Static Analysis of Serverless Applications with a Semantic Code Search Engine

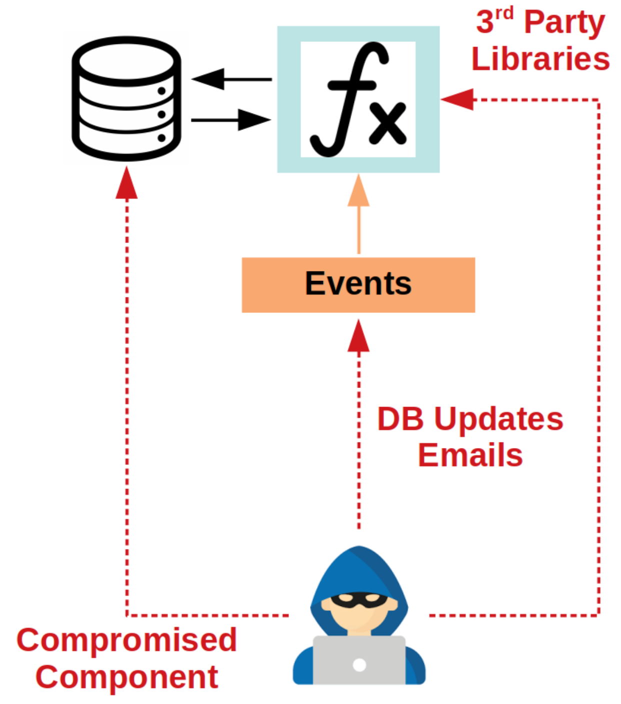
Giuseppe Raffa, Jorge Blasco, Daniel O'Keeffe
Royal Holloway University of London

Serverless Computing Paradigm

- Enterprises adopting the serverless paradigm can focus their attention on implementing the **business logic** of their software applications.
- Cloud providers manage the **underlying infrastructure** and charge their customers **only** for the resources actually used.

Motivation

- Serverless applications have a **large attack surface**, as code execution can be triggered by many events (e.g., database updates).
- Attackers can also rely on 3rd party libraries and compromised components.
- Traditional security solutions, e.g., WAFs, are unsuitable, and **security tools for serverless are still evolving**.
- Since serverless applications include both user-developed code and infrastructure configuration, their static analysis is **challenging**.



Our Work

- We have explored a novel approach to statically analysing serverless applications with a **general-purpose semantic code search engine**.

Infrastructure Code Analysis

- The infrastructure code specifies the infrastructure **elements** and their **configuration**.
- In our work, the **YAML syntax**, which is frequently used to implement infrastructure code, is converted into a **Python dictionary**.
- **23 infrastructure code queries** mostly focused on **Identity and Access Management** were developed.

$$\begin{tikzcd}\text{Infrastructure Code} \arrow[r] & \text{Python Dictionary} & \text{Analysis Query} \arrow[l]\end{tikzcd}$$

Application Code Analysis

- Only **injection vulnerabilities** and applications implemented in **Python** were considered.
- **7 application code queries** that rely on **local taint analysis** were implemented.

Automated Static Analysis

- The **public API** of the chosen code search engine can be used to run queries.
- The command-line tool **Serverless Inspector** was implemented to automate the process.
- **77 infrastructure code files** and **38 application code files** from the Serverless Framework repository were statically analysed.

$$\begin{tikzcd}\text{Command-line Interface} \arrow[r] & \text{Test Execution Logic} \arrow[r] & \text{Search Engine API Interface}\end{tikzcd}$$

Results

- **75.3%** of the tested **infrastructure** code files were flagged by **4 queries**. The highest number of samples was reported by the query dedicated to **functions triggered via HTTP events**.
- **13.2%** of the tested **application** code files were flagged by the query focused on **event data returned not sanitized**.
- The **average query execution time** per file was **less than 1 s** (except for outliers).

Query	No. of Samples
External plugin	28
Functions triggered via http	51
Multiple action nested	2
Unprotected environment information	7

Serverless Framework samples detected by the infrastructure code queries

Conclusion

- Our approach can be integrated into a **CI/CD pipeline** and features **short execution times**.
- Because of the high number of **infrastructure and application code variants**, writing queries that cover all cases is unattainable. Adopting **coding standards** can mitigate this problem.

Future Work

- **Validation** of the obtained results with code inspection and dedicated test environments.
- **Integration** of information extracted from infrastructure and application code to reduce false positives rate.

https://github.com/giusepperaffa/serverless-inspector-tool
giuseppe.raffa.2018@live.rhul.ac.uk