GitLab

The One DevOps Platform

A Go micro language framework for building Domain Specific Languages

GitLab

whoami

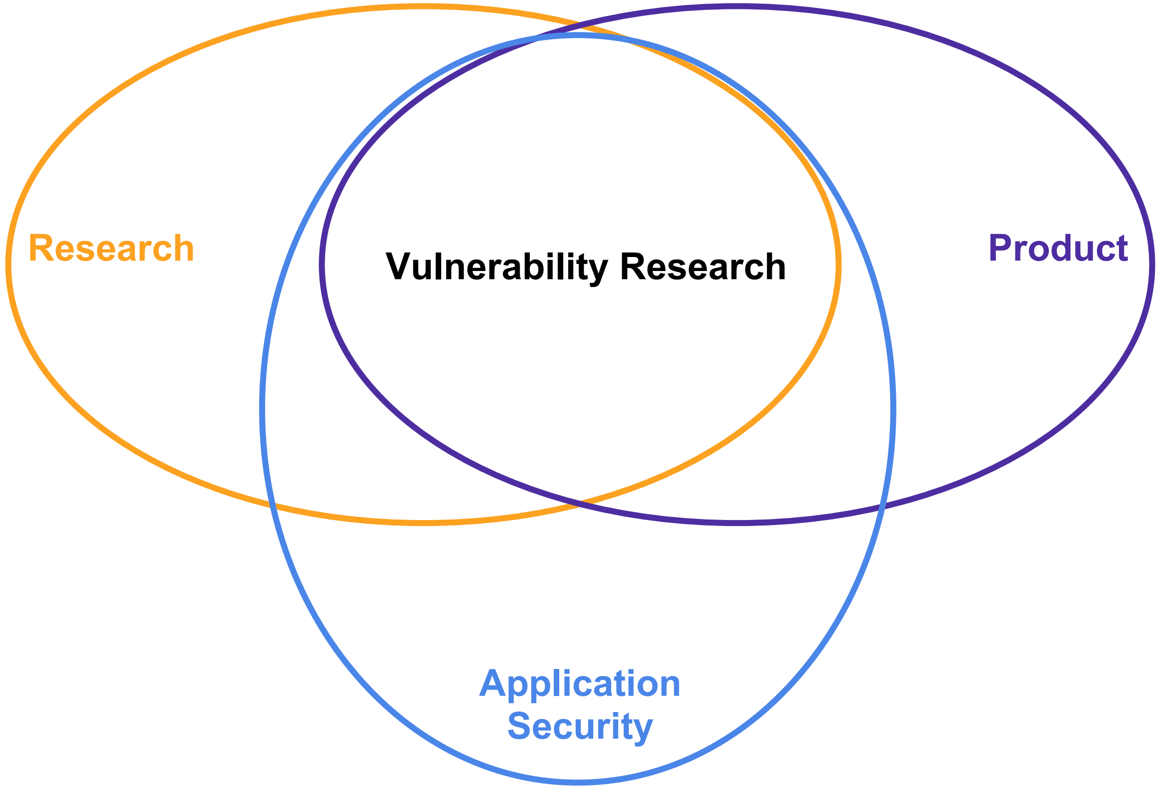
GitLab

$ whoami

- **Julian Thome**
  - Bitburg, Germany -> Zweibrücken (Dipl.-Inf. (FH)) -> Saarbrücken, Germany (MSc) -> Luxembourg (PhD)
  - Areas of (research/engineering) interests:
    - Security
    - Programming Languages and Compilers
    - Symbolic Execution
  - Vulnerability Research Engineer @ GitLab since 2019

Vulnerability Research @ GitLab

Vulnerability Research @ GitLab - Product Research and AppSec



Vulnerability Research @ GitLab

- **Mark Art - Vulnerability Research Manager**
  - Australia
  - Team & product direction

- **Isaac Dawson - Staff Vulnerability Research Engineer**
  - Japan
  - DAST/Browser specialist

Vulnerability Research @ GitLab

- **Dinesh Bolkensteyn - Sr. Vulnerability Research Engineer**
  - Switzerland
  - SAST specialist

- **Michael Henriksen - Sr. Vulnerability Research Engineer**
  - Denmark
  - Web application security specialist

Vulnerability Research @ GitLab

- Experiment with technology
  - Advance GitLab's capabilities in Security
  - Proof-of-concepts (PoC's) to improve the product
- Curate and maintain the GitLab Advisory Database
- Facilitate the CNA (CVE Numbering Authority) relationship
  - We curate software vulnerability feeds in and out of GitLab

LinGo: A Go micro language framework for building Domain Specific Languages

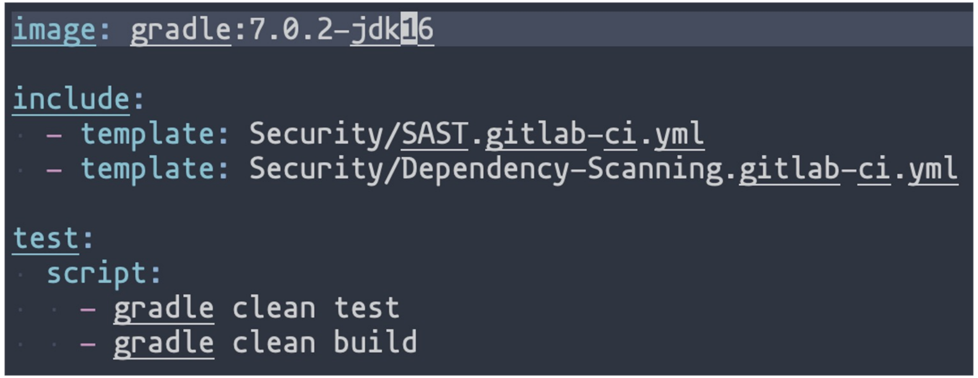
GitLab

Motivation - Context

- GitLab is a DevOps Platform.
- Main programming languages: Go, Ruby
- We are striving towards keeping the ECO systems we are working with as reduced as possible in order to prevent the build-up of technology or knowledge silos.

Motivation - Configuration-as-code

- GitLab ships a variety of different CI/CD tools (for example SAST/DAST) some of which are developed internally.
- We follow the configuration-as-code principle where our CI configuration format.



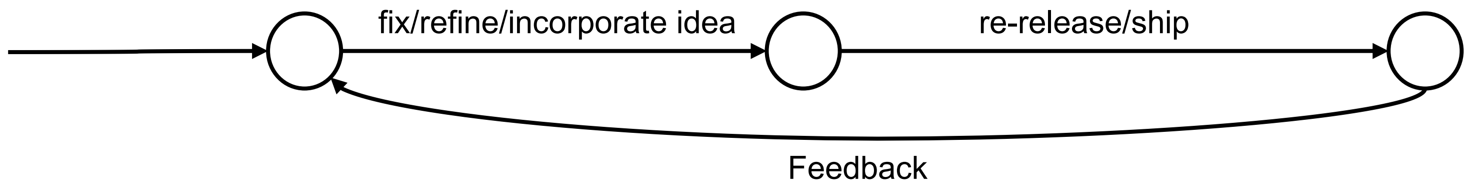
```yaml
image: gradle:7.0.2-jdk16

include:
  - template: Security/SAST.gitlab-ci.yml
  - template: Security/Dependency-Scanning.gitlab-ci.yml

test:
  script:
    - gradle clean test
    - gradle clean build
```

Motivation - Problem

- Tool for FP elimination.
- Iterator design.
- Parsing and evaluation potentially close.
- Language was developed iteratively from scratch.



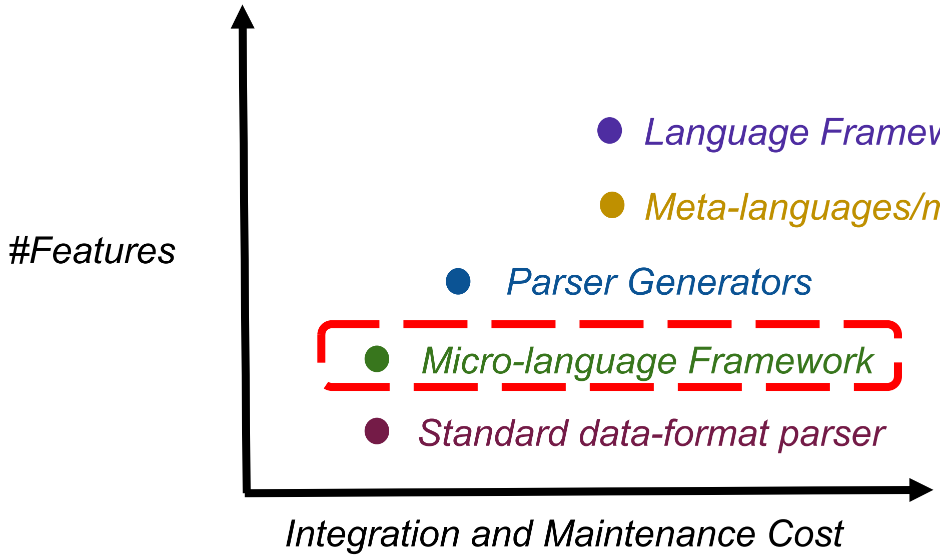
Motivation - “Micro-language” Requirements

1. **Stability:** Changes applied to the DSL should neither require any changes to the core lexer/parser implementation nor to the language processor implementation.
2. **Flexibility/Composability:** New DSL concepts (data-types, functions) can be integrated via a simple plug-in mechanism.
3. **Simplicity:** the language framework should have just enough features to provide a foundation that is powerful enough to implement and evolve a custom DSLs. In addition, the whole implementation of the micro language framework should be in pure Go so that it is easily embeddable in Go applications.

Motivation - Available DSL tools

There are great tools that help with DSL development

- **Language frameworks/workbenches:** Xtext, MPS
- **Parser generators:** ANTLR, bison, tree-sitter, text-mapper
- **Meta-languages and interpreter modules:** Racket, go-lua, yaegi, zygomys



GitLab

LinGo

GitLab

LinGo

1. Micro-language framework to design your **L**ISP-based Domain Specific Languages **in Go.**
2. ~3K lines of pure Go code.
3. S-expressions in prefix notation.
4. Macro support.
5. Evaluator uses depth-first traversal in post-order.
6. Lingo is designed in such a way that new functions can be plugged-in by implementing an interface and registering the newly implemented function to make it available.

Demo

Demo - **R**andom **T**ext **G**enerator: A data generation engine used for fuzzing

```
name, balance
Lisa, 100.30
Bert, 241.41
Maria, 151.13
```

balances.csv

```
#!/usr/bin/env ruby

exit(1) if ARGV.empty? || !File.exist?(ARGV[0])

sum = 0
File.foreach(ARGV[0]).each_with_index do |line, idx|
  next if idx == 0
  sum += Float(line.split(',')[1])
end

puts sum.round(2)
```

computebalance.rb

```
#!/usr/bin/awk -f

BEGIN{FS=","}{sum+=$2}END{print sum}
```

computebalance.awk

GitLab

Motivation - Language concepts to include

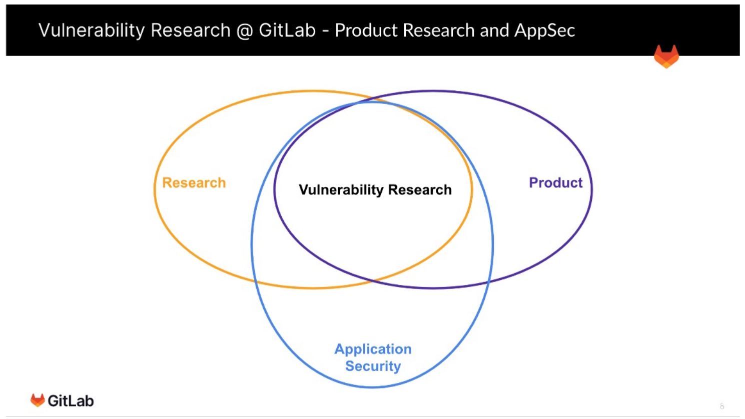
Our RTG language includes the following functions:

1. (**oneof** s0, s1, ..., sN): randomly returns one of the parameter strings.
2. (**join** s0, s1, ..., sN): joins all strings.
3. (**genfloat** min max): generates a random float number and returns it.
4. (**times** num exp): repeats the pattern generated by exp num times.

GitLab

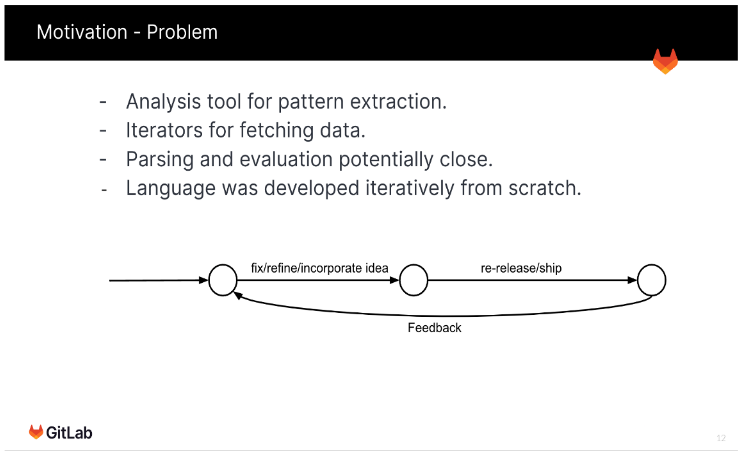
Summary

Summary



Vulnerability Research @ GitLab - Product Research and AppSec

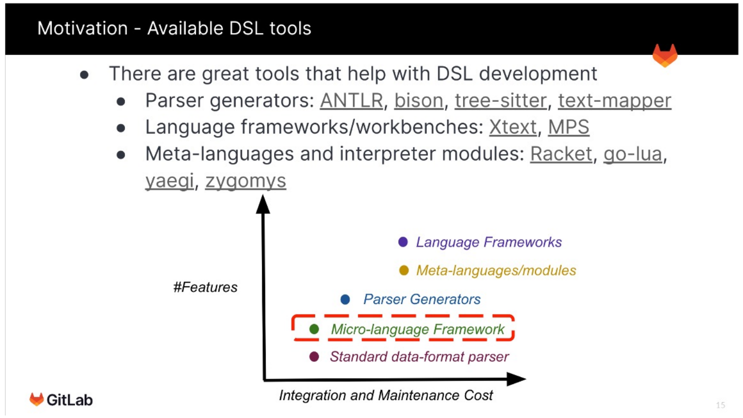
GitLab



Motivation - Problem

- Analysis tool for pattern extraction.
- Iterators for fetching data.
- Parsing and evaluation potentially close.
- Language was developed iteratively from scratch.

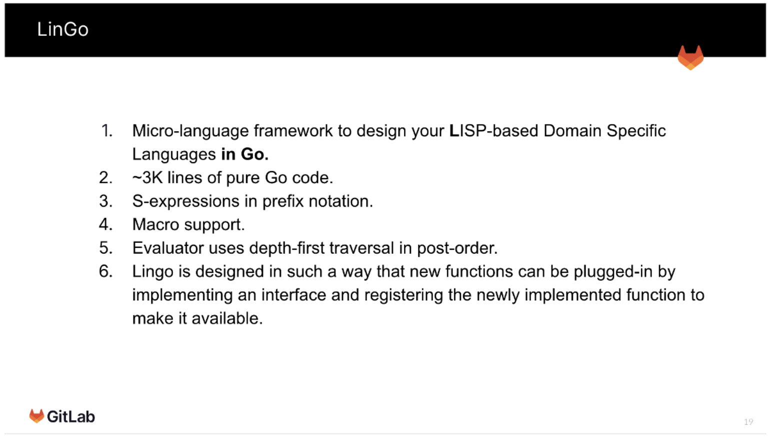
GitLab



Motivation - Available DSL tools

- There are great tools that help with DSL development
  - Parser generators: ANTLR, bison, tree-sitter, text-mapper
  - Language frameworks/workbenches: Xtext, MPS
  - Meta-languages and interpreter modules: Racket, go-lua, yaegi, zygomys

GitLab



LinGo

1. Micro-language framework to design your **L**ISP-based Domain Specific Languages **in Go.**
2. ~3K lines of pure Go code.
3. S-expressions in prefix notation.
4. Macro support.
5. Evaluator uses depth-first traversal in post-order.
6. Lingo is designed in such a way that new functions can be plugged-in by implementing an interface and registering the newly implemented function to make it available.

GitLab

GitLab

Thank You

GitLab