# Intro

Find yourself writing the same descriptions over and over again? Tired of typos coming up in your reports? Faraday provides a simple solution; unify criteria for naming vulnerabilities and save time and effort to yourself and your team.

> Write vulns once and use them forever!

Faraday Server comes with its own CWE Vulnerabilities DB for you to use. This is a simple **CSV** made using Open Source projects based in the **CWE standard** and allows you to create vulnerabilities without worrying about finding references, description, etc.

# Index

# Topics

## CSV

Faraday ships with a **CSV of the original Mitre project** included in its tree in `data/cwe.csv`. However, we also ship two different scripts to generate CSVs for **CFDB** and **VulnDB**. These scripts will download and parse the contents of those databases.

- CFDB Execute the following command to get a CSV for CFDB

```
$ ./helpers/cfdbToCsv.py
```

- VulnDb Execute the following command to get a CSV for VulnDB

```
$ ./helpers/vulndbToCsv.py
```

Next copy this CSV file (either cfdb.csv or vulndb.csv) to /data/cwe.csv.

## Upload CSV file

### Manual Import

Go to the **Web UI** and click on the **import** icon

A modal dialog will pop up asking you to choose a CSV file to upload, select it, click **ok** and you're done!

### Script

To upload the CSV to CouchDB using the script go to your Faraday Server installation root directory and run:

```
$ ./helpers/pushCwe.py
```

Use the paramater `-c` if you have a username and password for Faraday.

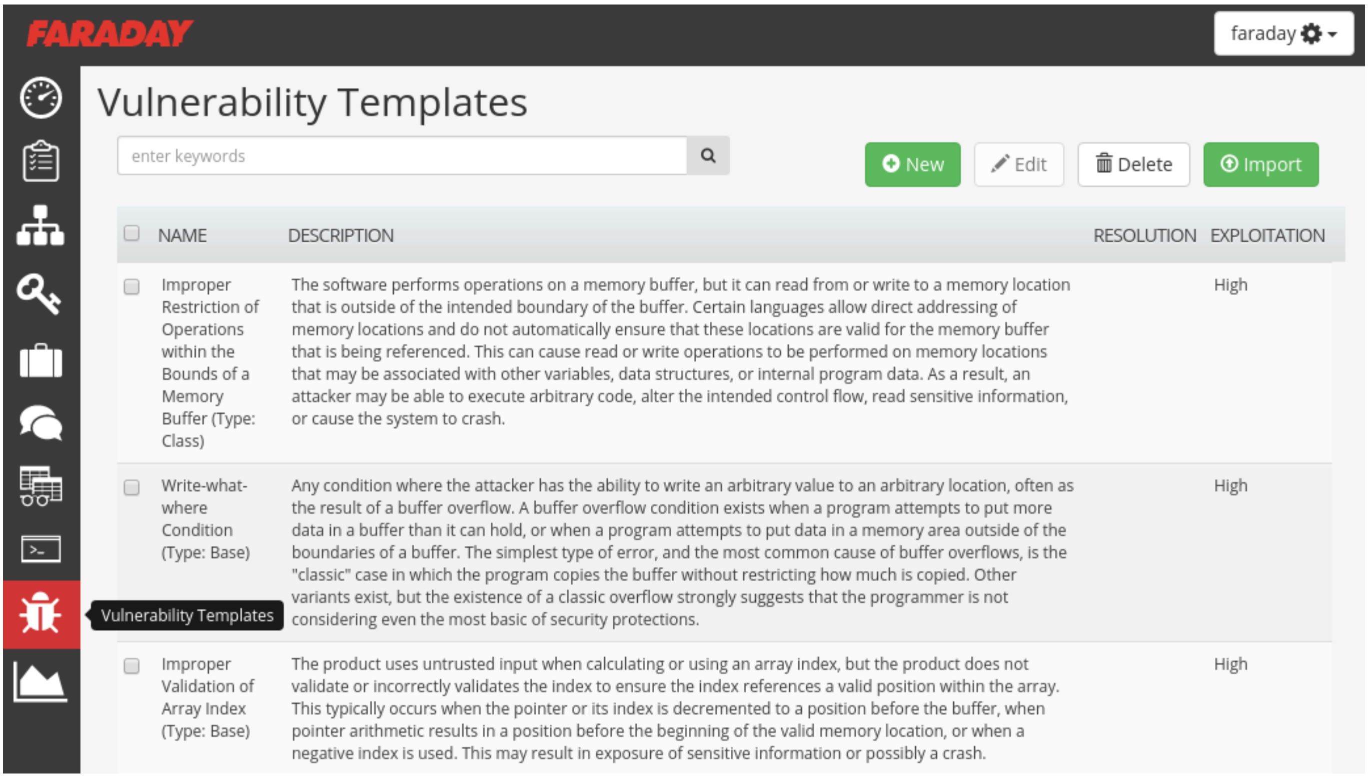
```
$ ./pushCwe.py -c 'http://USERNAME:PASSWORD@HOSTNAME:PORT/'
```

Also, if you need add your own CSV file, put the CSV inside `$FARADAY/data/cwe.csv`. And run pushCWE. Make sure you run the `pushCwe.py` script before use and that's it!

## Manually Adding Templates

You can also create templates manually in the Web UI. Click on the **vulnerability templates** icon

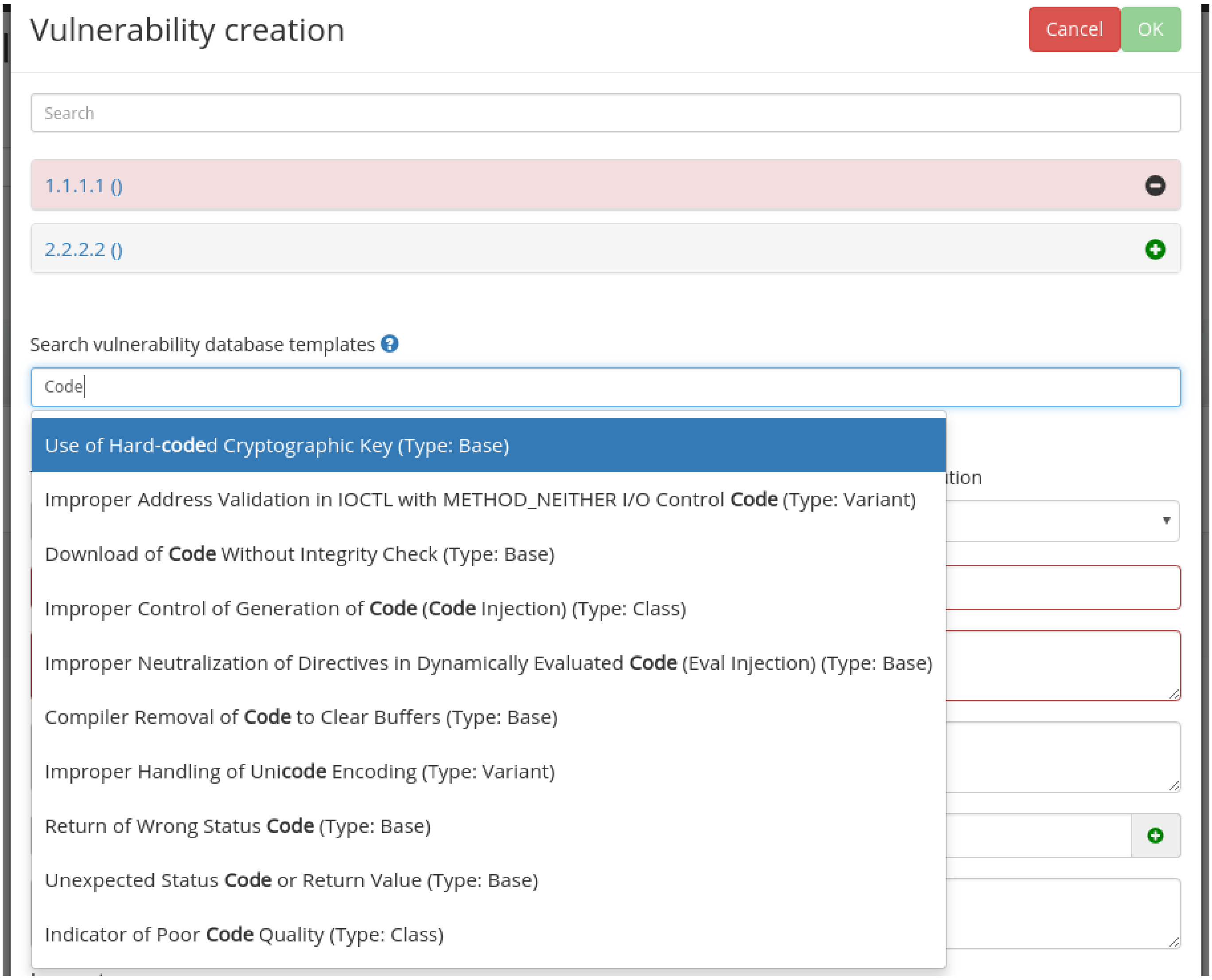
You will get a list of the existing templates in your installation



| □ | NAME | DESCRIPTION | RESOLUTION | EXPLOITATION |
|---|---|---|---|---|
| □ | Improper Restriction of Operations within the Bounds of a Memory Buffer (Type: Class) | The software performs operations on a memory buffer, but it can read from or write to a memory location that is outside of the intended boundary of the buffer. Certain languages allow direct addressing of memory locations and do not automatically ensure that these locations are valid for the memory buffer that is being referenced. This can cause read or write operations to be performed on memory locations that may be associated with other variables, data structures, or internal program data. As a result, an attacker may be able to execute arbitrary code, alter the intended control flow, read sensitive information, or cause the system to crash. | | High |
| □ | Write-what-where Condition (Type: Base) | Any condition where the attacker has the ability to write an arbitrary value to an arbitrary location, often as the result of a buffer overflow. A buffer overflow condition exists when a program attempts to put more data in a buffer than it can hold, or when a program attempts to put data in a memory area outside of the boundaries of a buffer. The simplest type of error, and the most common cause of buffer overflows, is the "classic" case in which the program copies the buffer without restricting how much is copied. Other variants exist, but the existence of a classic overflow strongly suggests that the programmer is not considering even the most basic of security protections. | | High |
| □ | Improper Validation of Array Index (Type: Base) | The product uses untrusted input when calculating or using an array index, but the product does not validate or incorrectly validates the index to ensure the index references a valid position within the array. This typically occurs when the pointer or its index is decremented to a position before the buffer, when pointer arithmetic results in a position before the beginning of the valid memory location, or when a negative index is used. This may result in exposure of sensitive information or possibly a crash. | | High |

## Usage

Login to your Faraday Web UI and create or edit a vulnerability. A search field will allow you to find your templates, as shown in the picture below.



Use of Hard-**code**d Cryptographic Key (Type: Base)

Improper Address Validation in IOCTL with METHOD_NEITHER I/O Control **Code** (Type: Variant)

Download of **Code** Without Integrity Check (Type: Base)

Improper Control of Generation of **Code** (**Code** Injection) (Type: Class)

Improper Neutralization of Directives in Dynamically Evaluated **Code** (Eval Injection) (Type: Base)

Compiler Removal of **Code** to Clear Buffers (Type: Base)

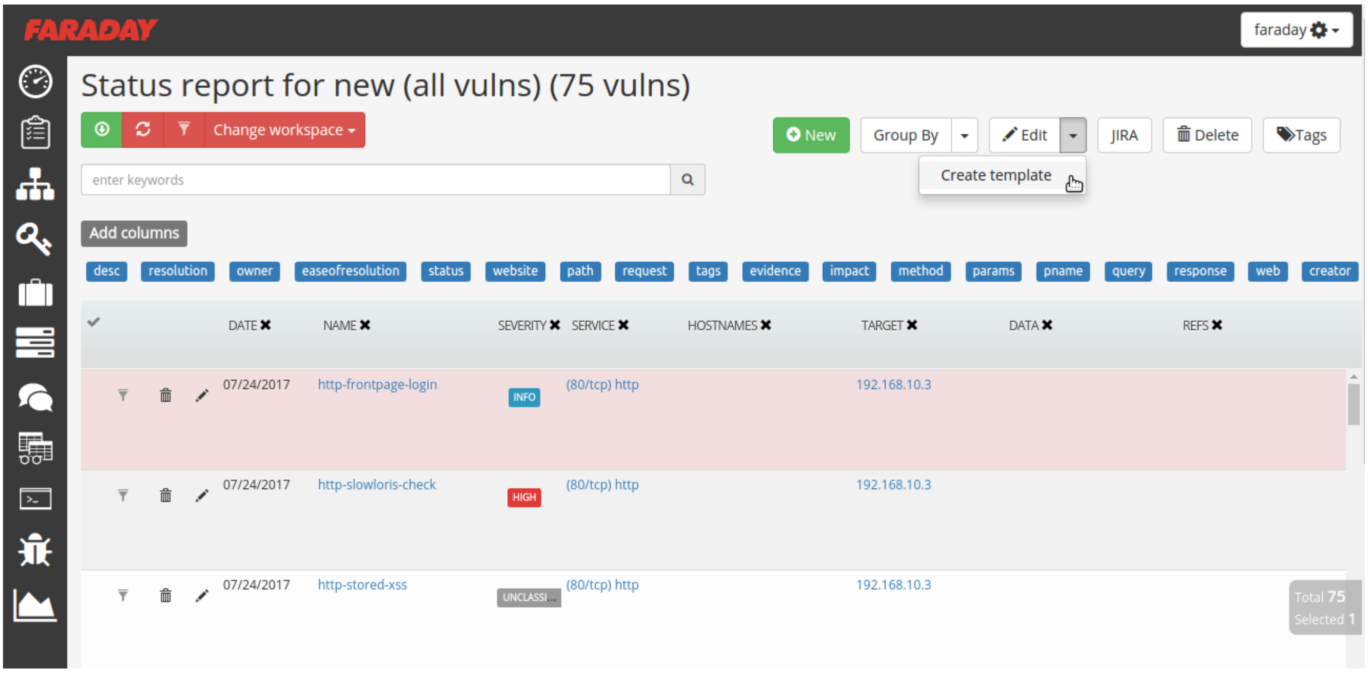
Improper Handling of Uni**code** Encoding (Type: Variant)

Return of Wrong Status **Code** (Type: Base)

Unexpected Status **Code** or Return Value (Type: Base)

Indicator of Poor **Code** Quality (Type: Class)

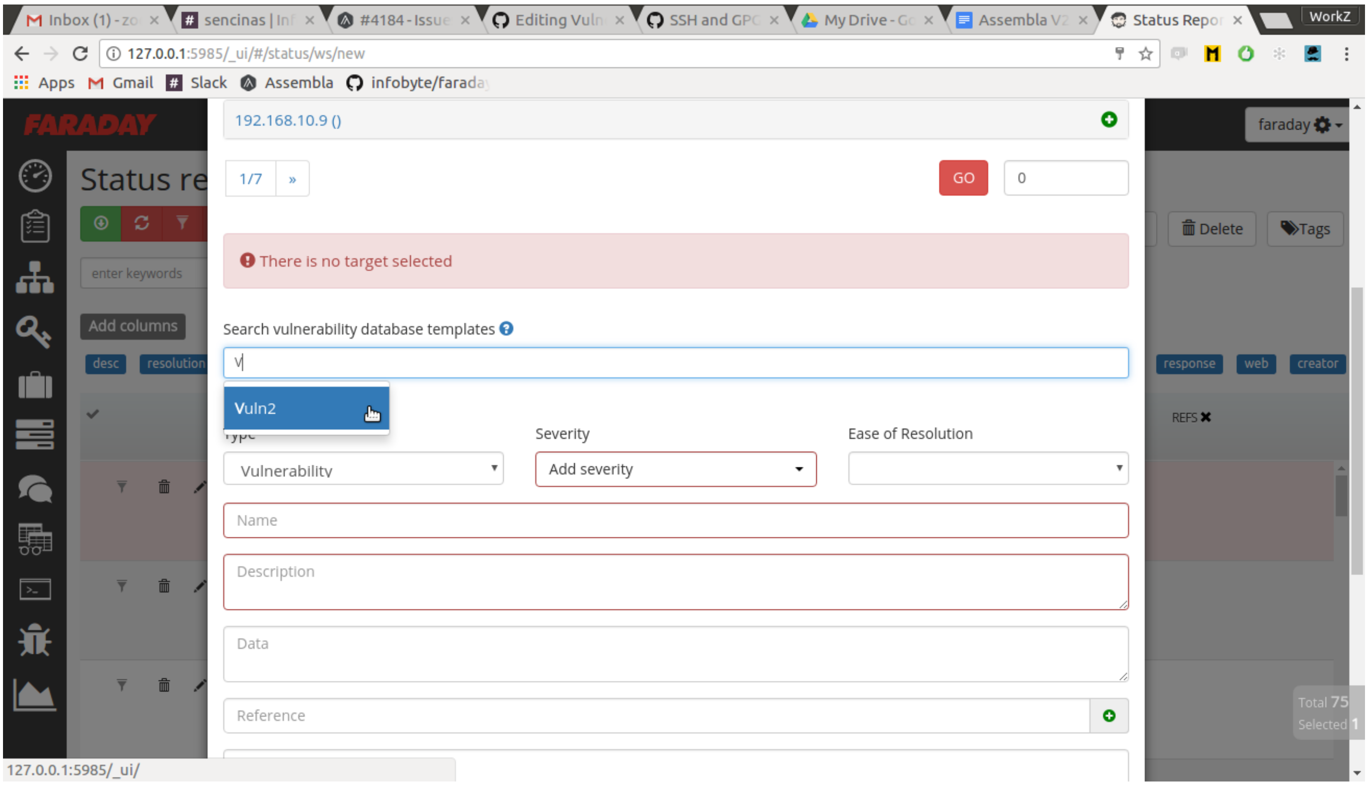
You can also duplicate vulnerabilities easily by saving them as a template.



Status report for new (all vulns) (75 vulns)

| DATE | NAME | SEVERITY | SERVICE | HOSTNAMES | TARGET | DATA | REFS |
|---|---|---|---|---|---|---|---|
| 07/24/2017 | http-frontpage-login | INFO | (80/tcp) http | | 192.168.10.3 | | |
| 07/24/2017 | http-slowloris-check | HIGH | (80/tcp) http | | 192.168.10.3 | | |
| 07/24/2017 | http-stored-xss | UNCLASSI... | (80/tcp) http | | 192.168.10.3 | | |

and later on importing the template:



192.168.10.9 ()

There is no target selected

Search vulnerability database templates

V

Vuln2

| Type | Severity | Ease of Resolution |
|---|---|---|
| Vulnerability | Add severity | |

Name

Description

Data

Reference

**Note:** Name, Description and Resolution fields are replaced with the information stored in the templates database.