

Semgrep

Adaptive Noise Cancelling Meets Code Scanning



Solutions Engineer, Strategic Accounts

David Whitlow

Recovering:

IT Security and Systems Admin

Daylight:

Improving AppSec with Semgrep

Moonlight:

2x Disney Princesses

7x League Kickball Champion



Semgrep AppSec Platform



Assistant

AI-powered context analysis and remediation guidance



Code

scan code you write (SAST)



Supply Chain

scan dependencies (SCA)



Secrets

scan for exposed credentials

Pro Engine

proprietary engine for 5-10x more true positives

Managed Scanning

1-click rollout of Semgrep across projects



Semgrep AppSec Platform

platform for shift left + secure guardrails using proprietary Semgrep engine and rules



Problems with Code Scanning

- Understanding code is hard
- Product bugs have direct customer impact
- Bad tools and poor implementations are both mind-numbing and soul-crushing
- Developers already have a day job



Semgrep Priority Findings



Semgrep-Demo

Dashboard

Projects

Code 8.7K

Secrets 67

Supply Chain 704

Rules

Policies

Editor

Registry

Feedback

Settings

Docs

Help

Code

Projects and branches 2
All projects with primary branches

Teams
All teams

Tags
internal x

Status
Open (22)

Severity
High Medium Low

Confidence
High Medium Low

Pro findings only

Category
All categories

Component
All components

Recommendation
Fix Ignore

Action

Group by Rule

All time

Sort by highest severity

Analyze (0)

Triage (0)

22 Matching Findings

1

tainted-ssrf-spring-add

Pro Security </> Java

Untrusted input might be used to build an HTTP request, which can lead to a Server-side request forgery (SSRF) vulnerability. SSRF allows an attacker to send crafted requests from the server side to other internal or external systems. SSRF can lead to unauthorized

Show more

5mo

src/main/java/com/.../LinkList.java:16

vulnado master Details

1

tainted-system-command

Pro Security </> Java

Untrusted input might be injected into a command executed by the application, which can lead to a command injection vulnerability. An attacker can execute arbitrary commands, potentially gaining complete control of the system. To prevent this vulnerability, avoid

Show more

5mo

src/main/java/com/.../Cowsay.java:11

vulnado master Details

4

regexp-redos

Pro Security </> JavaScript

Detected \$REQ argument enters calls to RegExp. This could lead to a Regular Expression Denial of Service (ReDoS) through catastrophic backtracking. If the input is attacker controllable, this vulnerability can lead to systems being non-responsive or may

Show more

3mo

routes/vulnCodeSnippet.ts:104

js-app main Details

3mo

routes/vulnCodeSnippet.ts:108

js-app main Details

3mo

routes/vulnCodeSnippet.ts:123

js-app main Details

3mo

routes/vulnCodeSnippet.ts:125

js-app main Details

3

cookie-secure-flag-false

Pro Security </> Java

A cookie was detected without setting the secure flag. The secure flag for cookies

Semgrep Priority Findings



Semgrep-Demo

Dashboard

Projects

Code8.7K

Secrets67

Supply Chain704

Rules

Editor

Registry

Feedback

Settings

Docs

Help

Supply Chain

Vulnerabilities

Advisories

Dependencies

License configuration

Group by Rule

All time

Projects and branches

All projects with primary branches

Teams

All teams

Tags

All project tags

Status

Open (55)

Severity

Critical

High

Medium

Low

Transitivity

Direct

Transitive

Undetermined

EPSS probability

High

Medium

Low

None

Reachability

Reachable

Always Reachable

Conditionally Reachable

No Reachability Analysis

55 Matching Findings

Sort by highest severity

Triage (0)

Critical

5

jsonwebtoken: Improper Input Validation

EPSS: 0.6% (Low)

CVE

2015-9235

Affected version of jsonwebtoken are vulnerable to Improper Input Validation leading to a verification bypass vulnerability. This is a result of weak validation of the JWT algorithm type, occurring when an attacker is allowed to arbitrarily specify the JWT algorithm.

4mo

lib/insecurity.ts:207

cryptography

Direct

Reachable

js-app

main

Details

4mo

routes/chatbot.ts:160

Direct

Reachable

js-app

main

Details

4mo

routes/chatbot.ts:221

Direct

Reachable

js-app

main

Details

4mo

routes/verify.ts:104

Direct

Reachable

js-app

main

Details

4mo

routes/verify.ts:115

Direct

Reachable

juice-shop

main

Details

Critical

1

com.guidedee.services:log4j-core: Improper Input Validation

EPSS: 96.5% (High)

CVE

2021-44228

org.apache.logging.log4j:log4j-core >= 2.4, < 2.12.2, < 2.3.1, >= 2.13.0, < 2.15.0 are vulnerable to remote code execution when logging untrusted or user controlled data.

14d

vulnerable-application/src/main/java/.../LoginServlet.java:35

user authentication

Direct

Reachable

JavaLog4J

main

Details

Critical

1

minimist: Prototype Pollution

EPSS: 3.5% (Low)

CVE

2021-44906

Minimist <=1.2.5 is vulnerable to Prototype Pollution via file index.js, function setKey() (lines 69-95).

4mo

index.js:19

Direct

Reachable

supply-chain-demo

main

Details

Critical

1

pyyaml: Improper Input Validation

EPSS: 0.6% (Low)

CVE

2020-1747

pyyaml before 5.3.1 is vulnerable to remote code injection. Use `yaml.safe_load` or `Loader=SafeLoader`, or upgrade to pyyaml 5.3.1.

Semgrep Priority Findings



Semgrep-Demo

Dashboard

Projects

Code8.7K

Secrets67

Supply Chain704

Rules

Policies

Editor

Registry

Feedback

Settings

Docs

Help

Secrets

Projects

All projects

Branches

All branches

Teams

All teams

Tags

All project tags

Status

Open (20)

Severity

Critical

High

Medium

Low

Validation state

Confirmed valid

Confirmed invalid

Validation error

No validator

Repository visibility

Public

Private

Group by Rule

All time

Sort by highest severity

Triage (0)

20 Matching Findings

7

semgrep

Semgrep

4mo

src/secrets-pr-validated.js:5

juice-shop

PR #6

Confirmed valid

Details

4mo

src/secrets-pr-validated.js:5

CodeSnippets

PR #7

Confirmed valid

Details

4mo

src/secrets-pr-validated.js:5

supply-chain-second-demo

PR #9

Confirmed valid

Details

4mo

src/secrets-pr-validated.js:5

pro-engine-demo

PR #9

Confirmed valid

Details

4mo

src/secrets-pr-validated.js:5

supply-chain-demo

PR #14

Confirmed valid

Details

Show 2 more findings

3

github_pat

GitHub

1mo

githubtest.txt:1

secrets-demo

main

Confirmed invalid

Details

1mo

test.txt:17

secrets-demo

main

Confirmed invalid

Details

1yr

web/js/salesforce.js:3

leaky-repo

master

Confirmed invalid

Details

2

vercel

Vercel

1mo

randall-demo2.txt:1

secrets-demo

main

Confirmed invalid

Details

1mo

test.txt:21

secrets-demo

main

Confirmed invalid

Details

1

alchemy_new

Alchemy

Semgrep Notification Control



Semgrep-Demo

Dashboard

Projects

Code 8.7k

Secrets 67

Supply Chain 704

Rules

Policies

Editor

Registry

Feedback

Settings

Docs

Help

Policies

Code

Secrets

Modes

Monitor 1044

Comment 811

Block 573

Disabled 235

Category

All categories

Severities

High Medium Low

Confidence

High Medium Low

Source

Pro Community Custom

Available rule upgrades

Secrets

Ruleset

All rulesets

Language

All languages

Minimum count of findings

No minimum

2428 Matching Rules

Active Debug Code

Code Injection

Command Injection

Configuration

Cookie Security

Rule name

Labels

cookie-missing-httponly java, servlets, security, audit

cookie-samesite-none csharp, dotnet-core, cookies

flask-cookie-app-config-httponly-false python, flask, web

flask-cookie-app-config-samesite-none python, flask, web

Show 50 more rules

Cross-Site Request Forgery (CSRF)

Cross-Site-Scripting (XSS)

Edit rule modes

By assigning rules to a mode, you define what happens when a finding is generated by a rule.

Monitor

Track findings in the Semgrep UI without notifying developers.

Notifications

No configured notifications

Comment

Notify developers of vulnerabilities in their merge requests and pull requests.

Notifications


leaves a PR comment

Block

Prevent high risk vulnerabilities from being merged by failing the build when findings are present.

Notifications

leaves a blocking PR comment

Semgrep



But what about the AI?

Focus on Fixes with Assistant

Semgrep Pro Engine Findings

Pro Engine -> Semgrep Pro Engine delivers best-in-breed, deterministic findings with more coverage than any other product.

True Positives

AI Autotriage -> Assistant can tell you which findings are FPs. Out of the box - Assistant reduces noise by as much as 30%.

Important Findings


AI Contextualization -> Of those TPs, it turns out lots are still unimportant (e.g. in tests, dev scripts, or mitigated by another control).

Fixed Findings


AI Autofix -> All important findings receive step by step remediation guidance with suggested code fixes. Fixes are automatically rescanned for assurance.

Meet Developers Where They Are



 semgrep-appsec-platform bot reviewed on Jul 7 View reviewed changes

```
src/assistant-fix-sqli-sequelize.ts
2 +   return (req: Request, res: Response, next: NextFunction) => {
3 +     let criteria: any = req.query.q === 'undefined' ? '' : req.query.q ?? ''
4 +     criteria = (criteria.length <= 200) ? criteria : criteria.substring(0, 200)
5 +     models.sequelize.query("SELECT * FROM Products WHERE ((name LIKE '%" + criteria + "%' OR des
```

 semgrep-appsec-platform bot on Jul 7 ...

Detected a sequelize statement that is tainted by user-input. This could lead to SQL injection if the variable is user-controlled and is not properly sanitized. In order to prevent SQL injection, it is recommended to use parameterized queries or prepared statements.

▼ View Dataflow Graph

Source

Traces

Sink


```
[Line: 3] req.query
```

```
[Line: 3] criteria
```

```
[Line: 5] "SELECT * FROM Pr
```

Navigation icons: back, forward, search, etc.

[Ignore this finding](#) from [express-sequelize-injection](#).



AI Autofix



semgrep-appsec-platform bot on Jul 7



Semgrep Assistant suggests the following fix: Use Sequelize parameterized queries instead of string concatenation.

► View step-by-step instructions

This code change should be a good starting point:

Suggested change

```
5 -     models.sequelize.query("SELECT * FROM Products WHERE ((name LIKE '%" + criteria + "%' OR
description LIKE '%" + criteria + "%') AND deletedAt IS NULL) ORDER BY name")
5 +     models.sequelize.query(
6 +         "SELECT * FROM Products WHERE ((name LIKE :criteria OR description LIKE :criteria)
AND deletedAt IS NULL) ORDER BY name",
7 +         {
8 +             replacements: { criteria: `${criteria}%` },
9 +             type: models.sequelize.QueryTypes.SELECT
10 +         }
11 +     )
```

Commit suggestion ▼

Add suggestion to batch

AI-generated comment. Please review the response carefully and leave feedback in the form of a 👍 or 🙏 reaction



2



1

AI Autofix



semgrep-appsec-platform bot on Jul 7

Semgrep Assistant suggests the following fix: Use Sequelize parameterized queries instead of string concatenation.

View step-by-step instructions

1. Change the query string to use parameterized query syntax by replacing the dynamic values with placeholders `:criteria`.

```
const query = "SELECT * FROM Products WHERE ((name LIKE :criteria OR description LIKE :crit
```

2. Pass the dynamic value in an object as the second parameter of the `sequelize.query` method.

```
models.sequelize.query(query, {  
  replacements: { criteria: `:${criteria}` }  
})
```

3. Replace the original query execution line with the updated parameterized query.

```
models.sequelize.query(query, {  
  replacements: { criteria: `:${criteria}` }  
})  
.then(([products]: any) => {  
  const dataString = JSON.stringify(products)  
  for (let i = 0; i < products.length; i++) {  
    products[i].name = req.__(products[i].name)  
    products[i].description = req.__(products[i].description)  
  }  
  res.json(utils.queryResultToJson(products))  
}).catch((error: ErrorWithParent) => {  
  next(error.parent)  
})
```

Using parameterized queries helps prevent SQL injection by ensuring that user input is properly escaped before being included in the SQL statement.

Autotriage

Some things are safe to ignore





AI Autotriage

```
9  app.post('/proxy', (req, res) => {
10    const { url } = req.query;
11
12    const errorResponse = util.validateProxyRequest(url, res);
13    if (errorResponse) {
14      return errorResponse;
15    }
16
17    request.post(
18      {
19        url,
20        headers: req.headers,
21        body: req.body,
22        json: true,
23      },
24      (error, response, body) => {
25        if (error) {
26          return res.status(500).send(`Request failed: ${error.message}`);
27        }
28        res.status(response.statusCode).send(body);

```




AI Autotriage

```
9  app.post('/proxy', (req, res) => {
10    const { url } = req.query;
11
12    const errorResponse = util.validateProxyRequest(url, res);
13    if (errorResponse) {
14      return errorResponse;
15    }
16
17    request.post(
18      {
19        url,
20        headers: req.headers,
21        body: req.body,
22        json: true,
23      },
24      (error, response, body) => {
25        if (error) {
26          return res.status(500).send(`Request failed: ${error.message}`);
27        }
28        res.status(response.statusCode).send(body);

```




AI Autotriage

```
9  app.post('/proxy', (req, res) => {
10    const { url } = req.query;
11
12    const errorResponse = util.validateProxyRequest(url, res);
13    if (errorResponse) {
14      return errorResponse;
15    }
16
17    request.post(
18      {
19        url,
20        headers: req.headers,
21        body: req.body,
22        json: true,
23      },
24      (error, response, body) => {
25        if (error) {
26          return res.status(500).send(`Request failed: ${error.message}`);
27        }
28        res.status(response.statusCode).send(body);

```



AI Autotriage

```
9  app.post('/proxy', (req, res) => {
10    const { url } = req.query;
11
12    const errorResponse = util.validateProxyRequest(url, res);
13    if (errorResponse) {
14      return errorResponse;
15    }
16
17    request.post(
18      {
19        url,
20        headers: req.headers,
21        body: req.body,
22        json: true,
23      },
24      (error, response, body) => {
25        if (error) {
26          return res.status(500).send(`Request failed: ${error.message}`);
27        }
28        res.status(response.statusCode).send(body);

```

AI Autotriage

```
9   app.post('/proxy', (req, res) => {
10     const { url } = req.query;
11
12     const errorResponse = util.validateProxyRequest(url, res);
13     if (errorResponse) {
14       return errorResponse;
```



semgrep-appsec-platform bot 5 minutes ago

...

Untrusted input might be used to build an HTTP request, which can lead to a Server-side request forgery (SSRF) vulnerability. SSRF allows an attacker to send crafted requests from the server side to other internal or external systems. SSRF can lead to unauthorized access to sensitive data and, in some cases, allow the attacker to control applications or systems that trust the vulnerable service. To prevent this vulnerability, avoid allowing user input to craft the base request. Instead, treat it as part of the path or query parameter and encode it appropriately. When user input is necessary to prepare the HTTP request, perform strict input validation. Additionally, whenever possible, use allowlists to only interact with expected, trusted domains.

► View Dataflow Graph

💬 To ignore this, reply with:

- `/fp <comment>` for false positive
- `/ar <comment>` for acceptable risk
- `/other <comment>` for all other reasons

Alternatively, triage in [Semgrep AppSec Platform](#) to ignore the finding created by [ssrf-deepsemgrep](#).

status open

```
`${error.message}`);
```

AI Autotriage

```
9   app.post('/proxy', (req, res) => {  
10     const { url } = req.query;  
11  
12     const errorResponse = util.validateProxyRequest(url, res);  
13     if (errorResponse) {  
14       return errorResponse;  
15     }  
16   });  
17  
18   res.status(200).send(`Success: ${error.message}`);  
19 }  
20  
21 export default app;
```



semgrep-appsec-platform (bot) 5 minutes ago

...

Untrusted input might be used to build an HTTP request, which can lead to a Server-side request forgery (SSRF) vulnerability. SSRF allows an attacker to send crafted requests from the server side to other internal or external systems. SSRF can lead to unauthorized access to sensitive data and, in some cases, allow the attacker to control applications or systems that trust the vulnerable service. To prevent this vulnerability, avoid allowing user input to craft the base request. Instead, treat it as part of the path or query parameter and encode it appropriately. When user input is necessary to prepare the HTTP request, perform strict input validation. Additionally, whenever possible, use allowlists to only interact with expected, trusted domains.

► View Dataflow Graph

💬 To ignore this, reply with:

- `/fp <comment>` for false positive
- `/ar <comment>` for acceptable risk
- `/other <comment>` for all other reasons

Alternatively, triage in [Semgrep AppSec Platform](#) to ignore the finding created by [ssrf-deepsemgrep](#).

status open

AI Autotriage

```
9   app.post('/proxy', (req, res) => {
10     const { url } = req.query;
11
12     const errorResponse = util.validateProxyRequest(url, res);
13     if (errorResponse) {
14       return errorResponse;
```



semgrep-appsec-platform (bot) 5 minutes ago

...

Untrusted input might be used to build an HTTP request, which can lead to a Server-side request forgery (SSRF) vulnerability. SSRF allows an attacker to send crafted requests from the server side to other internal or external systems. SSRF can lead to unauthorized access to sensitive data and, in some cases, allow the attacker to control applications or systems that trust the vulnerable service. To prevent this vulnerability, avoid allowing user input to craft the base request. Instead, treat it as part of the path or query parameter and encode it appropriately. When user input is necessary to prepare the HTTP request, perform strict input validation. Additionally, whenever possible, use allowlists to only interact with expected, trusted domains.

► View Dataflow Graph

💬 To ignore this, reply with:

- /fp <comment> for false positive
- /ar <comment> for acceptable risk
- /other <comment> for all other reasons

Alternatively, triage in [Semgrep AppSec Platform](#) to ignore the finding created by [ssrf-deepsemgrep](#).

status open

```
`${error.message}`);
```

AI Autotriage

```
9   app.post('/proxy', (req, res) => {  
10     const { url } = req.query;  
11  
12     const errorResponse = util.validateProxyRequest(url, res);  
13     if (errorResponse) {  
14       return errorResponse;  
15     }  
16   });  
17  
18   res.status(200).send(`Success: ${error.message}`);  
19 }
```



semgrep-appsec-platform (bot) 5 minutes ago

...

Untrusted input might be used to build an HTTP request, which can lead to a Server-side request forgery (SSRF) vulnerability. SSRF allows an attacker to send crafted requests from the server side to other internal or external systems. SSRF can lead to unauthorized access to sensitive data and, in some cases, allow the attacker to control applications or systems that trust the vulnerable service. To prevent this vulnerability, avoid allowing user input to craft the base request. Instead, treat it as part of the path or query parameter and encode it appropriately. When user input is necessary to prepare the HTTP request, perform strict input validation. Additionally, whenever possible, use allowlists to only interact with expected, trusted domains.

► View Dataflow Graph

💬 To ignore this, reply with:

- `/fp <comment>` for false positive
- `/ar <comment>` for acceptable risk
- `/other <comment>` for all other reasons

Alternatively, triage in [Semgrep AppSec Platform](#) to ignore the finding created by [ssrf-deepsemgrep](#).

status open

AI Autotriage

```
9   app.post('/proxy', (req, res) => {  
10     const { url } = req.query;  
11  
12     const errorResponse = util.validateProxyRequest(url, res);  
13     if (errorResponse) {  
14       return errorResponse;  
15     }  
16   });  
17  
18   res.json({error.message});  
19 }
```



semgrep-appsec-platform (bot) 5 minutes ago

...

Untrusted input might be used to build an HTTP request, which can lead to a Server-side request forgery (SSRF) vulnerability. SSRF allows an attacker to send crafted requests from the server side to other internal or external systems. SSRF can lead to unauthorized access to sensitive data and, in some cases, allow the attacker to control applications or systems that trust the vulnerable service. To prevent this vulnerability, avoid allowing user input to craft the base request. Instead, treat it as part of the path or query parameter and encode it appropriately. When user input is necessary to prepare the HTTP request, perform strict input validation. Additionally, whenever possible, use allowlists to only interact with expected, trusted domains.

► View Dataflow Graph

To ignore this, reply with:

- /fp <comment> for false positive
- /ar <comment> for acceptable risk
- /other <comment> for all other reasons

Alternatively, triage in [Semgrep AppSec Platform](#) to ignore the finding created by [ssrf-deepsemgrep](#).

status open

AI Autotriage














Leave feedback with a 👍 / 👎.

😊 1 👍 1



Write

Preview

 | **H** **B** *I* |    |    |  @   

/fp

We generally will use some sort of validation on user input. If there are functions that are named some variation of validate, sanitize, clean, etc that process user input - those checks that will likely validate the data before it's executed. In this case util.validateProxyRequest() is one of those tools that are making this finding safe to ignore.

 Markdown is supported  Paste, drop, or click to add files

Cancel

Add single comment

Start a review

AI Autotriage



Leave feedback with a 👍 / 👎.

😊 👍 1 👎 1

Write Preview

📎 H B I ≡ < > 🔗 ☰ ≡ ≡ 📎 @ ↗ ↶ 📐

/fp

We generally will use some sort of validation on user input. If there are functions that are named some variation of validate, sanitize, clean, etc that process user input - those checks that will likely validate the data before it's executed. In this case util.validateProxyRequest() is one of those tools that are making this finding safe to ignore.

📄 Markdown is supported 🖼️ Paste, drop, or click to add files

Cancel Add single comment Start a review

AI Autotriage



Leave feedback with a 👍 / 👎.

😊 1 👍 1

Write Preview

📎 H B I ≡ <> 🔗 ≡ ≡ ≡ 📎 @ ↗ ↶

/fp

We generally will use some sort of validation on user input. If there are functions that are named some variation of validate, sanitize, clean, etc that process user input - those checks that will likely validate the data before it's executed. In this case util.validateProxyRequest() is one of those tools that are making this finding safe to ignore.

📄 Markdown is supported 🖼️ Paste, drop, or click to add files

Cancel Add single comment Start a review

AI Autotriage



Leave feedback with a 👍 / 👎.

😊 1 👍 1

Write Preview

📎 H B I ≡ <> 🔗 ≡ ≡ ≡ 📎 @ ↗ ↶

/fp

We generally will use some sort of validation on user input. If there are functions that are named some variation of validate, sanitize, clean, etc that process user input - those checks that will likely validate the data before it's executed. In this case `util.validateProxyRequest()` is one of those tools that are making this finding safe to ignore.

📄 Markdown is supported 🖼️ Paste, drop, or click to add files

Cancel Add single comment Start a review

AI Autotriage



r2c-david 4 minutes ago

Author



/fp

We generally will use some sort of validation on user input. If there are functions that are named some variation of validate, sanitize, clean, etc that process user input - those checks that will likely validate the data before it's executed. In this case `util.validateProxyRequest()` is one of those tools that are making this finding safe to ignore.



semgrep-appsec-platform bot 4 minutes ago



Status updated to `ignored - false positive` by [@r2c-david](#).

Reply with `/open` to re-open this finding.




Reply...

Resolve conversation

AI Autotriage




 **r2c-david** 4 minutes ago Author ...

/fp


We generally will use some sort of validation on user input. If there are functions that are named some variation of validate, sanitize, clean, etc that process user input - those checks that will likely validate the data before it's executed. In this case `util.validateProxyRequest()` is one of those tools that are making this finding safe to ignore.



 **semgrep-appsec-platform** bot 4 minutes ago ...

Status updated to ignored – false positive by [@r2c-david](#).

Reply with `/open` to re-open this finding.





Resolve conversation





AI Autotriage

```
const errorResponse = util.validateProxyRequest(url, res);  
if (errorResponse) {  
  return errorResponse;  
}
```

```
const errorResponse = utility.checkSafe(url, res);  
if (errorResponse) {  
  return errorResponse;  
}
```



AI Autotriage

```
const errorResponse = util.validateProxyRequest(url, res);  
if (errorResponse) {  
  return errorResponse;  
}
```

```
const errorResponse = utility.checkSafe(url, res);  
if (errorResponse) {  
  return errorResponse;  
}
```




AI Autotriage

```
const errorResponse = util.validateProxyRequest(url, res);  
if (errorResponse) {  
  return errorResponse;  
}
```

```
const errorResponse = utility.checkSafe(url, res);  
if (errorResponse) {  
  return errorResponse;  
}
```

AI Autotriage



semgrep-appsec-platform bot 7 minutes ago



Semgrep Assistant thinks this might be safe to ignore. The code uses a function named `utility.checksafe()` to validate the url before making the request, indicating that there is likely a validation mechanism in place to ensure the url is safe. this makes the finding less of a security concern.

AI-generated comment; review carefully.

Leave a 👍 reaction to ignore the finding. Reacting with 👍 or 👎 also provides feedback to improve Assistant's future comments.



AI Autotriage



semgrep-appsec-platform bot 7 minutes ago



Semgrep Assistant thinks this might be safe to ignore. The code uses a function named `utility.checksafe()` to validate the url before making the request, indicating that there is likely a validation mechanism in place to ensure the url is safe. this makes the finding less of a security concern.

AI-generated comment; review carefully.

Leave a 👍 reaction to ignore the finding. Reacting with 👍 or 👎 also provides feedback to improve Assistant's future comments.



AI Autotriage



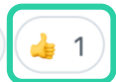
semgrep-appsec-platform bot 7 minutes ago



Semgrep Assistant thinks this might be safe to ignore. The code uses a function named `utility.checksafe()` to validate the url before making the request, indicating that there is likely a validation mechanism in place to ensure the url is safe. this makes the finding less of a security concern.

AI-generated comment; review carefully.

Leave a 👍 reaction to ignore the finding. Reacting with 👍 or 👎 also provides feedback to improve Assistant's future comments.



Assistant Memories

Never forget a great tip



AI Memories



returntocorp

Dashboard

Projects

Code181

Supply Chain432

Rules

Enable blocking for high or critical severity

Secrets

Find out immediately if

Run Secrets in

Historical scan

Historical scanning contained secrets t

Assistant

Get automated recommendations

Semgrep is enhanced

Allow code sni

Required to auto-tr with code context,

Weekly priority

Send organization

Auto-triage for

Get notified when ,

Customize with memories

Assistant uses the following memories when generating guidance. You can add memories by clicking **Improve fix** near Assistant's remediation guidance on finding details pages.


Active

Drafts (1 pending)

Search by memory content, username, or project name

Add memory

Project ↕	Rules ↕	Memory	Created ↕
All Projects 3.7K projects	All Secrets rules 854 rules	Encrypted secrets in code are decrypted at runtime by our internal framework Decryptonite. Use of Decryptonite near secrets indicat...	Nov 4, 2024 By: margaret@semgr...
Tagged External 459 projects	All SQL Injection Rules 154 rules	SQL Injection is one of our biggest concerns for external facing applications. These should never be considered false positives.	October 30th, 2024 By: raj@semgrep.com
Tagged Internal 3.2K projects	All SSRF Rules 106 rules	We generally will use some sort of validation on user input. If there are functions that are named some variation of validate, sanitize, cl...	October 28th, 2024 By: julia@semgrep.c...
Tagged Internal 3.2K projects	All Rules 3.4K rules	Data fetched from other internal services is considered trusted and is safe when flowing into potentially dangerous execution sites.	October 20th, 2024 By: emma@semgrep...
corp/decryptonite-proxy 1 project	Dockerfile.last.user.is.root 1 rule	Decryptonite-proxy requires Docker to run as root to access restricted network interfaces, so these should be considered safe...	October 18th, 2024 By: bence@semgrep...
Tagged MigrationPending 640 projects	Migration.ui.incomplete.references 1 rule	The new UI framework requires defined references for each of the components after migration. Use the following information when s...	October 18th, 2024 By: vivek@semgrep.c...

Semgrep

AI Memories



returntocorp

Dashboard

Projects

Code181

Supply Chain432

Rules

Enable blocking for high or critical severity

Secrets

Find out immediately if secrets are present in your code

Run Secrets in

Historical scan

Assistant

Get automated recommendations. Semgrep is enhanced with AI.

Allow code sni

Required to auto-triage with code context, project name, and rules.

Weekly priority

Send organization-wide alerts for high priority findings.

Auto-triage for

Get notified when a finding is automatically triaged.

Customize with memories

Assistant uses the following memories when generating guidance. You can add memories by clicking **Improve fix** near Assistant's remediation guidance on finding details pages.

Active

Drafts (1 pending)

Search by memory content, username, or project name

Add memory

Project ↕	Rules ↕	Memory	Created ↕
All Projects 3.7K projects	All Secrets rules 854 rules	Encrypted secrets in code are decrypted at runtime by our internal framework Decryptonite. Use of Decryptonite near secrets indicat...	Nov 4, 2024 By: margaret@semgr...
Tagged External 459 projects	All SQL Injection Rules 154 rules	SQL Injection is one of our biggest concerns for external facing applications. These should never be considered false positives.	October 30th, 2024 By: raj@semgrep.com
Tagged Internal 3.2K projects	All SSRF Rules 106 rules	We generally will use some sort of validation on user input. If there are functions that are named some variation of validate, sanitize, cl...	October 28th, 2024 By: julia@semgrep.c...
Tagged Internal 3.2K projects	All Rules 3.4K rules	Data fetched from other internal services is considered trusted and is safe when flowing into potentially dangerous execution sites.	October 20th, 2024 By: emma@semgrep...
corp/decryptonite-proxy 1 project	Dockerfile.last.user.is.root 1 rule	Decryptonite-proxy requires Docker to run as root to access restricted network interfaces, so these should be considered safe...	October 18th, 2024 By: bence@semgrep...
Tagged MigrationPending 640 projects	Migration.ui.incomplete.references 1 rule	The new UI framework requires defined references for each of the components after migration. Use the following information when s...	October 18th, 2024 By: vivek@semgrep.c...

Semgrep

AI Memories



returnto corp

Dashboard

Projects

Code181

Supply Chain432

Rules

Enable blocking for high or critical severity

Secrets

Find out immediately if secrets are present in code

Run Secrets in

Historical scan

Historical scanning contained secrets

Assistant

Get automated recommendations

Semgrep is enhanced

Allow code sni

Required to auto-tri

with code context,

Weekly priority

Send organization

Auto-triage for

Get notified when

Customize with memories

Assistant uses the following memories when generating guidance. You can add memories by clicking **Improve fix** near Assistant's remediation guidance on finding details pages.

Active Drafts (1 pending)

Search by memory content, username, or project name

Add memory

Project ↕	Rules ↕	Memory	Created ↕
All Projects 3.7K projects	All Secrets rules 854 rules	Encrypted secrets in code are decrypted at runtime by our internal framework Decryptonite. Use of Decryptonite near secrets indicat...	Nov 4, 2024 By: margaret@semgr...
Tagged External 459 projects	All SQL Injection Rules 154 rules	SQL Injection is one of our biggest concerns for external facing applications. These should never be considered false positives.	October 30th, 2024 By: raj@semgrep.com
Tagged Internal 3.2K projects	All SSRF Rules 106 rules	We generally will use some sort of validation on user input. If there are functions that are named some variation of validate, sanitize, cl...	October 28th, 2024 By: julia@semgrep.c...
Tagged Internal 3.2K projects	All Rules 3.4K rules	Data fetched from other internal services is considered trusted and is safe when flowing into potentially dangerous execution sites.	October 20th, 2024 By: emma@semgrep...
corp/decryptonite-proxy 1 project	Dockerfile.last.user.is.root 1 rule	Decryptonite-proxy requires Docker to run as root to access restricted network interfaces, so these should be considered safe...	October 18th, 2024 By: bence@semgrep...
Tagged MigrationPending 640 projects	Migration.ui.incomplete.references 1 rule	The new UI framework requires defined references for each of the components after migration. Use the following information when s...	October 18th, 2024 By: vivek@semgrep.c...

AI Memories



returnto corp

Dashboard

Projects

Code181

Supply Chain432

Rules

Enable blocking tool for high or critical severity

Secrets

Find out immediately if secrets are exposed

Run Secrets in CI

Historical scanning of code repositories contained secrets to help find secrets that have been removed

Assistant

Get automated recommendations. Semgrep is enhanced with AI

Allow code sniffs

Required to auto-triage with code context, and can be configured to auto-remediate

Weekly priority

Send organization-wide alerts

Auto-triage for

Get notified when a new rule is added or updated

Customize with memories

Assistant uses the following memories when generating guidance. You can add memories by clicking **Improve fix** near Assistant's remediation guidance on finding details pages.

Active

Drafts (1 pending)

Search by memory content, username, or project name

Add memory

Project ↕	Rules ↕	Memory	Created ↕
All Projects 3.7K projects	All Secrets rules 854 rules	Encrypted secrets in code are decrypted at runtime by our internal framework Decryptonite. Use of Decryptonite near secrets indicat...	Nov 4, 2024 By: margaret@semgr...
Tagged External 459 projects	All SQL Injection Rules 154 rules	SQL Injection is one of our biggest concerns for external facing applications. These should never be considered false positives.	October 30th, 2024 By: raj@semgrep.com
Tagged Internal 3.2K projects	All SSRF Rules 106 rules	We generally will use some sort of validation on user input. If there are functions that are named some variation of validate, sanitize, cl...	October 28th, 2024 By: julia@semgrep.c...
Tagged Internal 3.2K projects	All Rules 3.4K rules	Data fetched from other internal services is considered trusted and is safe when flowing into potentially dangerous execution sites.	October 20th, 2024 By: emma@semgrep...
corp/decryptonite-proxy 1 project	Dockerfile.last.user.is.root 1 rule	Decryptonite-proxy requires Docker to run as root to access restricted network interfaces, so these should be considered safe...	October 18th, 2024 By: bence@semgrep...
Tagged MigrationPending 640 projects	Migration.ui.incomplete.references 1 rule	The new UI framework requires defined references for each of the components after migration. Use the following information when s...	October 18th, 2024 By: vivek@semgrep.c...

Semgrep

AI Memories



returnto corp

Dashboard

Projects

Code181

Supply Chain432

Rules

Enable blocking for high or critical severity

Secrets

Find out immediately if

Run Secrets in

Historical scan

Historical scanning contained secrets t

Assistant

Get automated recom Semgrep is enhanced

Allow code sni

Required to auto-tr with code context,

Weekly priority

Send organization

Auto-triage for

Get notified when ,

Customize with memories

Assistant uses the following memories when generating guidance. You can add memories by clicking **Improve fix** near Assistant's remediation guidance on finding details pages.

Active Drafts (1 pending)

Search by memory content, username, or project name

Add memory

Project ↕	Rules ↕	Memory	Created ↕
All Projects 3.7K projects	All Secrets rules 854 rules	Encrypted secrets in code are decrypted at runtime by our internal framework Decryptonite. Use of Decryptonite near secrets indicat...	Nov 4, 2024 By: margaret@semgr...
Tagged External 459 projects	All SQL Injection Rules 154 rules	SQL Injection is one of our biggest concerns for external facing applications. These should never be considered false positives.	October 30th, 2024 By: raj@semgrep.com
Tagged Internal 3.2K projects	All SSRF Rules 106 rules	We generally will use some sort of validation on user input. If there are functions that are named some variation of validate, sanitize, cl...	October 28th, 2024 By: julia@semgrep.c...
Tagged Internal 3.2K projects	All Rules 3.4K rules	Data fetched from other internal services is considered trusted and is safe when flowing into potentially dangerous execution sites.	October 20th, 2024 By: emma@semgrep...
corp/decryptonite-proxy 1 project	Dockerfile.last.user.is.root 1 rule	Decryptonite-proxy requires Docker to run as root to access restricted network interfaces, so these should be considered safe...	October 18th, 2024 By: bence@semgrep...
Tagged MigrationPending 640 projects	Migration.ui.incomplete.references 1 rule	The new UI framework requires defined references for each of the components after migration. Use the following information when s...	October 18th, 2024 By: vivek@semgrep.c...

Semgrep

Semgrep Assistant Features

AI-powered workflows that cut through the noise and **fix**
the most critical vulnerabilities in your backlog.

✦✦ **Noise filtering** saves dev and AppSec time by flagging findings that likely not exploitable

✦✦ **Remediation guidance** tells your developers how to fix vulnerabilities in PR comments

✦✦ **Memories** tailor triage and remediation to your organization's standards and secure defaults

✦✦ **Rule generation** writes high-signal, low-FP custom rules using natural language instructions

✦✦ **Priority Inbox** surfaces critical findings on your most sensitive repositories

Supported in 30+ languages



Thank you!

Questions?