

SAST für TYPO3 Extensions

#T3CRR21 #security

TYPO3camp RheinRuhr 2021
6. November 2021

Oliver Hader
oliver@typo3.org
 [@ohader](https://twitter.com/ohader)

Oliver Hader

- Research & Development
- TYPO3 Security Team Lead
- 50% TYPO3 GmbH / 50% Freelancer
- #hof #cycling #paramedic #in.die.musik

 @ohader



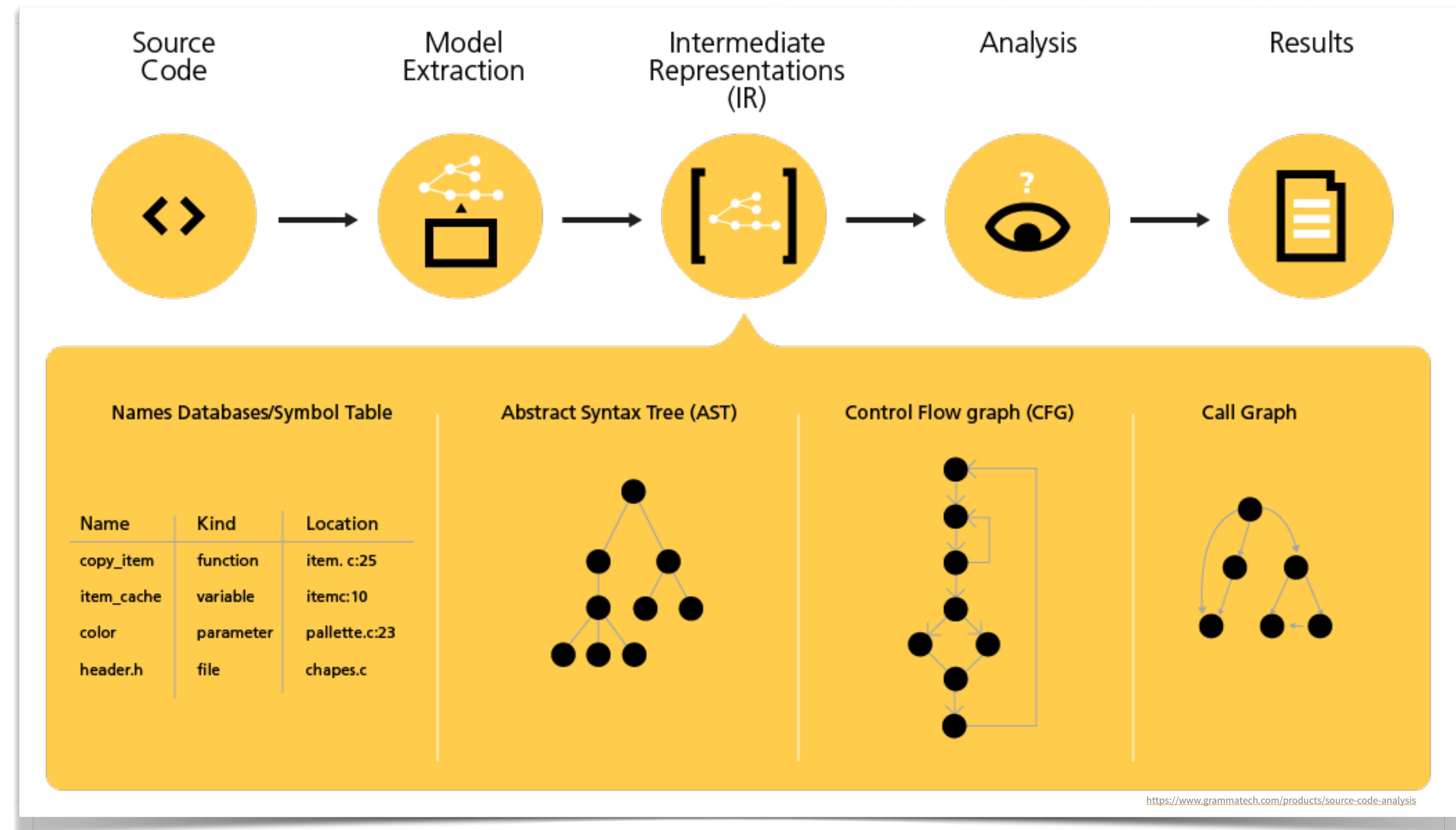
- Statische Code Analyse #basics
- Orientierung im Security Bereich #context
- Verwendung für TYPO3 #nähkästchen
- MüssteMalJemand™ #zukunft

- Bitte, bitte, bitte: **Security Schwachstellen nicht öffentlich einstellen/diskutieren**
 - **nicht** auf GitHub/Forge
 - **nicht** in Slack Channels
 - **nicht** auf Twitter, Twitch, ...
- Meldung an **security@typo3.org**
- Security Team unterstützt Community



#basics

Statische Code Analyse



Statische Code Analyse - Schematischer Ablauf

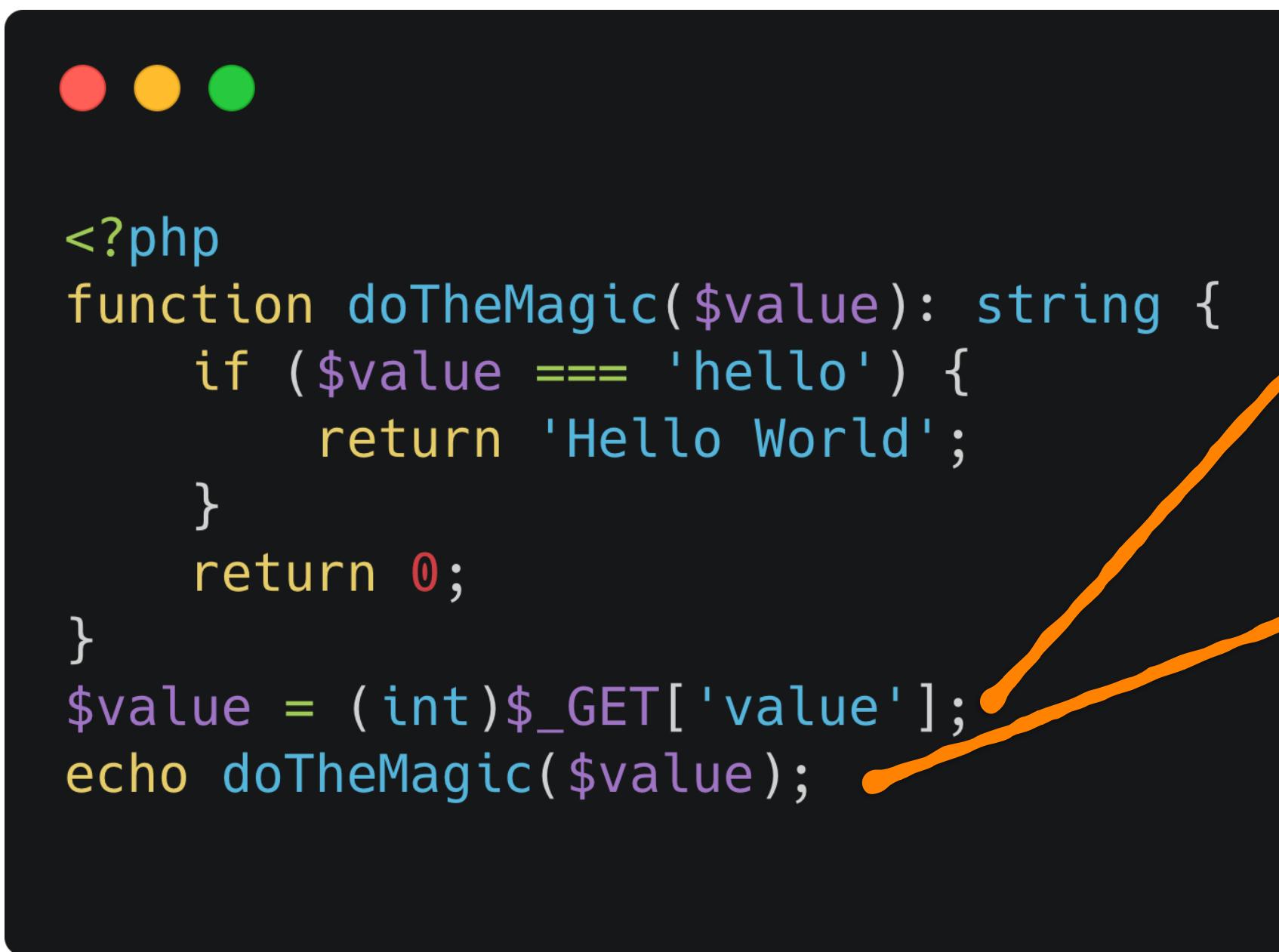
- Abstract Syntax Tree (AST)
- Composer Package
nikic/php-parser

```
bin/php-parse sast/magic.php

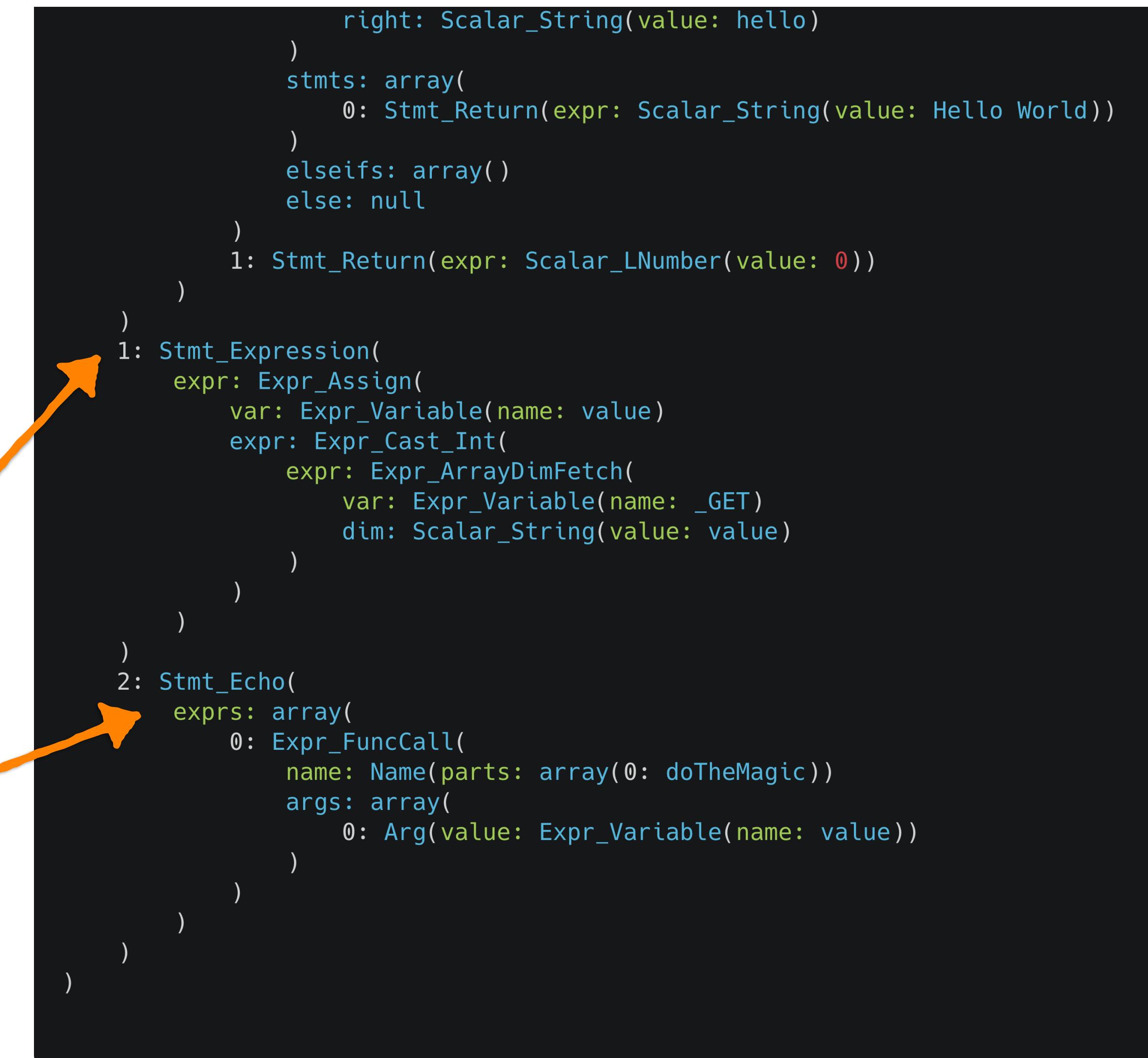
====> File sast/magic.php:
==> Node dump:
array(
    0: Stmt_Function(
        name: Identifier(name: doTheMagic)
        params: array(
            0: Param(
                flags: 0
                type: null
                var: Expr_Variable(name: value)
                default: null
            )
        )
        returnType: Identifier(name: string)
        stmts: array(
            0: Stmt_If(
                cond: Expr_BinaryOp_Identical(
                    left: Expr_Variable(name: value)
                    right: Scalar_String(value: hello)
                )
                stmts: array(
                    0: Stmt_Return(expr: Scalar_String(value: Hello World))
                )
                elseifs: array()
                else: null
            )
            1: Stmt_Return(expr: Scalar_LNumber(value: 0))
        )
    )
)
```

```
<?php
function doTheMagic($value): string {
    if ($value === 'hello') {
        return 'Hello World';
    }
    return 0;
}
$value = (int)$_GET['value'];
echo doTheMagic($value);
```

- Abstract Syntax Tree (AST)
- Composer Package
nikic/php-parser

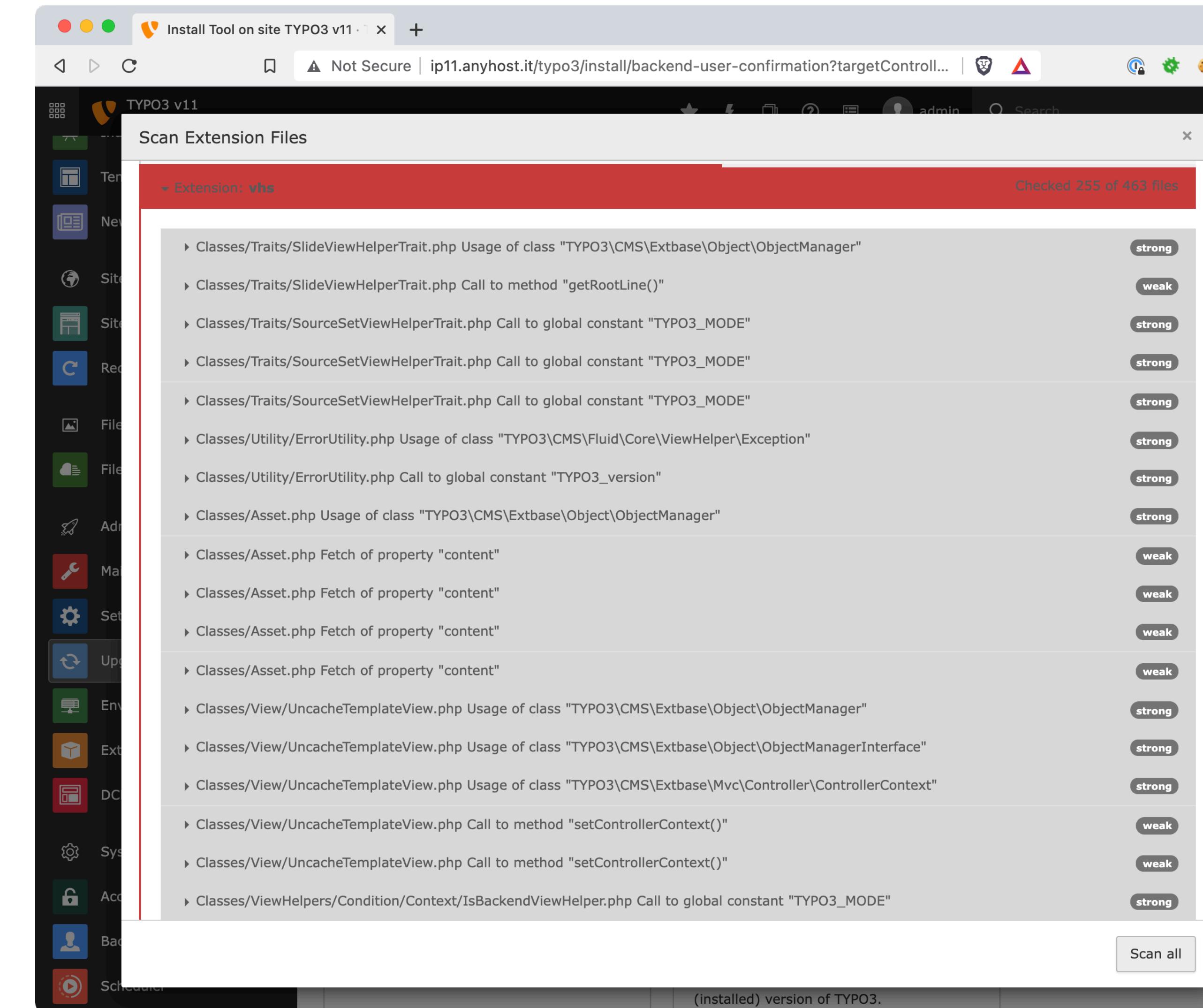


```
<?php
function doTheMagic($value): string {
    if ($value === 'hello') {
        return 'Hello World';
    }
    return 0;
}
$value = (int)$_GET['value'];
echo doTheMagic($value);
```



```
{
    "stmts": [
        {
            "stmt": "Expr_Assign",
            "expr": "Expr_Cast_Int",
            "var": "Expr_Variable",
            "value": "Scalar_String"
        },
        {
            "stmt": "Stmt_Echo",
            "exprs": [
                "Expr_FuncCall"
            ]
        }
    ],
    "elseifs": [
        {
            "stmt": "Stmt_Return"
        }
    ],
    "else": null
}
```

- basierend auf Abstract Syntax Tree (AST)
- **Extension Scanner** in TYPO3
- **PhpStan** - Core CI Builds
- **Rector** - TYPO3 Code Upgrade
- **Psalm** - Security Taint Graph



#context

Orientierung im

Security Bereich

Ebenenmodell zur Sicherheitskonzeption (BSI)

Ebene		Inhalt (Beispiele)
5	Semantik	<p>Schutz vor Täuschung und Betrug</p> <ul style="list-style-type: none"> - Informationen ermöglichen Social Engineering-Angriffe - Gebrauch von Popups u.ä. erleichtern Phishing-Angriffe - Keine Absicherung für den Fall der Fälschung der Webseite
4	Logik	<p>Absicherung von Prozessen und Workflows als Ganzes</p> <ul style="list-style-type: none"> - Verwendung unsicherer Email in einem ansonsten gesicherten Workflow - Angreifbarkeit des Passworts durch nachlässig gestaltete "Passwort vergessen"-Funktion - Die Verwendung sicherer Passworte wird nicht erzwungen
3	Implementierung	<p>Vermeiden von Programmierfehlern, die zu Schwachstellen führen</p> <ul style="list-style-type: none"> - Cross-Site Scripting - SQL-Injection - Session Riding
2	Technologie	<p>Richtige Wahl und sicherer Einsatz von Technologie</p> <ul style="list-style-type: none"> - unverschlüsselte Übertragung sensitiver Daten - Authentisierungsverfahren, die nicht dem Schutzbedarf angemessen sind - Ungenügende Randomness von Token
1	System	<p>Absicherung der auf der Systemplattform eingesetzten Software</p> <ul style="list-style-type: none"> - Fehler in der Konfiguration des Webservers - "Known Vulnerabilities" in den eingesetzten Softwareprodukten - Mangelnder Zugriffsschutz in der Datenbank
0	Netzwerk & Host	Absicherung von Host und Netzwerk

#phishing
#SocialEngineering

#workflows
SAST
#bugs

#infrastructure


**Bundesamt
für Sicherheit in der
Informationstechnik**



Dokument ist noch aktuell. (Stand 2020)

Sicherheit von Webanwendungen
Maßnahmenkatalog und Best Practices

Im Auftrag des
**Bundesamtes für Sicherheit
in der Informationstechnik**

erstellt von:
SecureNet

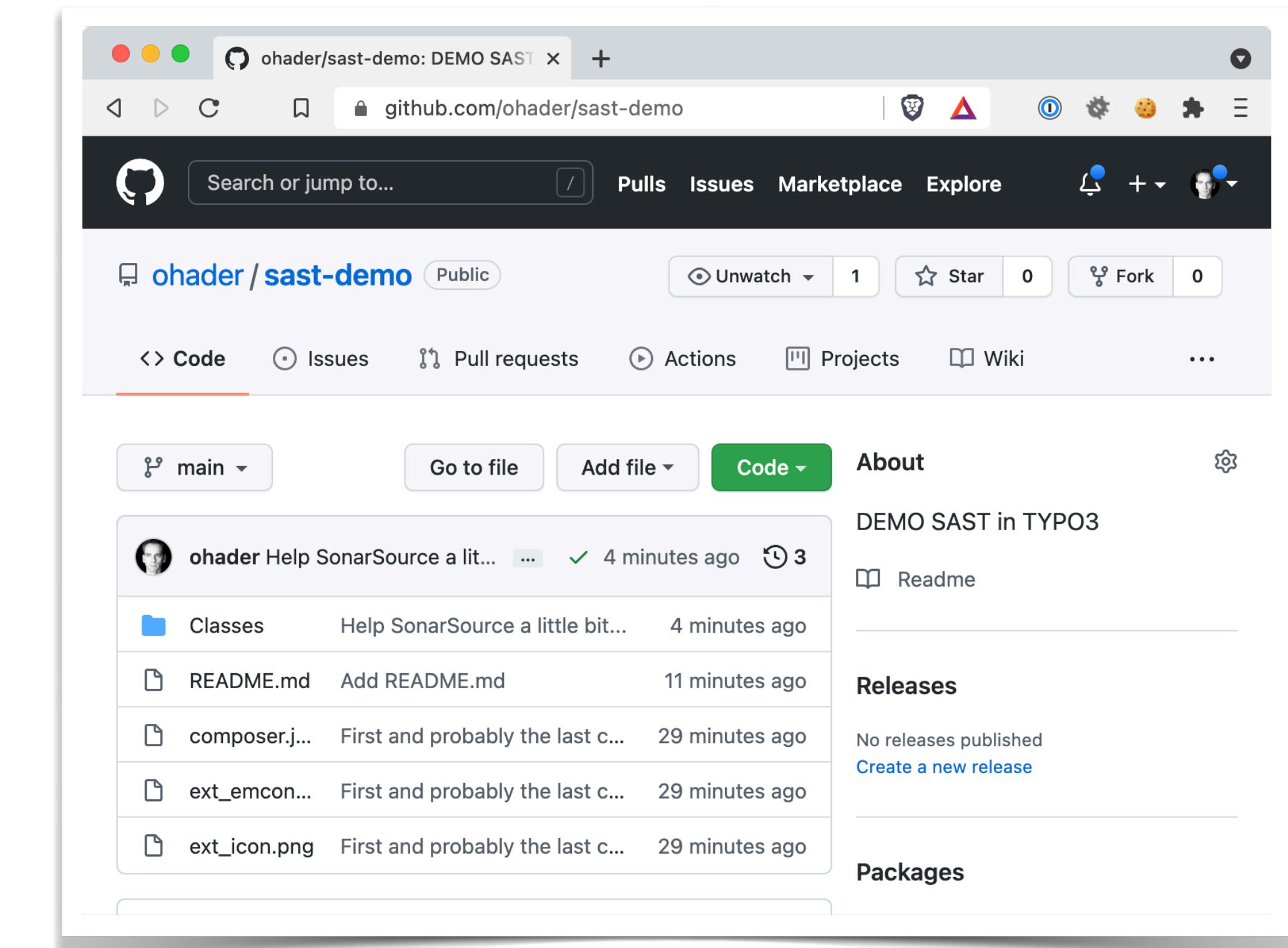
Version 1, August 2006

Bundesamt für Sicherheit in der Informationstechnik
Godesberger Allee 185-189, 53175 Bonn • Postfach 200363, 53133 Bonn
Tel.: 01888 9582-0 • +49 (0)3018 9582-0
Fax: 01888 9582-5400 • +49 (0)3018 9582-5400
Internet: www.bsi.bund.de



<https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Informationen-und-Empfehlungen/Empfehlungen-nach-Angriffszielen/Webanwendungen/node.html>

- **SonarSource, SonarCloud, SonarCube**
- **RIPS Tech Scanner**
- **DeepCode.ai**
- **PsalmPHP**



<https://github.com/ohader/sast-demo>

Demo Anwendung (vereinfacht)



```
<?php
declare(strict_types=1);
namespace Olly\SastDemo;

class SimpleController
{
    /** @var mixed */
    private $uid;
    /** @var \PDO */
    private $pdo;

    public function __construct()
    {
        $this->uid = $_GET['uid'];
    }
    public function executeAction(): void
    {
        $headers = [];
        $statement = $this->pdo->query(
            'SELECT header FROM tt_content WHERE uid=' . $this->uid
        );
        foreach ($statement->fetchAll(\PDO::FETCH_ASSOC) as $row) {
            $headers[] = $row['header'];
        }
        echo 'UID: ' . $this->uid;
        echo implode(', ', $headers);
    }
}
```

The screenshot shows a GitHub repository page for 'ohader/sast-demo'. The repository is public and has 1 pull request, 0 stars, and 0 forks. The 'Code' tab is selected, showing the main file. The repository contains several files: Classes, README.md, composer.json, ext_emcon.php, and ext_icon.png. The 'About' section indicates it's a 'DEMO SAST in TYPO3'. The 'Releases' section shows no releases published, with a link to 'Create a new release'. The 'Packages' section is also present.

<https://github.com/ohader/sast-demo>

Demo Anwendung (vereinfacht)



```
<?php
declare(strict_types=1);
namespace Olly\SastDemo;

class SimpleController
{
    /** @var mixed */
    private $uid;
    /** @var \PDO */
    private $pdo;

    public function __construct()
    {
        $this->uid = $_GET['uid']; ← Benutzereingaben
    }
    public function executeAction(): void
    {
        $headers = [];
        $statement = $this->pdo->query(
            'SELECT header FROM tt_content WHERE uid=' . $this->uid
        );
        foreach ($statement->fetchAll(\PDO::FETCH_ASSOC) as $row) {
            $headers[] = $row['header'];
        }
        echo 'UID: ' . $this->uid;
        echo implode(', ', $headers);
    }
}
```

SQLi

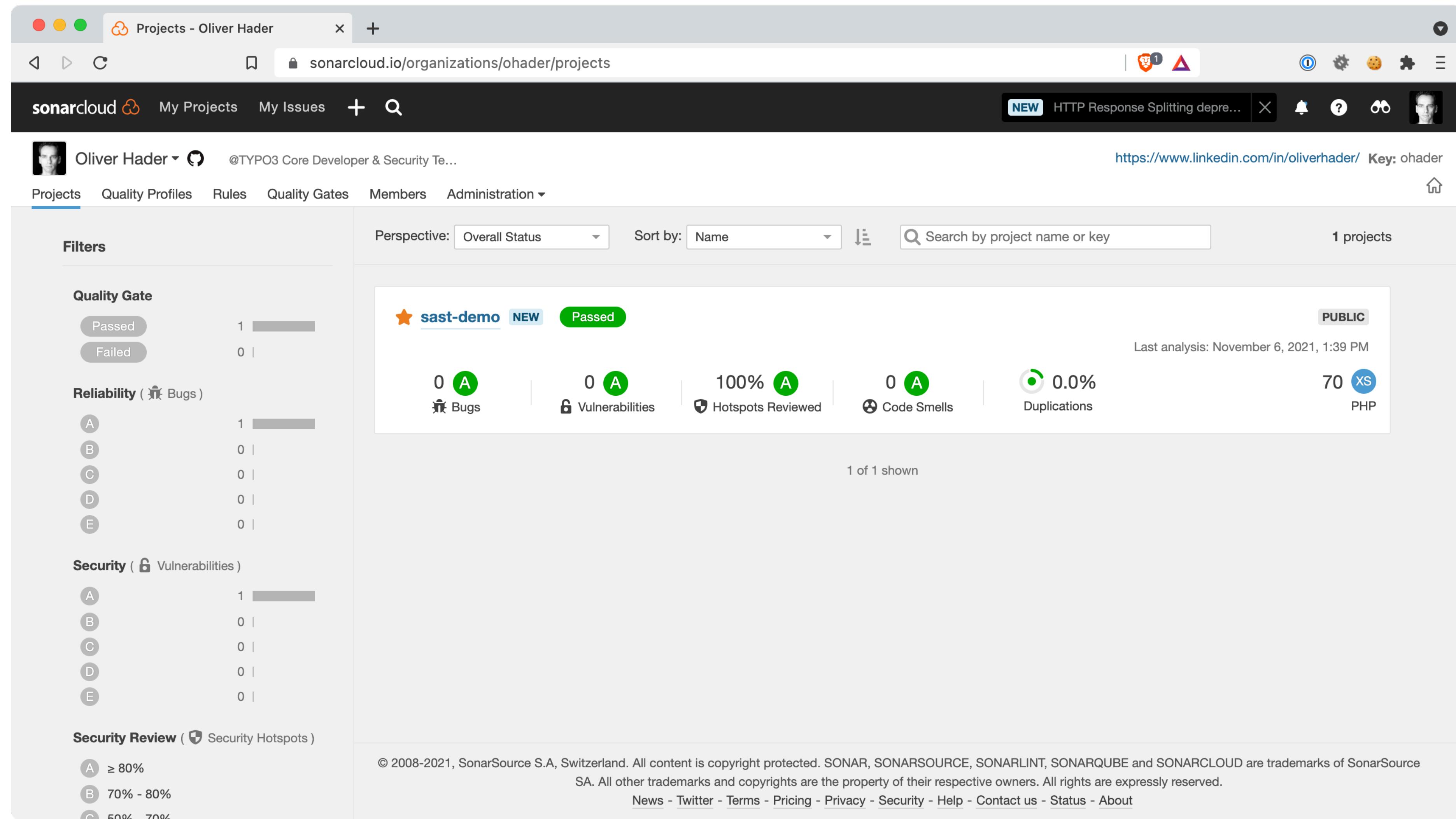
XSS

The screenshot shows a GitHub repository page for 'ohader/sast-demo'. The repository is public and has 1 pull request, 0 stars, and 0 forks. The 'Code' tab is selected, showing the main branch. The code listing includes the PHP file from the left panel. Below the code, there's a list of recent commits by 'ohader':

Commit	Message	Time	Actions
Help SonarSource a lit...	Help SonarSource a little bit...	4 minutes ago	3
Classes	Add README.md	4 minutes ago	
README.md	Add README.md	11 minutes ago	
composer.j...	First and probably the last c...	29 minutes ago	
ext_emcon...	First and probably the last c...	29 minutes ago	
ext_icon.png	First and probably the last c...	29 minutes ago	

On the right side, there are sections for 'About', 'DEMOSAST in TYPO3', 'Readme', 'Releases' (no releases published), and 'Packages'.

<https://github.com/ohader/sast-demo>



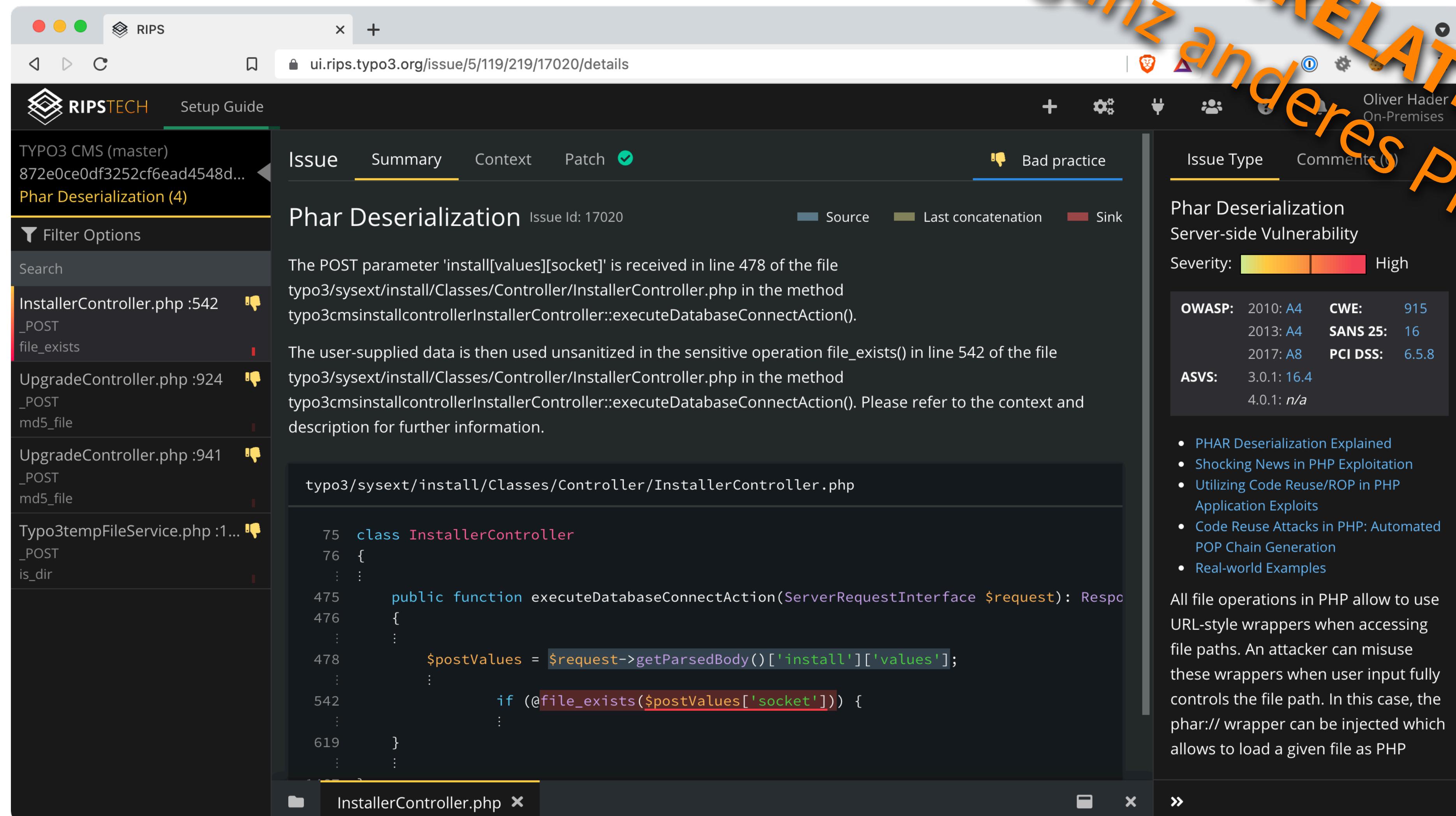
The screenshot shows the SonarCloud interface for the project 'sast-demo'. The main dashboard displays various quality metrics:

- Quality Gate:** Passed (1), Failed (0)
- Reliability (Bug):** A (1), B (0), C (0), D (0), E (0)
- Security (Vulnerability):** A (1), B (0), C (0), D (0), E (0)
- Security Review (Hotspots):** ≥ 80% (A), 70% - 80% (B), 50% - 70% (C)
- Overall Status:** Overall Status: Overall Status, Sort by: Name, Perspective: Overall Status
- Project Summary:** PUBLIC, Last analysis: November 6, 2021, 1:39 PM, 1 project.
- Metrics:** 0 Bugs (A), 0 Vulnerabilities (A), 100% Hotspots Reviewed (A), 0 Code Smells (A), 0.0% Duplications (A), 70 PHP (xs).

At the bottom, there is a copyright notice: © 2008-2021, SonarSource S.A, Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCLOUD are trademarks of SonarSource SA. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved.

SonarCloud (SonarSource) → „Alles supi!“ #nicht

ganz anderes Projekt



The screenshot shows the RIPS Tech Scanner interface. On the left, a sidebar lists issues found in "TYPO3 CMS (master)" with the commit hash "872e0ce0df3252cf6ead4548d...". The first issue is "Phar Deserialization (4)". The main panel displays the details for "Phar Deserialization" (Issue Id: 17020). It describes a POST parameter 'install[values][socket]' being received in line 478 of "typo3/sysext/install/Classes/Controller/InstallerController.php" and used unsanitized in "file_exists" in line 542. The code snippet shows the relevant PHP code. The right sidebar provides additional context, including severity (High), OWASP, CWE, SANS, and PCI DSS references, and a list of related articles.

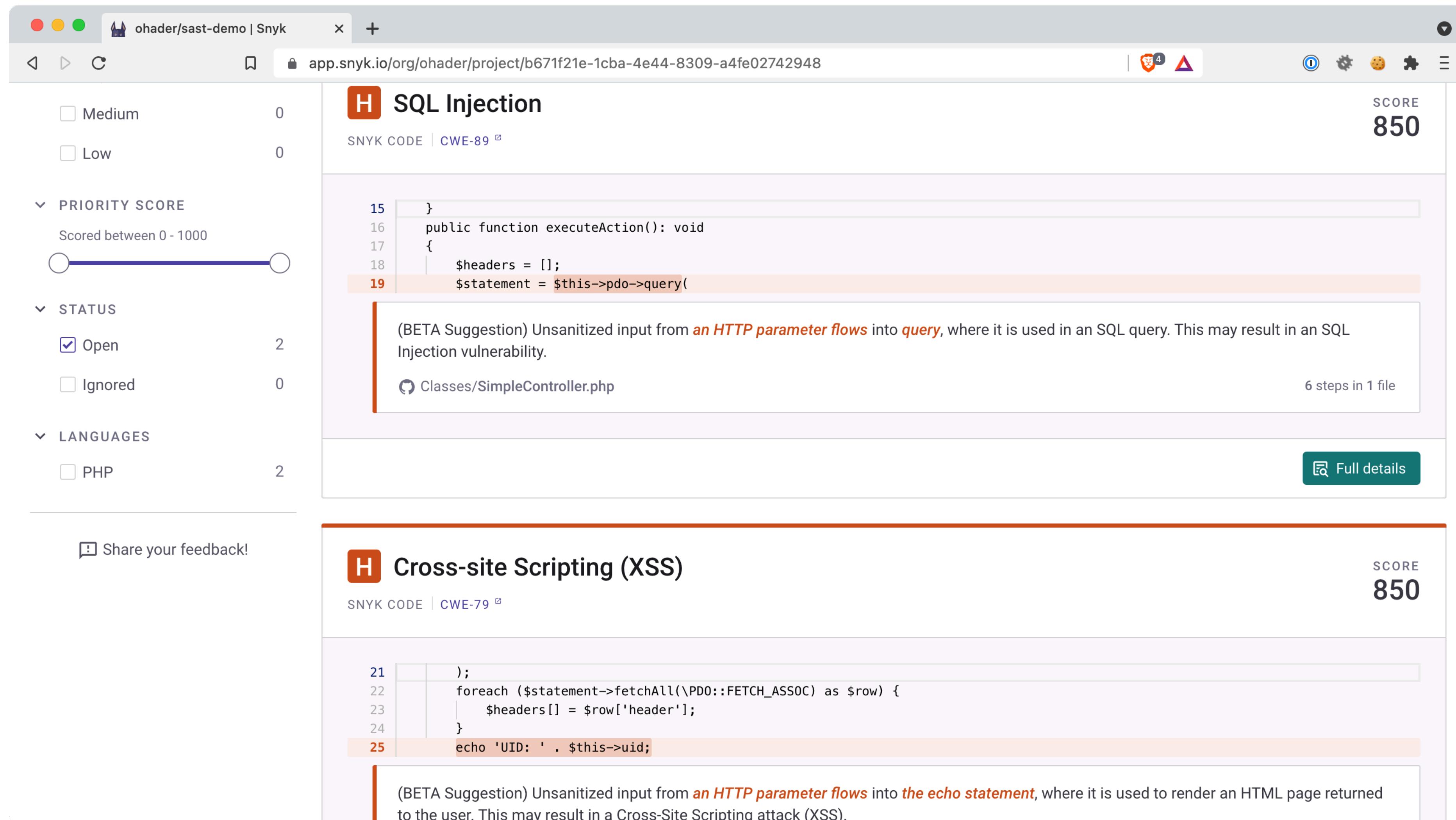
Issue Type: Phar Deserialization
Server-side Vulnerability
Severity: High

OWASP: 2010: A4 **CWE:** 915
 2013: A4 **SANS 25:** 16
 2017: A8 **PCI DSS:** 6.5.8
ASVS: 3.0.1: 16.4 4.0.1: n/a

- PHAR Deserialization Explained
- Shocking News in PHP Exploitation
- Utilizing Code Reuse/ROP in PHP Application Exploits
- Code Reuse Attacks in PHP: Automated POP Chain Generation
- Real-world Examples

All file operations in PHP allow to use URL-style wrappers when accessing file paths. An attacker can misuse these wrappers when user input fully controls the file path. In this case, the phar:// wrapper can be injected which allows to load a given file as PHP

RIPS Tech Scanner → übernommen von SonarSource



The screenshot shows the Snyk.io interface with two main findings:

- SQL Injection**: Score 850. Found in `Classes/SimpleController.php` at line 19. The code snippet is:

```
15 }
16     public function executeAction(): void
17     {
18         $headers = [];
19         $statement = $this->pdo->query(
```

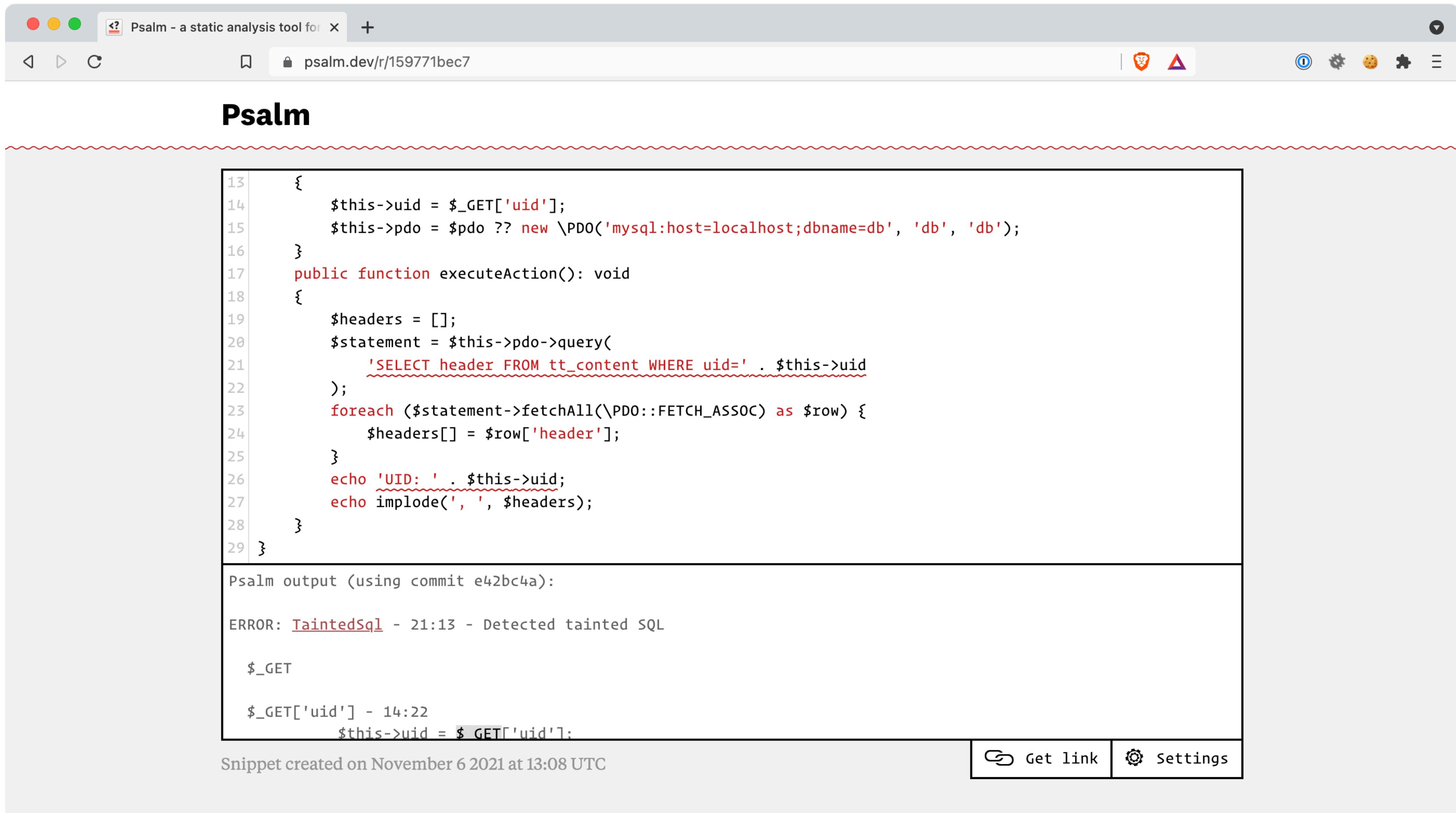
A tooltip provides a BETA suggestion: "Unsanitized input from *an HTTP parameter flows* into *query*, where it is used in an SQL query. This may result in an SQL Injection vulnerability."
- Cross-site Scripting (XSS)**: Score 850. Found in `Classes/SimpleController.php` at line 25. The code snippet is:

```
21 );
22     foreach ($statement->fetchAll(\PDO::FETCH_ASSOC) as $row) {
23         $headers[] = $row['header'];
24     }
25     echo 'UID: ' . $this->uid;
```

A tooltip provides a BETA suggestion: "Unsanitized input from *an HTTP parameter flows* into *the echo statement*, where it is used to render an HTML page returned to the user. This may result in a Cross-Site Scripting attack (XSS)."

On the left sidebar, there are filters for Medium (0), Low (0), Priority Score (Scored between 0 - 1000), Status (Open checked, Ignored 0), and Languages (PHP 2).

DeepCode.ai → übernommen von Snyk.io → Schwachstellen gefunden



The screenshot shows a web-based interface for Psalm, a static analysis tool. The title bar reads "Psalm - a static analysis tool for PHP". The URL in the address bar is "psalm.dev/r/159771bec7". The main content area is titled "Psalm" and contains a code editor with the following PHP code:

```
13     {
14         $this->uid = $_GET['uid'];
15         $this->pdo = $pdo ?? new \PDO('mysql:host=localhost;dbname=db', 'db', 'db');
16     }
17     public function executeAction(): void
18     {
19         $headers = [];
20         $statement = $this->pdo->query(
21             'SELECT header FROM tt_content WHERE uid=' . $this->uid
22         );
23         foreach ($statement->fetchAll(\PDO::FETCH_ASSOC) as $row) {
24             $headers[] = $row['header'];
25         }
26         echo 'UID: ' . $this->uid;
27         echo implode(', ', $headers);
28     }
29 }
```

Below the code editor, a section titled "Psalm output (using commit e42bc4a):" displays the analysis results:

```
ERROR: TaintedSql - 21:13 - Detected tainted SQL

$_GET

$_GET['uid'] - 14:22
    $this->uid = $_GET['uid'];
```

At the bottom left, it says "Snippet created on November 6 2021 at 13:08 UTC". At the bottom right are "Get link" and "Settings" buttons.

PsalmPHP (<https://psalm.dev/r/159771bec7>) → Schwachstellen gefunden

- **SonarSource**, SonarCloud, SonarCube - nichts gefunden 😵
- **RIPS Tech Scanner** - Schwachstellen vermutlich gefunden 😐
 - im Mai 2020 von **SonarSource** übernommen
- **DeepCode.ai** - Schwachstellen gefunden 😅
 - im August 2020 von **Snyk.io** übernommen
- **PsalmPHP** - Schwachstellen gefunden 😅

Demo Anwendung (realistischer)



```
<?php
declare(strict_types=1);
namespace Olly\SastDemo;

use Psr\Http\Message\ServerRequestInterface;
use TYPO3\CMS\Core\Database\ConnectionPool;
use TYPO3\CMS\Core\Utility\GeneralUtility;
use TYPO3\CMS\Fluid\View\StandaloneView;

class ModernController
{
    /** @var ConnectionPool */
    private $connectionPool;
    /** @var StandaloneView */
    private $view;

    public function __construct()
    {
        $this->connectionPool = GeneralUtility::makeInstance(ConnectionPool::class);
        $this->view = GeneralUtility::makeInstance(StandaloneView::class);
    }
    public function executeAction(ServerRequestInterface $request): void
    {
        $headers = [];
        $uid = $request->getQueryParams()['uid'];
        $statement = $this->connectionPool
            ->getConnectionForTable('tt_content')
            ->select(['header'], 'tt_content', ['uid' => $uid]);
        foreach ($statement->getIterator() as $row) {
            $headers[] = $row['header'];
        }
        echo 'UID: ' . $uid;
        echo implode(', ', $headers);
        $this->view->assign('uid', $uid);
        $this->view->assign('headers', $headers);
    }
}
```

The screenshot shows a GitHub repository page for 'ohader/sast-demo'. The repository is public, has 1 pull request, 0 stars, and 0 forks. The 'Code' tab is selected. A list of recent commits is shown:

- ohader Help SonarSource a lit... 4 minutes ago
- Classes Help SonarSource a little bit... 4 minutes ago
- README.md Add README.md 11 minutes ago
- composer.j... First and probably the last c... 29 minutes ago
- ext_emcon... First and probably the last c... 29 minutes ago
- ext_icon.png First and probably the last c... 29 minutes ago

<https://github.com/ohader/sast-demo>

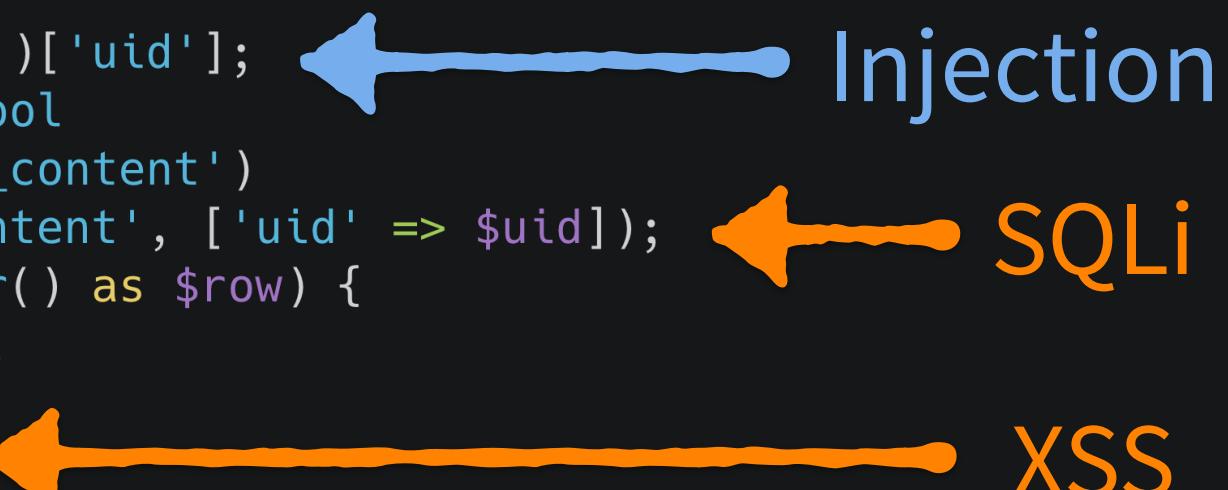


```
<?php
declare(strict_types=1);
namespace Olly\SastDemo;

use Psr\Http\Message\ServerRequestInterface;
use TYPO3\CMS\Core\Database\ConnectionPool;
use TYPO3\CMS\Core\Utility\GeneralUtility;
use TYPO3\CMS\Fluid\View\StandaloneView;

class ModernController
{
    /** @var ConnectionPool */
    private $connectionPool;
    /** @var StandaloneView */
    private $view;

    public function __construct()
    {
        $this->connectionPool = GeneralUtility::makeInstance(ConnectionPool::class);
        $this->view = GeneralUtility::makeInstance(StandaloneView::class);
    }
    public function executeAction(ServerRequestInterface $request): void
    {
        $headers = [];
        $uid = $request->getQueryParams()['uid'];
        $statement = $this->connectionPool
            ->getConnectionForTable('tt_content')
            ->select(['header'], 'tt_content', ['uid' => $uid]);
        foreach ($statement->getIterator() as $row) {
            $headers[] = $row['header'];
        }
        echo 'UID: ' . $uid;
        echo implode(', ', $headers);
        $this->view->assign('uid', $uid);
        $this->view->assign('headers', $headers);
    }
}
```

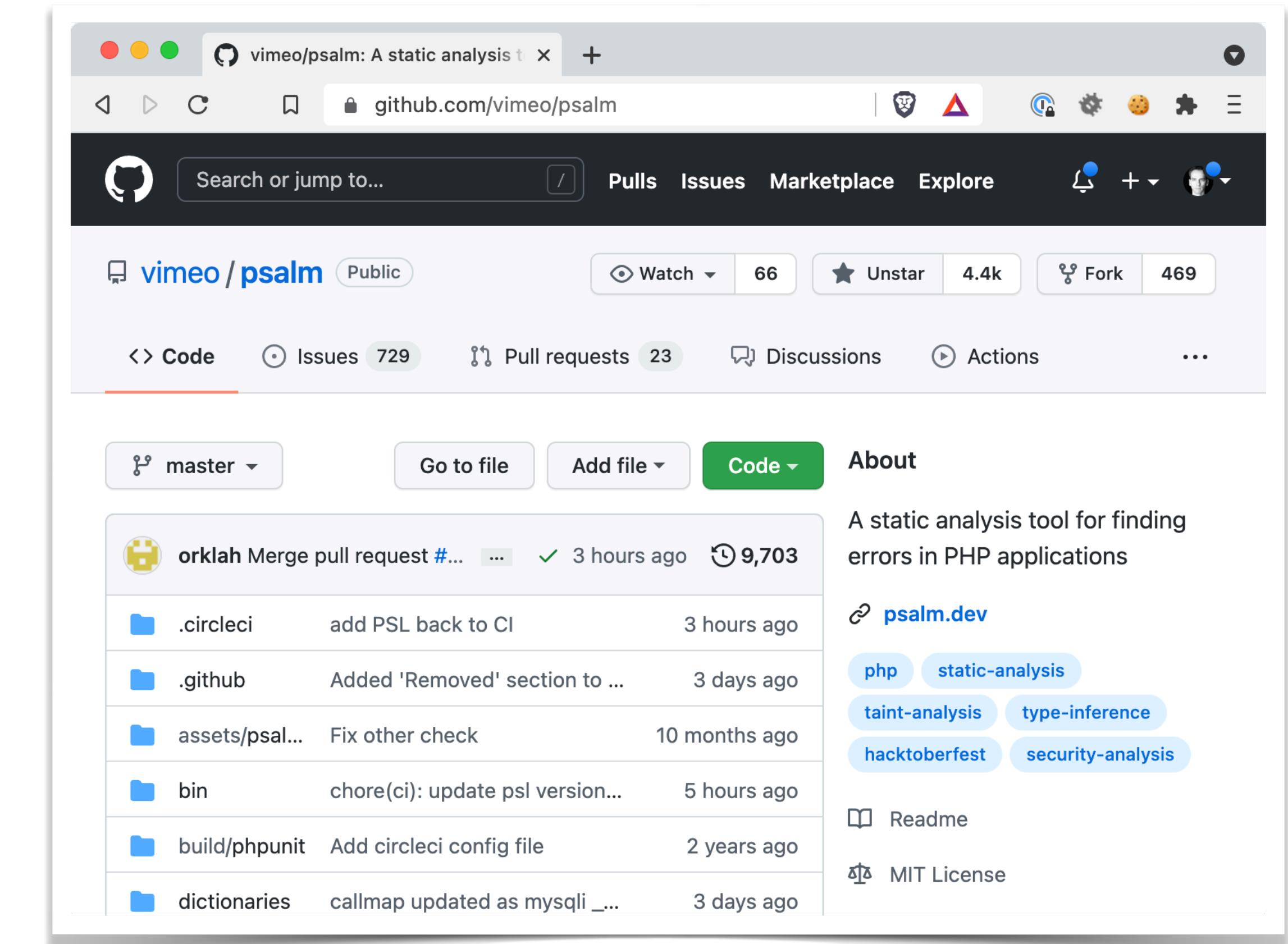


- Injection: Points to the line `$uid = $request->getQueryParams()['uid'];`
- SQLi: Points to the line `$statement = $this->connectionPool
 ->getConnectionForTable('tt_content')
 ->select(['header'], 'tt_content', ['uid' => $uid]);`
- XSS: Points to the line `echo implode(', ', $headers);`

- TYPO3 ist **speziell!**
- fehlende **Unterstützung** für **Framework Komponenten**
- **PSR: Request/Response**
- **PSR: Dependency Injection**
- **Fluid: Templating Engine**
- **DBAL: Datenbank Schicht**
- **TYPO3: Hooks, TypoScript**

#nähkästchen Verwendung für TYPO3

- **PsalmPHP für TYPO3**
- **Open-Source Project (MIT)**
- **Implementiert in PHP**
- **Erweiterbar durch Plugins**
- **unterstützt sources, sinks, escapes, unescapes, flow**



```
● ● ●  
  
class ModernController  
{  
    /** @var ConnectionPool */  
    private $connectionPool;  
    /** @var StandaloneView */  
    private $view;  
  
    public function __construct()  
    {  
        $this->connectionPool = GeneralUtility::makeInstance(  
            ConnectionPool::class);  
        $this->view = GeneralUtility::makeInstance(  
            StandaloneView::class);  
    }  
    public function executeAction(ServerRequestInterface $request): void  
    {  
        $headers = [];  
        $uid = $request->getQueryParams()['uid'];  
        $statement = $this->connectionPool  
            ->getConnectionForTable('tt_content')  
            ->select(['header'], 'tt_content', ['uid' => $uid]);  
        foreach ($statement->getIterator() as $row) {  
            $headers[] = $row['header'];  
        }  
        $this->view->assign('uid', $uid);  
        $this->view->assign('headers', $headers);  
    }  
}
```

```
● ● ●  
  
<?php  
namespace TYP03\CMS\Core\Utility;  
  
class GeneralUtility  
{  
    /**  
     * @template T  
     * @psalm-param class-string<T> $className  
     * @psalm-pure  
     * @return T  
     */  
    public static function makeInstance($className) {}
```

- **@template**
- **liefert Instanz von \$className**

```
● ● ●  
  
class ModernController  
{  
    /** @var ConnectionPool */  
    private $connectionPool;  
    /** @var StandaloneView */  
    private $view;  
  
    public function __construct()  
    {  
        $this->connectionPool = GeneralUtility::makeInstance(  
            ConnectionPool::class);  
        $this->view = GeneralUtility::makeInstance(  
            StandaloneView::class);  
    }  
    public function executeAction(ServerRequestInterface $request): void  
    {  
        $headers = [];  
        $uid = $request->getQueryParams()['uid'];  
        $statement = $this->connectionPool  
            ->getConnectionForTable('tt_content')  
            ->select(['header'], 'tt_content', ['uid' => $uid]);  
        foreach ($statement->getIterator() as $row) {  
            $headers[] = $row['header'];  
        }  
        $this->view->assign('uid', $uid);  
        $this->view->assign('headers', $headers);  
    }  
}
```

```
● ● ●  
  
<?php  
namespace Psr\Http\Message;  
  
interface ServerRequestInterface extends RequestInt  
{  
    /**  
     * @psalm-taint-source input  
     * @return array<string, mixed>  
     */  
    public function getqueryParams();
```

- **@psalm-taint-source**
- Rückgabewerte sind **Benutzereingaben**
- wie **\$_GET, \$_POST, \$_COOKIE, HTTP Header**

```
● ● ●  
class ModernController  
{  
    /** @var ConnectionPool */  
    private $connectionPool;  
    /** @var StandaloneView */  
    private $view;  
  
    public function __construct()  
{  
        $this->connectionPool = GeneralUtility::makeInstance(  
            ConnectionPool::class);  
        $this->view = GeneralUtility::makeInstance(  
            StandaloneView::class);  
    }  
    public function executeAction(ServerRequestInterface $request): void  
{  
        $headers = [];  
        $uid = $request->getQueryParams()['uid'];  
        $statement = $this->connectionPool  
            ->getConnectionForTable('tt_content')  
            ->select(['header'], 'tt_content', ['uid' => $uid]);  
        foreach ($statement->getIterator() as $row) {  
            $headers[] = $row['header'];  
        }  
        $this->view->assign('uid', $uid);  
        $this->view->assign('headers', $headers);  
    }  
}
```

```
● ● ●  
<?php  
namespace Doctrine\DBAL;  
  
use Doctrine\DBAL\Query\Expression\ExpressionBuilder;  
use Doctrine\DBAL\Query\QueryBuilder;  
  
class Connection  
{  
    /**  
     * @param string $query  
     * @return \Doctrine\DBAL\Driver\Statement  
     * @psalm-taint-sink sql $query  
     * @psalm-taint-specialize  
     */  
    public function query(string $query) {}
```

- **@psalm-taint-sink**
- verwundbar in Aspekt **sql**,
wenn Parameter **\$query**
aus unsicherer
@psalm-taint-source

```
● ● ●

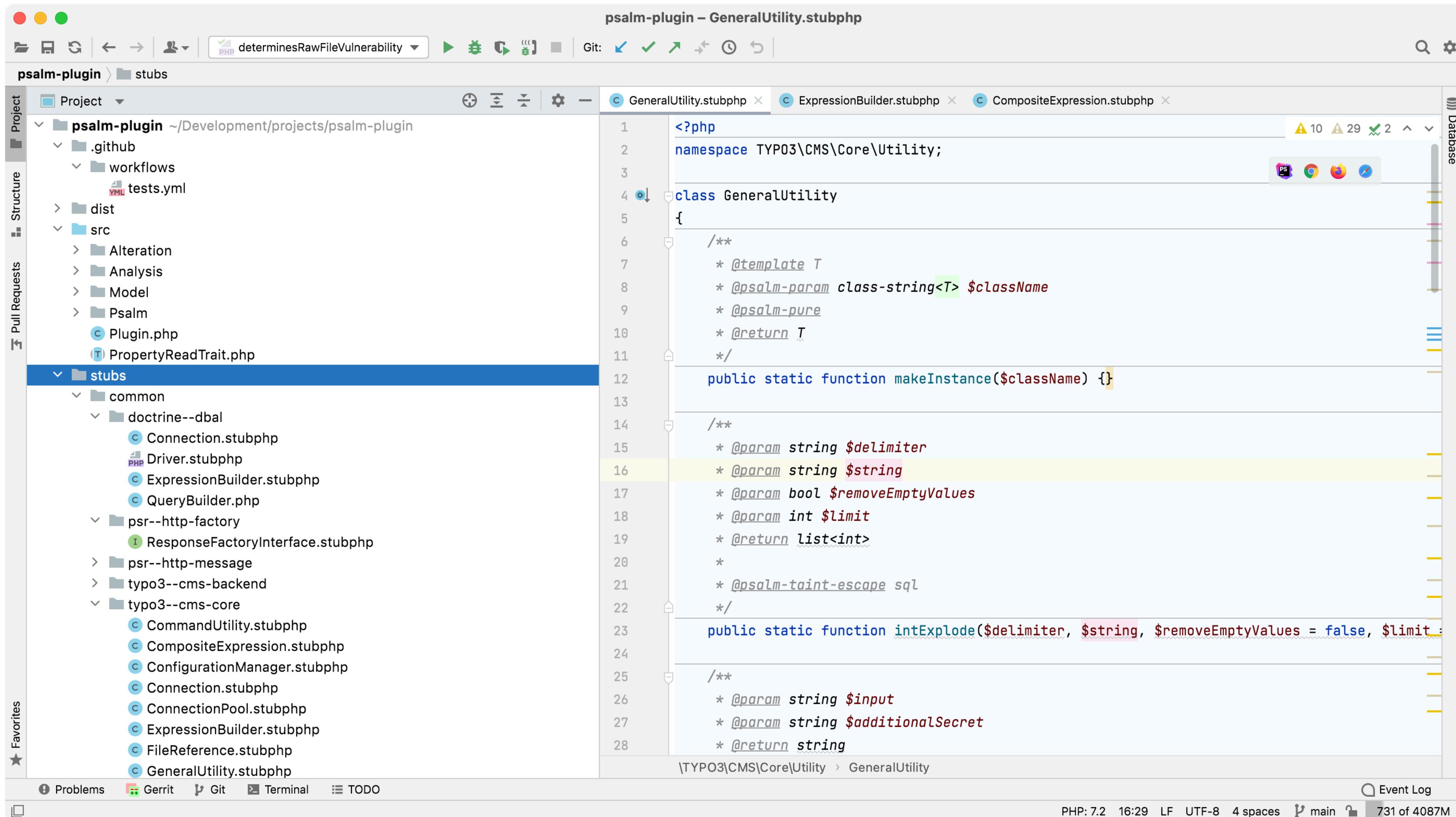
<?php
namespace Doctrine\DBAL;

use Doctrine\DBAL\Query\Expression\ExpressionBuilder;
use Doctrine\DBAL\Query\QueryBuilder;

class Connection
{
    /**
     * @param scalar $str
     * @psalm-taint-escape sql
     * @psalm-taint-specialize
     */
    public function quoteIdentifier($str) {}

    /**
     * @param scalar $value
     * @param int $type
     * @psalm-taint-escape sql
     * @psalm-taint-specialize
     */
    public function quote($value, $type = ParameterType::STRING) {}
```

- **@psalm-taint-escape**
- markiert Parameter **\$str** für Aspekt **sql** als **bereinigt**
- **@psalm-taint-unescape**
- ... das Gegenteil (unsicher)



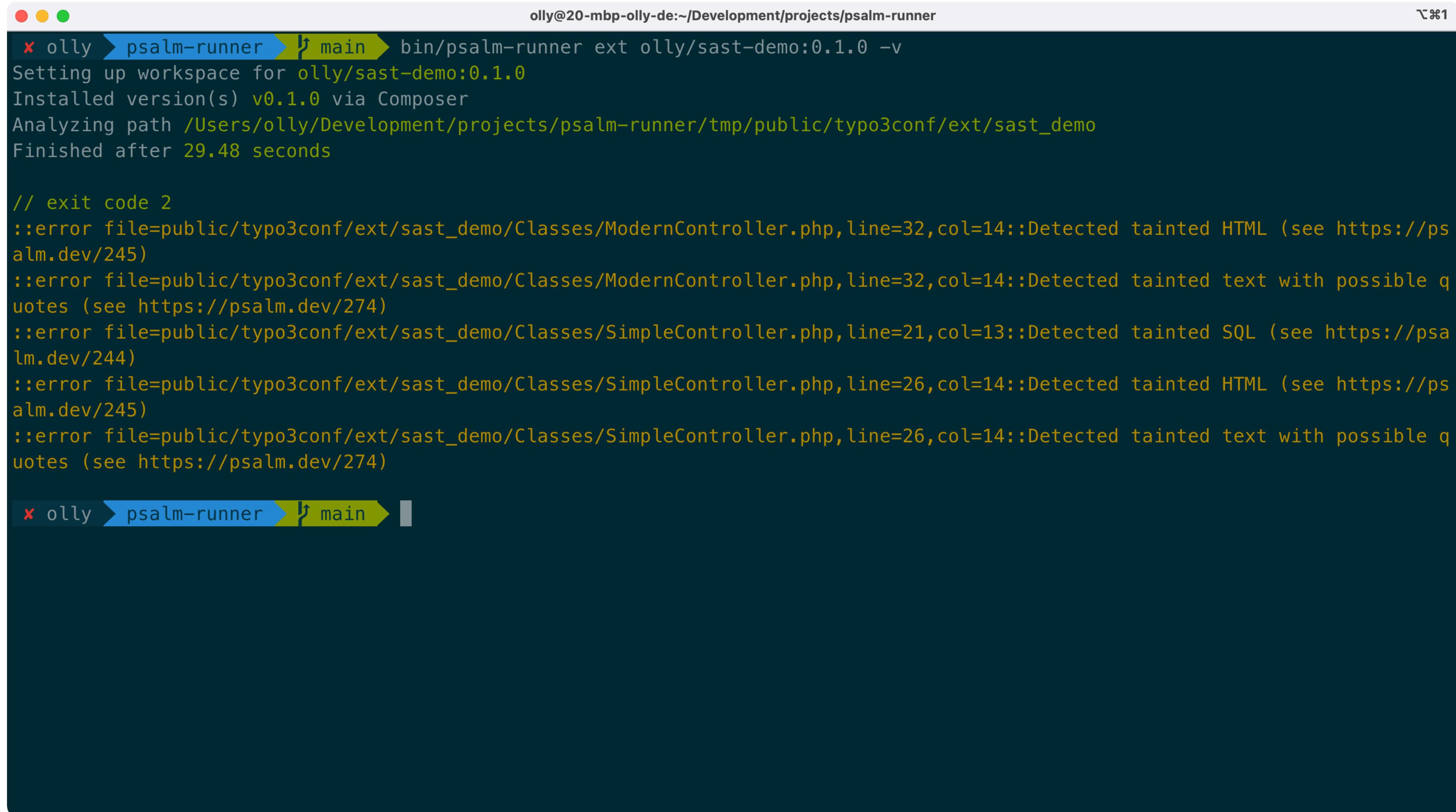
```
<?php
namespace TYPO3\CMS\Core\Utility;

class GeneralUtility
{
    /**
     * @template T
     * @psalm-param class-string<T> $className
     * @psalm-pure
     * @return T
     */
    public static function makeInstance($className) {}

    /**
     * @param string $delimiter
     * @param string $string
     * @param bool $removeEmptyValues
     * @param int $limit
     * @return list<int>
     *
     * @psalm-taint-escape sql
     */
    public static function intExplode($delimiter, $string, $removeEmptyValues = false, $limit = 0) {}

    /**
     * @param string $input
     * @param string $additionalSecret
     * @return string
     */
}
```

typo3-security/psalm-plugin - nicht öffentlich



The screenshot shows a terminal window with the following content:

```
olly@20-mbp-olly-de:~/Development/projects/psalm-runner
x oly ➤ psalm-runner ➤ ↴ main ➤ bin/psalm-runner ext olly/sast-demo:0.1.0 -v
Setting up workspace for olly/sast-demo:0.1.0
Installed version(s) v0.1.0 via Composer
Analyzing path /Users/olly/Development/projects/psalm-runner/tmp/public/typo3conf/ext/sast_demo
Finished after 29.48 seconds

// exit code 2
::error file=public/typo3conf/ext/sast_demo/Classes/ModernController.php,line=32,col=14::Detected tainted HTML (see https://psalm.dev/245)
::error file=public/typo3conf/ext/sast_demo/Classes/ModernController.php,line=32,col=14::Detected tainted text with possible quotes (see https://psalm.dev/274)
::error file=public/typo3conf/ext/sast_demo/Classes/SimpleController.php,line=21,col=13::Detected tainted SQL (see https://psalm.dev/244)
::error file=public/typo3conf/ext/sast_demo/Classes/SimpleController.php,line=26,col=14::Detected tainted HTML (see https://psalm.dev/245)
::error file=public/typo3conf/ext/sast_demo/Classes/SimpleController.php,line=26,col=14::Detected tainted text with possible quotes (see https://psalm.dev/274)

x oly ➤ psalm-runner ➤ ↴ main ➤ |
```

Kompakte Ausgabe

```
olly@20-mbp-olly-de:~/Development/projects/psalm-runner
olly ➤ psalm-runner ➤ ↵ main ➤ bin/psalm-runner ext olly/sast-demo:0.1.0 -v --format=console
Setting up workspace for olly/sast-demo:0.1.0
Installed version(s) v0.1.0 via Composer
Analyzing path /Users/olly/Development/projects/psalm-runner/tmp/public/typo3conf/ext/sast_demo
Finished after 29.91 seconds

// exit code 2

ERROR: TaintedHtml - public/typo3conf/ext/sast_demo/Classes/ModernController.php:32:14 - Detected tainted HTML (see https://psalm.dev/245)
    Psr\Http\Message\ServerRequestInterface::getqueryParams - ../vendor/typo3-security/psalm-plugin/stubs/common/psr--http-message/ServerRequestInterface.stubphp:10:21
        public function getqueryParams();

arrayvalue-fetch - public/typo3conf/ext/sast_demo/Classes/ModernController.php:25:16
    $uid = $request->getqueryParams()['uid'];

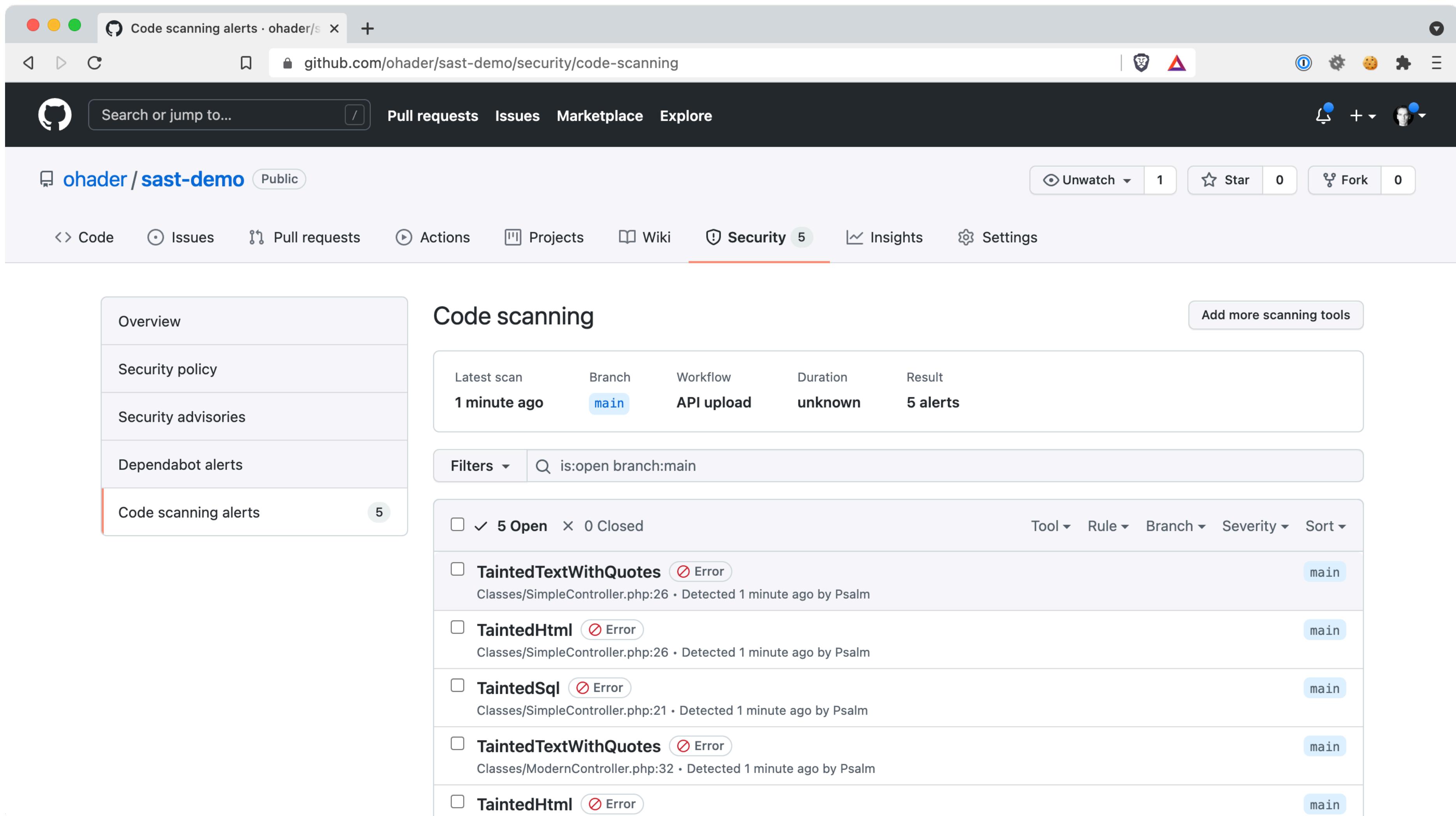
$uid - public/typo3conf/ext/sast_demo/Classes/ModernController.php:25:9
    $uid = $request->getqueryParams()['uid'];

concat - public/typo3conf/ext/sast_demo/Classes/ModernController.php:32:14
    echo 'UID: ' . $uid;

call to echo - public/typo3conf/ext/sast_demo/Classes/ModernController.php:32:14
    echo 'UID: ' . $uid;

ERROR: TaintedTextWithQuotes - public/typo3conf/ext/sast_demo/Classes/ModernController.php:32:14 - Detected tainted text with
possible quotes (see https://psalm.dev/274)
```

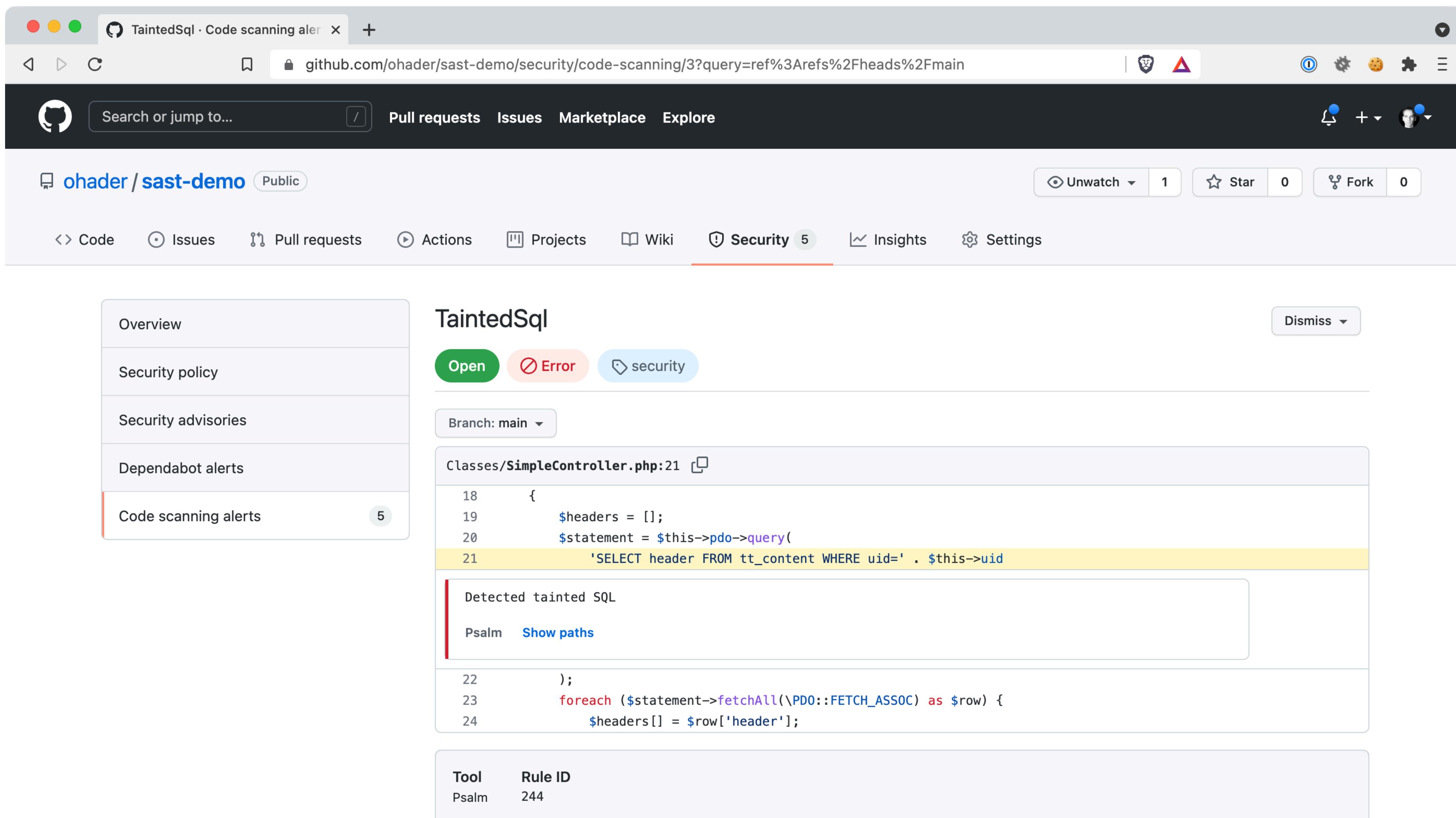
Ausführlichere Ausgabe



The screenshot shows a GitHub repository page for 'ohader/sast-demo'. The 'Security' tab is selected, displaying a 'Code scanning' section. The latest scan was performed 1 minute ago on the 'main' branch using the 'API upload' workflow, resulting in 5 alerts. A filter is applied to show only open alerts for the 'main' branch. The alerts listed are:

- TaintedTextWithQuotes (Error) - Classes/SimpleController.php:26 • Detected 1 minute ago by Psalm
- TaintedHtml (Error) - Classes/SimpleController.php:26 • Detected 1 minute ago by Psalm
- TaintedSql (Error) - Classes/SimpleController.php:21 • Detected 1 minute ago by Psalm
- TaintedTextWithQuotes (Error) - Classes/ModernController.php:32 • Detected 1 minute ago by Psalm
- TaintedHtml (Error)

SARIF Export & GitHub Integration

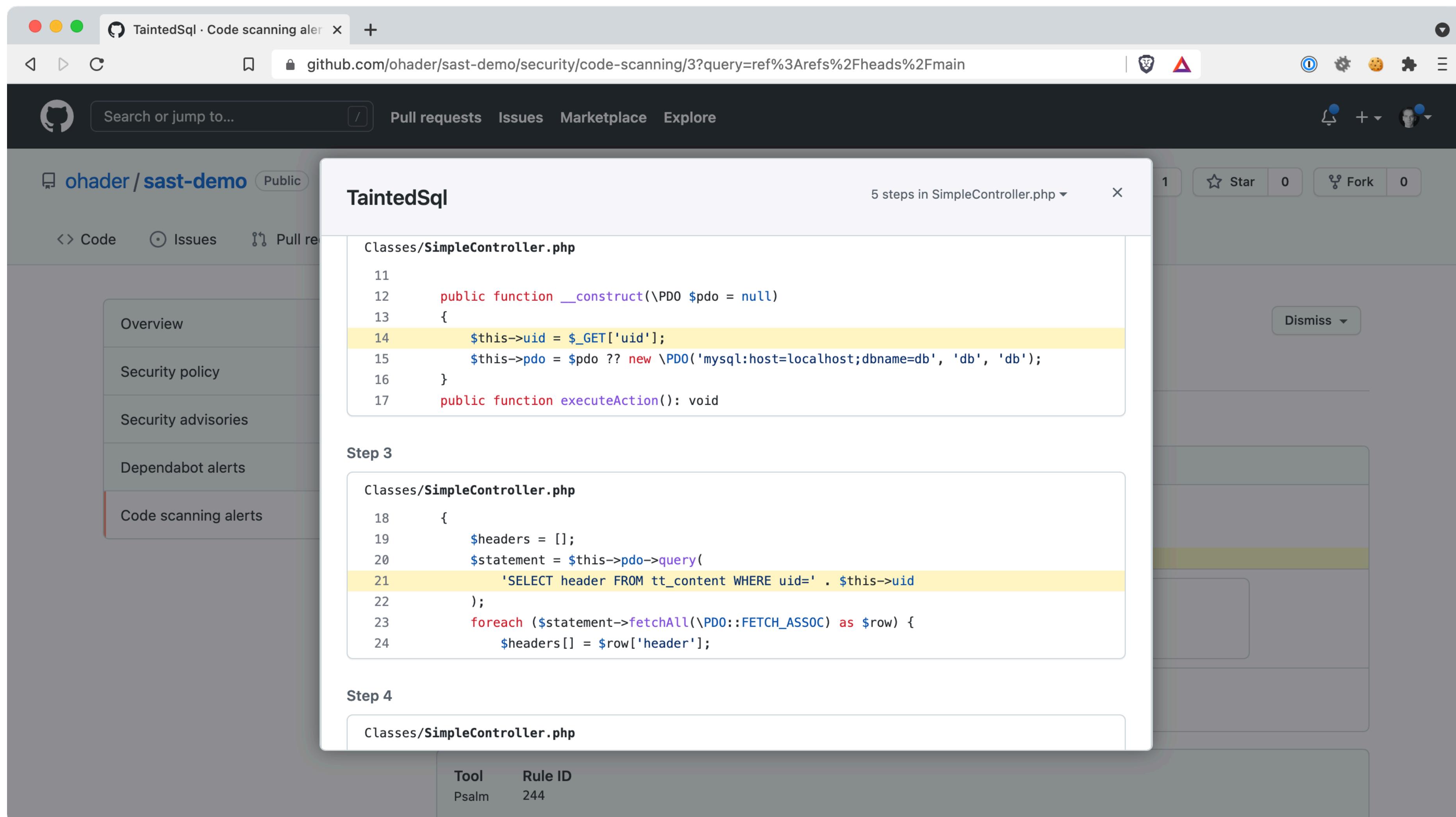


The screenshot shows a GitHub repository page for `ohader/sast-demo`. The user is viewing the `main` branch. The `Security` tab is selected, displaying 5 alerts. The first alert is for `TaintedSql`, which is an `Error` type related to `security`. The alert points to line 21 of `SimpleController.php`:

```
18     {
19         $headers = [];
20         $statement = $this->pdo->query(
21             'SELECT header FROM tt_content WHERE uid=' . $this->uid
```

A red box highlights the problematic line of code: `'SELECT header FROM tt_content WHERE uid=' . $this->uid`. Below the code, it says "Detected tainted SQL" and provides a "Show paths" link. The alert also includes a "Tool" section indicating "Psalm" and a "Rule ID" of 244.

SARIF Export & GitHub Integration



SARIF Export & GitHub Integration

#zukunft
MüssteMalJemand™

- **Erweiterung** der Stub-Deklarationen (Hooks, Fluid, TypoScript)
- **Erweiterung** für bekannte/bereinigte **Verwundbarkeiten**
- **Infrastruktur & Queue** für **regelmäßige** Extension Scans
- **Freigabe** für (echte) Extension Maintainer & Integration in **TER**
- (mögliche **Fundstellen** müssen natürlich auch **bereinigt** werden)

- TYPO3 Psalm-Plugin & Psalm-Runner vorerst **nicht öffentlich**
- evtl. **Public-Beta** über **Verified Extensions** (@VolkerGraubaum)
- **Entwicklung wird durch TYPO3 Association finanziert**
- **Interessierte/Supporter** bitte unter security@typo3.org melden

*thanks!
questions?*