SAST für TYPO3 Extensions #T3CRR21 #security

TYPO3camp RheinRuhr 2021
6. November 2021

Oliver Hader
oliver@typo3.org
@ohader

Oliver Hader

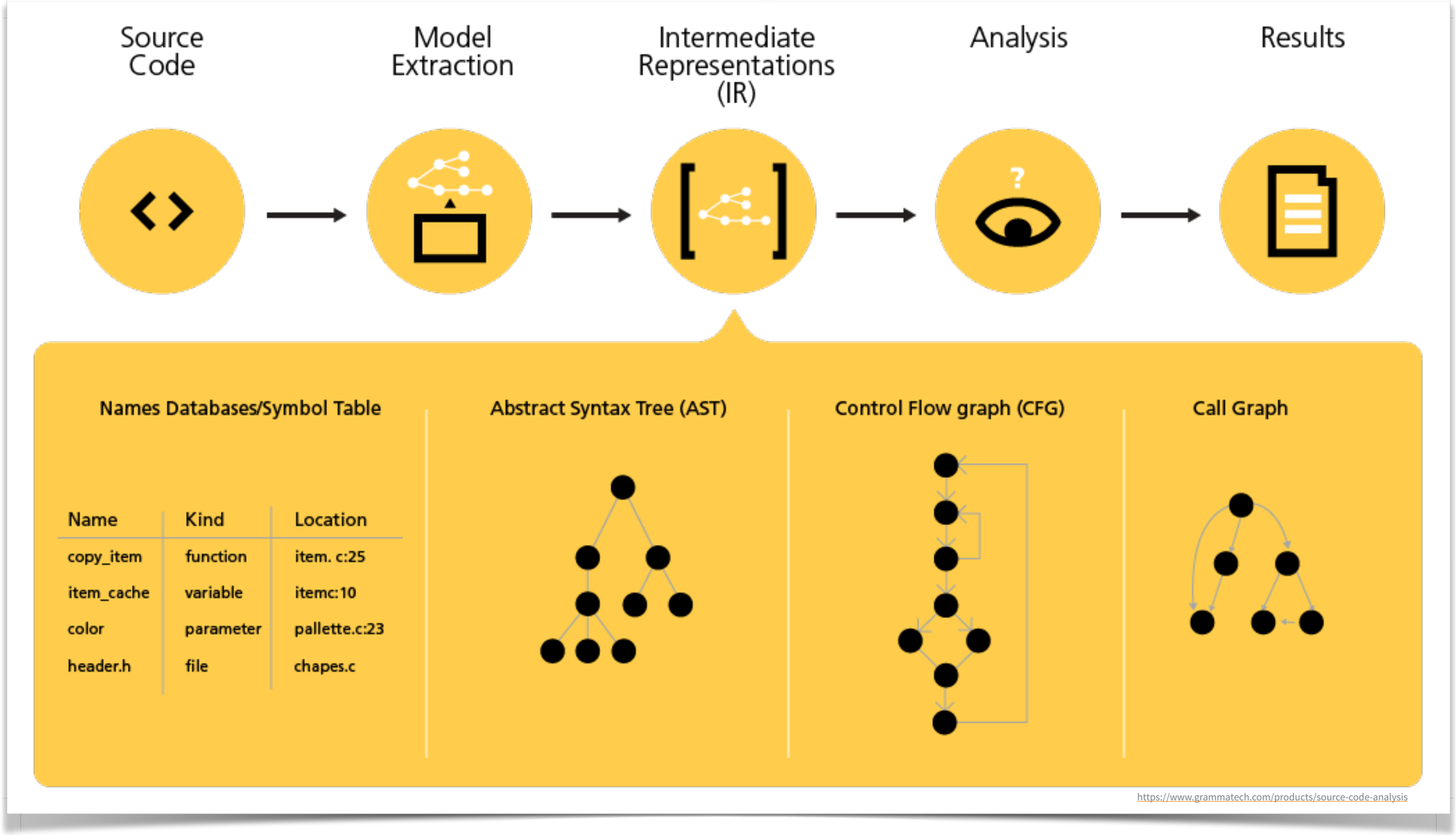
- Research & Development
- TYPO3 Security Team Lead
- 50% TYPO3 GmbH / 50% Freelancer
- #hof #cycling #paramedic #in.die.musik

@ohader

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

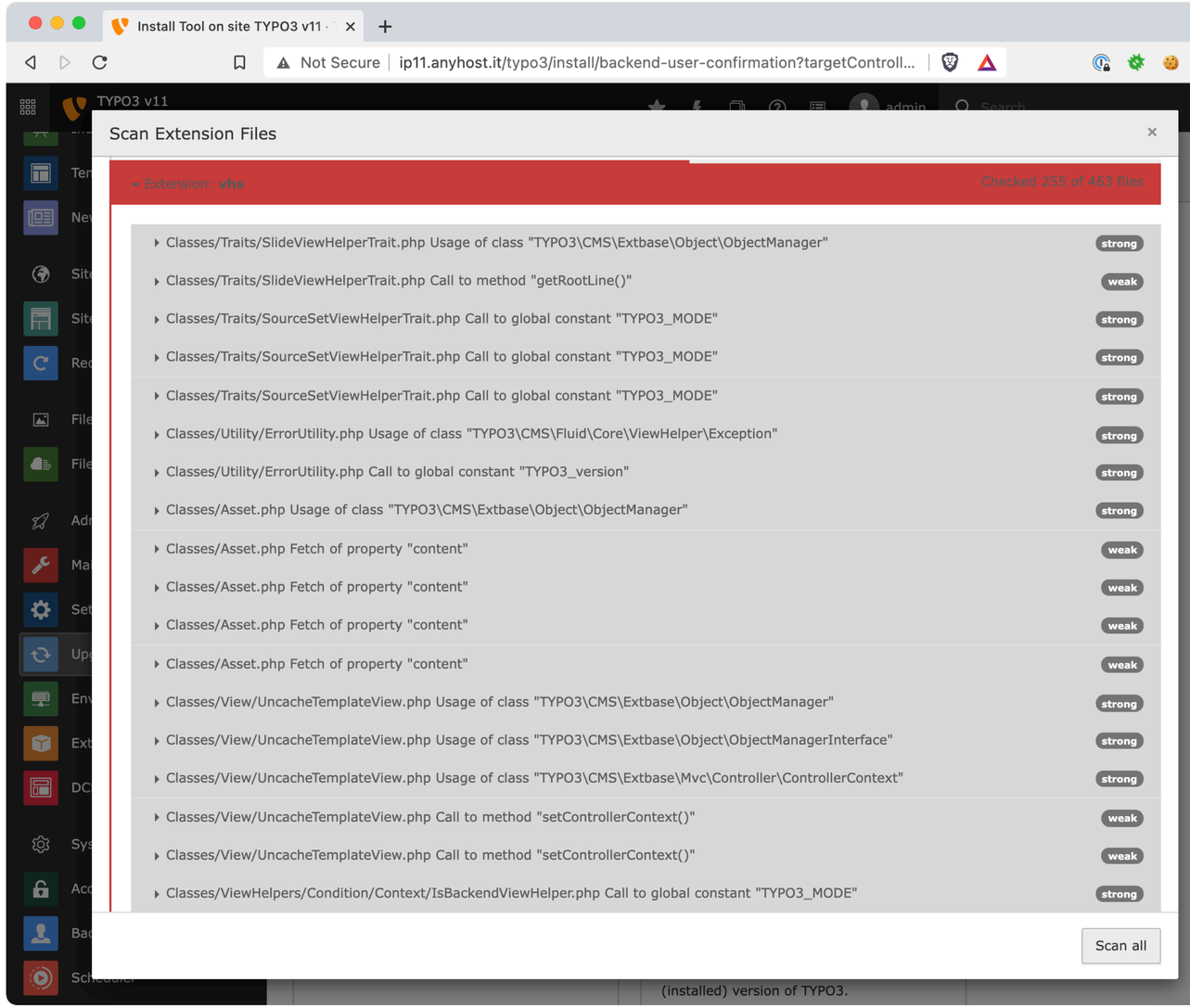
```php
<?php
function doTheMagic($value): string {
    if ($value === 'hello') {
        return 'Hello World';
    }
    return 0;
}
$value = (int)$_GET['value'];
echo doTheMagic($value);
```

```
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))
        )
    )
    1: Stmt_Expression(
```

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

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

- 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**	Schutz vor Täuschung und Betrug - Informationen ermöglichen Social Engineering-Angriffe - Gebrauch von Popups u.ä. erleichtern Phishing-Angriffe - Keine Absicherung für den Fall der Fälschung der Website
4	**Logik**	Absicherung von Prozessen und Workflows als Ganzes - 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**	Vermeiden von Programmierfehlern, die zu Schwachstellen führen - Cross-Site Scripting - SQL-Injection - Session Riding
2	**Technologie**	Richtige Wahl und sicherer Einsatz von Technologie - unverschlüsselte Übertragung sensitiver Daten - Authentisierungsverfahren, die nicht dem Schutzbedarf angemessen sind - Ungenügende Randomness von Token
1	**System**	Absicherung der auf der Systemplattform eingesetzten Software - 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

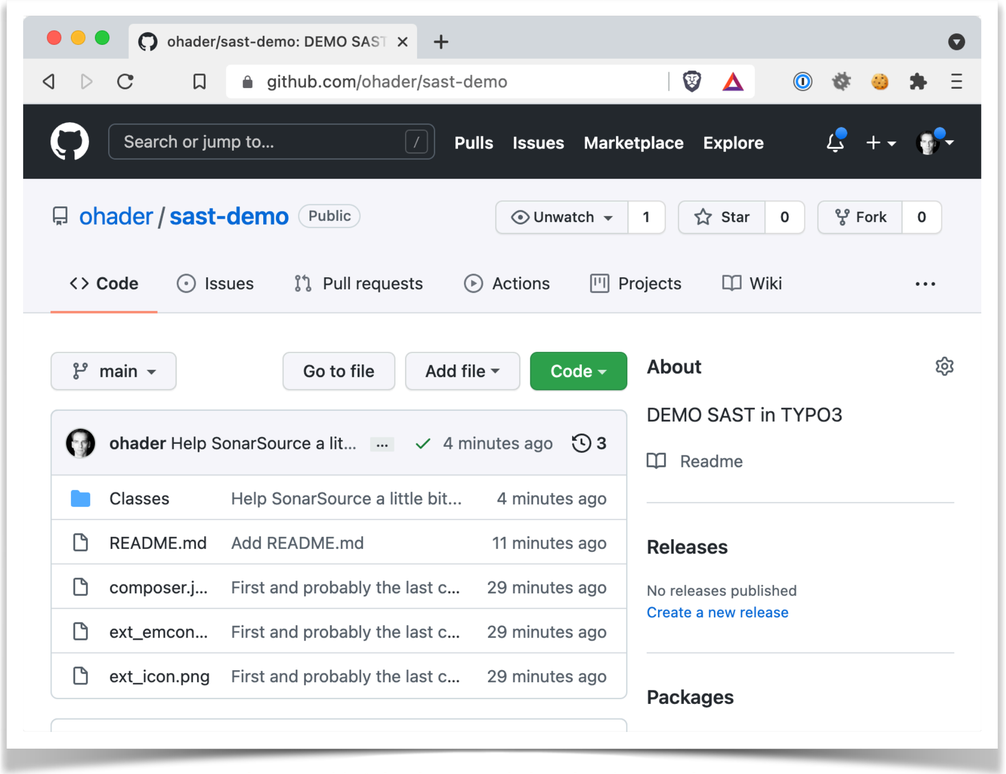
Version 1, August 2006

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/webanwendungen_node.html

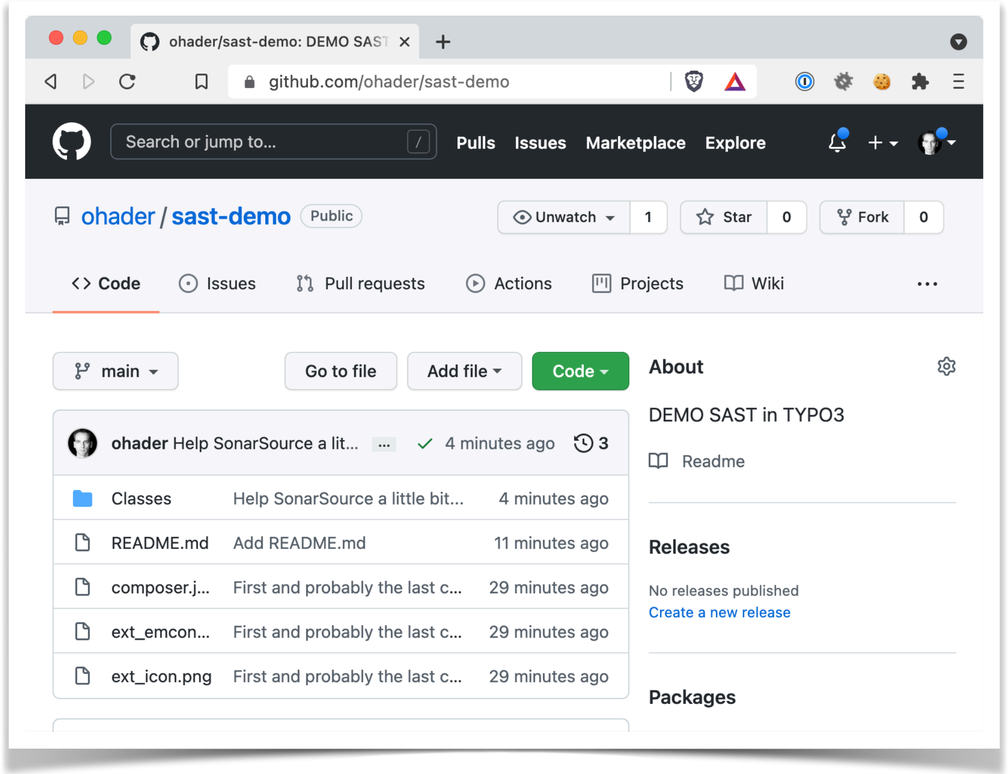
- **SonarSource**, SonarCloud, SonarCube
- **RIPS Tech Scanner**
- **DeepCode.ai**
- **PsalmPHP**



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

Demo Anwendung (vereinfacht)

```php
<?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);
    }
}
```



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

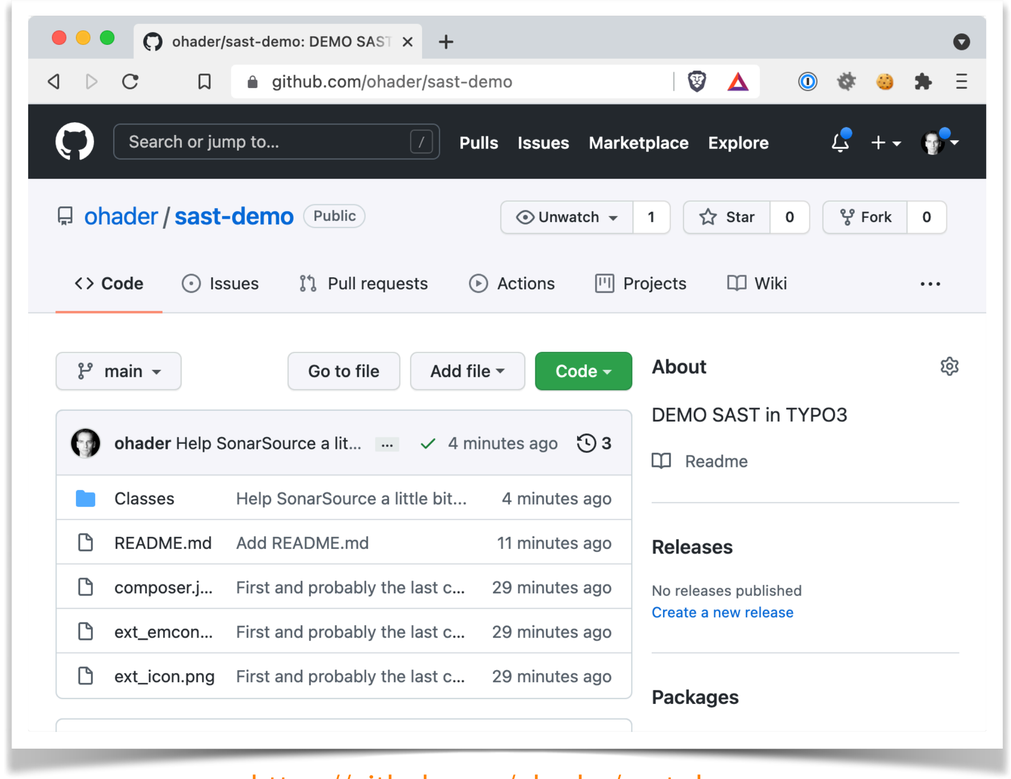
Demo Anwendung (vereinfacht)

```php
<?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);
    }
}
```

Benutzereingaben

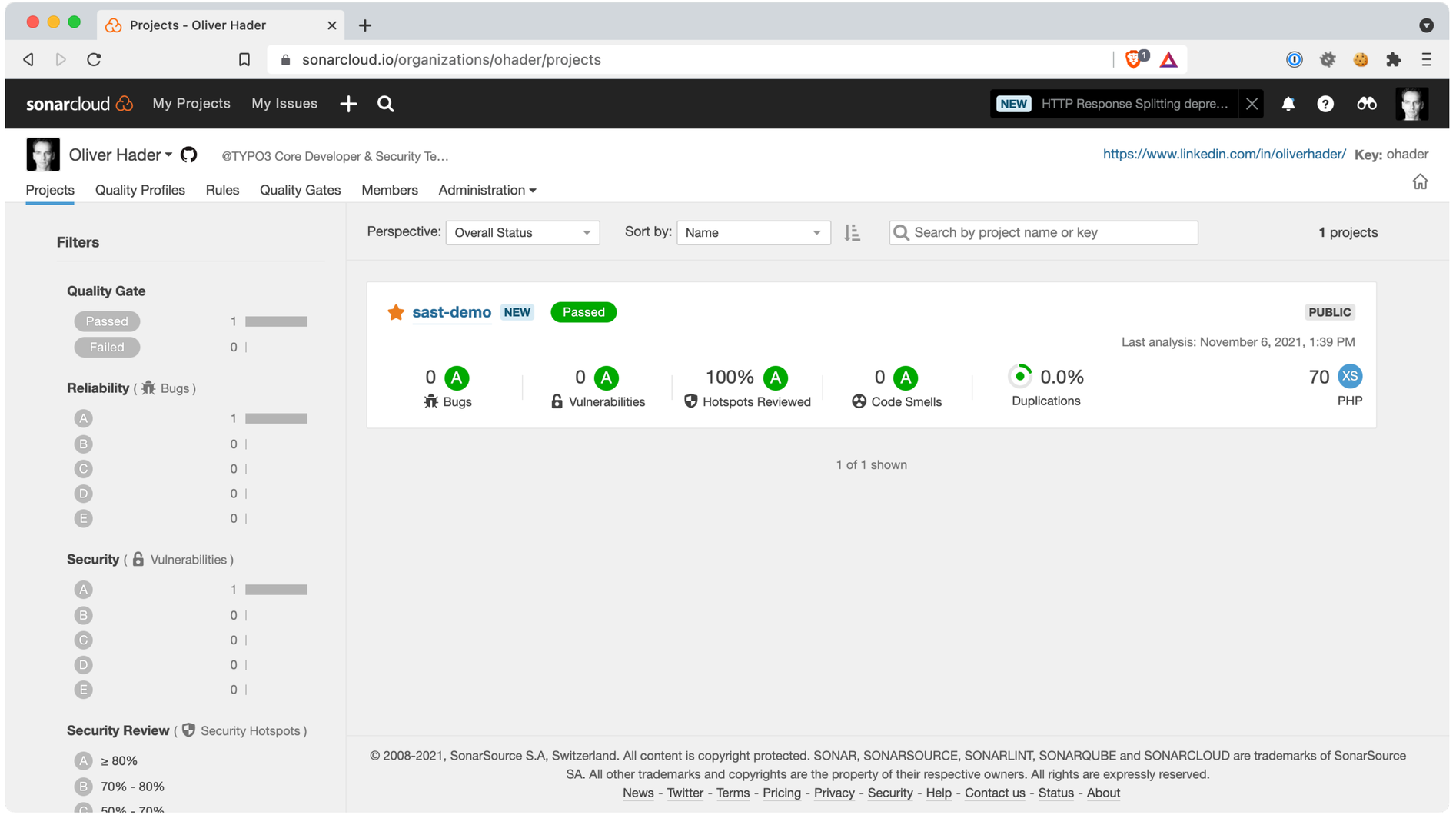
SQLi

XSS



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

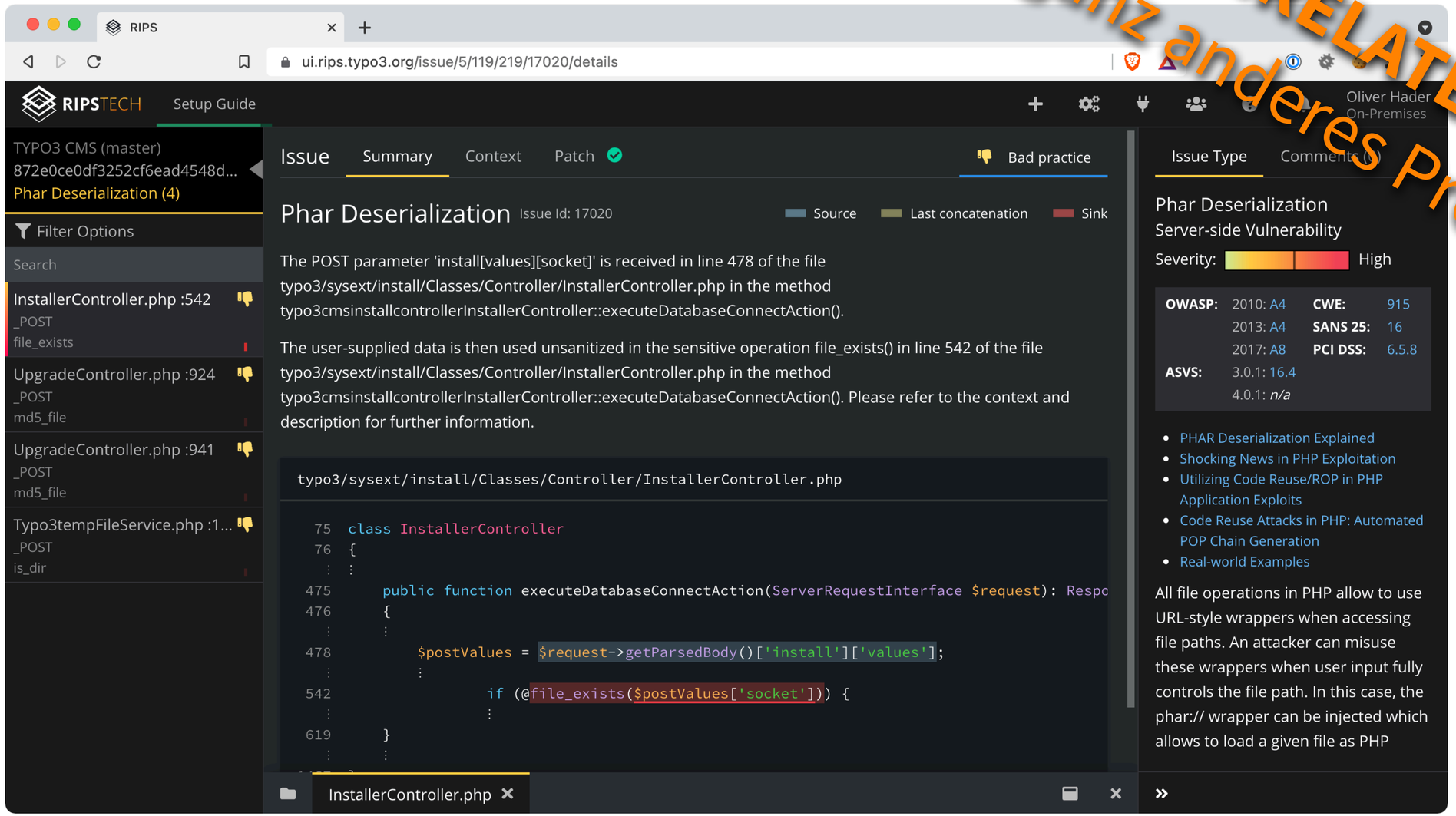
SonarCloud



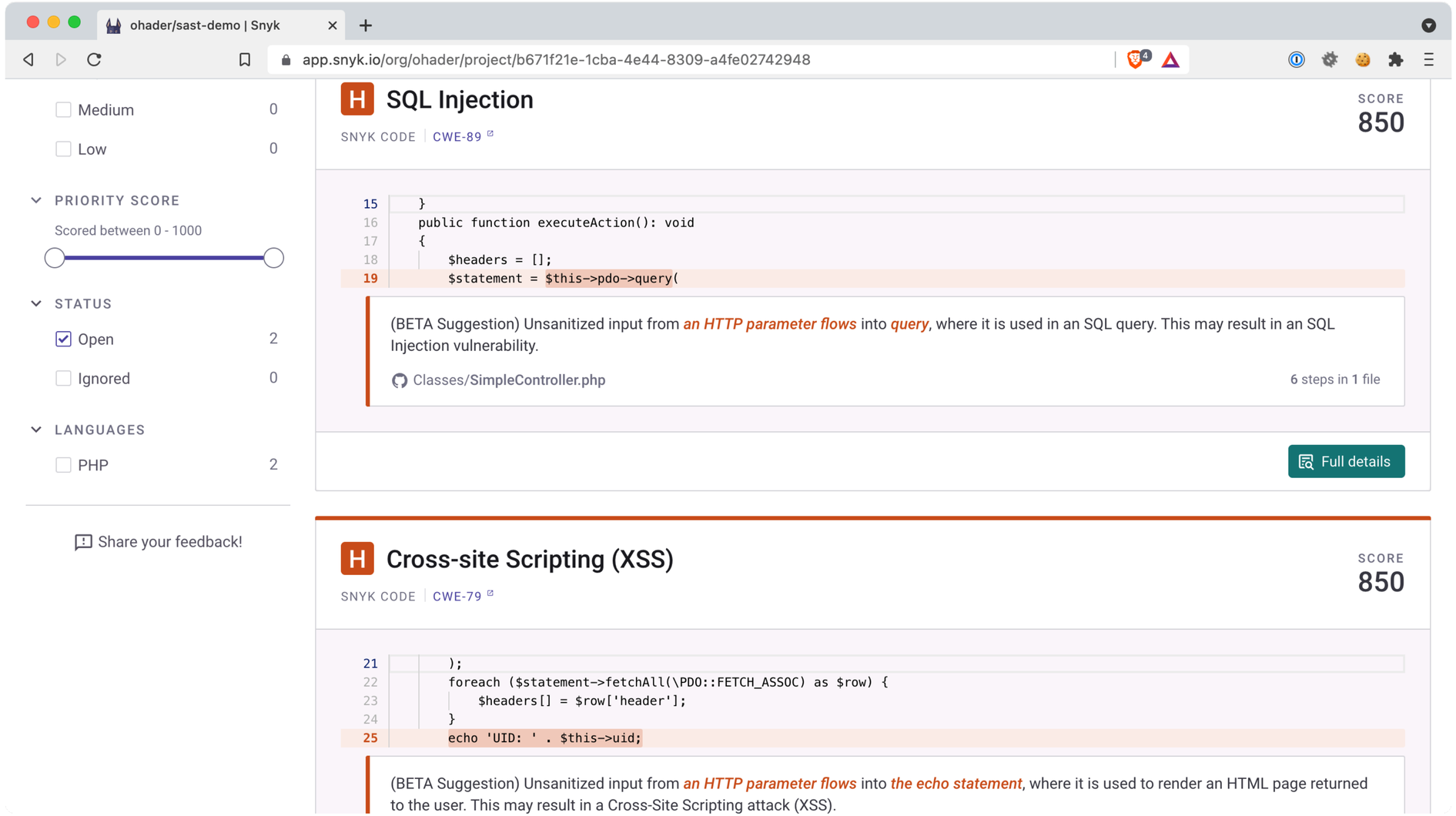
SonarCloud (SonarSource) → **„Alles supi!“** #nicht

RIPS Tech Scanner

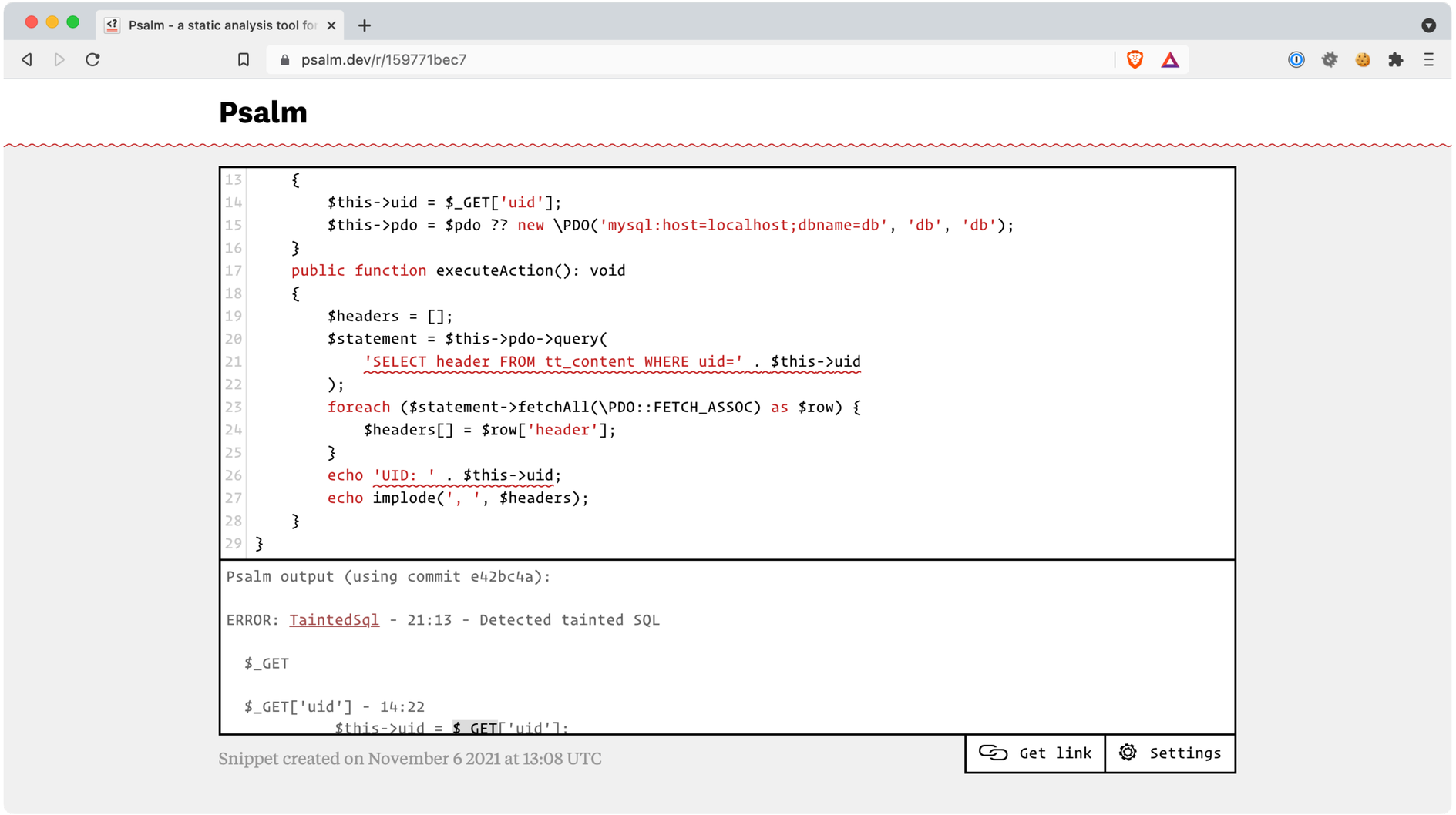
UNRELATED!
ganz anderes Projekt



RIPS Tech Scanner → übernommen von SonarSource



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

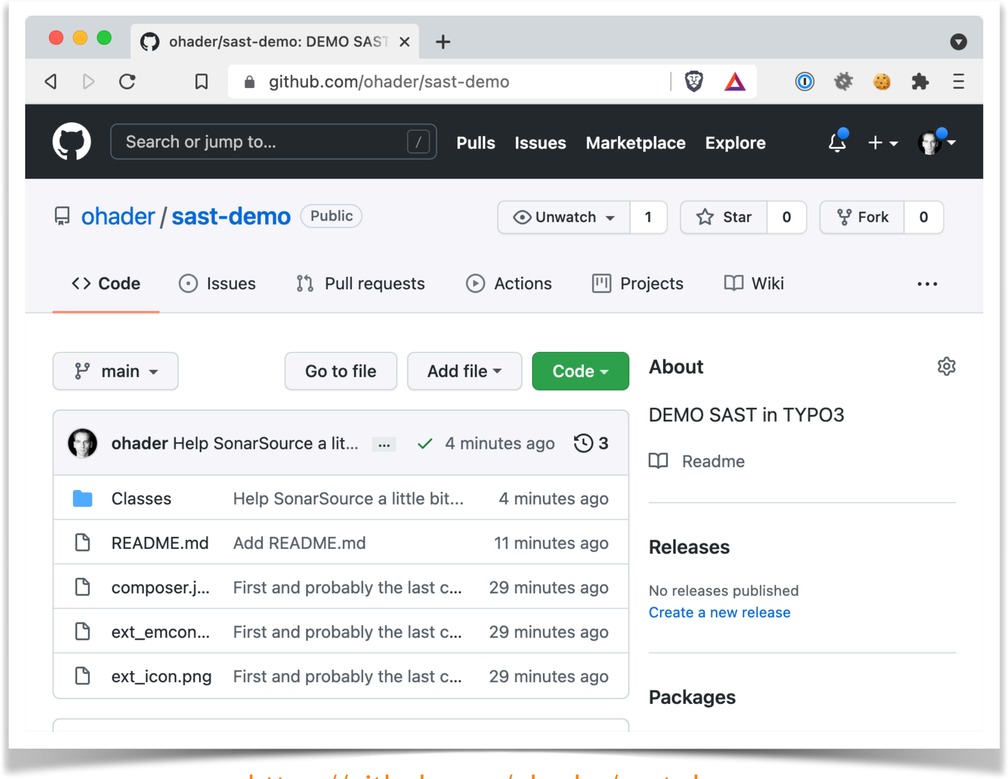


PsalmPHP (https://psalm.dev/r/159771bec7) → **Schwachstellen gefunden**

- **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
<?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);
    }
}
```



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

Demo Anwendung (realistischer)

```php
<?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

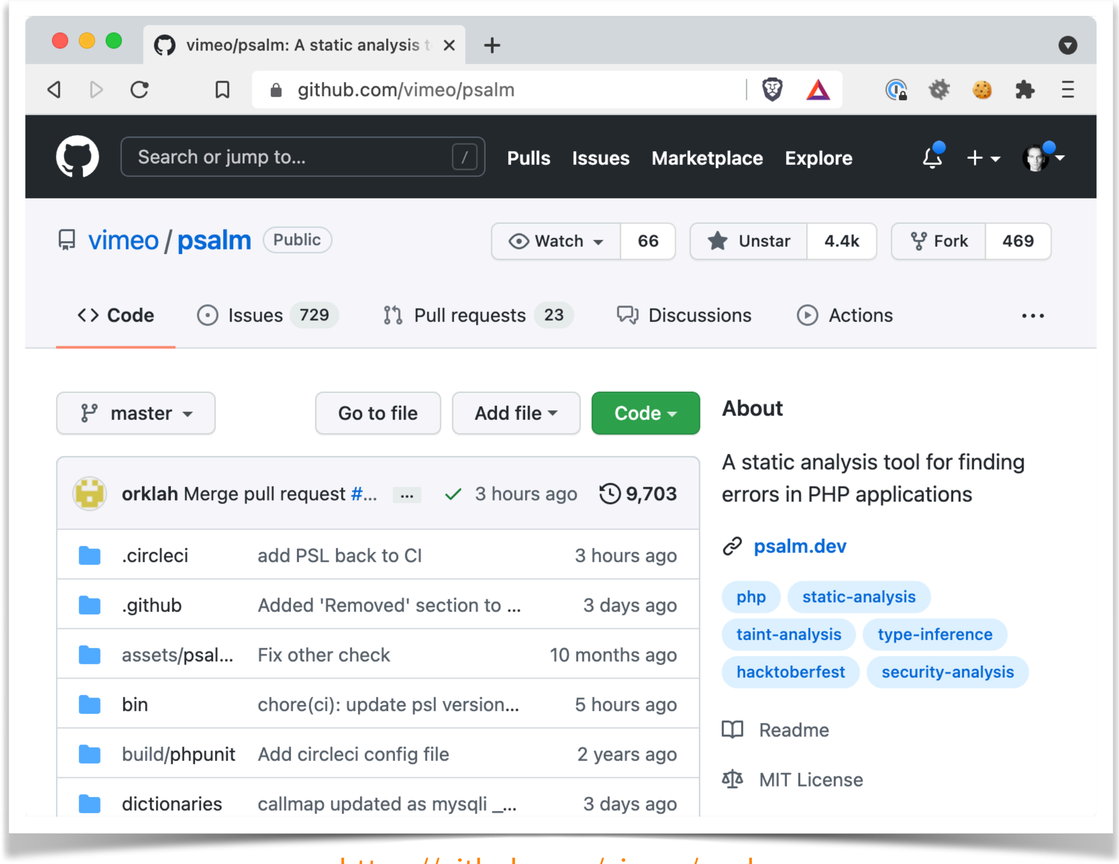
SQLi

XSS

- 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**



https://github.com/vimeo/psalm

PsalmPHP Stubs

```php
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
<?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) {}
```

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

PsalmPHP Stubs

```php
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
<?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

PsalmPHP Stubs

```php
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);
    }
}
```

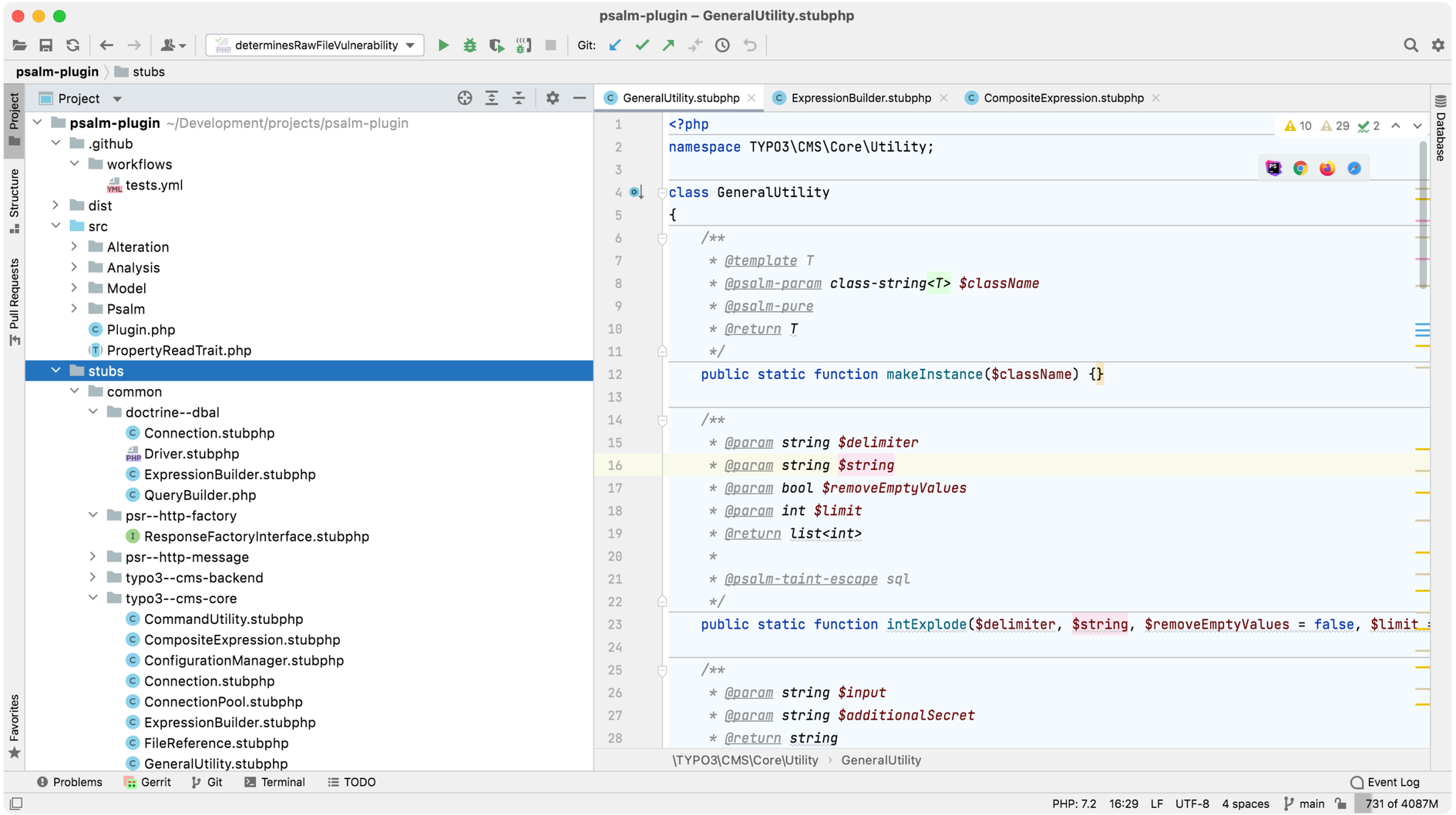
```php
<?php
namespace Doctrine\DBAL;

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
<?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) {}
```

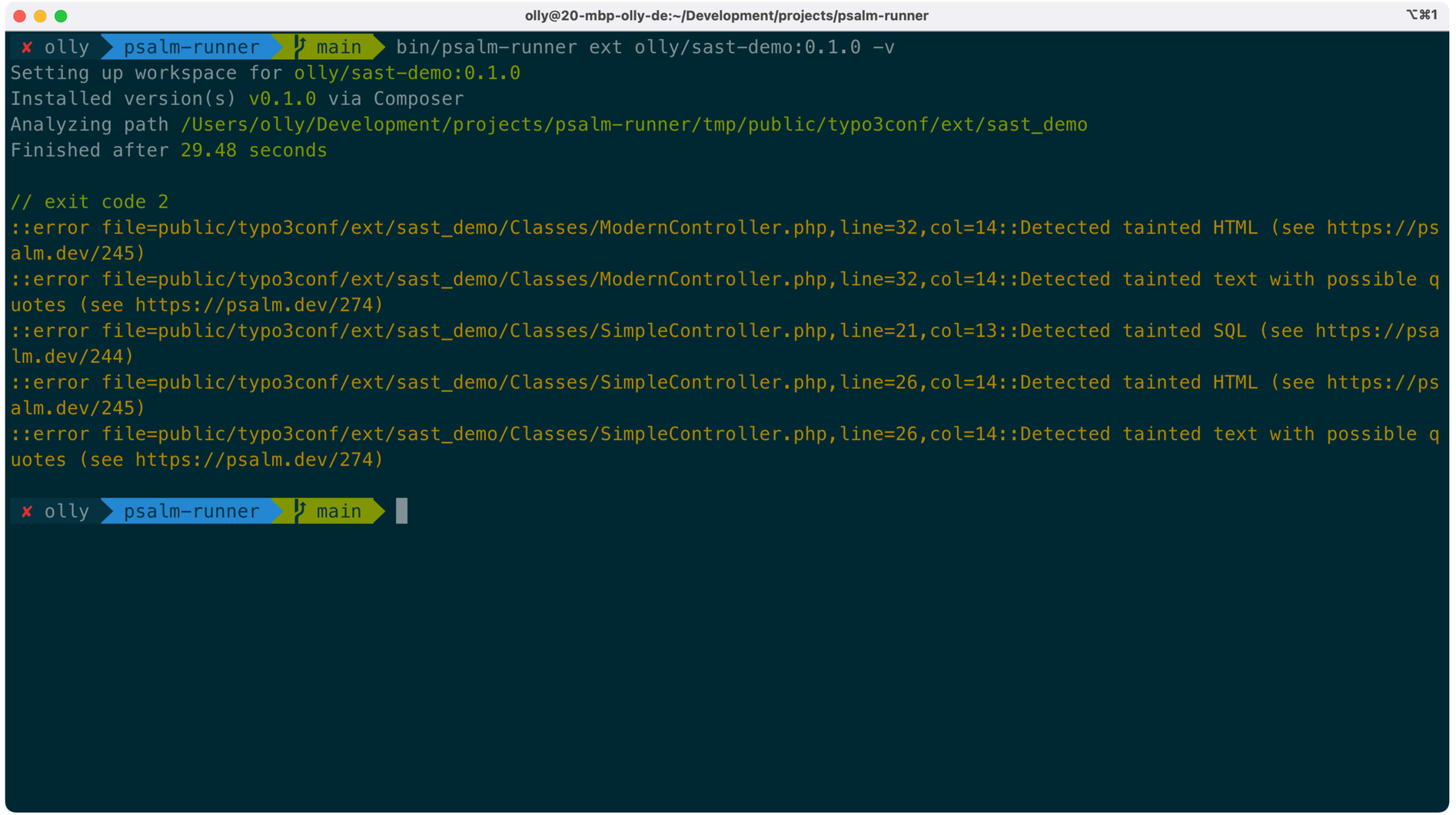
- **@psalm-taint-escape**
- markiert Parameter **$str** für Aspekt **sql** als **bereinigt**
- **@psalm-taint-unescape**
- … das Gegenteil (unsicher)

TYPO3 Plugin für PsalmPHP



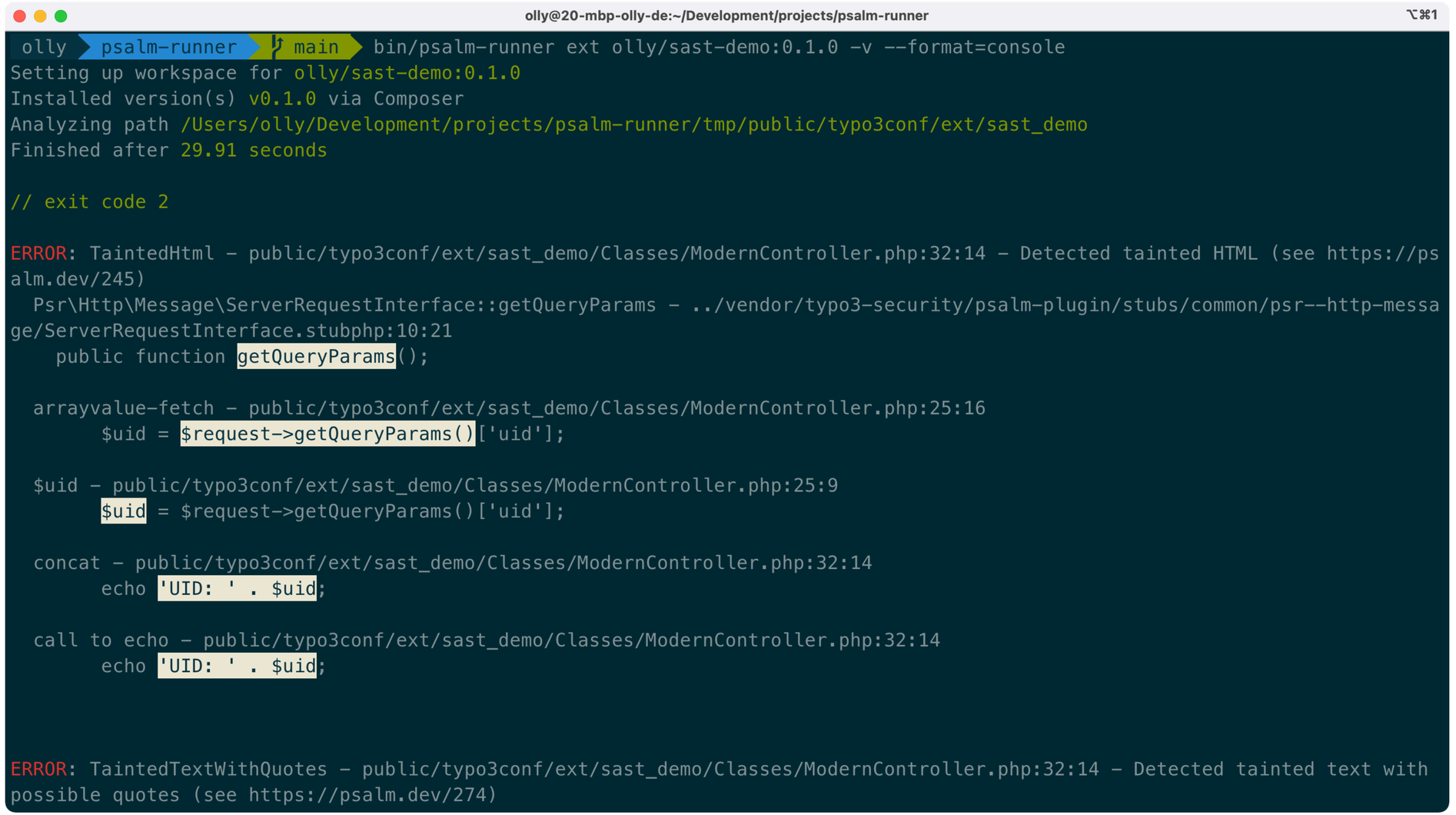
typo3-security/psalm-plugin - nicht öffentlich

Demo TYPO3 Psalm-Runner



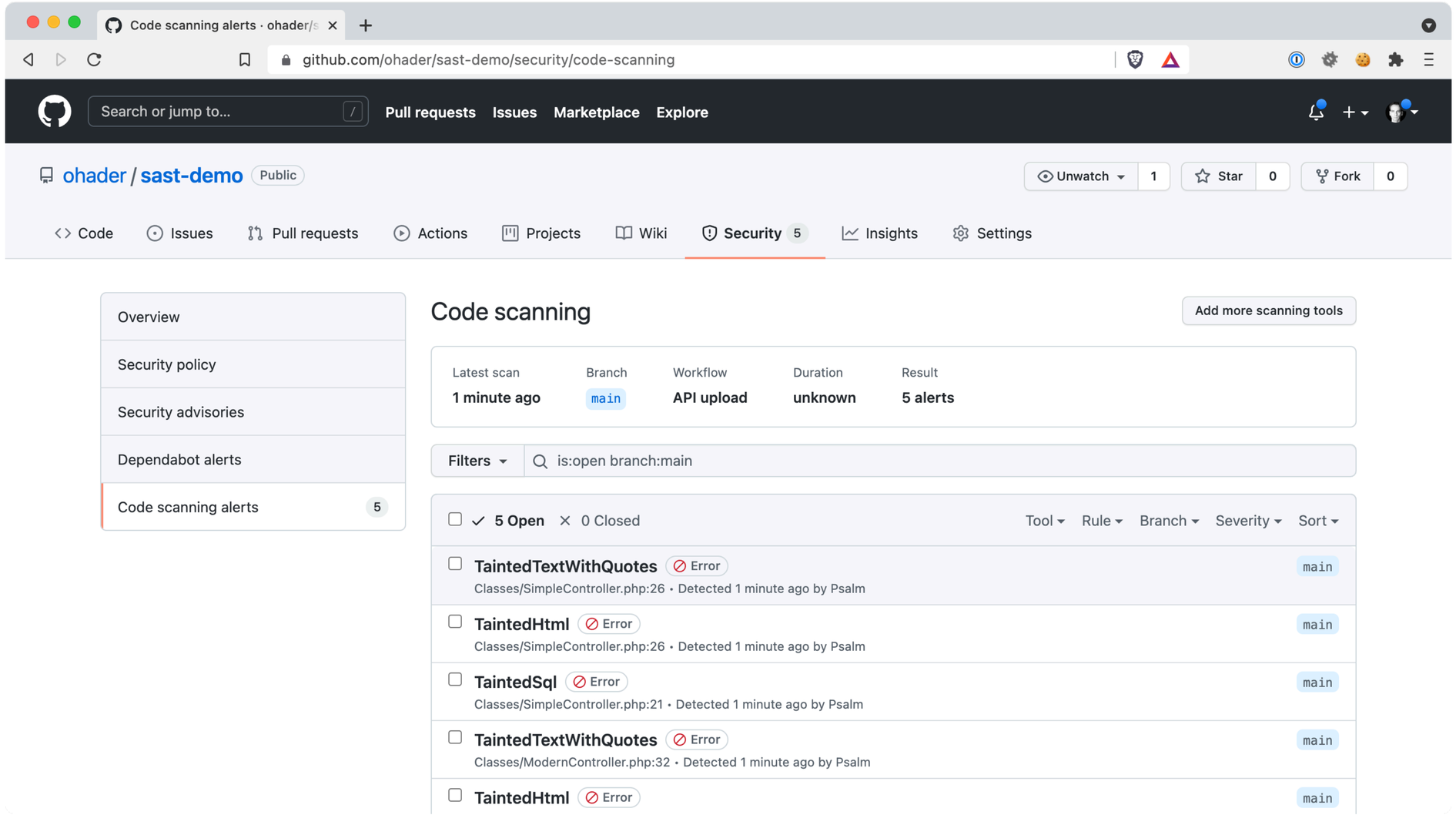
Kompakte Ausgabe

Demo TYPO3 Psalm-Runner



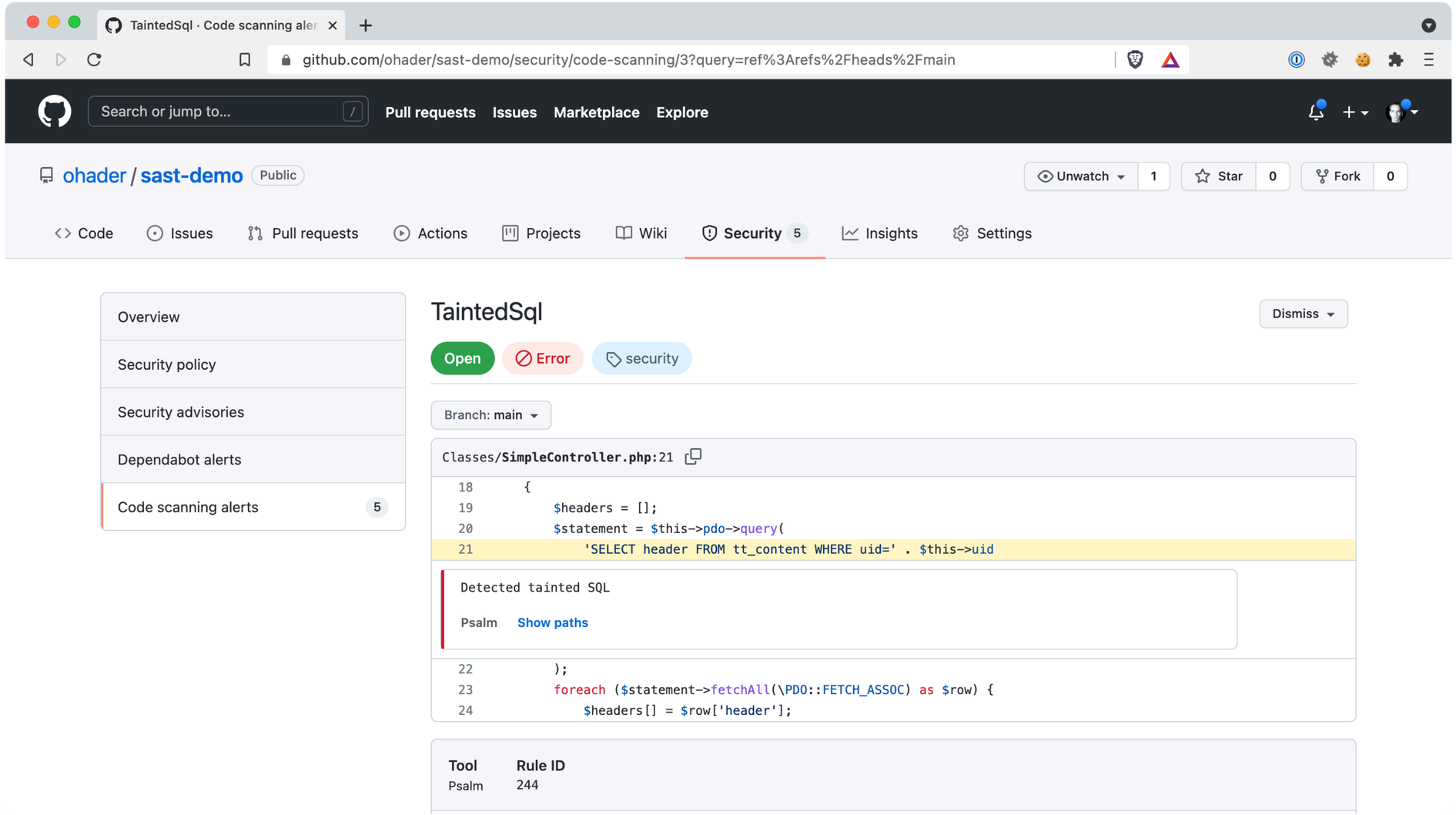
Ausführlichere Ausgabe

Demo TYPO3 Psalm-Runner



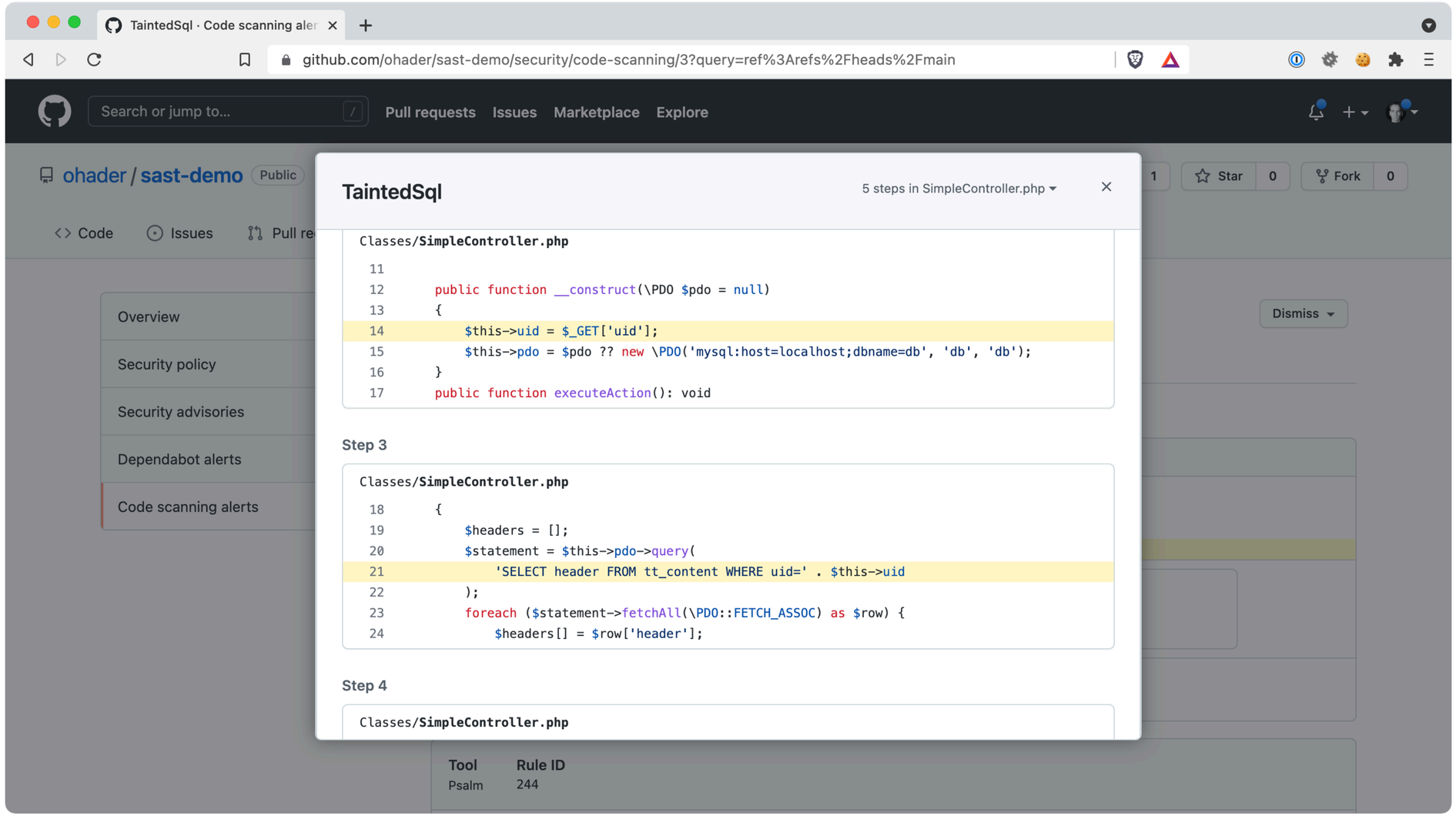
SARIF Export & GitHub Integration

Demo TYPO3 Psalm-Runner



SARIF Export & GitHub Integration

Demo TYPO3 Psalm-Runner



SARIF Export & GitHub Integration

#zukunft

MüsseMalJemand™

- **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

TYPO3

thanks!

questions?