Life in the fAST Lane

DRYRUN .SECURITY

https://github.com/cktricky/Life-in-the-fAST-Lane

Common language / terminology

Dynamic Application Security Testing (DAST)	Sending requests to running web applications and then observing & analyzing the application's behavior.
Static Application Security Testing (SAST)	Analysis of the source code, in its a non-execution environment, for patterns & anti-patterns that indicate a potential security issue.
Contextual Security Analysis (CSA)	A modern, comprehensive risk assessment of software changes using a multitude of factors/data-points.
AI / LLM	Witch magic

SAST

- A Brief History: SAST & Me
- Practical use cases
- Language examples:
  - Node.js + Babel
  - Python + ast

CSA + AI

- Introducing Contextual Security Analysis
- S.L.I.D.E.
- Integrating patterns into AI
- Code examples of performing analysis

https://github.com/cktricky/Life-in-the-fAST-Lane

A Brief History:

Static Application Security Testing (SAST)
&
Me

DRYRUN
.SECURITY

https://github.com/cktricky/Life-in-the-fAST-Lane

Ken Johnson
@cktricky

Co-Host of Absolute AppSec
CTO of DryRun Security
BJJ Nerd

DRYRUN
.SECURITY

A Brief History: SAST

A Brief History: Me

1995 JavaScript released	I was released… into 7th grade 🤮
The first SQL Injection vulnerability discovered in 1998	Thankfully, Gateway PCs were all the rage. Discovered how much I loved computers,
2001 OWASP founded, Mark Curphey	2001 I go into the US Navy to become an IT
2002 Ounce Labs formed, sold to IBM in 2009. 2003 Fortify formed, sold to HP in 2010. 2006 Veracode, sold in 2022 to TA Associates. 2006 CheckMarx Founded. Sold in 2020 to Hellman & Friedman.	2001-2007 Learn a bunch, leave the navy, work professionally in IT. Code/Hack at night/weekends for a hobby. Learned SQL, HTML, Ruby, etc.

A Brief History: SAST + Me

2008 - 2012 AppScan Source, Fortify, and Veracode dominate the SAST space	2008-2010 - Offered a contracting job for Pentagon (AppSec… before appsec) 2010-2012 - Consultant at Fishnet Security using Ounce Labs / AppScan Source
2012 Dependency check released, added to OWASP 2013 Top 10 (A9) 2013 Brakeman released as OSS, SAST for Ruby on Rails 2015? Retire.js OSS is released for JavaScript	2012-2013 LivingSocial AppSec Manager 2013 - 2017 CTO of nVisium

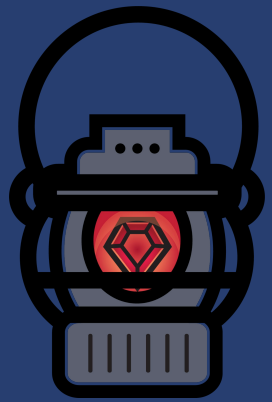
IMPORTANT!!!

DRYRUN .SECURITY

https://github.com/cktricky/Life-in-the-fAST-Lane

Why is this important?

- FREAKING OSS?! AMAZING 😍
- **OWASP Top 10 A9**: Using Known Vulnerable Components.
  - Dependencies are now being analyzed.
  - Retire.js contributes to this as well… mostly JQuery focused but still
- Brakeman really demonstrates the power of combining flow control AST with a highly opinionated framework (Ruby on Rails)



A Brief History: SAST

A Brief History: Me

2015 Snyk founded, 2017 started really gaining momentum with developers	2017 I ended my journey as CTO, began my journey as an IC on GitHub's AppSec team
2019 r2c is a company, 2020 rebrands sgrep to Semgrep	2018 Seth & Ken's excellent adventures course launched; preaching an agnostic approach to code review. Clint Gibler & Leif Dreizler are students in the first course. Both go on to join Semgrep.
Semmle (CodeQL) acquired by GitHub 2019	GitHub team begins uses this tooling and CodeQL becomes a part of GitHub Advanced Security.

Takeaways…

👩‍💻SAST Tools built & marketed with developers in mind

⚡ Some tooling provides a somewhat language agnostic approach to SAST, quick way to write rules (Semgrep)

⚙️Others are very highly tuned (CodeQL) and accurate but difficult to extend

🖥️ Integrated more closely into the development pipeline

🙅‍♂️Perfection in SAST does NOT exist, use it for what its worth

DRYRUN .SECURITY

https://github.com/cktricky/Life-in-the-fAST-Lane

Conclusion

Write something:

✅ Familiar to developer's so that they can easily extend & use

✅ Harnesses existing tooling & pipeline

✅ Balance between speed and perfection

DRYRUN .SECURITY

https://github.com/cktricky/Life-in-the-fAST-Lane

Practical Use Cases

Practical Use Cases

Awareness:

Looking for changes in security impacting policies and configurations

Variant Analysis:

Looking for similar issues that have cropped up in the past

Regression analysis:

Looking for areas where regression may occur

Write your own

https://github.com/cktricky/Life-in-the-fAST-Lane

Overview / Outline

JavaScript & TypeScript + Babel.js

Python + AST

Ruby + sexp

DRYRUN .SECURITY

https://github.com/cktricky/Life-in-the-fAST-Lane

JS/TS - Babel.js

Babel.js

Features:

Modular - You can run multiple rules at once

Handles various major versions of JS

Includes transpiling when conversion is necessary

Home Page: https://babeljs.io/

Important Terminology

Abstract Syntax Tree	Converts source code into chunks with specific instructions and format that can be traversed to look for patterns.
Parse - @babel/parse	Parses the source code and converts to an AST. Supports JSX, Flow, and Typescript.
Traversal - @babel/traversal	Traverses the Abstract Syntax Tree generated by parsing the source code. You provide rules (ie - patterns) to this function and it returns any discovered matching patterns.
Types - @babel/types	Convenience functions for traversing AST and identifying relevant nodes

Helpful tool to visualize an AST in JS/TS

ASTExplorer.net

Homepage: https://astexplorer.net

astexplorer.net

AST Explorer | Snippet | JavaScript | @babel/parser | Transform | default | ? Parser: @babel/parser-7.19.0

```
1 app.get('/', (req, res) => {
2     res.redirect(req.query.url);
3 });
```

Tree | JSON 12ms

☑ Autofocus ☑ Hide methods ☐ Hide empty keys ☐ Hide location data ☐ Hide type keys

```
    end: 65
  + loc: {start, end, filename, identifierName}
    sourceType: "module"
    interpreter: null
  - body: [
    - ExpressionStatement {
        type: "ExpressionStatement"
        start: 0
        end: 65
      + loc: {start, end, filename, identifierName}
      - expression: CallExpression {
          type: "CallExpression"
          start: 0
          end: 64
        + loc: {start, end, filename, identifierName}
        + callee: MemberExpression {type, start, end, loc, object, ... +2}
        + arguments: [2 elements]
        }
      }
    ]
    directives: [ ]
  }
  comments: [ ]
}
```

Built with React, Babel, Font Awesome, CodeMirror, Express, and webpack | GitHub | Build: 38c0d37

DRYRUN .SECURITY

https://github.com/cktricky/Life-in-the-fAST-Lane

Parse source code

```
const { parse } = require('@babel/parser');
// Usage example
const directoryName = './exampleApp'; // Replace with your directory name
aggregateContents(directoryName)
  .then(content => {
    var ast = parse(content, {
        sourceFilename: "index.js"
        // OPTIONAL - Add plugins like tsc, jsx, flow, etc yourself
        //plugins: ["jsx"]
    });
```

DRYRUN .SECURITY

https://github.com/cktricky/Life-in-the-fAST-Lane

Traverse the AST

```
1 app.get('/', (req, res) => {
2     res.redirect(req.query.url);
3 });
```

```
- callee: MemberExpression {
    type: "MemberExpression"
    start: 33
    end: 45
  + loc: {start, end, filename, identifierName}
  - object: Identifier {
      type: "Identifier"
      start: 33
      end: 36
    + loc: {start, end, filename,
      identifierName}
      name: "res"
    }
    computed: false
  - property: Identifier {
      type: "Identifier"
      start: 37
      end: 45
    + loc: {start, end, filename,
      identifierName}
      name: "redirect"
    }
  }
```

```
      + loc: {start, end, filename,
        identifierName}
        name: "req"
    }
    computed: false
  - property: Identifier = $node {
      type: "Identifier"
      start: 50
      end: 55
    + loc: {start, end, filename,
      identifierName}
      name: "query"
    }
  }
  computed: false
- property: Identifier {
    type: "Identifier"
    start: 56
    end: 59
  - loc: {
    + start: {line, column, index}
    + end: {line, column, index}
      filename: undefined
      identifierName: "url"
```

DRYRUN .SECURITY

https://github.com/cktricky/Life-in-the-fAST-Lane

Traverse the AST

```
const redirectCheckPlugin = require("./redirectCheckPlugin");
// Usage example
const directoryName = './exampleApp'; // Replace with your directory name
aggregateContents(directoryName)
  .then(content => {
    var ast = parse(content, {
        sourceType: "module"
        // OPTIONAL - Add plugins like tsc, jsx, flow, etc yourself
        //plugins: ["jsx"]
    });
    traverse(ast, redirectCheckPlugin().visitor );
  })
  .catch(error => {
    console.error('Error reading contents:', error);
  });
```

DRYRUN .SECURITY

https://github.com/cktricky/Life-in-the-fAST-Lane

Traverse the AST

```
// redirectCheckPlugin.js

const t = require("@babel/types");
module.e
    retu
        visitor: {
            CallExpression(path) {
                if (
                    t.isMemberExpression(path.node.callee) &&
                    t.isIdentifier(path.node.callee.object, { name: "res" }) &&
                    t.isIdentifier(path.node.callee.property, { name: "redirect" }) &&
                    t.isMemberExpression(path.node.arguments[0]) &&
                    t.isMemberExpression(path.node.arguments[0].object) &&
                    t.isIdentifier(path.node.arguments[0].object.object, { name: "req" }) &
                    t.isIdentifier(path.node.arguments[0].object.property, { name: "query"
                    t.isIdentifier(path.node.arguments[0].property, { name: "url" })
                ) {
                    console.log("Detected use of req.query.url inside res.redirect()");
                }
            }
        }
    };
};
```

```
(property) visitor: {
    CallExpression(path: any): void;
}
```

```
- callee: MemberExpression {
    type: "MemberExpression"
    start: 33
    end: 45
  + loc: {start, end, filename, identifierName}
  - object: Identifier {
      type: "Identifier"
      start: 33
      end: 36
    + loc: {start, end, filename, identifierName}
      name: "res"
    }
    computed: false
  - property: Identifier {
      type: "Identifier"
      start: 37
      end: 45
    + loc: {start, end, filename, identifierName}
      name: "redirect"
    }
  }
```

DRYRUN .SECURITY

https://github.com/cktricky/Life-in-the-fAST-Lane

cktricky@Kens-MacBook-Pro: ~/code/Life-in-the-fAST-Lane/js

..-fAST-Lane/js (-zsh)

~/code/Life-in-the-fAST-Lane/js main ?1

ok fAST py 18.13.0 node 11:50:20

DRYRUN .SECURITY

https://github.com/cktricky/Life-in-the-fAST-Lane

ast - Python

Important Terminology

Visitor	The Python/ast equivalent of traversal in Node/Babel.js. Allows you to walk the AST looking for nodes that match a specific pattern.
node	Same as JS, this represents a chunk of code
node.func	References a function call

ast

Features:

Built-in! Makes deployment easy :-)

Modular - You can run multiple rules at once

Can be run from an interactive session (terminal)

Similar to Babel, parses source code, generates an AST, and allows you to traverse (visit) the tree

Home Page:
https://docs.python.org/3/library/ast.html

Helpful tool to visualize an AST in Python

Python AST Explorer

Homepage:
https://python-ast-explorer.com/

python-ast-explorer.com

Python AST explorer

Star 240 Fork 31

Examples: Simple class Decorator Lambda with ... as

Options: Show null attrs

```
@login_required
def search(request):
    query = request.GET.get('q', '')

    my_project_list = Project.objects.filter(
        users_assigned=request.user.id).filter(
            title__icontains=query).order_by('title')

    # TODO Task list query is complicated, switch to using ORM
sometime soon.
    task_query = "%%" + query + "%%"
    sql = "select * from taskManager_task as t INNER JOIN
taskManager_task_users_assigned as a ON t.id = a.task_id WHERE
t.text LIKE '%s' OR t.title LIKE '%s' AND a.user_id = %d" %
(task_query,task_query,request.user.id)
    my_task_list = Task.objects.raw(sql)

    #my_task_list = Task.objects.filter(
    #    users_assigned=request.user.id).filter(
    #        title__icontains=query).order_by('title')
    return render(request,
                  'taskManager/search.html',
                  {'q': query,
                   'task_list': my_task_list,
                   'project_list': my_project_list,
                   'user': request.user})
```

```
▼ Module: {} 2 keys
  ▼ body: [] 1 item
    ▼ 0: {} 1 key
      ▼ FunctionDef: {} 6 keys
        ▼ args: {} 1 key
          ▼ arguments: {} 7 keys
            ▼ args: [] 1 item
              ▼ 0: {} 1 key
                ▼ arg: {} 3 keys
                    annotation: null
                    arg: "request"
                    type_comment: null
              defaults: [] 0 items
              kw_defaults: [] 0 items
              kwarg: null
              kwonlyargs: [] 0 items
              posonlyargs: [] 0 items
              vararg: null
        ▼ body: [] 6 items
```

DRYRUN .SECURITY

https://github.com/cktricky/Life-in-the-fAST-Lane

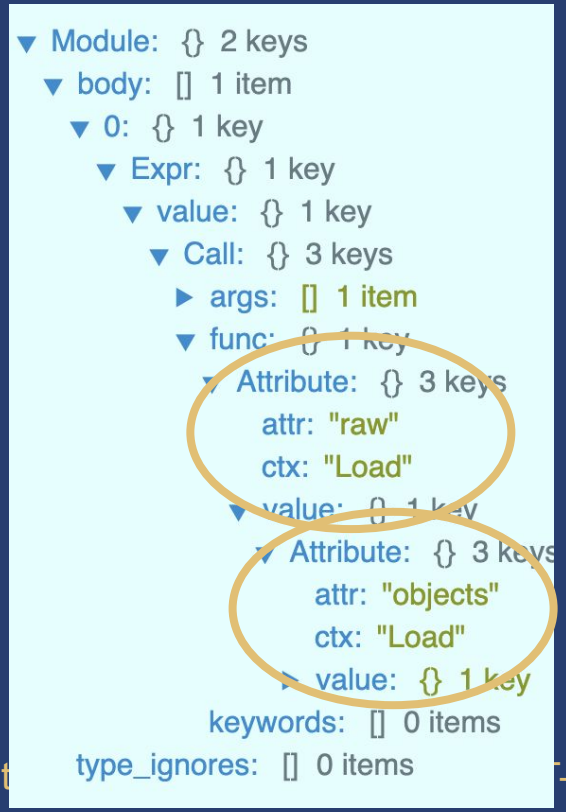
Parse source code

```
# Example
path = input("Enter a file or directory path: ")
code = get_contents(path)



tree = ast.parse(code)
```

Traverse (Visit) the AST

Python AST Explorer

```
Task.objects.raw(sql)
```



DRYRUN .SECURITY

https://git -Lane

Traverse the AST

```
class RawSQLVisitor(ast.NodeVisitor):
    def __init__(self):
        self.has_raw_sql = False

    def visit_Call(self, node):
        if (isinstance(node.func, ast.Attribute) and
            isinstance(node.func.value, ast.Attribute) and
            node.func.value.attr == 'objects' and
            node.func.attr == 'raw'):
            self.has_raw_sql = True
        self.generic_visit(node)

    import os
```

DRYRUN .SECURITY

https://github.com/cktricky/Life-in-the-fAST-Lane

Loading multiple rules

```
tree = ast.parse(code)
visitors = [ExampleVisitor(), RawSQLVisitor()]

for VisitorClass in visitors:
    visitor = VisitorClass        You, 1 second ag
    visitor.visit(tree)
```

DRYRUN .SECURITY

https://github.com/cktricky/Life-in-the-fAST-Lane

cktricky@Kens-MacBook-Pro: ~/code/Life-in-the-fAST-Lane/python

..T-Lane/python (-zsh)

~/code/Life-in-the-fAST-Lane/python main

ok fAST py 18.13.0 node 11:59:53

DRYRUN .SECURITY

https://github.com/cktricky/Life-in-the-fAST-Lane

Final thoughts on building your own SAST

🔮 Know what you're trying to use it for

🚧 Integrate into existing tooling and processes

📊 SAST is a very useful data point but it is but one of many available to us

🥊 Combine SAST results with additional factors that represent risk

DRYRUN .SECURITY

https://github.com/cktricky/Life-in-the-fAST-Lane

Contextual Security Analysis

Context > Control

Control

(S,D,I)AST embodies this:

😭Limited data points

😭Enforcement of rules

😭Blocking checkpoints

😭Pattern matching

Context

Next Gen:

😍Combines many data points

😍Warnings & guidance over enforcement

😍 Remediation guidance

😍 Risk-focused

Download Link: https://www.dryrun.security/resources/csa-guide

DRYRUN .SECURITY

https://github.com/cktricky/Life-in-the-fAST-Lane

CSA - SLIDE □

Surface	How the surface of the application changes
Language	Language and framework that the application is written in
Intent	Evaluates the person making the change, both in their patterns and their purpose
Detection	Tooling in place to detect vulnerabilities and security issues
Environment	Purpose of the application and service in the organization

SLIDE - Surface

⚠️Sensitive Code Paths

Depending on your language and framework, this could be configuration files, middleware, controllers, or how your application and service handles auth.

🚧 HTTP Routes

Routes refers to an exposed endpoint on an application that will take user-supplied information and perform an action. Expansion or contractions of these routes in an application inherently change the risk profile of the application.

DRYRUN .SECURITY

https://github.com/cktricky/Life-in-the-fAST-Lane

SLIDE - Language

💬Source Language

This weighs the language used (ex: Golang, Rust, PHP, etc.) as each has their own security issues and known vulnerabilities.

🧱Web Framework

Each web framework contains their own unique variations of sensitive areas, configurations, and components. Framework nuances are subtle and highly specific security issues that are akin to “footguns” 🦶🔫.

DRYRUN .SECURITY

https://github.com/cktricky/Life-in-the-fAST-Lane

SLIDE - Intent

📡 Authorship frequency

If the author rarely commits or has never committed to this repo this indicates the author may be unfamiliar with the development standards of the code base.

Author is not an owner

Code changes to a portion of code in which the author is not an owner or on a team that owns this portion of the code.

DRYRUN .SECURITY

https://github.com/cktricky/Life-in-the-fAST-Lane

SLIDE - Detection

🤐 Secrets

Secret scanning technology has enabled the detection of sensitive keys when pushed in source code. This is a major indicator of risk in a code change.

👩‍🔬 Testing

Consuming (S|D)AST, dependency, docker image, etc. scanner tool output. Past bug bounty submissions as a historical reference to known dangerous anti-patterns. Similar concept with vulnerabilities Identified by internal/external security contributors.

DRYRUN .SECURITY

https://github.com/cktricky/Life-in-the-fAST-Lane

SLIDE - Environment

💪 Change protection

Does the application utilize branch protection features or other testing-relevant gating mechanisms to ensure evaluation of new code changes?

👩‍💼 Business Risk

Critical factors include the level of risk a service poses through its business function, the type of data it processes/stores/transmits, and its overall importance to the business.

DRYRUN .SECURITY

https://github.com/cktricky/Life-in-the-fAST-Lane

AI / LLMs

Personal recommendations

🔬🧪Start experimentation with Baseplate or something similar

📖 Read this article:

https://engineering.peerislands.io/extending-openai-gpt-4-using-langchain-and-pinecone-for-q-a-over-your-own-content-using-1f3e9dc10e91

📖 Read this book:

https://www.amazon.com/What-ChatGPT-Doing-Does-Work/dp/1579550819

Langchain

Modular Components

- LLMs
- Compressors
- Retrievers
- Prompts
- Chains
- … and way more

+

DRYRUN .SECURITY

https://github.com/cktricky/Life-in-the-fAST-Lane

My AI learnings…

Contextual Compression + Documents, Chains, LLMs, Prompts… they are all highly configurable and can be very accurate

Created the following:

- AI Chat bot assistant that can dig through your organization's secure code docs to answer questions
- Analysis agent that provides composition of the application (framework, language, security centric libraries, datastore, etc.)
- Analysis agent that looks for known (in your app's specific code base) security bug patterns

https://github.com/cktricky/Life-in-the-fAST-Lane

Conclusion

https://github.com/cktricky/Life-in-the-fAST-Lane

Things are changing… they must

💡 Too many technology choices to continue testing in isolation using a single data point to determine risk

💡 The power is in your hands to build your own highly customized tooling

💡 Meet developers where they live

💡 AI / LLM can be a for multiplier; is NOT a silver bullet

Contact Info

@cktricky

ken@dryrun.security

youtube.com/@AbsoluteAppSec

DRYRUN
.SECURITY