ATTACK DEFENSE

by PentesterAcademy

Name	CodeSake dawn: Finding bugs in Ruby Code
URL	https://www.attackdefense.com/challengedetails?cid=2157
Type	DevSecOps Basics: Static Application Security Testing

Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic. Also, note that the IP addresses and domain names might be different in your lab.

Challenge Description

The Codesake dawn tool is used to find vulnerabilities in Ruby applications.

A Kali CLI machine (kali-cli) is provided to the user with dawnscanner installed on it. The source code for two sample applications is provided in the home directory of the root user.

Objective: Use the dawnscanner utility to find vulnerabilities in the applications!

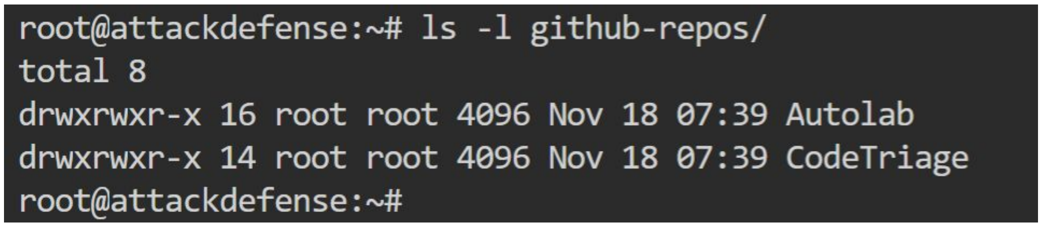
Instructions:

- The source code of applications is provided at /root/github-repos

Solution

Step 1: Check the provided applications.

Command: ls -l github-repos/



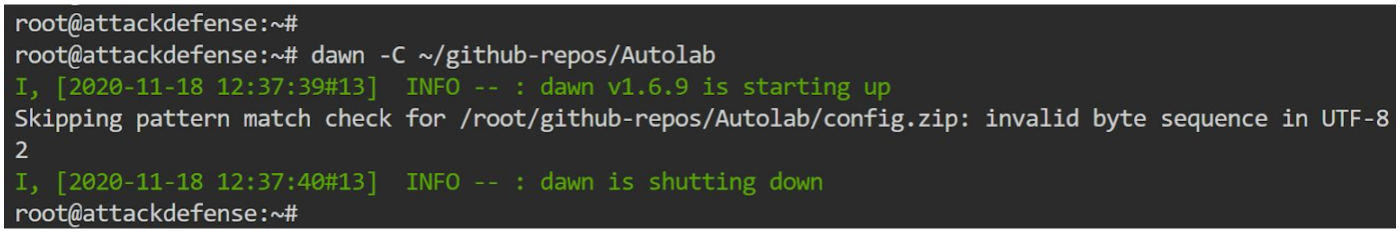
```
root@attackdefense:~# ls -l github-repos/
total 8
drwxrwxr-x 16 root root 4096 Nov 18 07:39 Autolab
drwxrwxr-x 14 root root 4096 Nov 18 07:39 CodeTriage
root@attackdefense:~#
```

We will take one example at a time and run the tool on that.

Example 1: Autolab

Step 1: Run the dawn tool on the Autolab directory to check the number of available vulnerabilities in the project.

Command: dawn -C ~/github-repos/Autolab

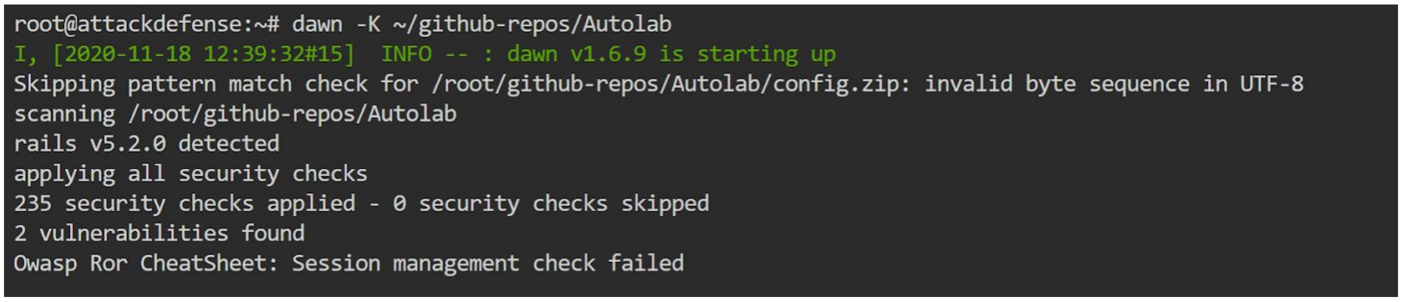


```
root@attackdefense:~#
root@attackdefense:~# dawn -C ~/github-repos/Autolab
I, [2020-11-18 12:37:39#13]  INFO -- : dawn v1.6.9 is starting up
Skipping pattern match check for /root/github-repos/Autolab/config.zip: invalid byte sequence in UTF-8
2
I, [2020-11-18 12:37:40#13]  INFO -- : dawn is shutting down
root@attackdefense:~#
```

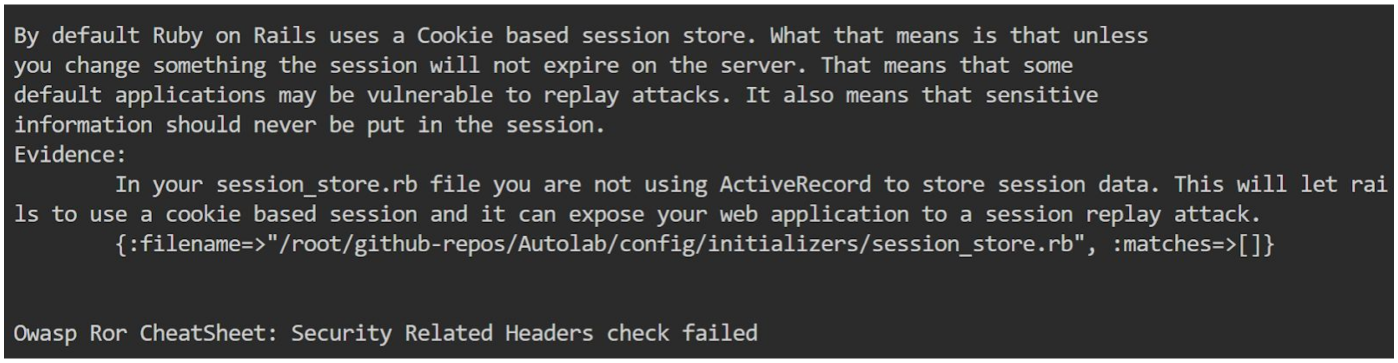
There are a total of 2 possible vulnerabilities found by the dawn scanner.

Step 2: Run the dawn tool in order to identify the two vulnerabilities in the project.

Command: dawn -K ~/github-repos/Autolab

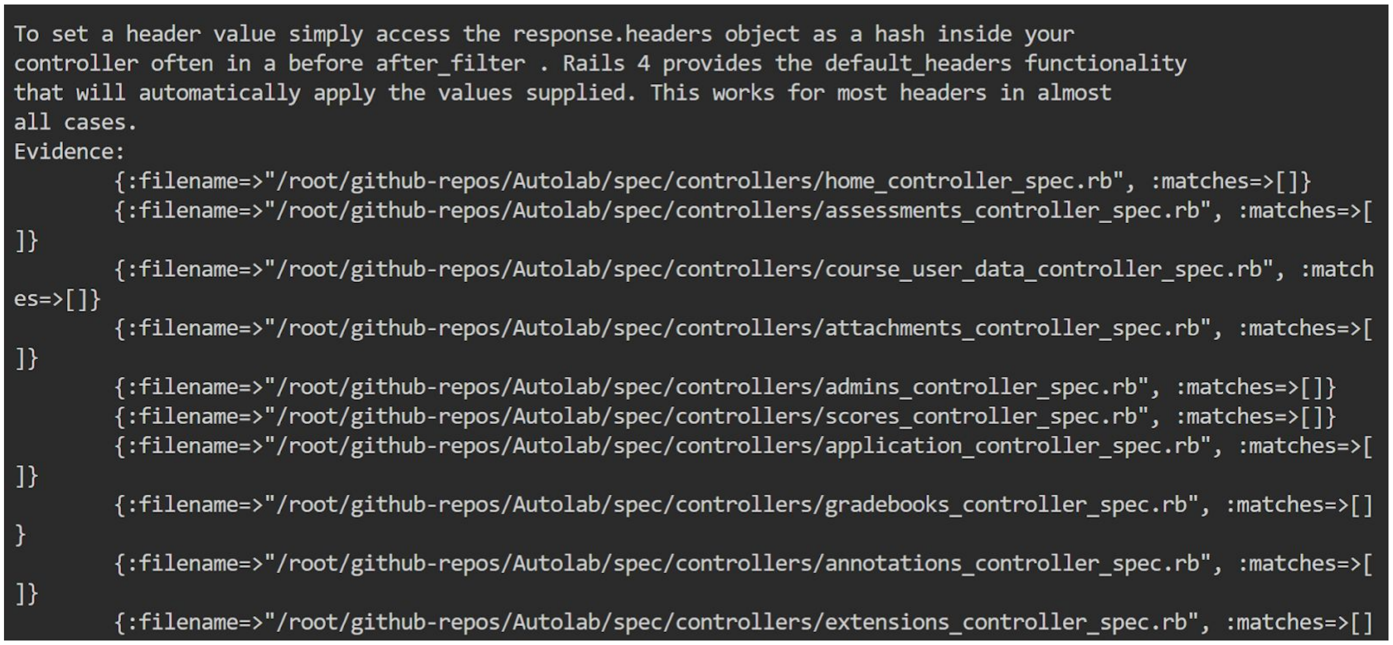


```
root@attackdefense:~# dawn -K ~/github-repos/Autolab
I, [2020-11-18 12:39:32#15]  INFO -- : dawn v1.6.9 is starting up
Skipping pattern match check for /root/github-repos/Autolab/config.zip: invalid byte sequence in UTF-8
scanning /root/github-repos/Autolab
rails v5.2.0 detected
applying all security checks
235 security checks applied - 0 security checks skipped
2 vulnerabilities found
Owasp Ror CheatSheet: Session management check failed
```

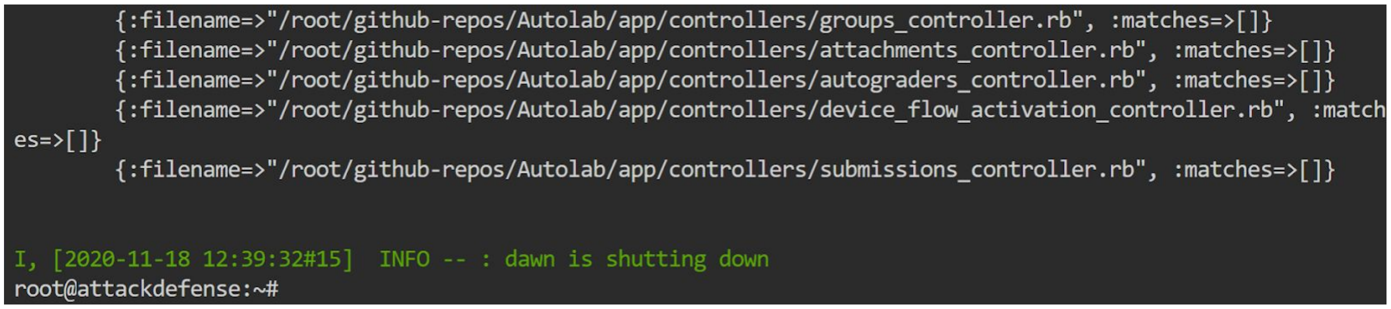


```
By default Ruby on Rails uses a Cookie based session store. What that means is that unless
you change something the session will not expire on the server. That means that some
default applications may be vulnerable to replay attacks. It also means that sensitive
information should never be put in the session.
Evidence:
        In your session_store.rb file you are not using ActiveRecord to store session data. This will let rai
ls to use a cookie based session and it can expose your web application to a session replay attack.
        {:filename=>"/root/github-repos/Autolab/config/initializers/session_store.rb", :matches=>[]}


Owasp Ror CheatSheet: Security Related Headers check failed
```



```
To set a header value simply access the response.headers object as a hash inside your
controller often in a before after_filter . Rails 4 provides the default_headers functionality
that will automatically apply the values supplied. This works for most headers in almost
all cases.
Evidence:
        {:filename=>"/root/github-repos/Autolab/spec/controllers/home_controller_spec.rb", :matches=>[]}
        {:filename=>"/root/github-repos/Autolab/spec/controllers/assessments_controller_spec.rb", :matches=>[
]}
        {:filename=>"/root/github-repos/Autolab/spec/controllers/course_user_data_controller_spec.rb", :match
es=>[]}
        {:filename=>"/root/github-repos/Autolab/spec/controllers/attachments_controller_spec.rb", :matches=>[
]}
        {:filename=>"/root/github-repos/Autolab/spec/controllers/admins_controller_spec.rb", :matches=>[]}
        {:filename=>"/root/github-repos/Autolab/spec/controllers/scores_controller_spec.rb", :matches=>[]}
        {:filename=>"/root/github-repos/Autolab/spec/controllers/application_controller_spec.rb", :matches=>[
]}
        {:filename=>"/root/github-repos/Autolab/spec/controllers/gradebooks_controller_spec.rb", :matches=>[]
}
        {:filename=>"/root/github-repos/Autolab/spec/controllers/annotations_controller_spec.rb", :matches=>[
]}
        {:filename=>"/root/github-repos/Autolab/spec/controllers/extensions_controller_spec.rb", :matches=>[]
```



```
        {:filename=>"/root/github-repos/Autolab/app/controllers/groups_controller.rb", :matches=>[]}
        {:filename=>"/root/github-repos/Autolab/app/controllers/attachments_controller.rb", :matches=>[]}
        {:filename=>"/root/github-repos/Autolab/app/controllers/autograders_controller.rb", :matches=>[]}
        {:filename=>"/root/github-repos/Autolab/app/controllers/device_flow_activation_controller.rb", :match
es=>[]}
        {:filename=>"/root/github-repos/Autolab/app/controllers/submissions_controller.rb", :matches=>[]}


I, [2020-11-18 12:39:32#15]  INFO -- : dawn is shutting down
root@attackdefense:~#
```

The issues found in the application are present in the specified file path.

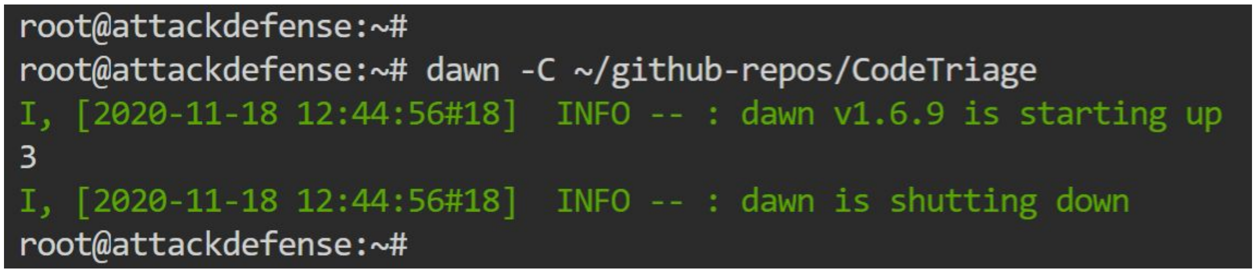
Issues Detected:

- Session management check failed
- Security Related Headers check failed

Example 2: Code Triage

Step 1: Run the dawn tool on the CodeTriage directory to check the number of available vulnerabilities in the project.

Command: dawn -C ~/github-repos/CodeTriage

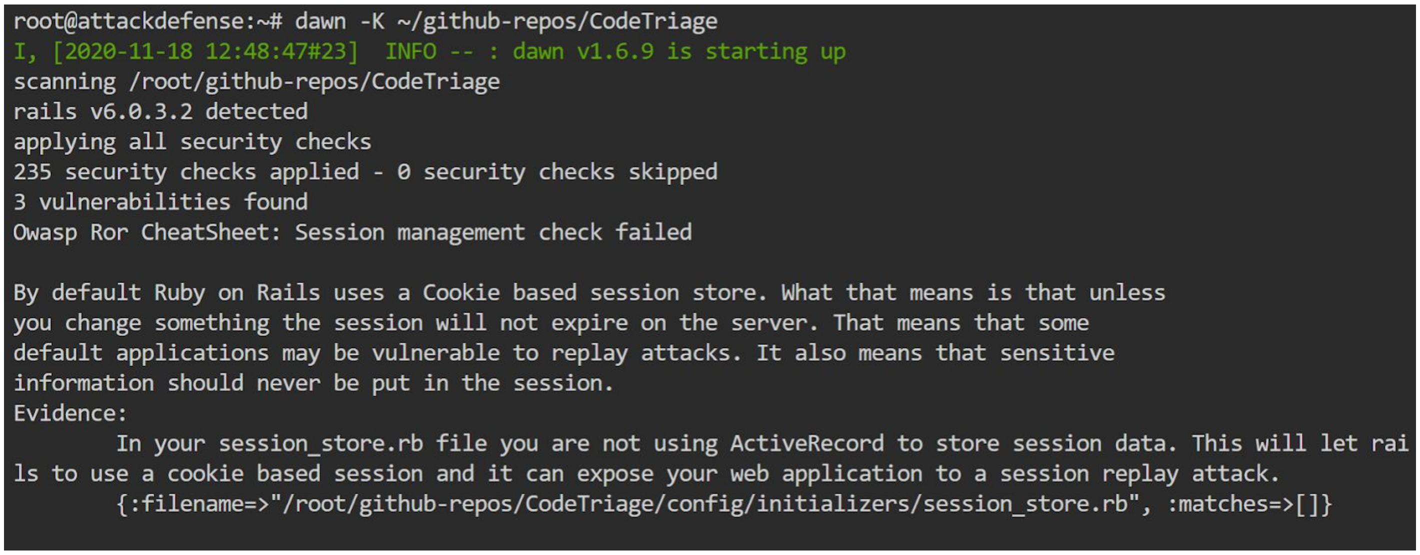


```
root@attackdefense:~#
root@attackdefense:~# dawn -C ~/github-repos/CodeTriage
I, [2020-11-18 12:44:56#18]  INFO -- : dawn v1.6.9 is starting up
3
I, [2020-11-18 12:44:56#18]  INFO -- : dawn is shutting down
root@attackdefense:~#
```

There are a total of 3 possible vulnerabilities found by the dawn scanner.

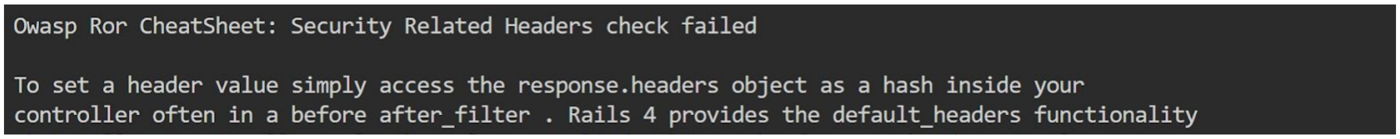
Step 2: Run the dawn tool in order to identify the two vulnerabilities in the project.

Command: dawn -K ~/github-repos/CodeTriage



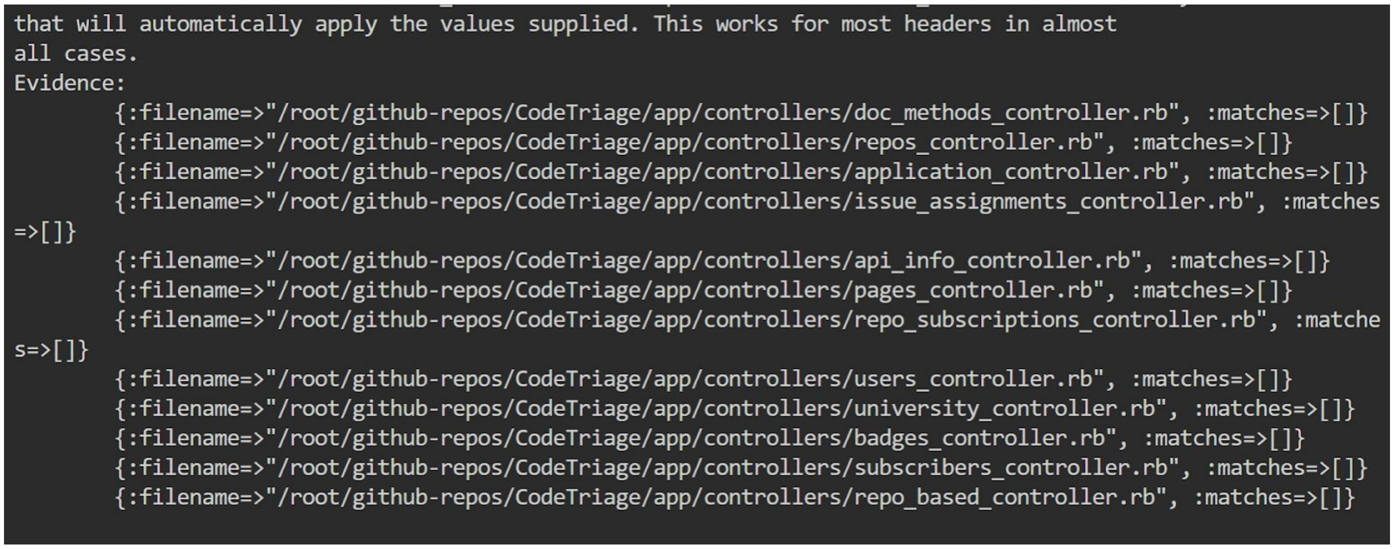
```
root@attackdefense:~# dawn -K ~/github-repos/CodeTriage
I, [2020-11-18 12:48:47#23]  INFO -- : dawn v1.6.9 is starting up
scanning /root/github-repos/CodeTriage
rails v6.0.3.2 detected
applying all security checks
235 security checks applied - 0 security checks skipped
3 vulnerabilities found
Owasp Ror CheatSheet: Session management check failed

By default Ruby on Rails uses a Cookie based session store. What that means is that unless
you change something the session will not expire on the server. That means that some
default applications may be vulnerable to replay attacks. It also means that sensitive
information should never be put in the session.
Evidence:
        In your session_store.rb file you are not using ActiveRecord to store session data. This will let rai
ls to use a cookie based session and it can expose your web application to a session replay attack.
        {:filename=>"/root/github-repos/CodeTriage/config/initializers/session_store.rb", :matches=>[]}
```

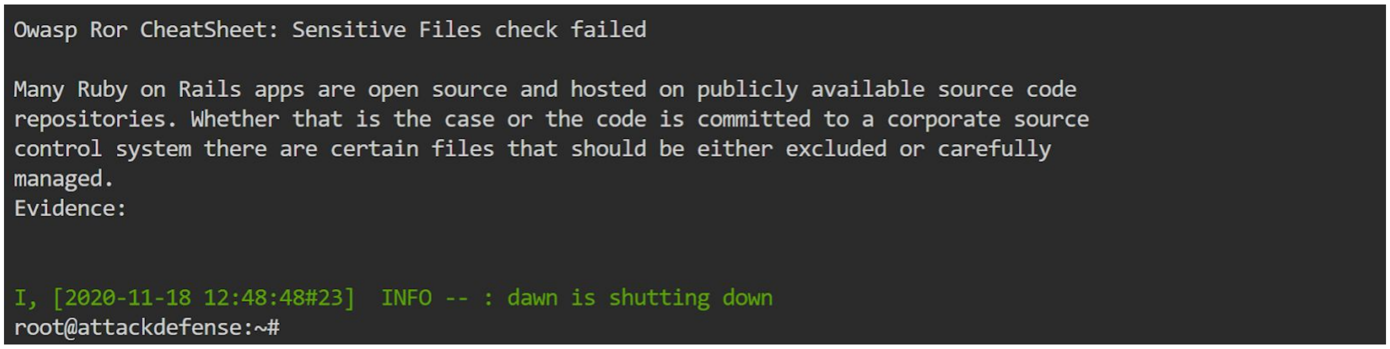


```
Owasp Ror CheatSheet: Security Related Headers check failed

To set a header value simply access the response.headers object as a hash inside your
controller often in a before after_filter . Rails 4 provides the default_headers functionality
```



```
that will automatically apply the values supplied. This works for most headers in almost
all cases.
Evidence:
        {:filename=>"/root/github-repos/CodeTriage/app/controllers/doc_methods_controller.rb", :matches=>[]}
        {:filename=>"/root/github-repos/CodeTriage/app/controllers/repos_controller.rb", :matches=>[]}
        {:filename=>"/root/github-repos/CodeTriage/app/controllers/application_controller.rb", :matches=>[]}
        {:filename=>"/root/github-repos/CodeTriage/app/controllers/issue_assignments_controller.rb", :matches
=>[]}
        {:filename=>"/root/github-repos/CodeTriage/app/controllers/api_info_controller.rb", :matches=>[]}
        {:filename=>"/root/github-repos/CodeTriage/app/controllers/pages_controller.rb", :matches=>[]}
        {:filename=>"/root/github-repos/CodeTriage/app/controllers/repo_subscriptions_controller.rb", :matche
s=>[]}
        {:filename=>"/root/github-repos/CodeTriage/app/controllers/users_controller.rb", :matches=>[]}
        {:filename=>"/root/github-repos/CodeTriage/app/controllers/university_controller.rb", :matches=>[]}
        {:filename=>"/root/github-repos/CodeTriage/app/controllers/badges_controller.rb", :matches=>[]}
        {:filename=>"/root/github-repos/CodeTriage/app/controllers/subscribers_controller.rb", :matches=>[]}
        {:filename=>"/root/github-repos/CodeTriage/app/controllers/repo_based_controller.rb", :matches=>[]}
```



```
Owasp Ror CheatSheet: Sensitive Files check failed

Many Ruby on Rails apps are open source and hosted on publicly available source code
repositories. Whether that is the case or the code is committed to a corporate source
control system there are certain files that should be either excluded or carefully
managed.
Evidence:


I, [2020-11-18 12:48:48#23]  INFO -- : dawn is shutting down
root@attackdefense:~#
```

The issues found in the application are present in the specified file path.

Issues Detected:

- Session management check failed
- Security Related Headers check failed
- Sensitive Files check failed

Learnings

Perform Static Code Analysis using the CodeSake dawn tool.

References:

- Autolab (https://github.com/autolab/Autolab.git)
- Code Triage (https://github.com/codetriage/CodeTriage.git)