

# Secure By Defaults with Semgrep



# Agenda

1. Understanding Secure by Defaults
2. Introduction to Semgrep
3. Semgrep Implementation Process
4. Semgrep Enforcement Process
5. Outcomes and Performance

# Secure by Defaults



- A holistic approach to solving security problems at the root cause
- Acts as a scale to reduce the overall harm to a system or type of component
- Secure by Default covers the long-term technical effort to ensure that the right security primitives are built into software and hardware
- It mainly covers the basic principles outlined by OWASP like HTTPS flag, not using weak hashing functions etc.,
- Takes an average of 30 hours to fix a critical vulnerability when not secured by default

OWASP TOP 10	Secure By Default Rules
Injection	<ul style="list-style-type: none"> <li>• Potential Command injections</li> <li>• Potential XPath injection</li> <li>• Potential SQL injection</li> <li>• Potential code injection</li> <li>• Potential template injection</li> </ul>
Broken Authentication	<ul style="list-style-type: none"> <li>• Missing Anti-CSRF protection</li> </ul>
Sensitive Data Exposure	<ul style="list-style-type: none"> <li>• Leaking of hardcoded credentials</li> </ul>
XML External Entities (XXE)	<ul style="list-style-type: none"> <li>• Potential XXE attack</li> </ul>
Broken Access Control	<ul style="list-style-type: none"> <li>• URL redirects</li> <li>• Weak SSL Contexts</li> </ul>
Security Misconfiguration	<ul style="list-style-type: none"> <li>• Persistent Cookie usage</li> <li>• Potential CORS attack</li> <li>• HTTP secure flag not set</li> <li>• Cookie without HTTPOnly flag</li> </ul>
Cross-Site Scripting (XSS)	<ul style="list-style-type: none"> <li>• Weak XSS protection</li> <li>• Potential XSS attack</li> </ul>
Insecure Deserialization	<ul style="list-style-type: none"> <li>• Struts and Spring files disclosure</li> <li>• HTTP parameters pollution</li> <li>• Potential HTTP Response splitting</li> <li>• Potential deserialization vulnerabilities</li> </ul>
Using Components with Known Vulnerabilities	<ul style="list-style-type: none"> <li>• Weak hash functions</li> <li>• Weak encryptions</li> </ul>
Insufficient Logging & Monitoring	<ul style="list-style-type: none"> <li>• Improper error handling</li> </ul>

# Semgrep

<https://semgrep.dev>

Semgrep is a  
lightweight,  
offline, open-source,  
static analysis tool

It supports multiple  
languages like Go, Java,  
Python, Ruby and many  
more

Takes lesser time to scan  
and reduces false  
positives

You can use community  
provided rules or build  
your custom rules to  
find vulnerabilities.

Semgrep is also  
recommended by  
OWASP

# Semgrep vs SAST

- Custom rulesets based on regular expression
- Fewer false positives
- Faster scans
- Mostly up to date

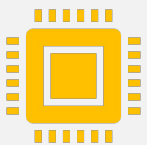
# What are the Benefits?



Helps developers reduce the repetitive errors



Custom rulesets which can be tailored to the languages

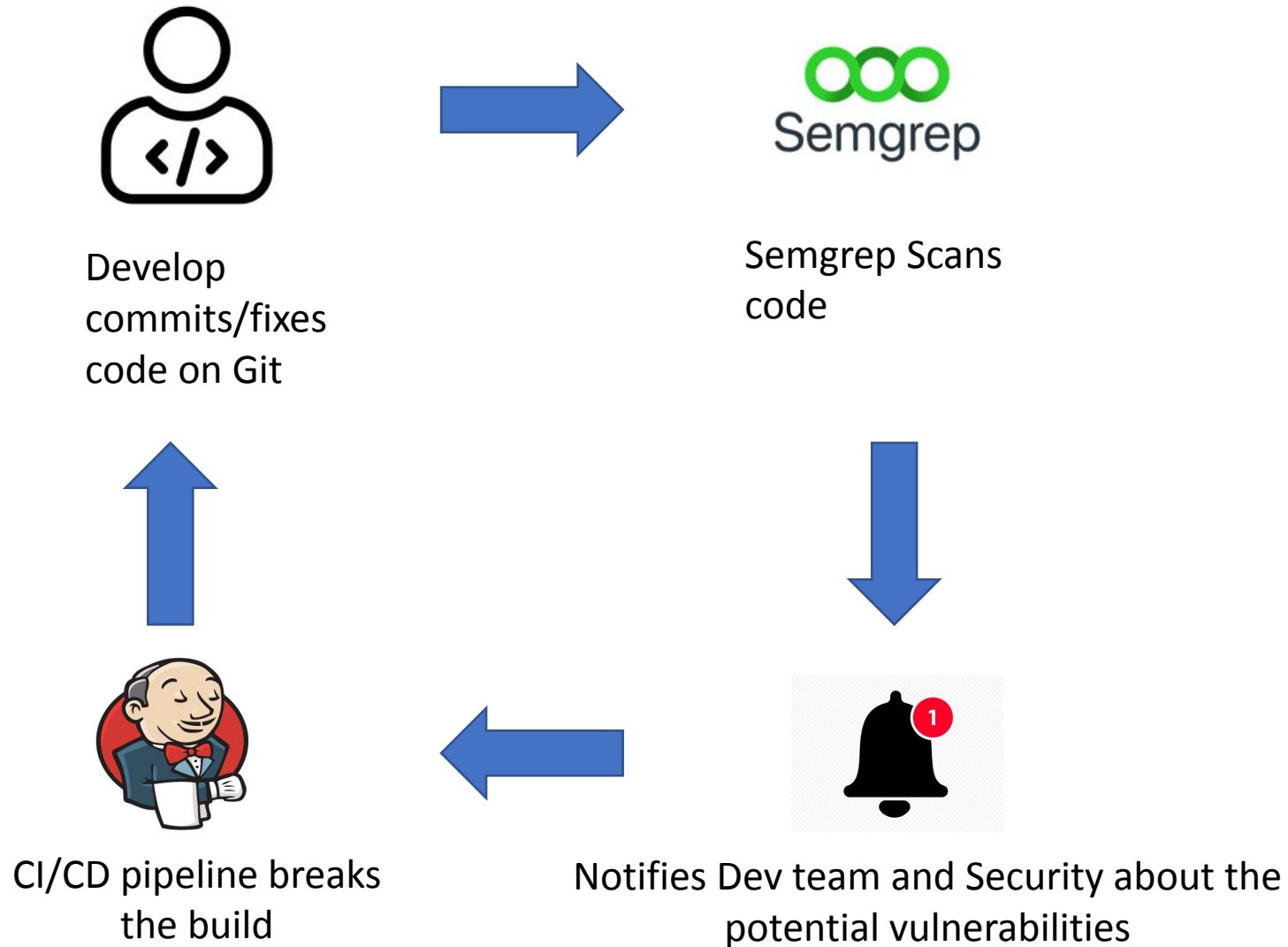


Implemented on the new code which reduces burden for developers in the future



Focus on remediating vulnerability classes

# High level Workflow





# IMPLEMENTATION PROCESS

# Script to Manage Commits

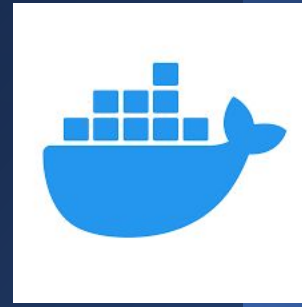
- Trigger for every commit on GitHub
- Commit Data sent using webhook created per repo for automated response
- Using the Git APIs to get the content of the file/files that was modified
- Formatting the contents to get the code that was modified
- Create a temp file to run Semgrep
- Capture the output for notifying developers
- Format the captured output to send alerts through email and add comments on the PR

# Custom Rulesets

- Community rulesets are available based on the language
- Rulesets can be tailored based on the requirements and goals
- Rulesets are written in YAML
- Primarily uses regular expression to detect patterns
- Here is an example of the custom ruleset
- Link to the community rules <https://semgrep.dev/r>

```
- id: missing_httponly_flag
  pattern: $COOKIE.setHttpOnly(false);
  languages: [java]
  message: Missing HTTP only flag at this line
  fix: $COOKIE.setHttpOnly(true);
  severity: WARNING
```

# Containerizing the Script



- Scaling up to manage the number outgoing commits every minute
- Makes debugging easier with logging
- Helps in modifying the script to add functionality
- Load balancing is efficient since the script needs to be up and listening
- Isolated environments

# Webhook Creation

- Using URL from the deployed script to create webhooks
- Webhooks can be created based on the language
- They in turn trigger the script which continues the cycle
- This must be added to every repo to get the alerts

## Webhooks / Manage webhook

Settings

Recent Deliveries

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).

Payload URL \*

Content type

Secret

If you've lost or forgotten this secret, you can change it, but be aware that any integrations using this secret will need to be updated. — [Change Secret](#)

Which events would you like to trigger this webhook?

- ☒ Just the push event.
- ☐ Send me **everything**.
- ☐ Let me select individual events.

☒ Active

We will deliver event details when this hook is triggered.

Update webhook

Delete webhook

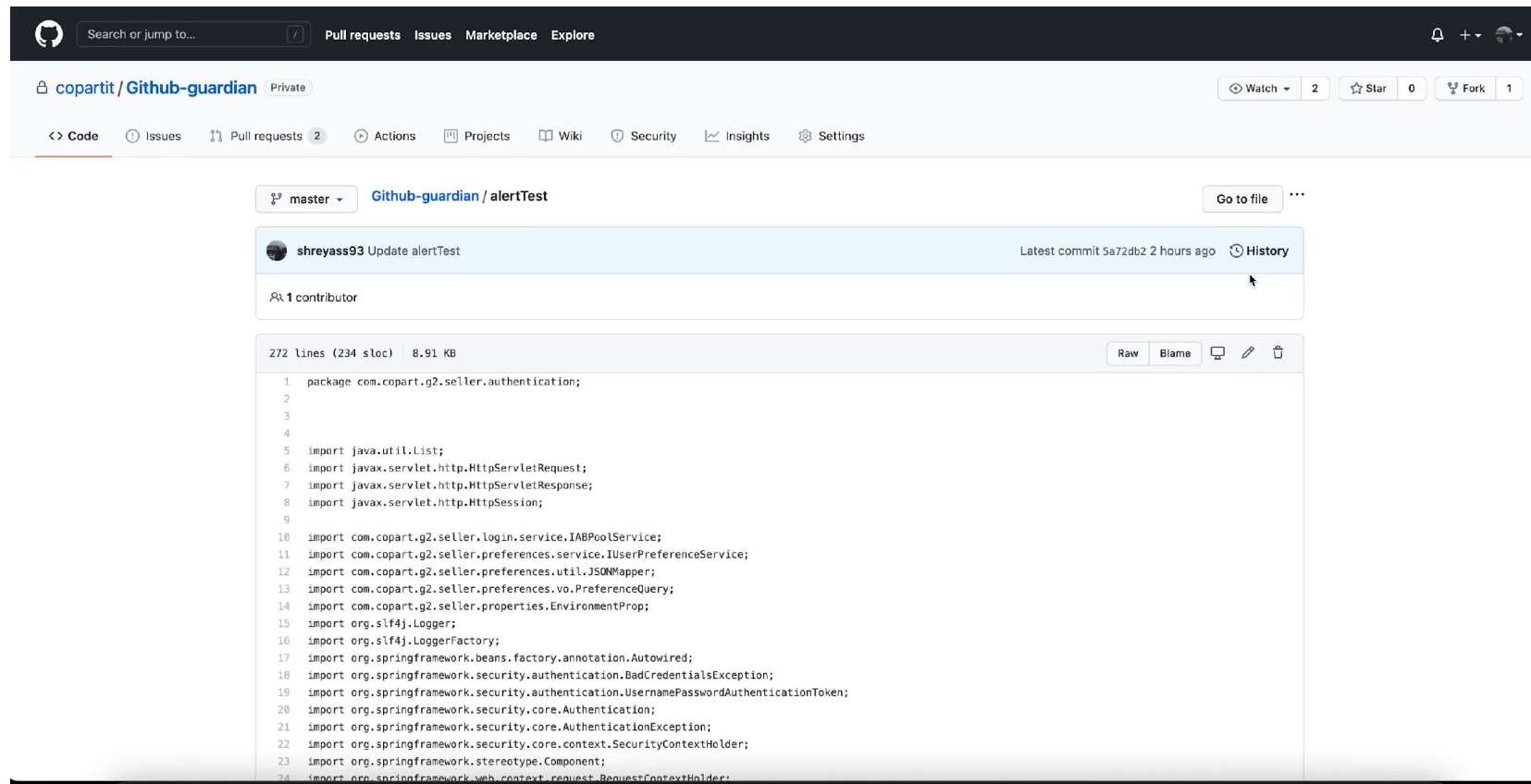
# CI/CD Integration

- Semgrep offers CLI support which can be directly integrated with the pipeline
- Add the semgrep command as part of requirements within pipeline automation
- Semgrep can be added as status check for the Pull Requests
- Vulnerable code won't move forward in pipeline unless they are fixed

This screenshot shows a GitHub Pull Request interface. At the top, there are tabs for Conversation (0), Commits (1), Checks (0), and Files changed (1). A comment from user 'shreyass93' is visible, stating 'No description provided.' Below the comment, a commit titled 'Update BackendSupport.java' is shown with a 'Verified' status and a green checkmark. A message indicates: 'Add more commits by pushing to the shreyass93-patch-2 branch on copartit/G2-Member.' The bottom section displays a green checkmark and the text 'All checks have passed' with '1 successful check'. Below this, two specific checks are listed: 'ci/jenkins/semgrep' (This commit looks good) and 'This branch has no conflicts with the base branch' (Merging can be performed automatically.). A green 'Merge pull request' button is at the bottom, with a note: 'You can also open this in GitHub Desktop or view command line instructions.'

This screenshot shows a GitHub Pull Request interface with failed checks. At the top, a message from user 'shreyass93' states 'reopened this 2 minutes ago'. A message indicates: 'Add more commits by pushing to the shreyass93-patch-1 branch on copartit/G2-Member.' The bottom section displays a red 'X' icon and the text 'All checks have failed' with '1 errored check'. Below this, two specific checks are listed: 'ci/jenkins/semgrep' (This commit cannot be built) and 'This branch is out-of-date with the base branch' (Merge the latest changes from master into this branch.). A red 'Merge pull request' button is at the bottom, with a note: 'You can also open this in GitHub Desktop or view command line instructions.'

# Snippet of our Implementation



The screenshot displays the GitHub interface for the repository 'copartit / Github-guardian'. The repository is marked as 'Private'. The navigation bar includes links for 'Code', 'Issues', 'Pull requests' (with a count of 2), 'Actions', 'Projects', 'Wiki', 'Security', 'Insights', and 'Settings'. On the right, there are buttons for 'Watch' (2), 'Star' (0), and 'Fork' (1). The main content area shows the 'alertTest' file under the 'master' branch. A commit by 'shreyass93' is highlighted, with the message 'Update alertTest' and a timestamp of 'Latest commit 5a72db2 2 hours ago'. Below the commit information, it states '1 contributor'. The file itself is 272 lines long (234 sloc) and 8.91 KB in size. The code is a Java file with the following content:

```
1 package com.copart.g2.seller.authentication;
2
3
4
5 import java.util.List;
6 import javax.servlet.http.HttpServletRequest;
7 import javax.servlet.http.HttpServletResponse;
8 import javax.servlet.http.HttpSession;
9
10 import com.copart.g2.seller.login.service.IABPoolService;
11 import com.copart.g2.seller.preferences.service.IUserPreferenceService;
12 import com.copart.g2.seller.preferences.util.JSONMapper;
13 import com.copart.g2.seller.preferences.vo.PreferenceQuery;
14 import com.copart.g2.seller.properties.EnvironmentProp;
15 import org.slf4j.Logger;
16 import org.slf4j.LoggerFactory;
17 import org.springframework.beans.factory.annotation.Autowired;
18 import org.springframework.security.authentication.BadCredentialsException;
19 import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
20 import org.springframework.security.core.Authentication;
21 import org.springframework.security.core.AuthenticationException;
22 import org.springframework.security.core.context.SecurityContextHolder;
23 import org.springframework.stereotype.Component;
24 import org.springframework.web.context.request.RequestContextHolder;
```



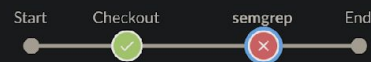
# Enforcing Secure by Default

- Semgrep is integrated to our Build tool (jenkins)
- Semgrep scans the code as part of prebuild process

📁 Jenkins	Update semgrep.groovy
📁 src	Merge branch 'master' into mds-develop-release-3.26
📄 README.md	Update README.md
📄 owner.md	Create owner.md
📄 pom.xml	Purge CSS plugin

```
    }
    steps {
        script {
            sh "cd ${app_workspace}"
            def target_commit = sh(returnStdout: true, script: "git rev-parse origin/${env.CHANGE_TARGET}").trim()
            // print files changed
            sh "git diff --diff-filter=ACM --name-only ${target_commit} HEAD"
            sh "git diff --diff-filter=ACM --name-only ${target_commit} HEAD | xargs docker run -v ${env.workspace}:/src returntocorp/semgrep:latest --config https://semgrep.dev/s/shreyass93:copartjavarules --error --lang
java --verbose"
        }
    }
}
```





semgrep - 19s

✓	> Maven 3.3.3 — Use a tool from a predefined Tool Installation	<1s
✓	> Fetches the environment variables for a given tool in a list of 'FOO=bar' strings suitable for the withEnv step.	<1s
✓	> OpenJDK 11.0.6 — Use a tool from a predefined Tool Installation	<1s
✓	> Fetches the environment variables for a given tool in a list of 'FOO=bar' strings suitable for the withEnv step.	<1s
✓	> cd . — Shell Script	<1s
✓	> git rev-parse origin/mds-develop-release-3.26 — Shell Script	<1s
✓	> git diff --diff-filter=ACM --name-only b9e556403f2de89f69fcd652c238df06c237d89f HEAD — Shell Script	<1s
✗	> git diff --diff-filter=ACM --name-only b9e556403f2de89f69fcd652c238df06c237d89f HEAD   xargs docker run -v /opt/copart/node/workspace/ecops-semgrep_G2-Member_PR-10631/src:returntocorp/semgrep:latest --config https://semgre... — Shell Script	18s
✓	> Delete workspace when build is done	<1s
✓	> Recursively delete the current directory from the workspace	<1s

```
[2021-09-29T20:08:46.652Z] severity:warning rule:improper_error_handling: Error handling is also important from an intrusion detection perspective. Certain attacks against your application may trigger errors whi
can help detect attacks in progress
[2021-09-29T20:08:46.652Z] autofix: log.error(msg, exception);
[2021-09-29T20:08:46.652Z] 123:             e.printStackTrace();
[2021-09-29T20:08:46.652Z]
[2021-09-29T20:08:46.652Z] src/main/java/com/copart/g2/member/exception/handlers/webservice/CreatePasswordWSEExceptionHandler.java
[2021-09-29T20:08:46.652Z] severity:warning rule:improper_error_handling: Error handling is also important from an intrusion detection perspective. Certain attacks against your application may trigger errors whi
can help detect attacks in progress
[2021-09-29T20:08:46.652Z] autofix: log.error(msg, exception);
[2021-09-29T20:08:46.652Z] 48:             e.printStackTrace();
[2021-09-29T20:08:46.652Z]
[2021-09-29T20:08:46.652Z] src/main/java/com/copart/g2/member/exception/handlers/webservice/SecurityQuestionsWSEExceptionHandler.java
[2021-09-29T20:08:46.653Z] severity:warning rule:improper_error_handling: Error handling is also important from an intrusion detection perspective. Certain attacks against your application may trigger errors whi
can help detect attacks in progress
[2021-09-29T20:08:46.653Z] autofix: log.error(msg, exception);
[2021-09-29T20:08:46.653Z] 49:             e.printStackTrace();
[2021-09-29T20:08:46.653Z]
```

# Outcomes and Performance

- 45% reduction in tickets within past 6 months
- About 60 potential vulnerabilities blocked in the past 6 months
- Early detection and rectification
- Better awareness among the new developers
- Secure coding practices
- Processes close to 1800 lines of code per second

# What are the Challenges?



CONTEXT OF THE  
COMMITTED CODE



MANAGING RULESETS OF  
MULTIPLE LANGUAGES



CREATING AND ADDING  
WEBHOOKS



CREATING AWARENESS  
AMONG DEV TEAMS

# What are the Gaps?

- Supports only the major languages
- Business logic flaws cannot be handled
- Affects the pipeline when Semgrep is down
- Pipeline automation script must be added/updated on every repo

# Next steps

## Adding Rulesets

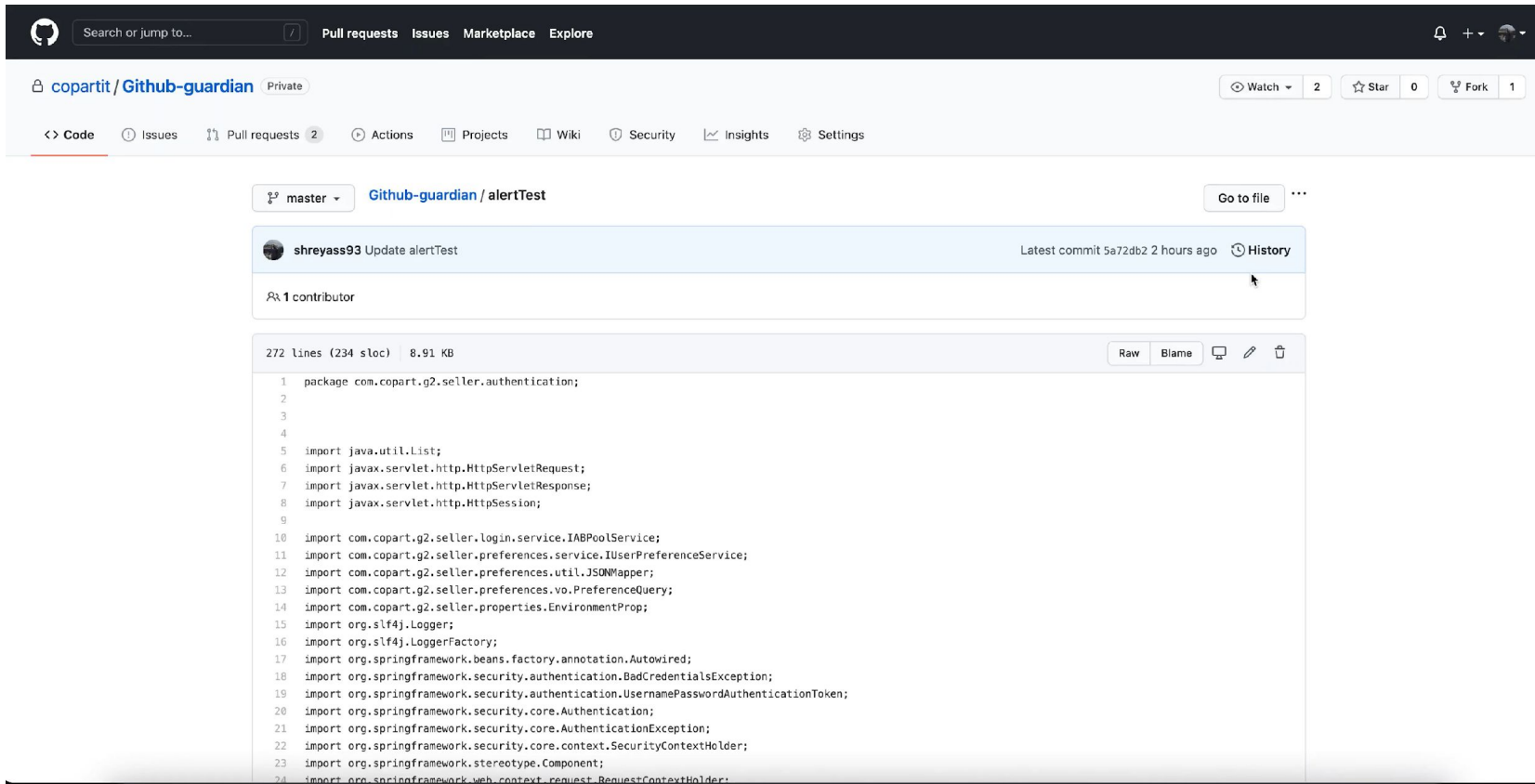
- Replace HTTP URLs with HTTPS on new commits
- Find unauthenticated routes on new commits
- Disallow URLs with usernames and passwords
- Extension to other languages

Questions



# How is it Implemented?

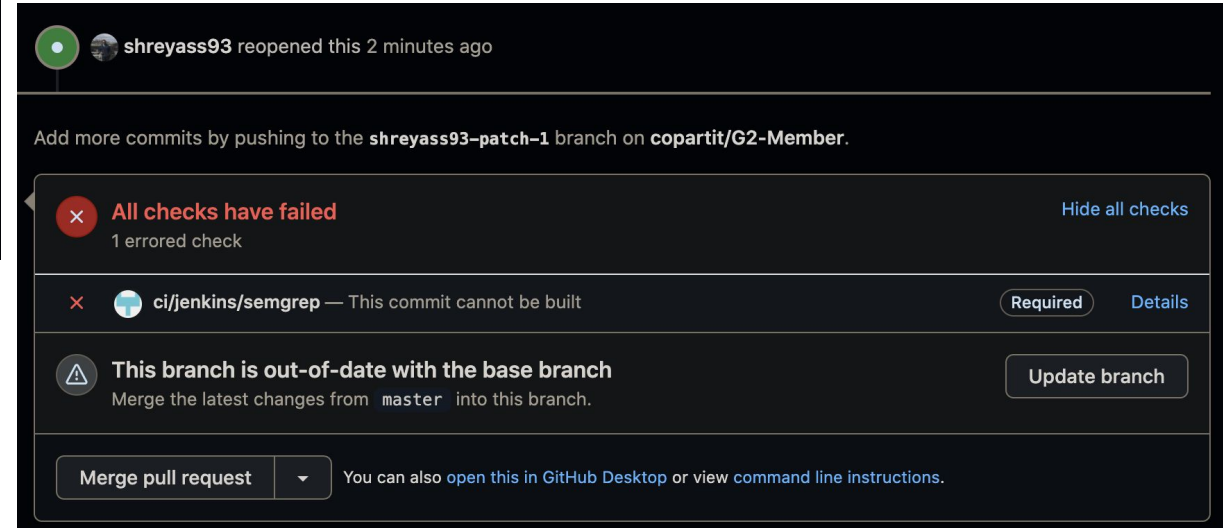
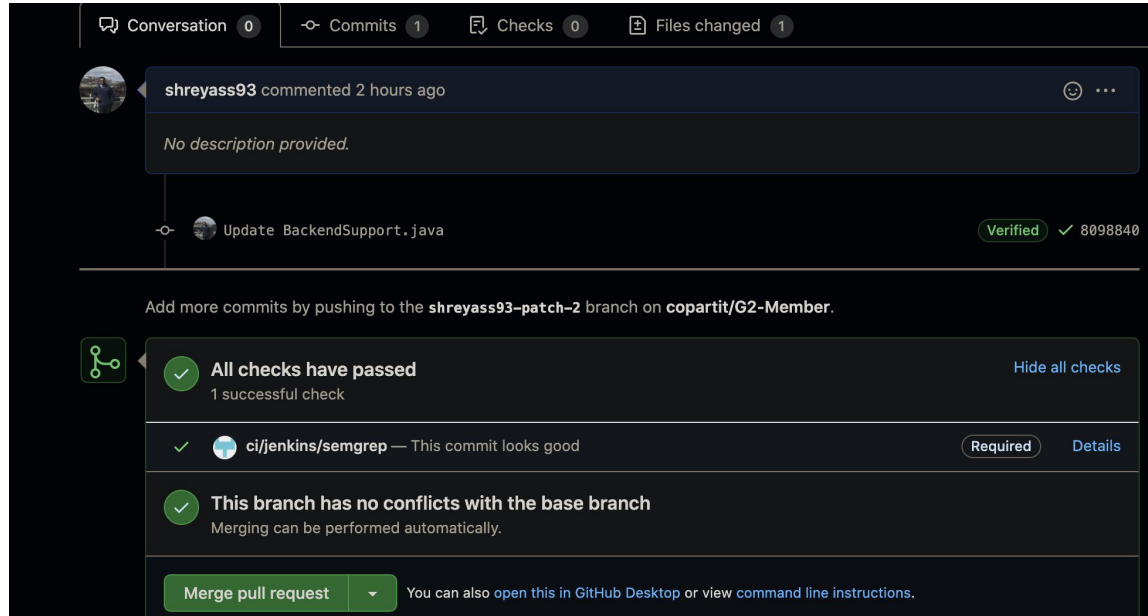
- Semgrep is initially implemented with alert mode



The screenshot shows the GitHub interface for the repository `copartit/Github-guardian`. The file `alertTest` is selected, showing its commit history and code. The code is a Java file with 272 lines (234 slots) and 8.91 KB. It contains several import statements and package declarations.

```
1 package com.copart.g2.seller.authentication;
2
3
4
5 import java.util.List;
6 import javax.servlet.http.HttpServletRequest;
7 import javax.servlet.http.HttpServletResponse;
8 import javax.servlet.http.HttpSession;
9
10 import com.copart.g2.seller.login.service.IABPoolService;
11 import com.copart.g2.seller.preferences.service.IUserPreferenceService;
12 import com.copart.g2.seller.preferences.util.JSONMapper;
13 import com.copart.g2.seller.preferences.vo.PreferenceQuery;
14 import com.copart.g2.seller.properties.EnvironmentProp;
15 import org.slf4j.Logger;
16 import org.slf4j.LoggerFactory;
17 import org.springframework.beans.factory.annotation.Autowired;
18 import org.springframework.security.authentication.BadCredentialsException;
19 import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
20 import org.springframework.security.core.Authentication;
21 import org.springframework.security.core.AuthenticationException;
22 import org.springframework.security.core.context.SecurityContextHolder;
23 import org.springframework.stereotype.Component;
24 import org.springframework.web.context.request.RequestContextHolder;
```

- Semgrep is enforced as a PR check which stops the code from merging if there are any errors
- It is only done on master branches for now





# Why do we need Semgrep?

- Semgrep helps developers reduce the repetitive errors by alerting them early in the pipeline
- Custom rulesets which are tailor made to Copart repos and are also based on previous tickets found within the repo
- Semgrep is only implemented on the new outgoing code which means it reduces burden for developers in the future
- Therefore, Semgrep is way efficient and faster than it's counterparts like Checkmarx and Sonarqube and helps mitigate vulnerability classes

Semgrep is a lightweight, offline, open-source, static analysis tool

It supports multiple languages like Go, Java, Python, Ruby and many more

Takes lesser time to scan and reduces false positives

You can use community provided rules or build your custom rules to find vulnerabilities.