[AI for Cloud Ops] RTQA: Real-time Code Feedback for Data Scientists

Ayse Kivilcim Coskun [1] Gianluca Stringhini [1] Alan Liu [1] Mert Toslali[2] Anthony Byrne [2] Saad Ullah [2] Lesley Zhou [2]

[1]Principal Investigator [2]PhD Student

Strong Foundations

- Rapid experimentation/iteration essential to modern data science and software development.
  - Initial PoCs often written as scripts or notebooks.
- Business pressure means production products often built atop development/research code.
  - Manual code reviews sometimes skipped/ineffective.
- As features are added, small cracks in the foundation can grow, causing big problems.

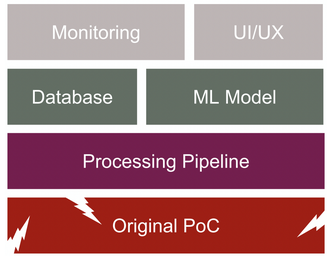

Figure 1. Strong foundations are needed for reliable production software.

Real-time Quality Assurance (RTQA)

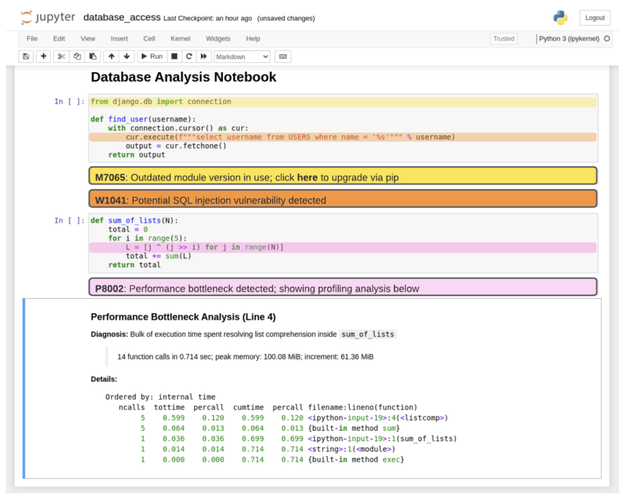

Figure 2. Mockup of the output for our RTQA JupyterLab plugin.

RTQA provides feedback to developers and scientists during development/experimentation. Real-time feedback could warn about outdated modules, security vulnerabilities, and performance bottlenecks, before the code even reaches the QA phase.

RTQA: JupyterLab Plugin

1. JuptyterLab is a popular multi-language IDE.
   - Often used with Python, R, and Julia.
2. Client/server architecture means extensions can be easily deployed to multiple users.
   - Ideal for maximizing availability of backend extensions/frameworks.

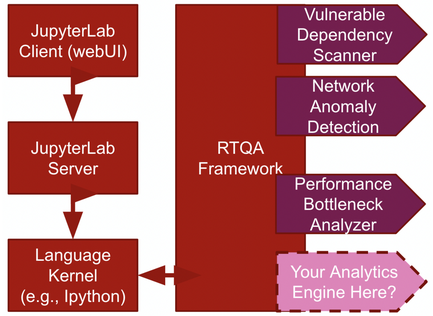

Figure 3. Envisioned architecture for the RTQA famework.

PoC Analytics Engine: Praxi

Praxi is a fully-automated, machine-learning-based method for discovering cloud software as it's installed.

- Learns to associate "filesystem footprints" to labeled events, e.g., "install apache2 v2.4.2".
- Achieves $> 96\%$ accuracy 14x faster than existing methods.

Praxi is **used with RTQA** to detect unsafe/vulnerable pre-built components, e.g., outdated Python modules, potentially-dangerous tools (e.g., popen()-ing package mgr.), and unsafe calls to foreign code (e.g., C-backed modules).

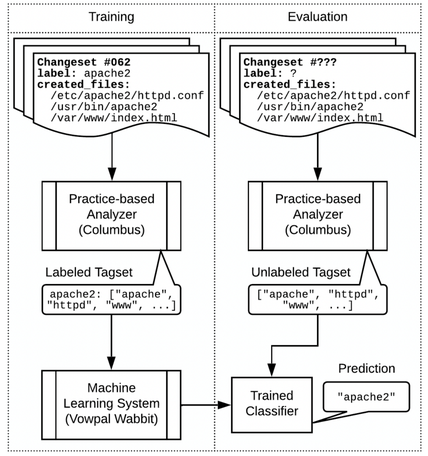

Figure 4. Praxi Architecture

Moving Forward

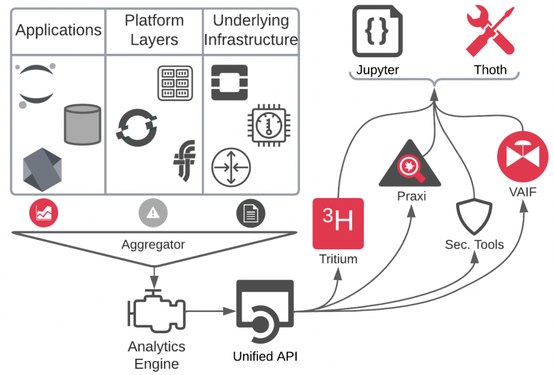

Figure 5. Overview of the RTQA Framework.

We are working on the following projects to be integrated in the RTQA framework:

1. Vulnerability Detection:
   - Our goal is to provide static code analysis (e.g., unsafe foreign code calls) at the line-of-code level to the developer in real-time.
   - Preventing vulnerabilities to pile up, becoming too complex to resolve, and reaching the deployment phase.
2. Code Performance Analysis:
   - Our goal is to help developers analyze their code's performance via profiling and tracing.
   - Automated tools to extract code execution flows, localize performance variations, pinpoint bugs and optimization points in code.
3. Privacy in Network Telemetry Data:
   - Our goal is to protect user's differential privacy inside network telemetry data.
   - *User privacy:* users does not want to release their private data to the third party. (e.g., GPS track location, IP address, etc.) Can we provide a systematic solution to encode telemetry data in a private way?
   - *Large storage:* network telemetry usually have large metrics and traces. Can we extract features that are useful to answer queries while compressing the data?
   - *Heavy computation demand:* while preserving the user privacy, we still need to be able to answer a lot of queries with high accuracy.
4. Machine Learning as a Service:
   - Praxi and other analytics engines require large, up-to-date ML models, so it is impractical to store copies on user's local machines.
   - Will be utilizing Red Hat's OpenShift Data Science platform and Kubeflow to build hosted ML pipeline. This can iteratively update models daily, e.g., with latest modules released on PyPI or vulnerabilities reported on SafetyDB.

And whatever else you'd like to add!

- github.com/operate-first/ai-for-cloud-ops
- PRs welcome!

SCAN ME