.
- `cssutils`, `beautifulsoup4`, and `defusedxml` to parse CSS, HTML, and XML.

#### A.2.5 Benchmarks

None.

### A.3 Set-up

#### A.3.1 Installation

Download the artifact from `https://github.com/Spider-Scents/dbfuzz/releases/tag/v0.3`.
Follow the instructions in the README; namely:
Install Python 3.10.
Install MariaDB or MySQL to provide the mysql and mysqldump commands.
Install the Chrome browser and Chromedriver.
Install Graphviz.
Install Docker and Docker Compose.
Install the Python environment with `pipenv install`.

### A.3.2 Basic Test

Unzip the Doctor Appointment Management System web application at `docker/doctor`. Inside the extracted contents, correct the permissions of the served Apache directory `dbfuzz` with:

```
chmod 755 dbfuzz
```

Start the web application with:

```
docker compose up --build
```

Restore the provided database file in `docker/doctor` through the PhpMyAdmin console at http://localhost:8080/ with `server:mariadb`, `username:root`, and `password:notSecureChangeMe`

Copy the provided `config_doctor.ini` to the root of the repository.

Update the cookies in the config file to allow the initial URL crawler to access authenticated pages, if necessary. Update the location of mysql and mysqldump, if necessary.

Run the script once to get crawl for webpage URLs with:

```
pipenv run script --config
    config_doctor.ini --insert-empty
     --reset-fuzzing --reset-
    scanning --sensitive-rows --
    primary-keys --traversal column
```

Though inapplicable to this application, the script stops at this point to prompt the user to manually remove any destructive URLs from the generated URL file `urls doctor app_config.insert_empty=True.txt`

Run the previous `pipenv run script` command as before to finish scanning the web application.

Mappings from database inputs to unprotected outputs in this web application should be generated in PDF and CSV form in the `output` folder.

## A.4 Evaluation workflow

### A.4.1 Major Claims

**(C1):** *Spider-Scents automatically finds unprotected outputs from the database of web applications. This claim is supported by experiment E1 described in Paper Section 5 whose results are included in Paper Table 1.*

**(C2):** *Spider-Scents has high coverage of database fields that can possibly contain XSS payloads. This claim is supported by experiment E2 described in Paper Section 5.3 whose results are illustrated in Paper Figure 5.*

**(C3):** *Unprotected outputs found by Spider-Scents are related to stored-XSS vulnerabilities and exploits. This claim is supported by experiment E3 described in Paper Section 5 whose results are illustrated in Paper Table 1 and Paper Table 2.*

Table 1: Unprotected outputs in Doctor Apt.

| Column |
|---|
| tblappointment.Email |
| tblappointment.Name |
| tbldoctor.Email |
| tbldoctor.FullName |
| tblspecialization.Specialization |
| tblpage.Timing |
| tblpage.PageDescription |
| tblpage.Email |

### A.4.2 Experiments

**(E1):** *Scan Doctor Apt.[10 human-minutes + 10 compute-minutes]: Scan the Doctor Appointment Management System for unprotected outputs.*
**Preparation:** Set up the Doctor Apt. web application.
**Execution:** Run the scanner script with the evaluation parameters on the Doctor Apt. web application.
**Results:** The script will have generated the mapping of database inputs to unprotected outputs in the web application. These are in `output/graph doctor.pdf` and `output/reflections doctor.csv`
The eight unprotected outputs in Table 1 should be present in the results.

**(E2):** *Doctor Apt. DB Coverage[5 human-minutes + 10 compute-minutes]: Scanning the Doctor Appointment Management System has high database coverage.*
**Preparation:** Set up the Doctor Apt. web application. Skip if experiment E1 already has been run.
**Execution:** Run the scanner script with the evaluation parameters on the Doctor Apt. web application. Skip if experiment E1 already has been run.
**Results:** The script will have generated a summary of database coverage in `output/coverage doctor.csv`
This summary should reflect that all 16 columns that could possibly contain an XSS payload string are tested by the scanner.
Note that this summary shows information about all string-type columns in the database; some may be too short to hold *any* XSS payload.

**(E3):** *Doctor Apt. Vulnerabilities[20 human-minutes + 10 compute-minutes]: Unprotected outputs in the Doctor Appointment Management System are related to stored XSS vulnerabilities.*
**Preparation:** Set up the Doctor Apt. web application. Skip if experiment E1 already has been run.
**Execution:** Run the scanner script with the evaluation parameters on the Doctor Apt. web application. Skip if experiment E1 already has been run.
**Results:** The script will have generated the mapping of

Table 2: Vulnerabilities (V) and Exploits (E) in the Doctor Appointment Management System

| Column | Input Protection | V | E |
|---|---|---|---|
| tblappointment.Email | none | true | true |
| tblappointment.Name | none | true | true |
| tbldoctor.Email | none | true | false |
| tbldoctor.FullName | none | true | false |
| tblspecialization.Specialization | no input | false | false |
| tblpage.Timing | no input | false | false |
| tblpage.PageDescription | no input | false | false |
| tblpage.Email | no input | false | false |

database inputs to unprotected outputs in `output/graph doctor.pdf` and `output/reflections doctor.csv`
Manual analysis of code relating to the database fields from unprotected outputs will show that four of these are vulnerable, while the remaining four do not have any input.
Furthermore, two of these vulnerabilities are directly exploitable given the permission system (users can perform XSS on doctors), while the remaining two represent doctor self-XSS.
These findings correspond to Table 2.

## A.5 Notes on Reusability

### A.5.1 Other web applications

In addition to the detailed description of how to run our experiments on the Doctor Appointment Management System, we also provide setups and expected results for other web applications used in our evaluation in the artifact.

With these, an interested reviewer can run experiments E1 and E2 for these other web applications.

Table 3 lists all Docker packaged web applications in the artifact. Compute time will vary for running scans on these applications, refer to Table 4 and Paper Table 3 for guidance.

### A.5.2 Docker performance

Packaging web applications as Docker containers prioritizes the ease-of-use and portability of this artifact, at the expense of its performance.

Web application performance has the largest impact on runtime for this scanner; we observe an overall 18% improvement in runtime by simply running a number of the evaluated applications natively, outside of Docker - see Table 4.

## A.6 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2024/.

Table 3: Docker-packaged web applications used in the evaluation

| Application | Date | Version | Location |
|---|---|---|---|
| CMSMS | 2022 | 2.2.16 | docker/cmsms |
| Doctor Apt. | 2023 | 2023/1/11 | docker/doctor |
| Hospital | 2022 | 2022/11/8 | docker/hospital |
| Hostel | 2021 | 2021/9/30 | docker/hostel |
| Joomla | 2023 | 4.2.8 | docker/joomla |
| MyBB | 2023 | 1.8.33 | docker/mybb |
| OpenCart | 2023 | 4.0.1.1 | docker/opencart |
| Piwigo | 2023 | 13.6.0 | docker/piwigo |
| User Login | 2021 | V3 | docker/userlogin |
| WordPress | 2023 | 6.1.1 | docker/wordpress |

Table 4: Runtime performance of Spider-Scents, Docker (D) vs. Native (N).

| Web application | Scan time (D) | Scan time (N) | Improvement |
|---|---|---|---|
| Doctor Apt. | 0:10 | 0:08 | 20% |
| Hospital | 0:31 | 0:22 | 29% |
| Hostel | 0:13 | 0:13 | 0% |
| MyBB | 6:06 | 4:21 | 29% |
| OpenCart | 4:29 | 1:39 | 63% |
| Piwigo | 0:59 | 1:07 | -14% |
| PrestaShop | 38:39 | 32:29 | 16% |
| User Login | 0:02 | 0:01 | 50% |
| WordPress | 2:22 | 1:55 | 19% |
| Overall | 51:31 | 42:15 | 18% |