Dawn - The raising security scanner for ruby web applications

Dawn is a source code scanner designed to review your ruby code for security issues.

Dawn is able to scan plain ruby scripts (e.g. command line applications) but all its features are unleashed when dealing with web applications source code. Dawn is able to scan major MVC (Model View Controller) frameworks, out of the box:

- Ruby on Rails
- Sinatra
- Padrino

gem version 1.3.1 build failing dependencies out-of-date coverage 100% TRENDING docs

Dawn version 1.3 has 180 security checks loaded in its knowledge base. Most of them are CVE bulletins applying to gems or the ruby interpreter itself. There are also some check coming from Owasp Ruby on Rails cheatsheet.

An overall introduction

When you run Dawn on your code it parses your project Gemfile.lock looking for the gems used and it tries to detect the ruby interpreter version you are using or you declared in your ruby version management tool you like most (RVM, rbenv, ...).

Then the tool tries to detect the MVC framework your web application uses and it applies the security check accordingly. There checks designed to match rails application or checks that are appliable to any ruby code.

Dawn can also understand the code in your views and to backtrack sinks to spot cross site scripting and sql injections introduced by the code you actually wrote. In the project roadmap this is the code most of the future development effort will be focused on.

Dawn security scan result is a list of vulnerabilities with some mitigation actions you want to follow in order to build a stronger web application.

Installation

dawnscanner rubygem is cryptographically signed. To be sure the gem you install hasn’t been tampered, you must first add `paolo@dawnscanner.org` public signing certificate as trusted to your gem specific keyring.

```
$ gem cert --add <(curl -Ls https://raw.githubusercontent.com/thesp0nge/dawnscanner/master/certs/paolo_at_dawnsca
```

You can install latest Dawn version, fetching it from Rubygems by typing: RootFile

```
$ gem install dawnscanner -P MediumSecurity
```

The MediumSecurity trust profile will verify signed gems, but allow the installation of unsigned dependencies. This is necessary because not all of Dawn’s dependencies are signed, so we cannot use HighSecurity.

In order to install a release candidate version, the gem install command line is the following:

```
$ gem install dawnscanner --pre -P MediumSecurity
```

If you want to add dawn to your project Gemfile, you must add the following:

```
group :development do
  gem 'dawnscanner', :require=>false
end
```

InsertIntoFile

And then upgrade your bundle

```
$ bundle install
```

RootFile

You may want to build it from source, so you have to check it out from github first:

```
$ git clone https://github.com/thesp0nge/dawnscanner.git
$ cd dawnscanner
$ bundle install
$ rake install
```

RootFile

And the dawnscanner gem will be built in a pkg directory and then installed on your system. Please note that you have to manage dependencies on your own this way. It makes sense only if you want to hack the code or something like that.

Usage

You can start your code review with Dawn very easily. Simply tell the tool where the project root directory.

Underlying MVC framework is autodetected by Dawn using target Gemfile.lock file. If autodetect fails for some reason, the tool will complain about it and you have to specify if it's a rails, sinatra or padrino web application by hand.

Basic usage is to specify some optional command line option to fit best your needs, and to specify the target directory where your code is stored.

```
$ dawn [options] target
```

RootFile

In case of need, there is a quick command line option reference running `dawn -h` at your OS prompt.

```
$ dawn -h
Usage: dawn [options] target_directory

Examples:
  $ dawn a_sinatra_webapp_directory
  $ dawn -C the_rails_blog_engine
  $ dawn -C --json a_sinatra_webapp_directory
  $ dawn --ascii-tabular-report my_rails_blog_ecommerce
  $ dawn --html -F my_report.html my_rails_blog_ecommerce

   -r, --rails          force dawn to consider the target a rails application
   -s, --sinatra        force dawn to consider the target a sinatra application
   -p, --padrino        force dawn to consider the target a padrino application
   -G, --gem-lock       force dawn to scan only for vulnerabilities affecting dependencies in Gemfile.lock
   -a, --ascii-tabular-report   cause dawn to format findings using table in ascii art
   -j, --json                   cause dawn to format findings using json
   -C, --count-only             dawn will only count vulnerabilities (useful for scripts)
   -z, --exit-on-warn           dawn will return number of found vulnerabilities as exit code
```

RootFile

```
   -F, --file filename          tells dawn to write output to filename
   -c, --config-file filename   tells dawn to load configuration from filename

Disable security check family

       --disable-cve-bulletins  disable all CVE security checks
       --disable-code-quality   disable all code quality checks
       --disable-code-style     disable all code style checks
       --disable-owasp-ror-cheatsheet   disable all Owasp Ruby on Rails cheatsheet checks
       --disable-owasp-top-10           disable all Owasp Top 10 checks

Flags useful to query Dawn

       -S, --search-knowledge-base [check_name]   search check_name in the knowledge base
           --list-knowledge-base                  list knowledge-base content
           --list-known-families                  list security check families contained in dawn's knowledge base
           --list-known-framework                 list ruby MVC frameworks supported by dawn

Service flags

   -D, --debug                                enters dawn debug mode
   -V, --verbose                              the output will be more verbose
   -v, --version                              show version information
   -h, --help                                 show this help
```

Rake task

To include Dawn in your rake task list, you simply have to put this line in your `Rakefile`

```
require 'dawn/tasks'
```

InsertIntoFile

Then executing `$ rake -T` you will have a `dawn:run` task you want to execute.

```
$ rake -T
...
rake dawn:run                  # Execute dawnscanner on the current directory
...
```

Interacting with the knowledge base

You can dump all security checks in the knowledge base this way

```
$ dawn --list-knowledge-base
```

RootFile

Useful in scripts, you can use `--search-knowledge-base` or `-S` with as parameter the check name you want to see if it's implemented as a security control or not.

```
$ dawn -S CVE-2013-6421
07:59:30 [*] dawn v1.1.0 is starting up
CVE-2013-6421 found in knowledgebase.

$ dawn -S this_test_does_not_exist
08:02:17 [*] dawn v1.1.0 is starting up
this_test_does_not_exist not found in knowledgebase
```

Dawn security scan in action

As output, Dawn will put all security checks that are failed during the scan.

This the result of Codedake::Dawn running against a Sinatra 1.4.2 web application wrote for a talk I delivered in 2013 at Railsberry conference.

As you may see, Dawn first detects MVC running the application by looking at Gemfile.lock, than it discards all security checks not appliable to Sinatra (49 security checks, in version 1.0, especially designed for Ruby on Rails) and it applies them.

```
$ dawn ~/src/hacking/railsberry2013
18:40:27 [*] dawn v1.1.0 is starting up
18:40:27 [$] dawn: scanning /Users/thesp0nge/src/hacking/railsberry2013
18:40:27 [$] dawn: sinatra v1.4.2 detected
18:40:27 [$] dawn: applying all security checks
18:40:27 [$] dawn: 109 security checks applied - 0 security checks skipped
18:40:27 [$] dawn: 1 vulnerabilities found
18:40:27 [!] dawn: CVE-2013-1800 check failed
18:40:27 [$] dawn: Severity: high
18:40:27 [$] dawn: Priority: unknown
18:40:27 [$] dawn: Description: The crack gem 0.3.1 and earlier for Ruby does not properly restrict casts of stri
18:40:27 [$] dawn: Solution: Please use crack gem version 0.3.2 or above. Correct your gemfile
18:40:27 [$] dawn: Evidence:
18:40:27 [$] dawn:      Vulnerable crack gem version found: 0.3.1
18:40:27 [*] dawn is leaving
```

When you run Dawn on a web application with up to date dependencies, it's likely to return a friendly *no vulnerabilities found* message. Keep it up working that way!

This is Dawn running against a Padrino web application I wrote for a scorecard quiz game about application security. Italian language only. Sorry.

```
18:42:39 [*] dawn v1.1.0 is starting up
18:42:39 [$] dawn: scanning /Users/thesp0nge/src/CORE_PROJECTS/scorecard
18:42:39 [$] dawn: padrino v0.11.2 detected
18:42:39 [$] dawn: applying all security checks
18:42:39 [$] dawn: 109 security checks applied - 0 security checks skipped
18:42:39 [*] dawn: no vulnerabilities found.
18:42:39 [*] dawn is leaving
```

If you need a fancy HTML report about your scan, just ask it to Dawn with the `--html` flag used with the `--file` since I wanto to save the HTML to disk.

```
$ dawn /Users/thesp0nge/src/hacking/rt_first_app --html --file report.html

09:00:54 [*] dawn v1.1.0 is starting up
09:00:54 [*] dawn: report.html created (2952 bytes)
09:00:54 [*] dawn is leaving
```

Useful links

Project homepage: http://dawnscanner.org

Twitter profile: @dawnscanner

Github repository: https://github.com/thesp0nge/dawnscanner

Mailing list: https://groups.google.com/forum/#!forum/dawnscanner

Support us

Feedbacks are great and we really love to hear your voice.

If you're a proud dawnscanner user, if you find it useful, if you integrated it in your release process and if you want to openly support the project you can put your reference here. Just open an issue with a statement saying how do you feel the tool and your company logo if any.

More easily you can drop an email to paolo@dawnscanner.org sending a statement about your success story and I'll put on the website.

Thank you.

Thanks to

saten: first issue posted about a typo in the README

presidentbeef: for his outstanding work that inspired me creating dawn and for double check comparison matrix. Issue #2 is yours :)

marinerJB: for misc bug reports and further ideas

Matteo: for ideas on API and their usage with github.com hooks

Contribute to Dawn

Are you interested in contributing to Dawn project? Great, here is some very basic rules in order to make rocking pull requests.

First of all, I use the branching model described in this post. There are two major branches:

- master: it contains in every moment the code for the latest dawnscanner released gem. You can't make branches from here unless you're working on a bugfix.
- development: it contains the unstable code that is going to be the next dawnscanner realease. You start from here. Pick a task on the Roadmap.md and create a separated branch to work on your feature to. When you're ready (remember to include also spec files), submit your pull request. If the code will be fine, it will be merged into the development tree ready to be include in upcoming gem version.

No branch from master it would be analyzed unless they are related to bugfix. In this case, the branch name must be something like *issue*\#xx_description_

LICENSE