ORACLE

Runtime and Software Supply Chain Security

Current research at Oracle Labs

François Gauthier, PhD
Senior Principal Researcher
Oracle Labs Australia

whoami

Name: François Gauthier

Education: PhD Software Engineering, MSc Bioinformatics 

Experience: 12 years of experience, 10 years as an industrial researcher

Interests: program (analysis|testing|repair|synthesis), fuzzing, cybersecurity

Lightning Talk #1: Synthesizing SQLi Protections with RASPunzel

K. Vorobyov, F. Gauthier and P. Krishnan: Synthesis of Allowlists for Runtime Protection against SQLi, to appear at ICSE NIER 2024

SQL Injection 101

SQLi still sits in 3rd position of the OWASP Top 10, despite 20+ years of research

```
String sql = "SELECT * FROM items WHERE owner='" + user_name +"';"
```

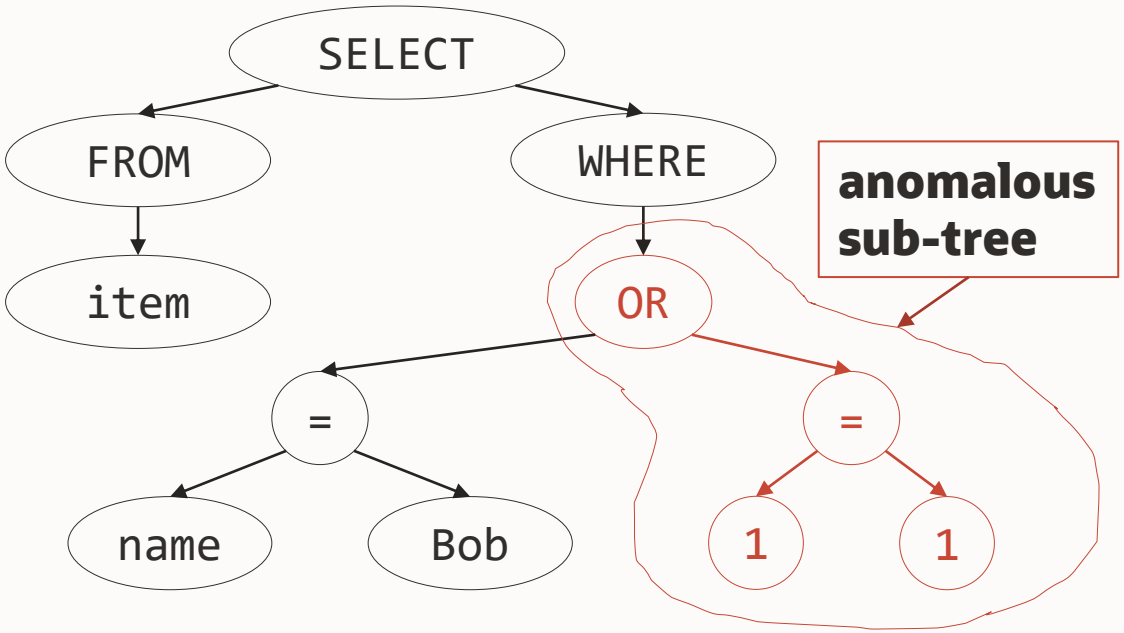
```
SELECT * FROM items WHERE owner='Bob';  //Select items owned by Bob
```

```
SELECT * FROM items WHERE owner='Bob' OR 1=1; --';  //Select all items
```

Runtime SQLi Prevention 101

State-of-the-art approaches detect syntactic anomalies

Anomalous sub-tree



Ceccato, Mariano, et al. "SOFIA: An automated security oracle for black-box testing of SQL-injection vulnerabilities." ASE. 2016.

Anomalous features

Type	SELECT
Tables	Items
Operators	=, OR
Functions	NA

anomalous operator

Jahanshahi, Rasoul, et al. "You shall not pass: Mitigating sql injection attacks on legacy web applications." AsiaCCS 2020.

Our Approach: Prevent Unwanted Information Disclosure

1. Deconstruct a benign query *b* into mappings $d \rightarrow <A,P>$

```
SELECT * FROM items WHERE owner='Bob';
```

Disclosed	Accessed	Predicate
*	owner	owner='Bob'

2. Deconstruct an incoming query *i*:

```
SELECT * FROM items WHERE owner='Bob' OR 1=1; --';
```

Disclosed	Accessed	Predicate
*	owner	owner='Bob' ∨ 1=1

3. Check that the incoming query *i* discloses no more information than benign query *b*.

- $d_i \subseteq D(b)$ ✓
- $A(i) \subseteq A(b)$ ✓
- $P(i) \Rightarrow P(b)$ ✗

More About Runtime Security In Our Papers

Synthesis of Allowlists for Runtime Protection against SQLi, ICSE NIER 2024

- How to formally deconstruct a SQL query into $d \rightarrow <A,P>$ mappings.
- How to generalise access and predicates mappings from a set of queries.

Synthesis of Java Deserialisation Filters from Examples, COMPSAC 2022

- Synthesis of regular expression filters with strong inclusion and exclusion guarantees to counter deserialization attacks.

Runtime prevention of deserialization attacks, ICSE NIER 2022

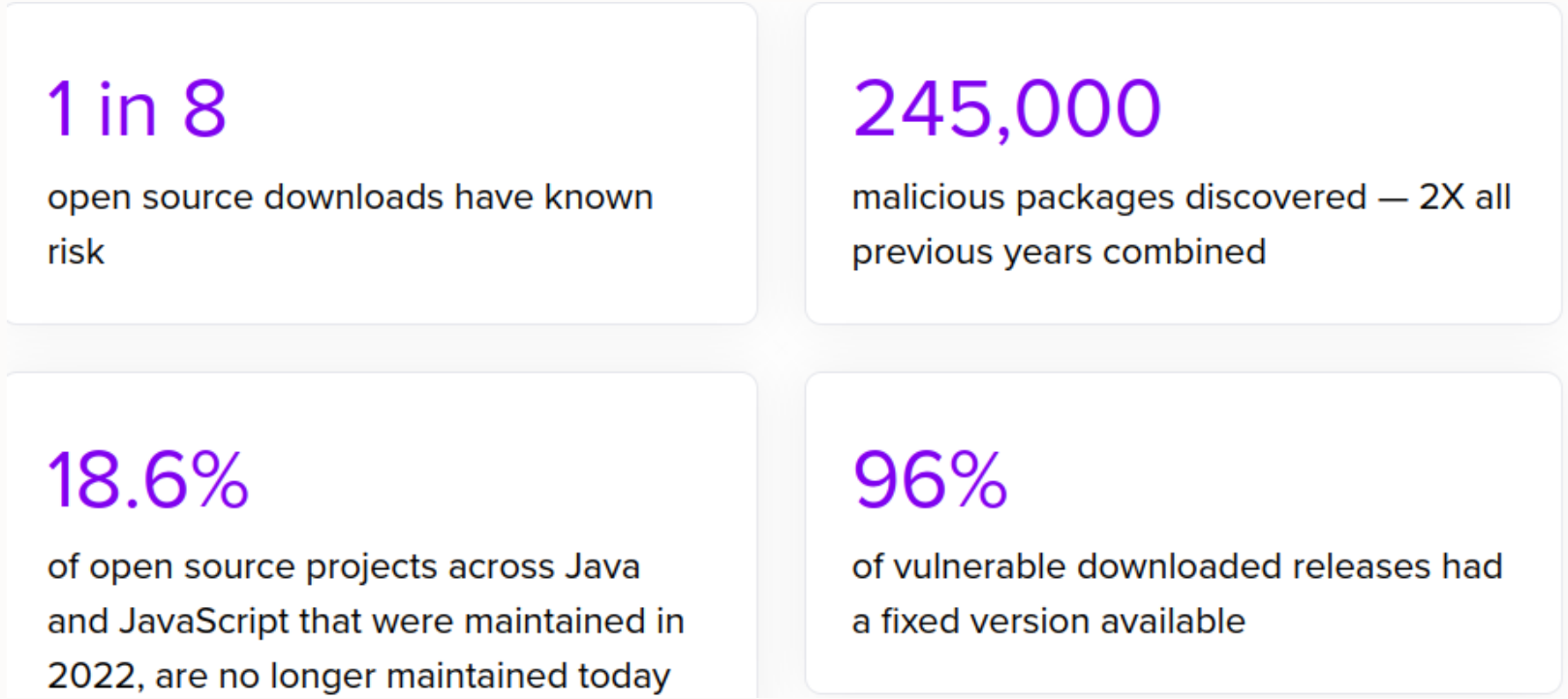
- Probabilistic learning of Markov-based deserialization filters.

Lightning Talk #2: Protecting the Software Supply Chain with Macaron

B. Hassanshahi, T.N. Mai , A. Michael, B. Selwyn-Smith, S. Bates, and P. Krishnan: Macaron: A logic-based framework for software supply chain security assurance, to appear at SCORED 2023

Software Supply Chain Attacks Are On The Rise

"In one year alone, we've seen twice as many supply chain attacks to the cumulative numbers in previous years."



https://www.sonatype.com/state-of-the-software-supply-chain/introduction

US Government Regulates The Software Supply Chain

Executive Order 14028, “Improving the Nation’s Cybersecurity”, mandates software supply chain security in Federal Software Systems built using closed source and/or open-sourced software.

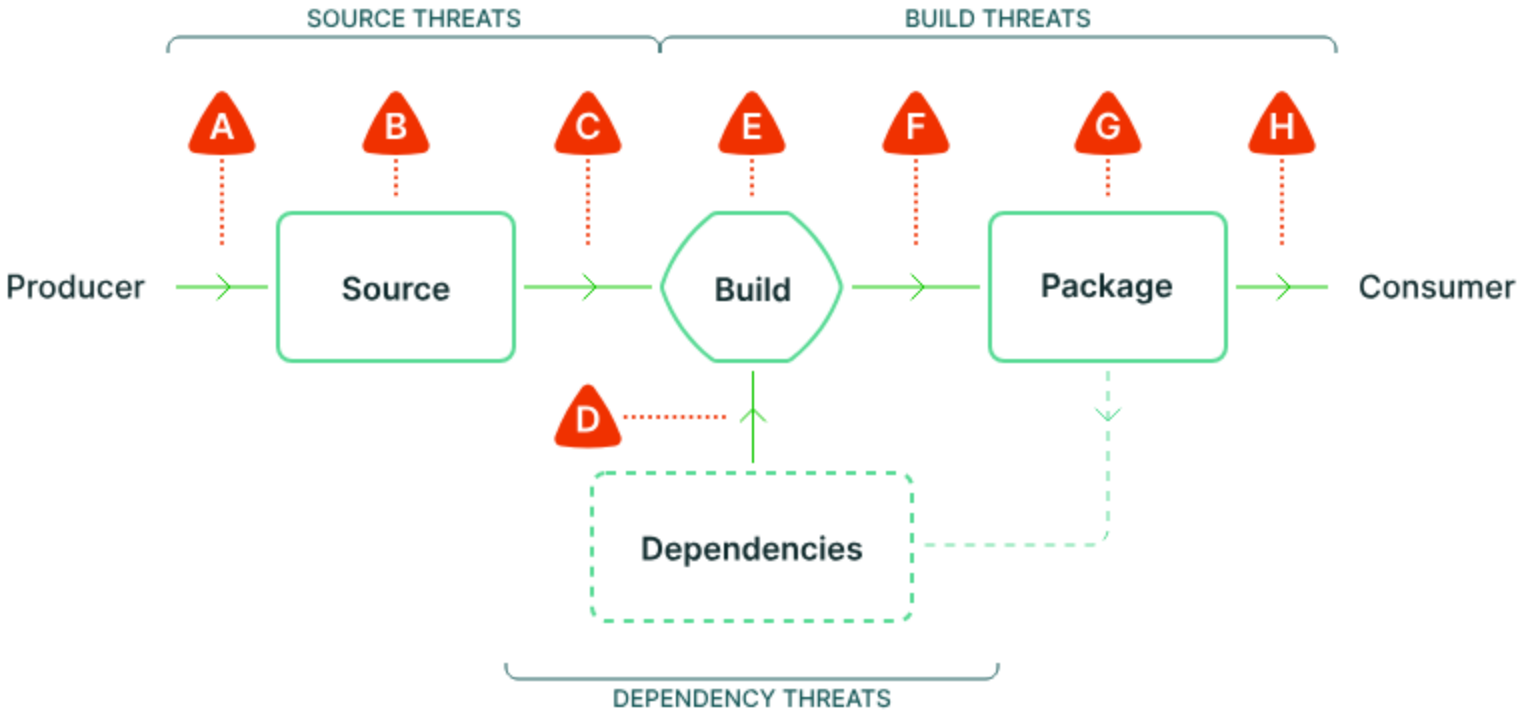
EO 14028 reiterates the need for software producers to follow a Secure Software Development Frame (SSDF) such as that created by the National Institute of Standards and Technology (NIST):

- A Software Bill of Materials (SBOM) associated with any software purchased by the Government.
- A way to check for and automate vulnerability remediation.
- **Provenance**, or the ability to be able to identify the origin for all software components

Supply Chain Threats and Remediations

https://slsa.dev/spec/v1.0/threats-overview

- A: Code analysis
- B/G: Protect core infrastructure
- C/D/E/F/H: Project Macaron
  - Analysis of build and deployment processes (E, F)
  - Provenance generation (H)
  - Provenance checking (C, D)



SOURCE THREATS

A Submit unauthorized change

B Compromise source repo

C Build from modified source

DEPENDENCY THREATS

D Use compromised dependency

BUILD THREATS

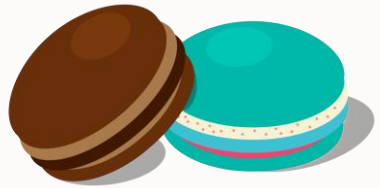
E Compromise build process

F Upload modified package

G Compromise package registry

H Use compromised package

Macaron To The Rescue



An extensible framework designed for supply chain security

Already comes with abstractions for development and infrastructure tooling

- build tools, versions controls, CI configurations, package registries

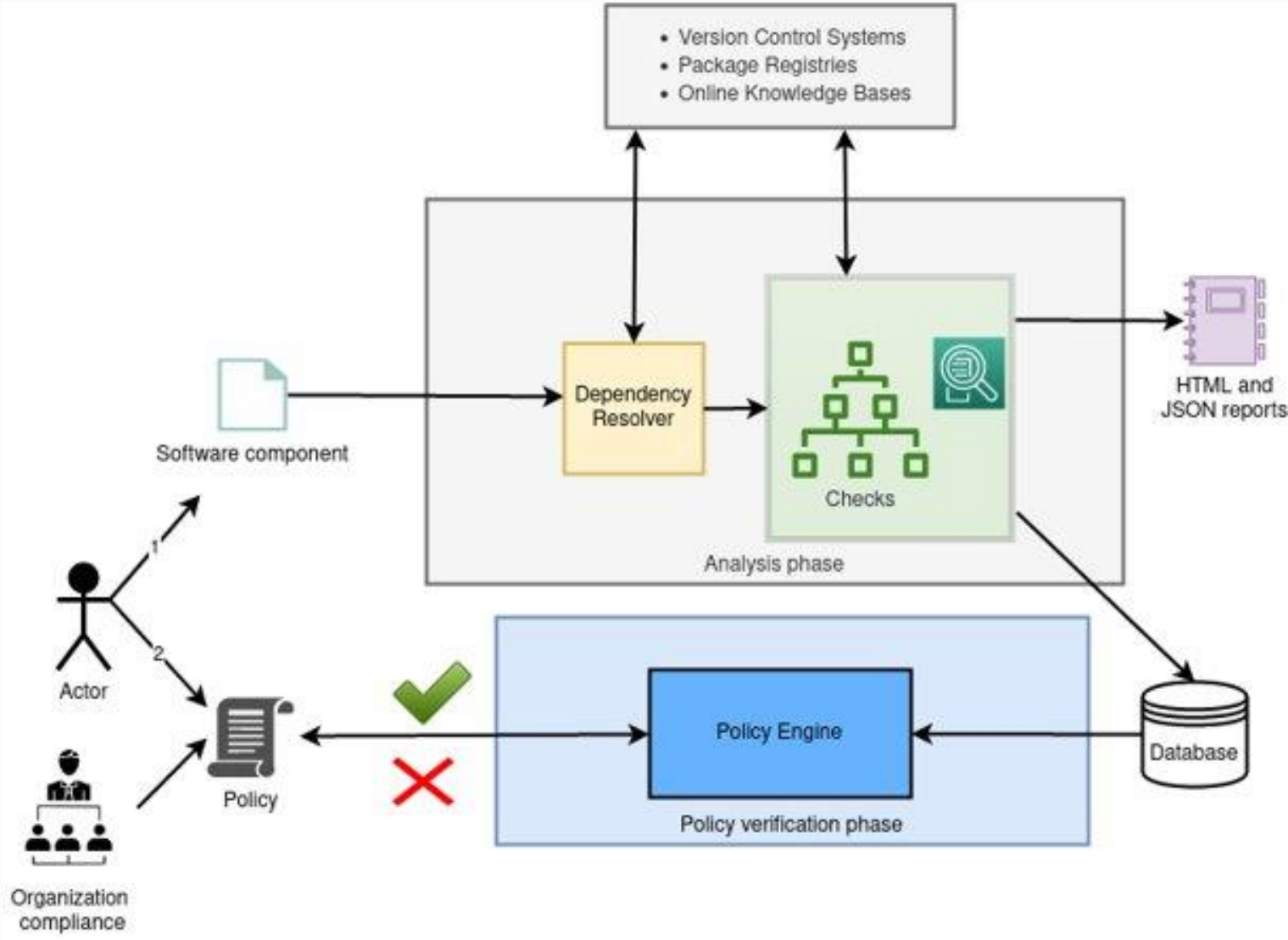
If you have an idea for a security property, it allows you to write a check easily in few lines of code, e.g., you don't need to worry about things like

- Finding a repository for an artifact and cloning
- Static analysis of GitHub Actions and bash scripts
- Discovering artifacts on registries
- Language-specific build commands

Automatically prepares the collected evidence to be used by a policy engine

Macaron: A Supply Chain Security Analysis Tool

- Inspired by the Supply Chain Levels for Software Artifacts (SLSA) specification.
- Two-tier build analysis architecture:
  - Low-level checks produce atomic facts about the build.
  - High-level policies, encoded as Soufflé Datalog programs, build on atomic facts to verify build properties.



Example Checks and Policies Provided by Macaron

`BuildPlatform` check is satisfied if:

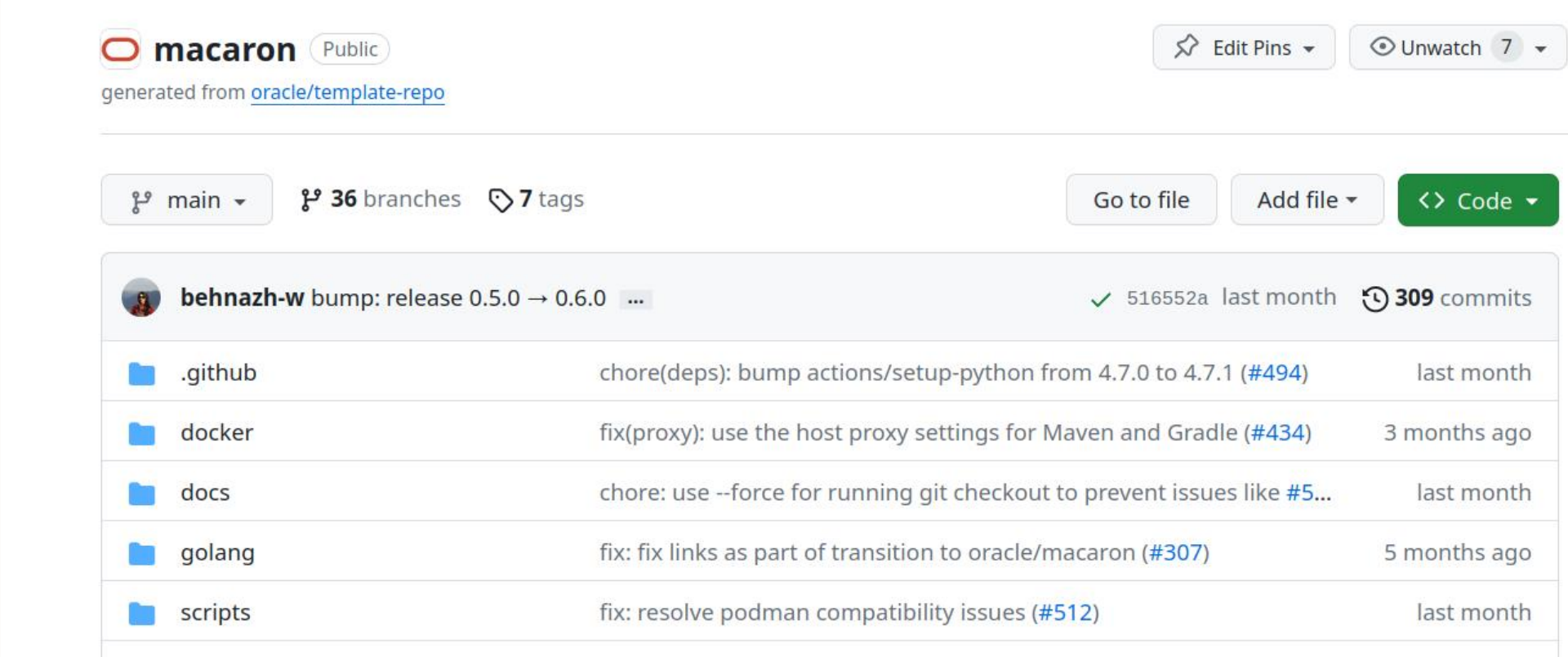
- The software component is built and deployed using a hosted build platform ✓
- The build artifact includes provenance information ✓
- The provenance information can be verified ✓

```
Policy("SLSA2-transitive",parent) :-
  Dependency(parent, child),
  SLSA2(parent),
  SLSA2(child).

SLSA2(component) :-
  ProvenanceAvailable(component, "SLSA2"),
  BuildPlatform(component, "passed").

apply_policy_to("SLSA2-transitive", component) :-
  is_component(component).
```

Macaron is open source and welcomes contributions!

https://github.com/oracle/macaron

https://oracle.github.io/macaron/



Let’s have a chat.

francois.gauthier@oracle.com
Oracle Labs Australia