DevSecOps and Securing SDLC

Introduction to the workshop

MODUS CREATE

- Team Intro
  - Andy Dennis - VP Platform and Cloud
  - Bill Reyor - Director of Security

- What is Modus Create?
- Course structure
  - PART 1 (IDE / LOCAL SECURITY)
  - BREAK (15 Minutes)
  - PART 2 (PIPELINE SECURITY)

BSidesCT

A Quick Class Survey

MODUS CREATE

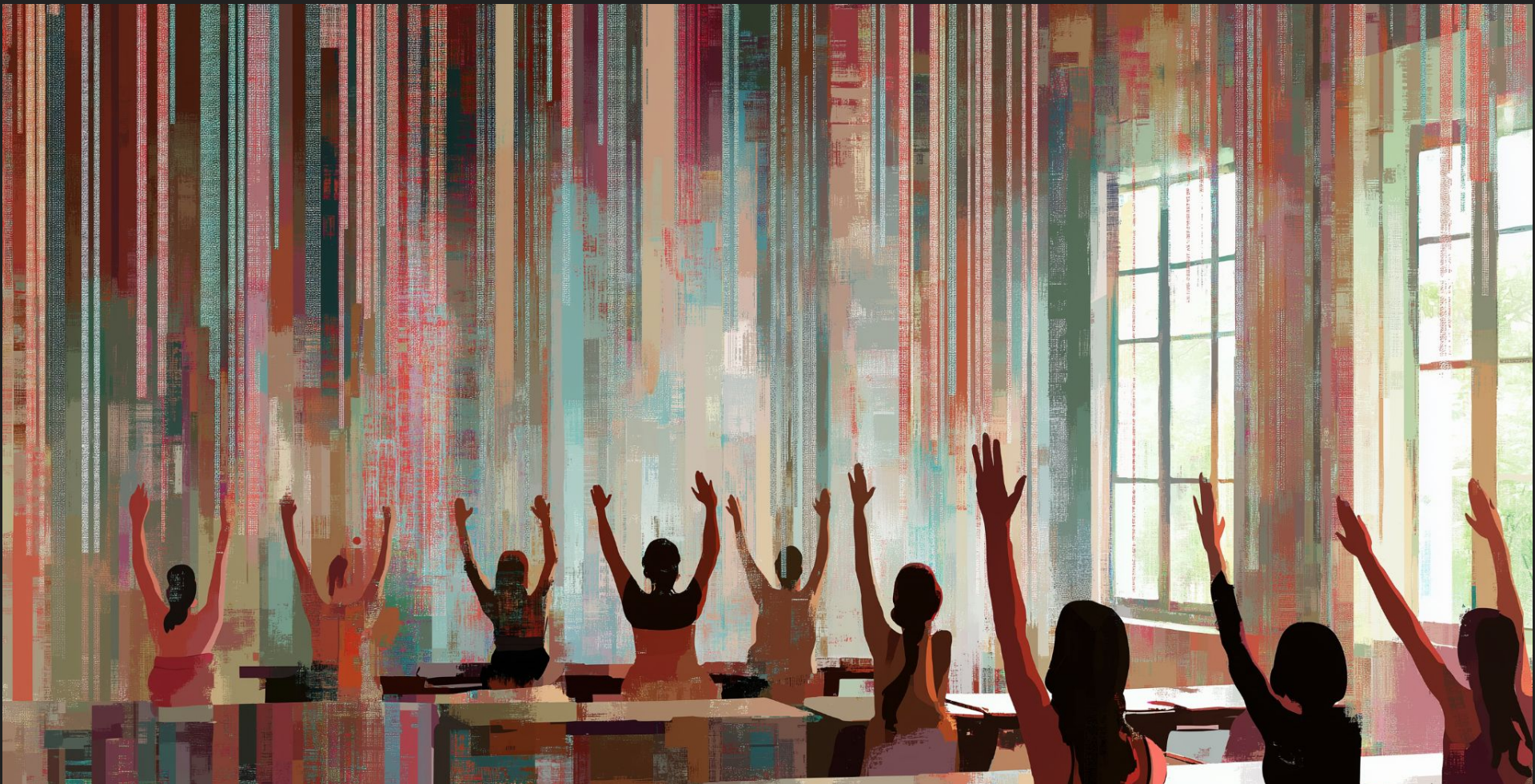
Have you ever?

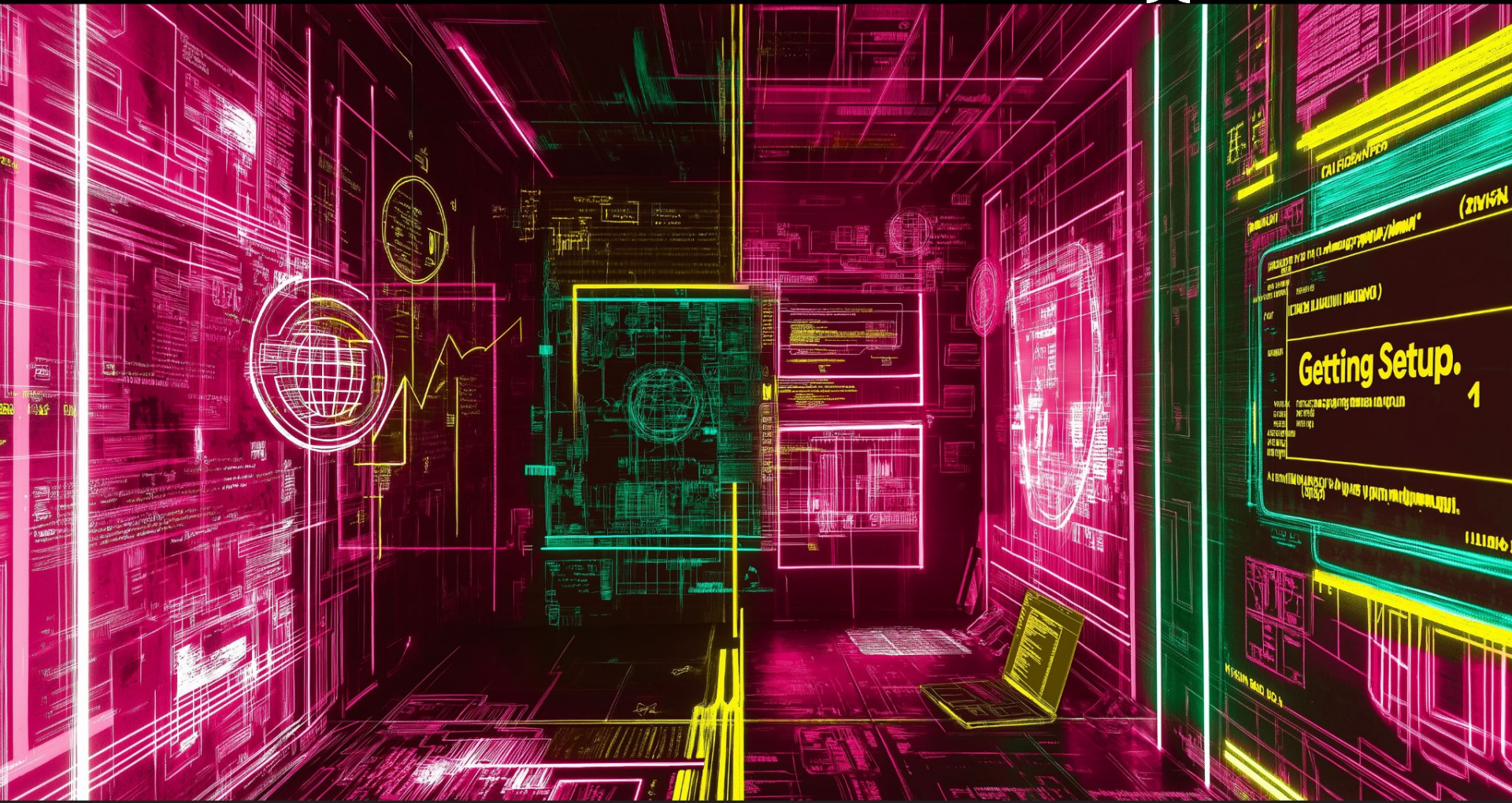
> Cloned a repository with Git?
> Merged a pull request
> Exploited an SQL injection vulnerability? (or had to fix one)

BSidesCT

If you need help raise your hand

MODUS CREATE




Getting Setup.
1

Getting setup - The tools

MODUS CREATE

You will need

- GitHub account
  https://www.github.com
- SonarLint/SonarCloud
  https://docs.sonarsource.com/sonarcloud/improving/sonarlint/
- IDE (located in Codespace)
  https://code.visualstudio.com/
- Copilot
  https://github.com/features/copilot

During the course we will use

- Talisman
  https://github.com/thoughtworks/talisman
- BFG
  https://rtyley.github.io/bfg-repo-cleaner/

BSidesCT

Getting setup - repository

MODUS CREATE

We've setup a repository for you to work with during this course.

- Located at Modus Create repository
  https://github.com/tweag/bsidesct-devsecops-sdlc

- Fork it
  https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/working-with-forks/fork-a-repo

BSidesCT


Need help
forking?
BSidesCT


MODUS CREATE
code spaces
BSidesCT

Follow Along with the Demos

MODUS CREATE

The lab instructions are divided into two parts, each with its own README file:

Part 1

1. Navigate to the **part1** directory on the main branch of your forked repository
2. Open the README.md file
3. Follow the step-by-step instructions to complete the Part 1 demo

Part 2

1. Navigate to the **part2** directory also in the main branch
2. Open the README.md file
3. Follow the step-by-step instructions to complete the Part 2 demo

The **README** files contain detailed guidance, code snippets, and examples to help you follow along with the live demos during the workshop.

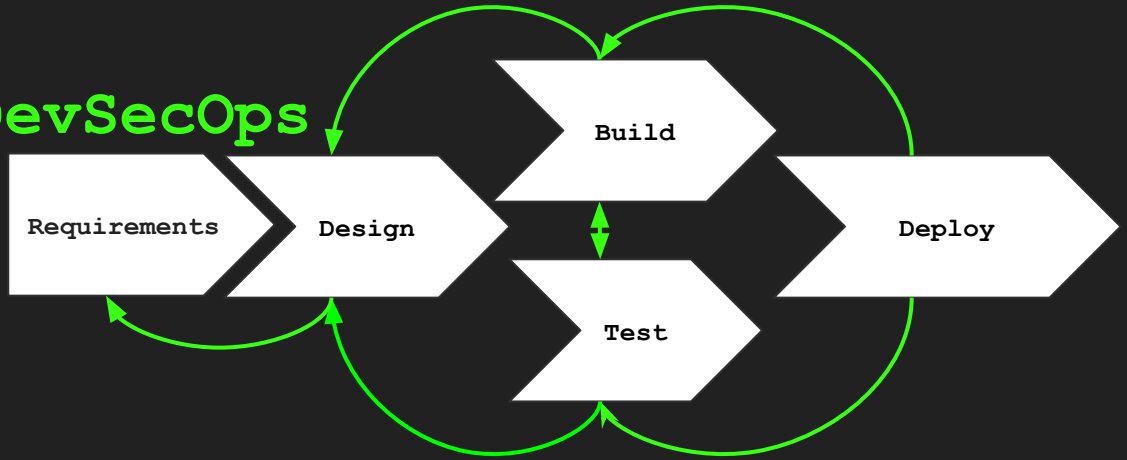
BSidesCT

What is DevSecOps?

MODUS CREATE

Traditional

DevSecOps

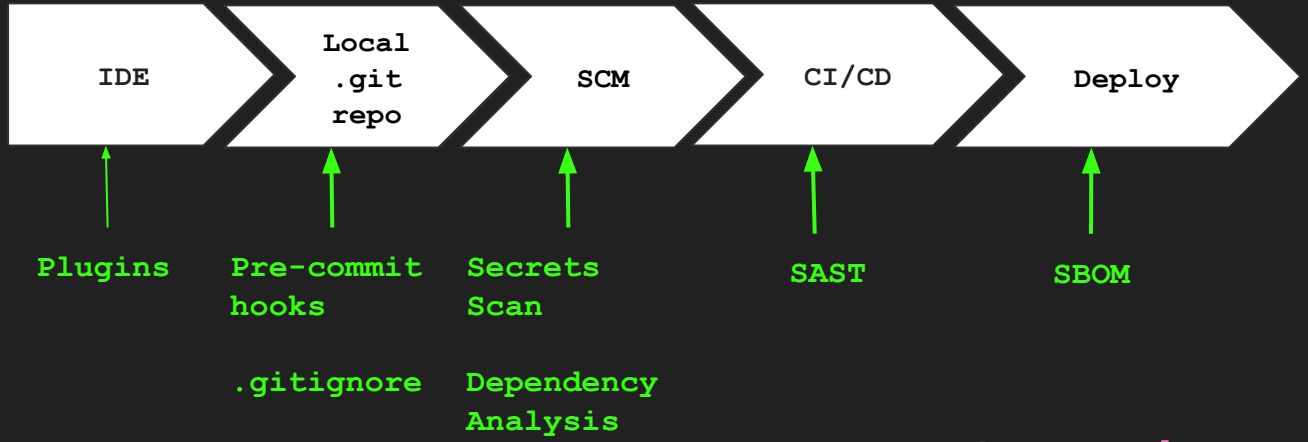


Security feedback loops

BSidesCT

What is Shifting Left?

MODUS CREATE



Security is constantly pushed further left in SDLC

BSidesCT

MODUS CREATE

Part 1

Local Security
IDEs and Codespace

BSidesCT

Introduction

MODUS CREATE

The first part of the course is dedicated to integrating security into your local development environment.

This covers:

> Security integration into the Codespace IDE
> Ignore files and clean up
> Pre-commit hooks

BSidesCT

Security in the IDE

MODUS CREATE

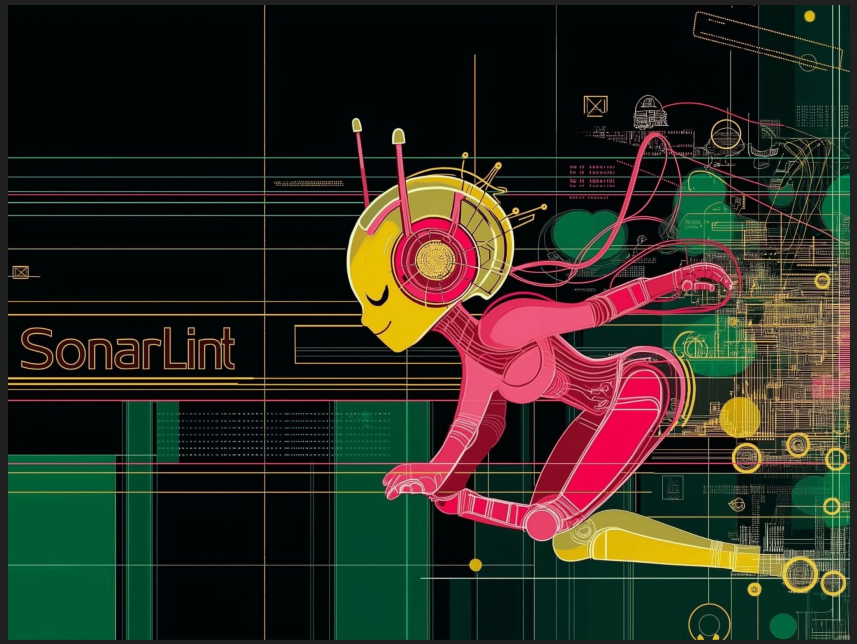
IDE plugins

- Acts as a first line of defense
- Catch errors in the code
- Detect secrets in the code e.g. passwords and API keys
- Uncover vulnerabilities like XSS and code that allows for SQL Injections

BSidesCT

SonarLint

MODUS CREATE



Setup instructions

SonarLint works with a number of popular IDEs including VSCode in Codespace

1. Follow the steps for VSCode to install it.
2. Setup a SonarCloud account via your GitHub user if you don't have one

(See README in REPO)

BSidesCT

SonarLint - Rules Configuration

MODUS CREATE

JavaScript

__proto__ property should not be used on

"===" and "!==" should be used instead of "==" and "!=" on

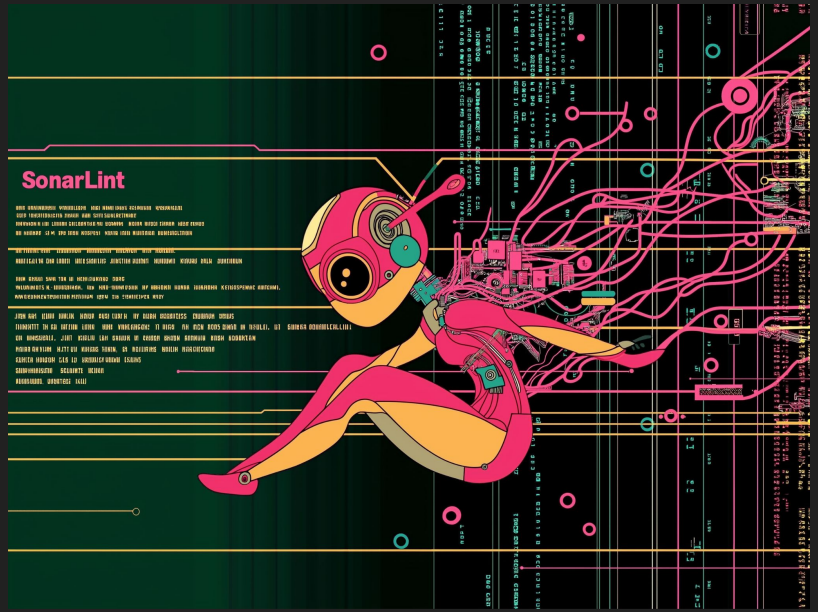
"alert(...)" should not be used on

BSidesCT

Demo 1 - Sonarlint

SonarLint

MODUS CREATE



We can see the plugin identified:

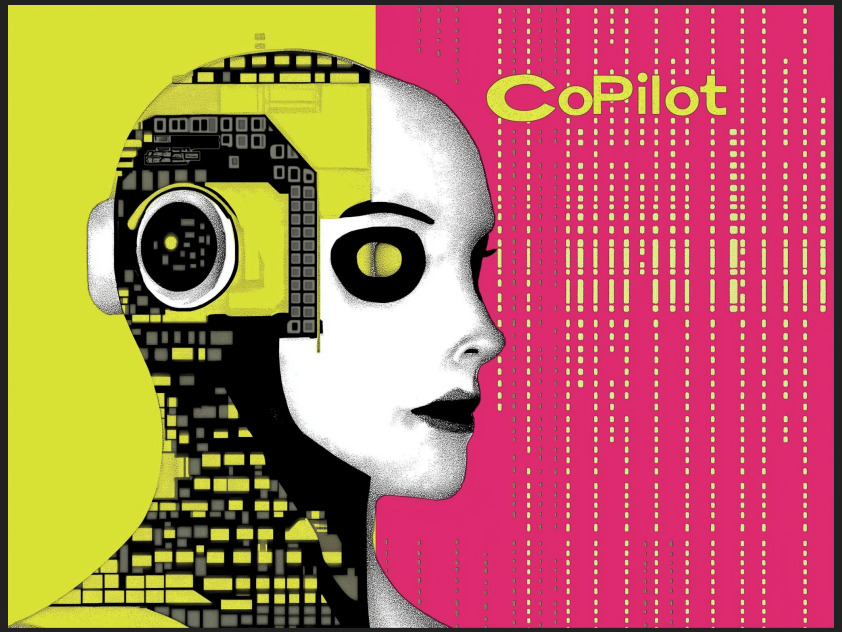
- > Use of the alert statement
- > Useless variable assignments
- > Use of the debugger statement
- > Use of var

Some of these have security implications!

BSidesCT

GitHub Copilot

MODUS CREATE



GitHub Copilot

Used 1 reference >

To prevent SQL injection, you should use parameterized queries or prepared statements. This ensures that user input is always treated as literal data and not part of the SQL command. Here's how you can modify your `saveToDatabase` method to use parameterized queries:

```
static saveToDatabase(blogPost) {
    const sql = `INSERT INTO blog_posts (title, content, author) VALUES (?, ?, ?
    // Pretend database call
    console.log(`Executing SQL: ${sql}`);
    var db = window.openDatabase("sqlcachedb", "1.0", "Cache result", 200000);
    db.transaction(function(tx){
        tx.executeSql(sql, [blogPost.title, blogPost.content, blogPost.author],
            console.log("Data inserted successfully");
        }, function(tx, error){
            console.log("Error occurred while inserting data");
        });
    });
}
```

In this code, the `?` are placeholders for the actual values. The values are passed in as an array in the second argument to `executeSql`. This ensures that the values are properly escaped, preventing SQL injection attacks.

BSidesCT

Demo 3 - Github Copilot

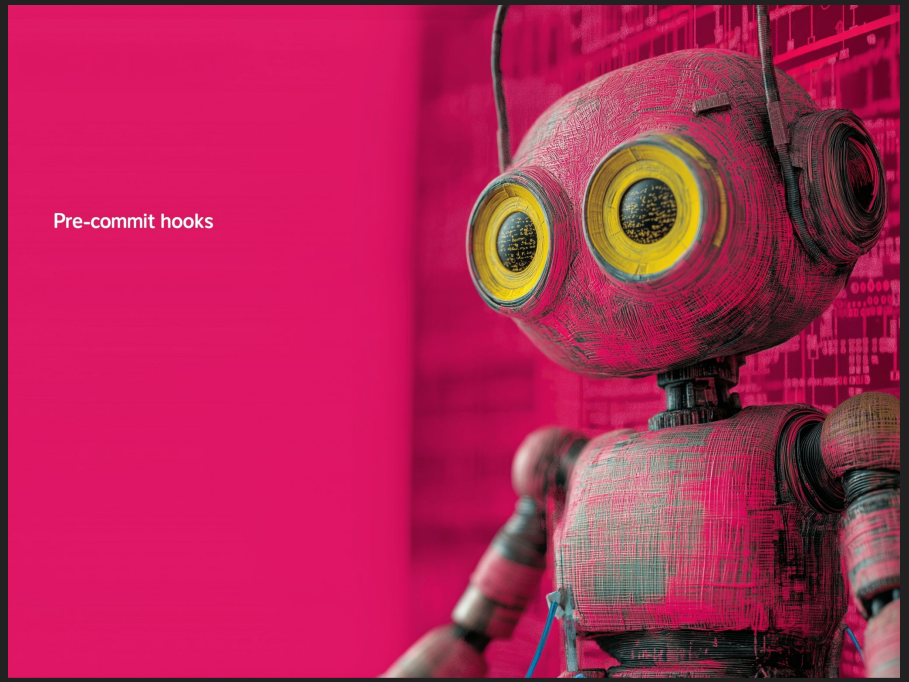
MODUS CREATE

Pre-commit Hooks

Pre-commit hooks

MODUS CREATE

- Scripts run before commiting
- Enforce code quality, catch issues
- Customizable, but can be skipped

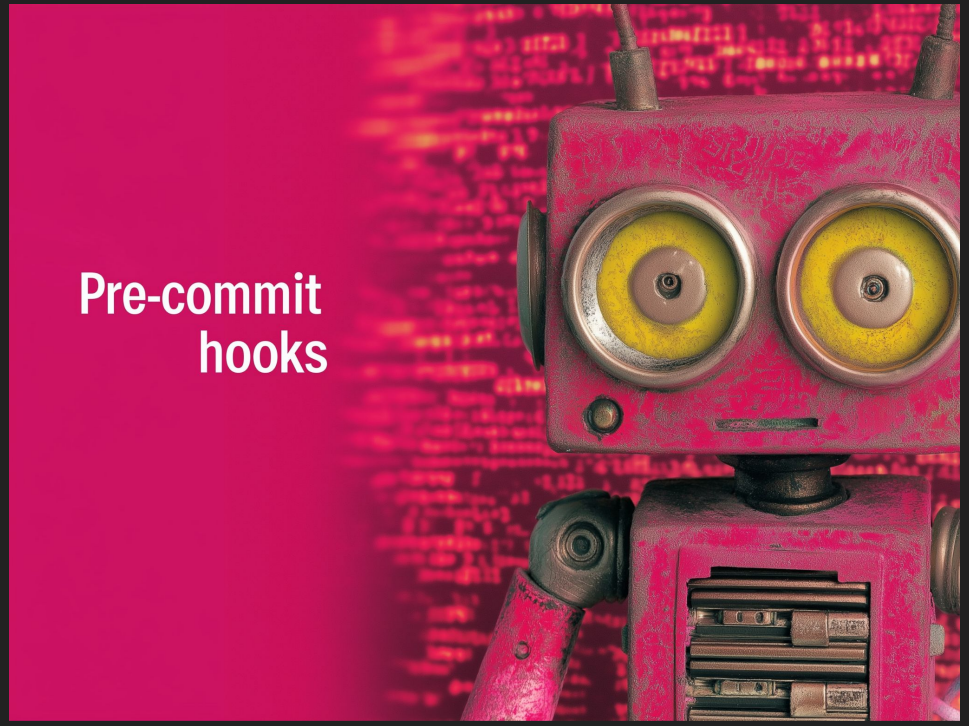


Old school but One of the better ways to prevent secrets from being committed.

BSidesCT

Pre-commit hooks - key features

MODUS CREATE



Used to perform tasks like

- Quality enforcement
- Prevent commits with problems e.g. security vulnerabilities

Key features include

- Exist in the .git/hooks https://git-scm.com/docs/githooks
- Not transferred with the repo
- Can be bypassed (--no-verify flag)

BSidesCT

Demo 4 - Talisman

Talisman

MODUS CREATE

Suite of pre-commit hooks w/security

Features of Talisman

- Overview of key features

Exploring Talisman

- Switch to the **README** file
- Command line experiments

Talisman

BSidesCT

.gitignore

MODUS CREATE

Using .gitignore files

MODUS CREATE

Pre-commit hooks are very handy for catching secrets and other code we do not want to push up to our repository.

How about if there are whole classes of file we want to avoid adding?

Well we can add an extra layer of defense using the `.gitignore` file.

The `.gitignore` file allows us to:

- Create an exclusion list of file types and directories so these are not tracked
- Apply fine grained controls over this exclusion list by adding to each repository
- Can be version controlled so every user inherits it. This is unlike the hooks we just explored

Using .gitignore files

MODUS CREATE

BSidesCT

Demo 5 - .gitignore

MODUS CREATE

Cleaning up secrets and files

MODUS CREATE

BFG is pre-installed in the Codespace.

Instructions on utilizing it can be found in the README.

The README provides an example of how we can use BFG to:

- Remove certificates
- Remove files with passwords
- Remove specific strings (passwords, API keys) from files

Let's walk through this.

BSidesCT

Demo 6 - BFG

MODUS CREATE

Part 1 wrap up

MODUS CREATE

Congratulations, you've now reached the end of **Part 1** and have a basic understanding of IDE plugins, pre-commit hooks and .gitignore files.

Some of the tools such as Talisman have many features we couldn't cover today. You may be interested in taking the time to dig into these after the course.

Let's take a short break, then we will move onto **Part 2**.

BSidesCT


15
15Minute
Break

Part 2

Security in the CI/CD pipeline

BSidesCT

Welcome to Part 2

MODUS CREATE

In this section we will cover:

- Secrets scanning
- Handling secrets in GitHub
- Detecting security vulnerabilities in the repository
- Vulnerable dependencies
- Static Analysis (SAST)
- Branch protection rules

BSidesCT

Secrets Scanning

Secret Scanning

MODUS CREATE

Useful Documentation

- **TruffleHog**
  https://github.com/trufflesecurity/trufflehog

- **GitHub Secrets Scanning**
  https://docs.github.com/en/code-security/secret-scanning/about-secret-scanning

- **Horusec**
  https://horusec.io/site/

BSidesCT

Demo 7 - Secrets Scanning

BSidesCT

Handling Secrets in GitHub

BSidesCT

Secrets in GitHub

MODUS CREATE

Best practices for handling secrets in source code management

- **Prepare** - use a secret manager (GitHub has one)
- **Prevent** - git-secrets, and the other tools we saw with the IDE/Pre-commit hooks
- **Detect** - pipeline checks, we can also use TruffleHog
- **Respond** - BFG / Git-filter

Let's look at secret storage in GitHub quickly

BSidesCT

Demo 8 - Handling Secrets

Detecting Security Vulnerabilities

Detecting security vulnerabilities

MODUS CREATE

Multiple tools for detecting vulnerabilities.

Secrets scanning - pattern/entropy detection

SAST - Static Analysis Security Testing

SCA - Software Composition Analysis

SBOMs - Software Bills of Materials

GitHub has the following features

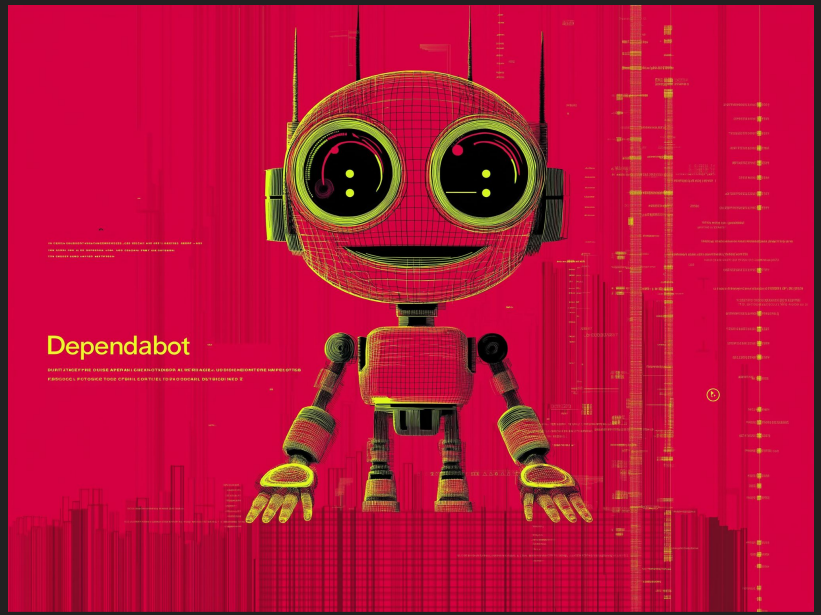
- SECURITY.md
- Secrets scanning
- Dependency analysis
- CodeQL(SAST)via Actions
- GitHub SBOM

BSidesCT

Dependency Vulnerabilities

Dependabot

MODUS CREATE



Features include

- Detect vulnerable and out of date packages
- Create automatic pull requests to remediate issues
- List vulnerabilities associated with the packages

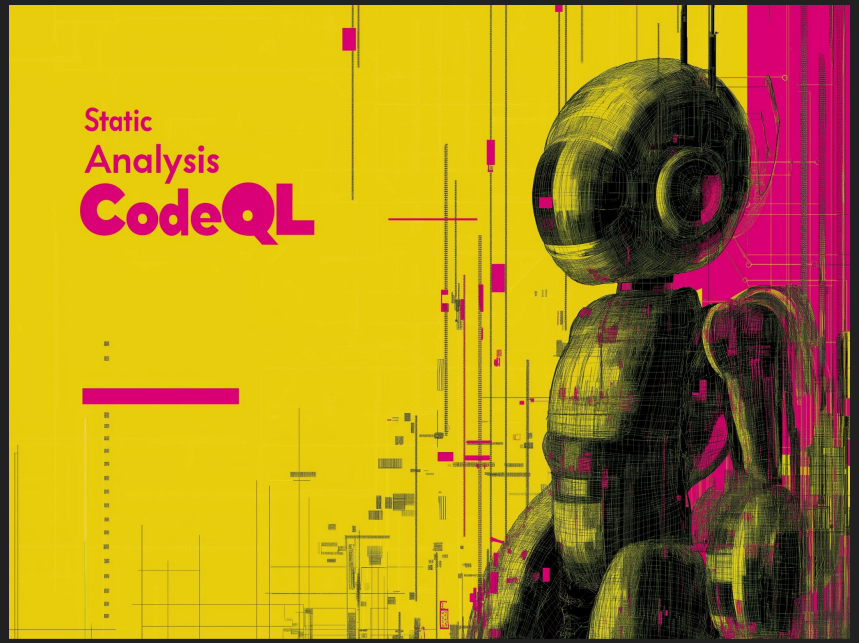
BSidesCT

Demo 9 - Dependencies

Static Analysis

Static Analysis (SAST/CodeQL)

MODUS CREATE



Features

- Multiple language support (Java, JS, Ruby etc.)
- Highlights security vulnerabilities
- Allows for custom queries/query packs
- Execute as a gating mechanism in your CI/CD pipeline

BSidesCT

Demo 10 - SAST

Branch Protection Rules

Branch protection rules

MODUS CREATE

Branch protection rules can be used to prevent code being merged until it passess all the relevant tests.

It also allows us to lock down who can commit to the branch.

Features include

- Block merges until security checks are passed
- Lockdown branch ownership via CODEOWNERS files
- Require multiple individuals to approve a pull request before it can be merged

BSidesCT

Demo 11 - Branch Protection

SBOMs

MODUS CREATE

Software Bill of Materials (SBOMS) are great way of getting a snapshot of all the components in your application.

GitHub supports SBOMs out of the box for repositories stored there.

SBOM formats and data enrichments include:

- **CycloneDX format** https://cyclonedx.org/
- **SPDX format** https://spdx.dev/
- **VEX** https://www.cisa.gov/sites/default/files/2023-01/VEX_Use_Cases_Aprill2022.pdf

BSidesCT

Demo 12 - SBOMS

ENDOR Place Holder
Reachability analysis demo

MODUS CREATE

Demo 14 - ENDOR SBOMS
How to Generate SBOM w/VEX reachability data

Conclusion

Recap

MODUS CREATE

In this workshop we covered a number of areas related to mastering security in development.

- Security in the IDE
- Pre-commit hooks
- Secrets scanning
- SAST
- Dependency scanning
- Branch protection rules
- SBOMS

BSidesCT

Next Steps

MODUS CREATE

After this workshop you can take the repository you forked and use it as a starting to point for your own projects.

Some ideas for other areas to explore are listed on the right.

- Try integrating with Infrastructure-as-Code. Check out Checkov https://www.checkov.io/
- Experiment with other third party tooling for SAST. For example, the open source Horusec https://github.com/ZupIT/horusec
- Look into container scanning https://github.com/quay/clair

BSidesCT

Thanks for attending!

MODUS CREATE

Thank you for attending the workshop today. A big thank you to the BSidesCT Staff and volunteers as well.

Questions?

william.reyor@moduscreate.com

Remember to check out:

https://moduscreate.com/careers/

For our latest job posts!

BSidesCT