Automatic code analysis with CodeQL

Axel Pettersson & Christopher Gustafson

Outline

What CodeQL is

Queries

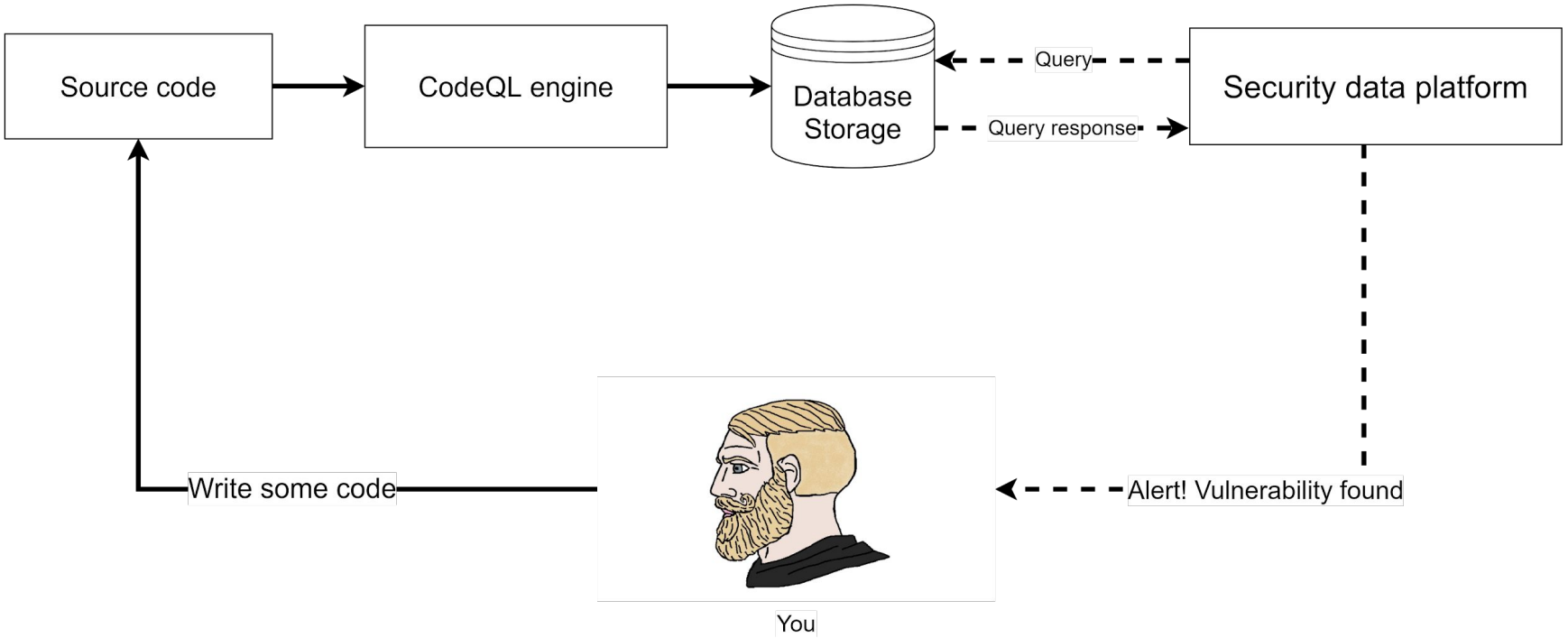
Utilising CodeQL

Contributing

Reflection & take home message

What is CodeQL?

- Code analysis engine developed by GitHub
- Your source code is compiled into a **relational database**
- **Queries** are run against this database
- Queries identify **vulnerable patterns**
- Query results mean that a **vulnerability** has been found



Example CodeQL flow

Queries: How to detect vulnerabilities

```
app.get("/", (req, res) => {
    const content = req?.query["content"] ?? "";
    res.send(template(content));
});
```

Example of vulnerable code

Source: vulnerable-express-app/app.js at main · Ackuq/vulnerable-express-app (github.com)

```
class XSSConfiguration extends TaintTracking::Configuration {
 XSSConfiguration() { this = "XSSConfiguration" }

 // Consider remote flow data (e.g. URL parameters)
 override predicate isSource(DataFlow::Node source) { source instanceof RemoteFlowSource }

 // First argument to res.send(...) is our sink
 override predicate isSink(DataFlow::Node sink) {
   sink =
     moduleImport("express")
         .getACall()
         .getAMemberCall("get")
         .getABoundCallbackParameter(1, 1)
         .getAMemberCall("send")
         .getArgument(0)
 }
}
```

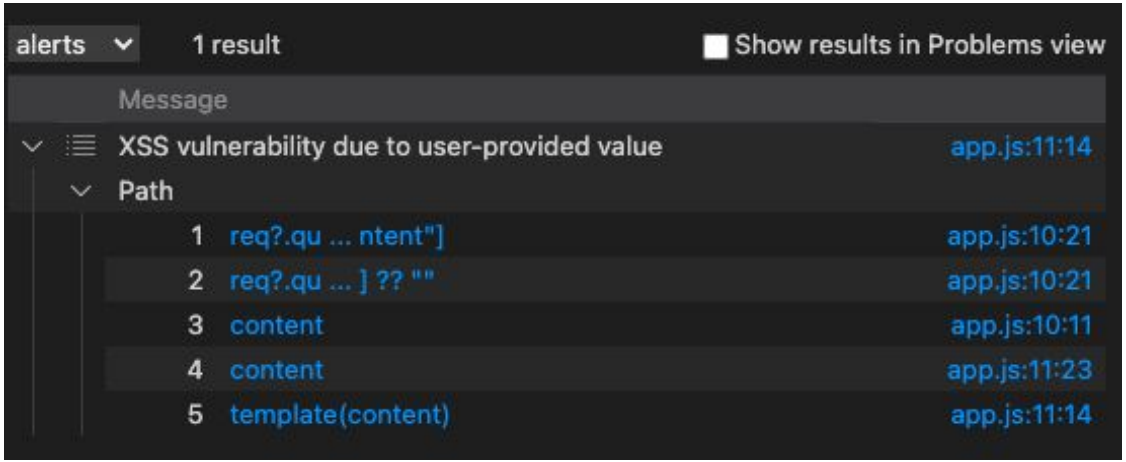
CodeQL configuration for detecting these flows

Source: vulnerable-express-app/ServerSideInjection.ql at main · Ackuq/vulnerable-express-app (github.com)

The query

```
from XSSConfiguration cfg, DataFlow::PathNode source, DataFlow::PathNode sink
where cfg.hasFlowPath(source, sink)
select sink.getNode(), source, sink, "XSS vulnerability due to user-provided value"
```

The query response



Query for alerting these problems

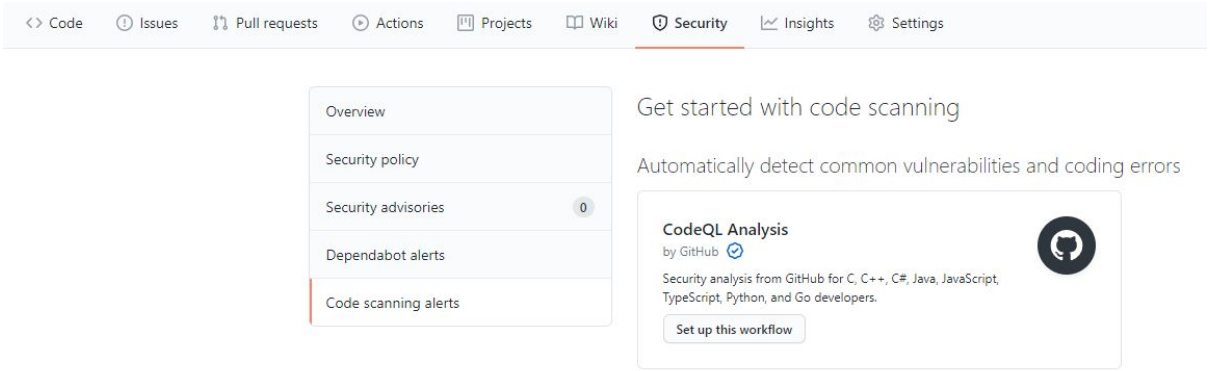
Source: vulnerable-express-app/ServerSideInjection.ql at main · Ackuq/vulnerable-express-app (github.com)

How to utilise CodeQL

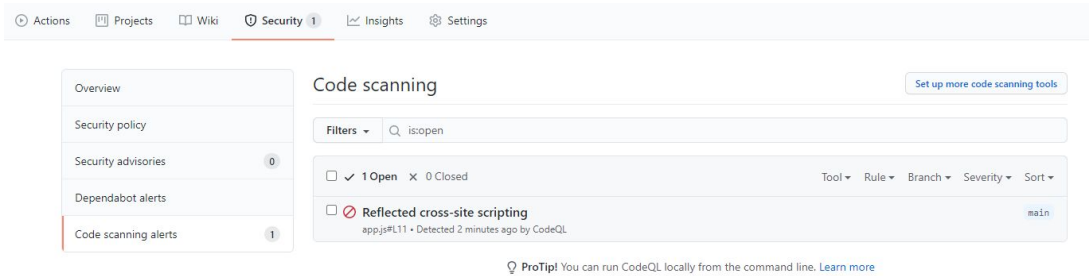
- Locally
  - Requires more manual setup
  - Primarily used for developing new queries
- CI tools:
  - GitHub Code Scanning
  - LGTM.com

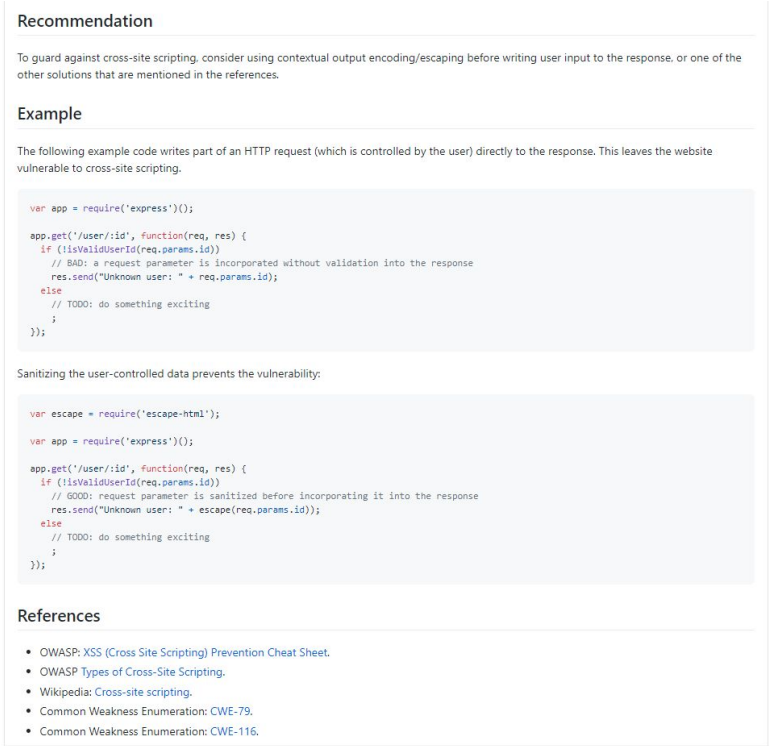
Integrating with your GitHub repository

- Easily integrated with GitHub using Code Scanning Alerts
- Uses set of open-source queries



Setting up CodeQL





Recommendation

To guard against cross-site scripting, consider using contextual output encoding/escaping before writing user input to the response, or one of the other solutions that are mentioned in the references.

Example

The following example code writes part of an HTTP request (which is controlled by the user) directly to the response. This leaves the website vulnerable to cross-site scripting.

```
var app = require('express')();

app.get('/user/:id', function(req, res) {
  if (!isValidUserId(req.params.id))
    // BAD: a request parameter is incorporated without validation into the response
    res.send("Unknown user: " + req.params.id);
  else
    // TODO: do something exciting
    ;
});
```

Sanitizing the user-controlled data prevents the vulnerability:

```
var escape = require('escape-html');

var app = require('express')();

app.get('/user/:id', function(req, res) {
  if (!isValidUserId(req.params.id))
    // GOOD: request parameter is sanitized before incorporating it into the response
    res.send("Unknown user: " + escape(req.params.id));
  else
    // TODO: do something exciting
    ;
});
```

References

- OWASP: XSS (Cross Site Scripting) Prevention Cheat Sheet.
- OWASP Types of Cross-Site Scripting.
- Wikipedia: Cross-site scripting.
- Common Weakness Enumeration: CWE-79.
- Common Weakness Enumeration: CWE-116.

Result of CodeQL action running

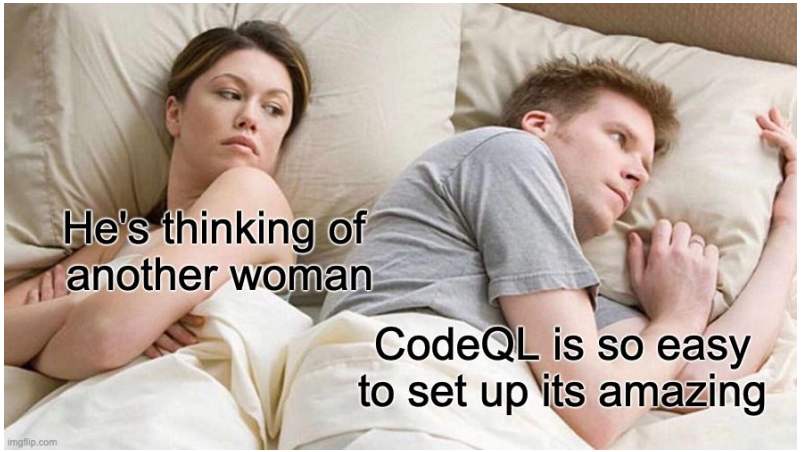
Source: Code scanning alerts · Ackuq/vulnerable-express-app (github.com)

Contribution to the Tool

- CodeQL in GitHub code scanning is open-source
- You can contribute through their repository `github/codeql`
- GitHub Bounty program

Reflection

- Simple to set up
- Potentially enormous benefits



- But…
- Never take the absence of vulnerabilities for granted

Take Home Message:

Ensuring the safety and security of your software can be very difficult, but introducing tools to help you with this task can be very simple.

Thanks for listening!