
Sobelow



Sobelow is a security-focused static analysis tool for Elixir & the Phoenix framework. For security researchers, it is a useful tool for getting a quick view of points-of-interest. For project maintainers, it can be used to prevent the introduction of a number of common vulnerabilities.

Currently Sobelow detects some types of the following security issues:

- Insecure configuration
- Known-vulnerable Dependencies
- Cross-Site Scripting
- SQL injection
- Command injection
- Code execution
- Denial of Service
- Directory traversal
- Unsafe serialization

Potential vulnerabilities are flagged in different colors according to confidence in their insecurity. High confidence is red, medium confidence is yellow, and low confidence is green.

A finding is typically marked “low confidence” if it looks like a function could be used insecurely, but it cannot reliably be determined if the function accepts user-supplied input. i.e. **If a finding is marked green, it may be critically insecure, but it will require greater manual validation.**

Note: This project is in constant development, and additional vulnerabilities will be flagged as time goes on. If you encounter a bug, or would like to request additional features or security checks, please open an issue!

Table of Contents

- Installation
 - To Use
- Options
- Configuration Files
- False Positives
- Modules
- Umbrella Apps
- Updates

Installation

To use Sobelow, you can add it to your application's dependencies.

```
1 def deps do
2   [
3     {:sobelow, "~> 0.13", only: [:dev, :test], runtime: false}
4   ]
5 end
```

You can also install Sobelow globally by executing the following from the command line:

```
1 $ mix escript.install hex sobelow
```

To install from the master branch, rather than the latest release, the following command can be used:

```
1 $ mix escript.install github nccgroup/sobelow
```

To Use

After installation, the simplest way to scan a Phoenix project is to run the following from the project root:

```
1 $ mix sobelow
```

Options

Note: Any path arguments should be absolute paths, or relative to the application root.

- `--root` or `-r` - Specify the application root directory. Accepts a path argument, e.g. `../my_project`.
- `--verbose` or `-v` - Print code snippets and additional finding details.
- `--ignore` or `-i` - Ignore given finding types. Accepts a comma-separated list of module names, e.g. `XSS.Raw, Traversal`.
- `--ignore-files` - Ignore files. Accepts a comma-separated list of file names, e.g. `config/prod.exs`.
- `--details` or `-d` - Get finding-type details. Accepts a single module name, e.g. `Config.CSRF`.
- `--all-details` - Get details of all finding-types.

-
- `--private` - Skip update checks.
 - `--router` - Specify router location. This only needs to be used if the router location is non-standard. Accepts a path argument, e.g. `my/strange/router.ex`.
 - `--exit` - Return non-zero exit status at or above a confidence threshold of `low`, `medium`, or `high`. Defaults to `false` which returns a zero exit status
 - `--threshold` - Return findings at or above a confidence level of `low` (default), `medium`, or `high`.
 - `--format` or `-f` - Specify findings output format. Accepts a format, e.g. `txt` or `json`.

Note that options such as `--verbose` will not work with the `json` format. All `json` formatted findings contain a `type`, `file`, and `line` key. Other keys may vary.

- `--quiet` - Return a single line indicating number of findings. Otherwise, return no output if there are no findings.
- `--compact` - Minimal, single-line findings with output colorised according to confidence.
- `--flycheck` - Minimal, single-line findings that are compatible with flycheck-based tooling.
- `--save-config` - Generates a configuration file based on command line options. See Configuration Files for more information.
- `--config` - Run Sobelow with configuration file. See Configuration Files for more information.
- `--mark-skip-all` - Mark all displayed findings as skippable.
- `--clear-skip` - Clear configuration created by `--mark-skip-all`.
- `--skip` - Ignore findings that have been marked for skipping. See False Positives for more information.
- `--version` - Outputs the current version of Sobelow. This is useful for CI steps or integration with other tools like Salus.

Configuration Files

Sobelow allows users to save frequently used options in a configuration file. For example, if you find yourself constantly running:

```
1 $ mix sobelow -i XSS.Raw,Traversal --verbose --exit Low
```

You can use the `--save-config` flag to create your `.sobelow-conf` config file:

```
1 $ mix sobelow -i XSS.Raw,Traversal --verbose --exit Low --save-config
```

This command will create the `.sobelow-conf` file at the root of your application. You can edit this file directly to make changes.

You can also run the command without any options:

```
1 $ mix sobelow --save-config
```

when you first start out using this package - the generated configuration file will be populated with the default values for each option. (This helps in quickly incorporating this package into a pre-existing codebase.)

Now if you want to run Sobelow with the saved configuration, you can run Sobelow with the `--config` flag.

```
1 $ mix sobelow --config
```

False Positives

Sobelow favors over-reporting versus under-reporting. As such, you may find a number of false positives in a typical scan. These findings may be individually ignored by adding a `# sobelow_skip` comment, along with a list of modules, before the function definition.

```
1 # sobelow_skip ["Traversable"]
2 def vuln_func(...) do
3   ...
4 end
```

When integrating Sobelow into a new project, there can be a large number of false positives. To mark all printed findings as false positives, run sobelow with the `--mark-skip-all` flag.

Once you have tagged the appropriate findings, run Sobelow with the `--skip` flag.

```
1 $ mix sobelow --skip
```

While `# sobelow_skip` comments can only mark function-level findings (and so cannot be used to skip configuration issues), the `--mark-skip-all` flag can be used to skip any finding type.

Modules

Findings categories are broken up into modules. These modules can then be used to either ignore classes of findings (via the `ignore` and `skip` options) or to get vulnerability details (via the `details` option).

This list, and other helpful information, can be found on the command line:

```
1 $ mix help sobelow
```

Umbrella Apps

In order to run Sobelow against all child apps within an umbrella app with a single command, you can add an alias for sobelow in your root `mix.exs` file:

```
1 defp aliases do
2   [
3     sobelow: ["cmd mix sobelow"]
4   ]
5 end
```

If you wish to use configuration files in an umbrella app, create a `.sobelow-conf` in each child application and use the `--config` flag.

Updates

When scanning a project, Sobelow will occasionally check for updates, and will print an alert if a new version is available. Sobelow keeps track of the last update-check by creating a `.sobelow` file in the root of the scanned project.

If this functionality is not desired, the `--private` flag can be used with the scan.