GitHub Constellation India 2022

# Securing Code Using CodeQL JavaScript

Presented by - @s-samadi and @zbazztian

# Speakers

Shadi Samadi - @s-samadi

Security Solution Architect

Sebastian Bauersfeld - @zbazztian

Code Scanning Architect

# Housekeeping

- All workshop material can be found in this repository - https://github.com/GitHubConstellation2022/codeql-workshop/tree/main
- Prerequisite: have VSCode environment set up. Instructions are in the repo and have been broadcast prior to the workshop
- We ask that everyone mutes their microphones and if you have any questions feel free to ask via the Zoom chat

# Agenda

- Vulnerability - CVE 2021-32685
- Introduction to CodeQL
- Workshop - VSCode

# CVE-2021-32685 - Improper Verification of Cryptographic Signature in tEnvoy

CVSS Score - 9.8

## Cryptographic node.js Library

Used by TogaTech for cryptographic privacy and authentication for data communication

## Impact

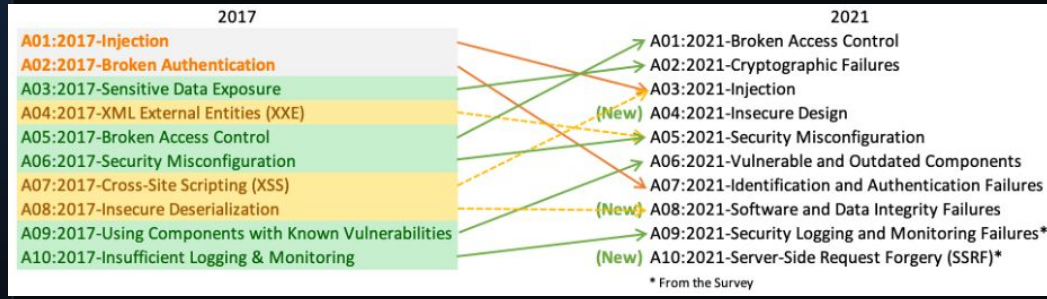
In versions prior to 7.0.3, the `verifyWithMessage` method of `tEnvoyNaClSigningKey` always returns `true` for any signature of a SHA-512 hash matching the SHA-512 hash of the message even if the signature is invalid.

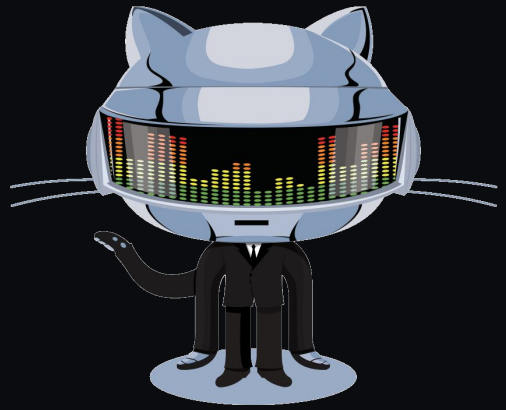
# OWASP Top 10

## #2 Cryptographic Failures

https://owasp.org/Top10/A02_2021-Cryptographic_Failures/

https://owasp.org/Top10/





# CVE-2021-32685
# Demo

Alice

$m_a$ = The CodeQL workshop starts tomorrow at 2pm IST.
Regards,
Alice

Bob

Eve

$m_e$ = The CodeQL workshop has been cancelled.
Cheers,
Alice.

Alice

$s = enc(hash(m_a))$

$dec(s) == hash(m_a)$

Bob

Eve

**Cryptographic Signatures**

- Message authenticity
- Message integrity
- Frequently based on public key cryptography

# Understanding the vulnerability

Let’s look at some code...
In `tEnvoyNaClSigningKey` class of `tEnvoy.js`

# tEnvoy.js :: sign method

```
48134  this.sign = (message, password = null) => {
48135    let assertion = _assertPassword("sign", password);
48136    if(assertion.proceed) {
48137      if(message == null) {
48138        throw "tEnvoyNaClSigningKey Fatal Error: argument message of method sign is required and does not have a default value.";
48139      }
48140      if(_type == "private") {
48141        message = _tEnvoy.util.pack(message);
48142        let hashed = _tEnvoy.util.bytesToHex(_nacl.hash(message)); // sha512 hash
48143        return {
48144          signature: hashed + "::" + _tEnvoy.util.bytesToHex(_nacl.sign.detached(_nacl.hash(message), this.getPrivate(_getPassword()))),
48145          hash: hashed
48146        };
48147      } else {
48148        throw "tEnvoyNaClSigningKey Fatal Error: Key does not have a private component.";
48149      }
48150    } else {
48151      throw assertion.error;
48152    }
48153  }
```

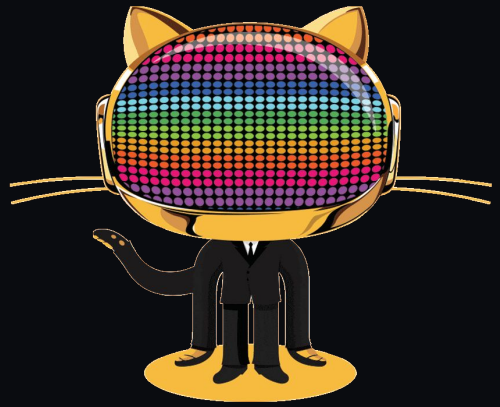
# tEnvoy.js :: verify method

```
this.verify = (signed, password = null) => {
  let assertion = _assertPassword("verify", password);
  if(assertion.proceed) {
    if(signed == null) {
      throw "tEnvoyNaClSigningKey Fatal Error: argument signed of method verify is required and does not have a default value.";
    }
    if(signed.split("::").length != 2) {
      throw "tEnvoyNaClSigningKey Fatal Error: Invalid signature.";
    }
    let hash = _tEnvoy.util.hexToBytes(signed.split("::")[0]);
    let signature = _tEnvoy.util.hexToBytes(signed.split("::")[1]);
    return {
      verified: _nacl.sign.detached.verify(hash, signature, this.getPublic(_getPassword())),
      hash: signed.split("::")[0]
    };
  } else {
    throw assertion.error;
  }
}
```

# tEnvoy.js :: verifyWithMessage method

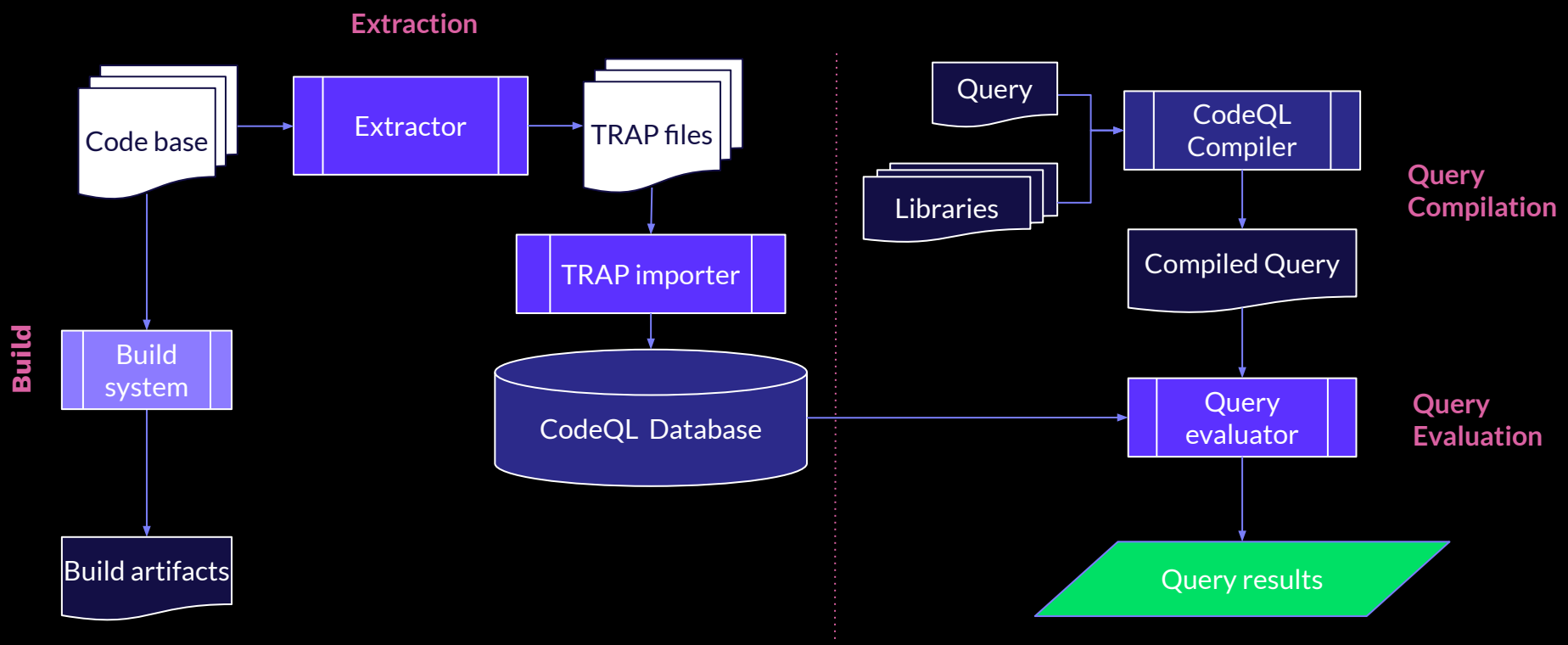
```
this.verifyWithMessage = (signed, message, password = null) => {
  if(signed == null) {
    throw "tEnvoyNaClSigningKey Fatal Error: argument signed of method verifyWithMessage is required and does not have a default value.";
  }
  if(message == null) {
    throw "tEnvoyNaClSigningKey Fatal Error: argument message of method verifyWithMessage is required and does not have a default value.";
  }
  if(signed.split("::").length != 2) {
    throw "tEnvoyNaClSigningKey Fatal Error: Invalid signature.";
  }
  let hash = _tEnvoy.util.hexToBytes(signed.split("::")[0]);
  return this.verify(signed, password) && _tEnvoy.util.bytesToHex(_nacl.hash(_tEnvoy.util.pack(message))) == _tEnvoy.util.bytesToHex(hash);
}
```

This will always be TRUE.



# Introduction to CodeQL

# How CodeQL Works



# CodeQL Language

The basic syntax will look familiar to anyone who has used SQL, but it is used somewhat differently.

- CodeQL is a logic programming language.
- CodeQL uses common logical connectives (such as and, or, and not).
- Quantifiers (such as forall and exists)
- CodeQL supports recursion and aggregates.
- CodeQL is object-oriented - supports classes so you can define new high level entities to represents or extend a given model.

Simple query that return `"hello world"`.

```
import <language>

select "hello world"
```

More complicated queries typically look like this:

```
from /* ... variable declarations ... */
where /* ... logical formulas ... */
select /* ... expressions ... */
```

For example, the result of this query is the number 42:

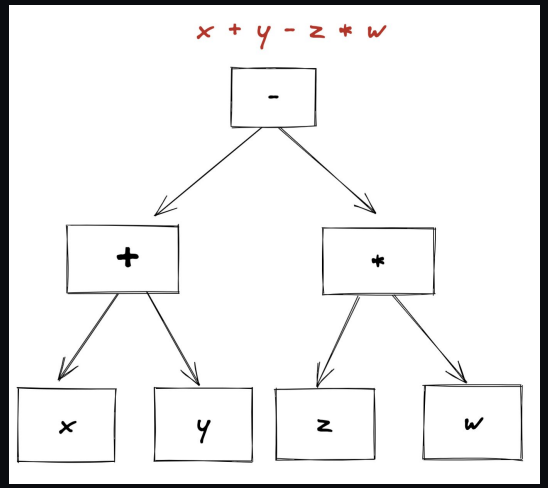
```
from int x, int y
where x = 6 and y = 7
select x * y
```

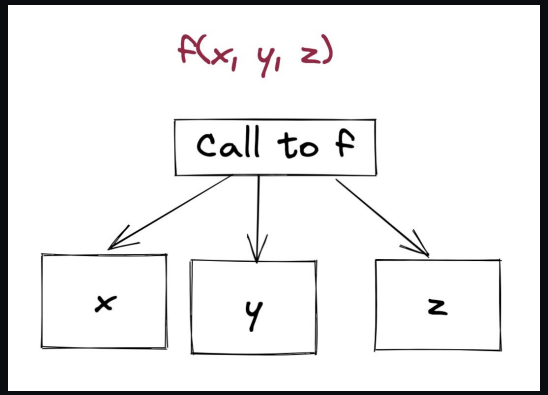
An example of a Class declaration

```
class SmallInt extends int {
  SmallInt() { this in [1..10] }
  int square() { result = this*this }
}

from SmallInt x, SmallInt y, SmallInt z
where x.square() + y.square() = z.square()
select x, y, z
```

# Abstract Syntax Tree (AST)



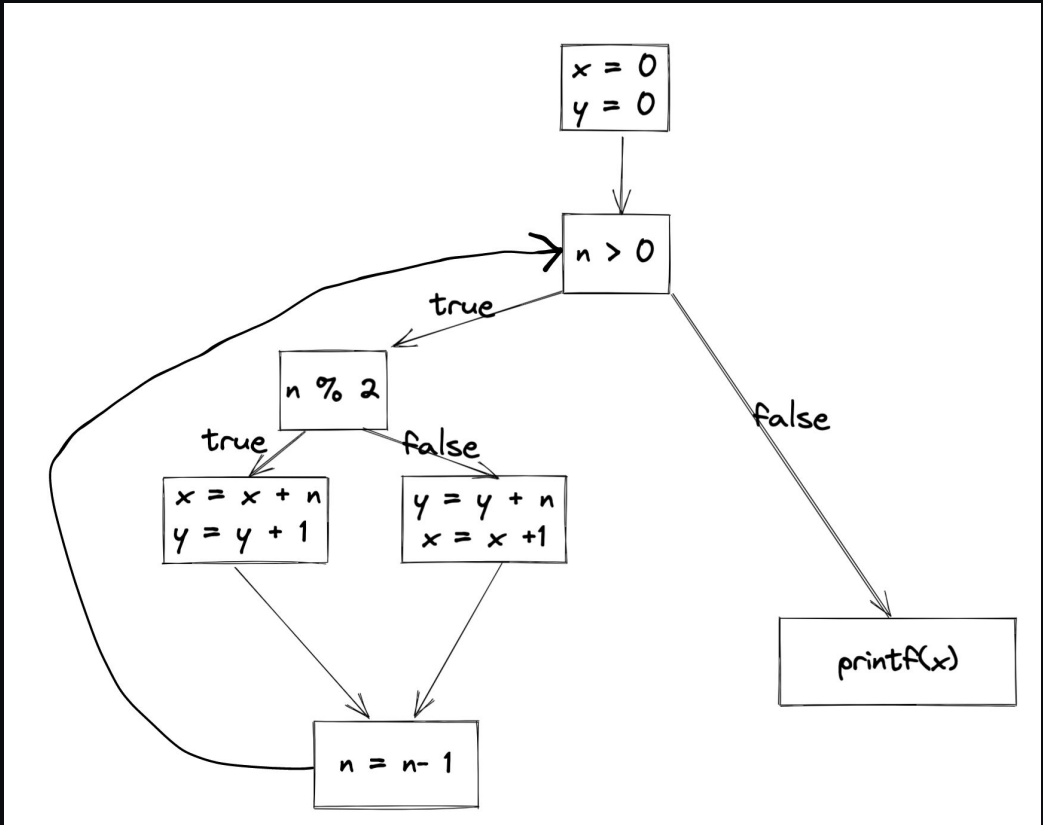

- The abstract syntax tree (AST) of a program represents the program's syntactic structure
- Each program element is represented as a node
- AST Viewer in VSCode
- AST documentation

# Control Flow Graph (CFG)

- Control flow creates a graph from AST
- Models the flow of control during execution
- Typically used to determine if something is evaluated before or after another AST node - predecessor or successor
- Modelling domination path - control flow X dominates control flow Y if all the control flow paths to Y have to go through X first
- Great for deriving whether a path always returns true or false

# Control Flow Graph (CFG)

```
1   int x = 0;
2   int y = 0;
3
4   while(n > 0) {
5           if( n %2 = 0) {
6                   x = x + n;
7                   y = y + 1;
8           }
9           else {
10                  y = y + n;
11                  x = x + 1;
12          }
13          n = n - 1;
14  }
15  printf(x);
16
```



# Control Flow Graph (CFG) Cheatsheet

### ControlFlowNode

A node in the control flow graph, which is an expression, a statement, or a synthetic node.

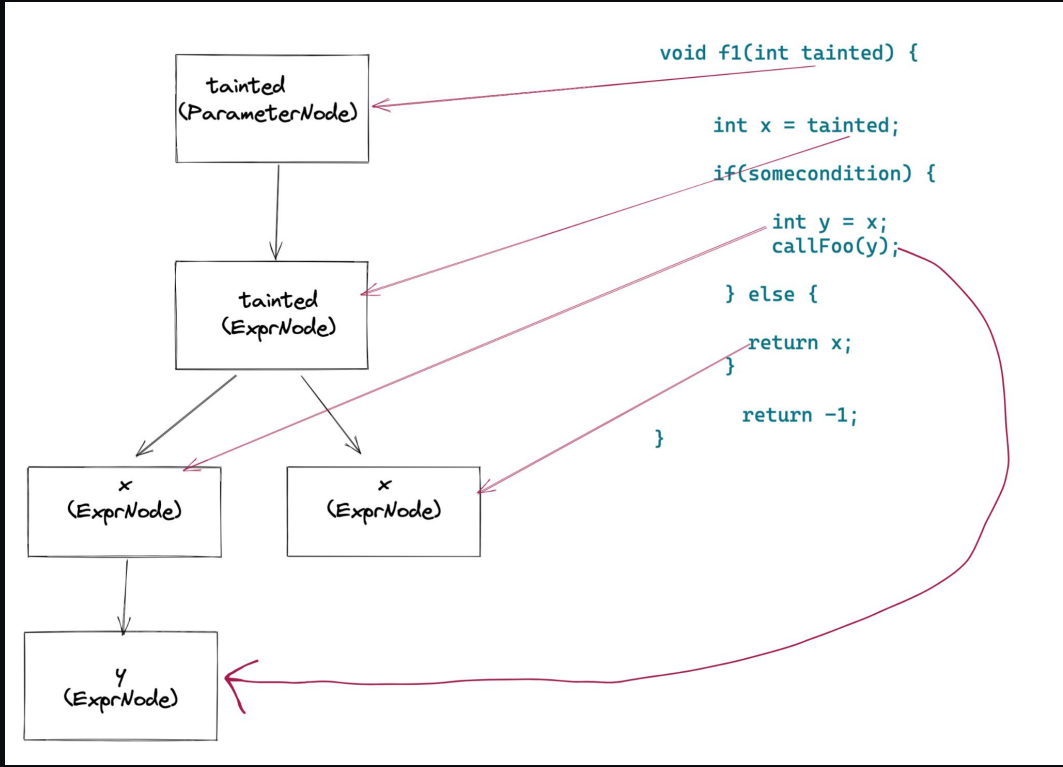
### ConditionGuardNode

A guard node recording that some condition is known to be truthy or falsy at this point in the program

# Data Flow Graph

- We model the flow of data through the program as a directed graph, where:
  - The *nodes* of the graph represent program elements that have a value, such as expressions and parameters.
  - The *edges* of the graph represent *flow* between those program elements.
- Local (“intra-procedural”) data flow models flow within one function; feasible to compute for all functions in a snapshot
- Global (“inter-procedural”) data flow models flow across function calls; not feasible to compute for all functions in a snapshot

# Data Flow Graphs



# Taint Tracking

```
public void foo() {
  string formatString = "hello";
  string tainted = "world!";

  formatString = formatString + tainted;

}
```

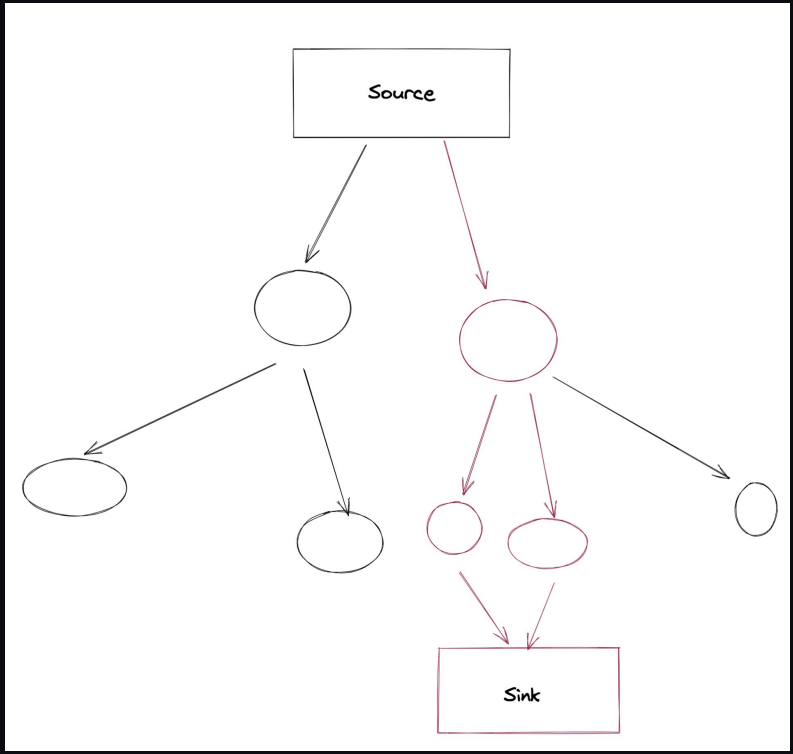
- Data flow analysis tells us how values flow *unchanged* through the program. Taint tracking analysis is slightly more general: it tells us how values flow through the program and may undergo minor changes, while still influencing (or tainting) the places where they end up.
- Would not be found by data flow alone because concatenation modifies the `formatString` by *appending* the tainted value.
- Taint tracking can be thought of as another type of data flow graph. It usually extends the standard data flow graph for a problem by adding edges between nodes where one one node influences or taints another.

# Data Flow Library

```
class Example extends DataFlow::Configuration {
  Quick Evaluation: Example
  Example() { this = "Example" }

  Quick Evaluation: isSource
  override predicate isSource(DataFlow::Node node) {
    none()
  }

  Quick Evaluation: isSink
  override predicate isSink(DataFlow::Node node) {
    none()
  }
}
```



# Using CodeQL - Problem Statement

This is a DataFlow problem - is there flow from Source to Sink?

## Source

In the `verify` function:

```
{
    verified: _nacl.sign.detached.verify(hash, signature, this.getPublic(_getPassword())),
    hash: signed.split("::")[0]
};
```

## Sink

In the `verifyWithMessage` function:

```
this.verify(signed, password)
```

# Brand guidelines & accessibility

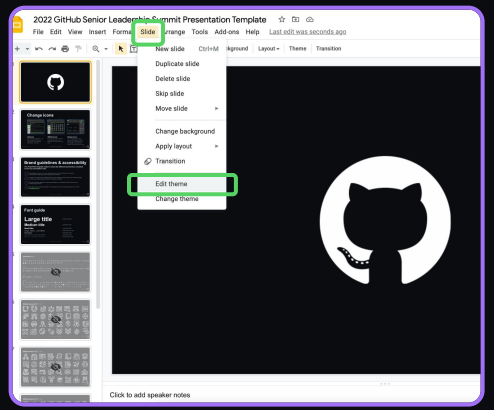
Our goal is to create a user friendly template so all Hubbers have the opportunity to create a presentation that looks as if a designer created it. Please be sure to follow guidelines and template structures to ensure brand and accessibility compliance.

This template is based on our core brand elements to assist Hubbers in producing communications that align with the GitHub brand system.

The colors and font sizing have been approved by the accessibility team to ensure we are actively being inclusive.

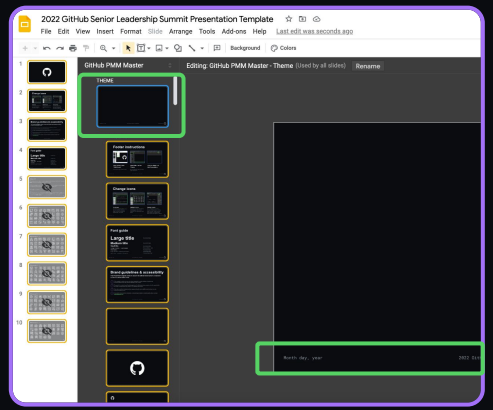
If you have issues with the template, please reach out to via our slack channel #visual-communications

# Footer instructions



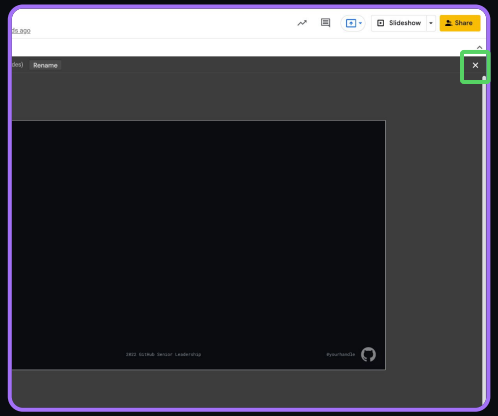

Click “Slide” then “Edit theme”

Found at the top of the tool bar




Edit the footer on Slide 0 of the theme

By changing the date, talk title, and GitHub handle of the speaker(s)

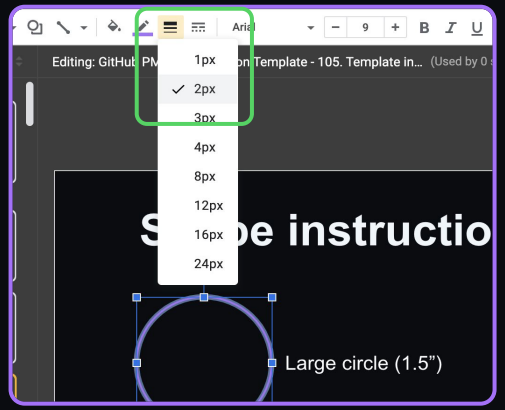



Click the X at the top right to close out and return to presentation

# Font guide

| | |
|---|---|
| **Large number** | Arial Bold 60pt |
| **Large title** | Arial Bold 44pt |
| **Medium title** | Arial Bold 36pt |
| **Small title** | Arial Bold 20pt |
| `Large dates, usernames` | Roboto Mono 16pt |
| Information | Arial Regular 16pt |
| Small information, supporting descriptions | Arial Regular 14pt |
| `Small dates, usernames, information` | Roboto Mono 12pt |

# Shape instructions



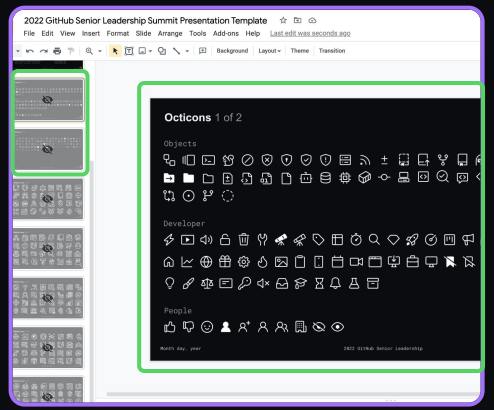
Large circle (1.5”)

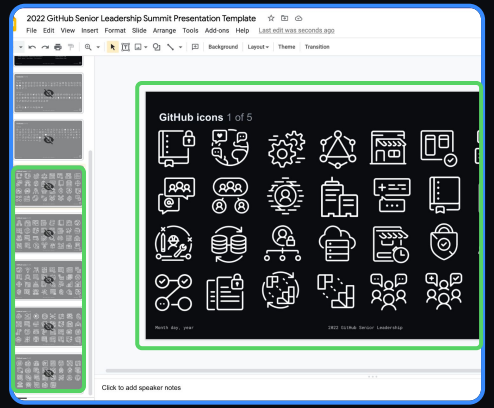
Small circle (0.6”)

When creating rounded corner rectangles, lines, or circles, all line widths or outlines should be 2px

Either create your own shape and follow the shape instructions or feel free to copy and paste from here! Click “Slide” then “Edit theme” and then copy and paste the intended shape. Click the “x” at the top right to exit theme and return to presentation view.

# Icon instructions





## Small circle (0.6”)

When you see the small circle, find an **Octicon** from the Octicon pages and copy/paste into the center of the circle. No need to resize.

## Large circle (1.5”)

When you see the large circle, find a **GitHub icon** from the GitHub icon pages and copy/paste into the center of the circle. No need to resize.

# **Octicons** 1 of 2

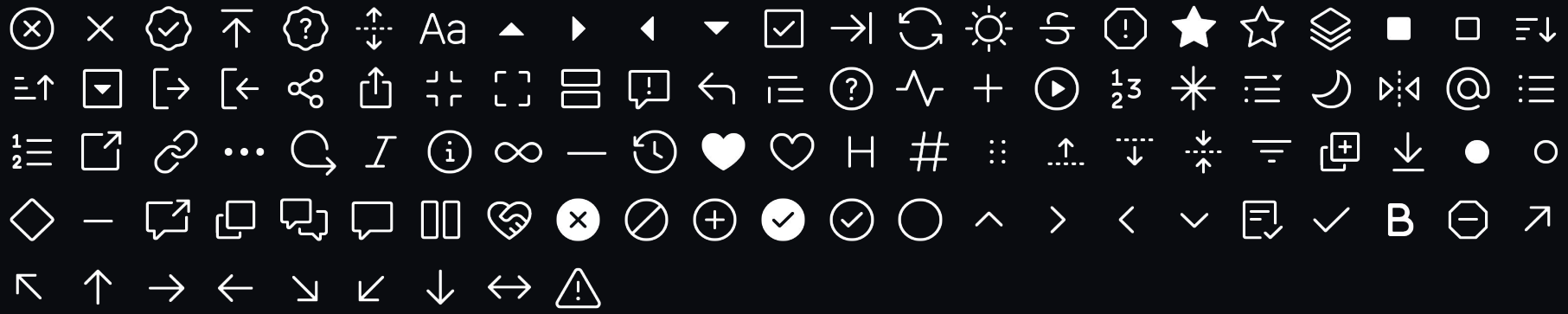
## Objects

## Developer

## People

# **Octicons** 2 of 2

Symbols



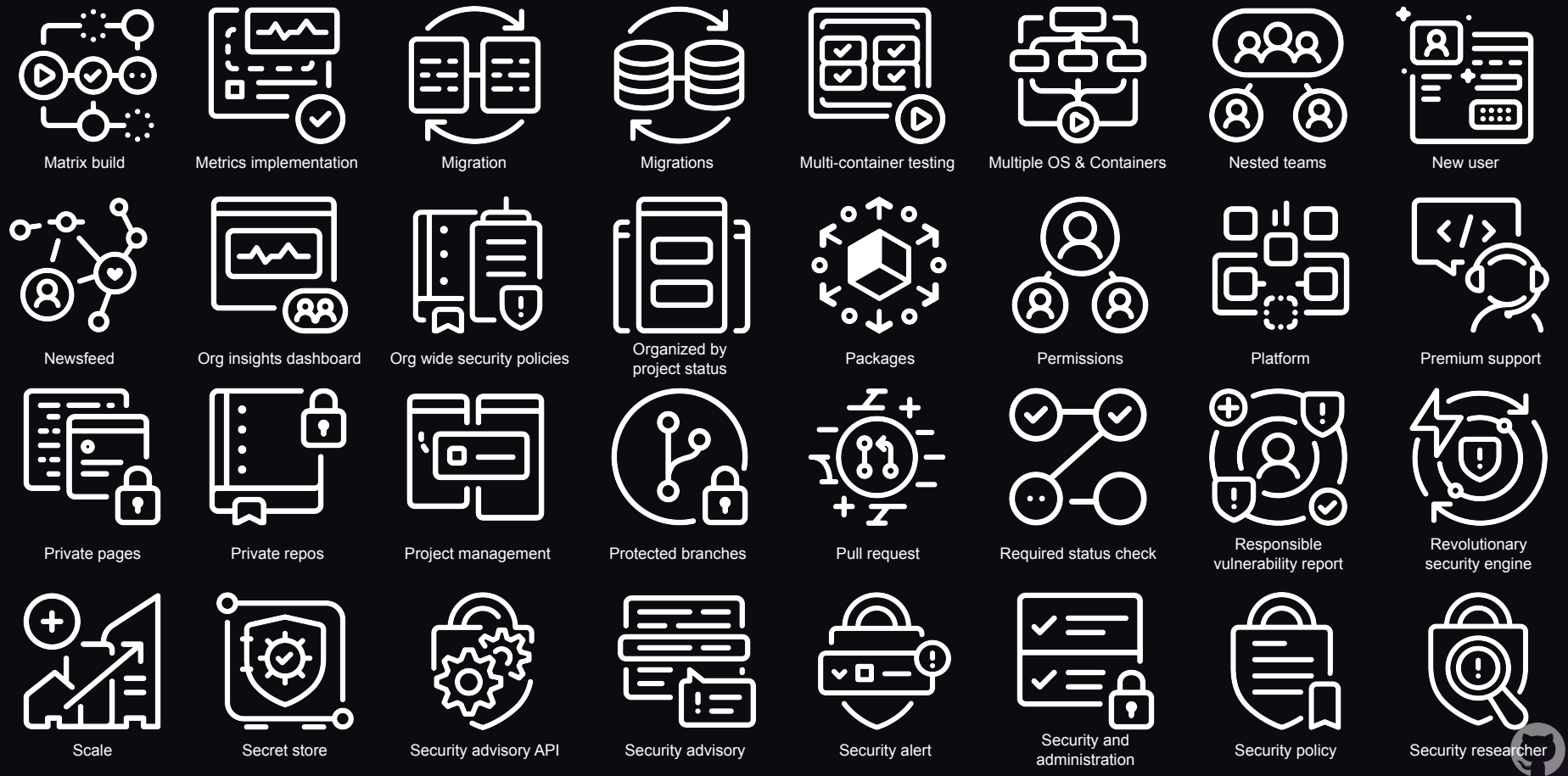
# GitHub icons 1 of 5

API
Accelerated SDLC
Access and permission
Actions
Add tasks to columns
Admin mentoring
Administration
All for one
Any language
Augmented reality
Automate your workflow
Autonomous coding
Blockchain
Bug Hunter
Build
Business
CLI easy creation
CLI seamless navigation
CLI status check
CVE record
Career Ladder
Celebration
Changelog
Code Nav
Code alert
Code as data
Code diff
Code hosting
Code
Collection
Community actions
Community forum

# GitHub icons 2 of 5

# GitHub icons 3 of 5

# GitHub icons 4 of 5



# GitHub icons 5 of 5

Security
See project activity
Service account engineer
Share work
Skywalker
Sponsors in product
Sponsors
Suggested changes
Team activity
Team discussions
Team mentions
Team metrics
Token scanning
Topic
Track progress
Train the trainer
Unified business
Unified contributions
Unified search
Used by
Virtual reality
Vulnerability data
Webhooks
Workflow consultation
Workflow settings

# We want your feedback!

Our goal is to offer you a template which makes it simple to create a beautiful presentation.Tell us if we succeeded or if there are ways in which we could improve this tool for you!

Feedback survey

# Control Flow Graph (CFG) in CodeQL JavaScript

- The CFG library in CodeQL provides classes for working with CFG-based program representation

- Located in codeql/javascript/ql/lib/semmle/javascript/CFG.qll