Developing Coverage Scripts

Introduction

Coverage Scripts generate payloads to exercise the workload, covering its intended use cases. They are ***not*** test scripts and don't test for the accuracy of results. Therefore, they don't need the preparation of data sets to compare results. They are ***not*** code coverage tests, either. They only need to ensure the executables, libraries, and scripts that the workload needs are invoked.

They are often quick and straightforward to write, and once developed and maintained, they help automate OSS vulnerability remediation once and forever. At RapidFort, the Coverage Script for our most complex workload took less than one day to write. Our simpler workloads took less than an hour.

In summary:

- Coverage scripts are NOT test scripts: No verification of the results.
- They generate payloads that exercise the workload, covering its different functionalities and configurations.
- They are quick to write.
- For a sample coverage script, visit: https://github.com/rapidfort/demos/tree/main/redis_coverage

A Software Attack Surface Management (SASM) Platform

RapidFort's SASM platform consists of 3 main product suites. The diagram below depicts these products and their benefits:

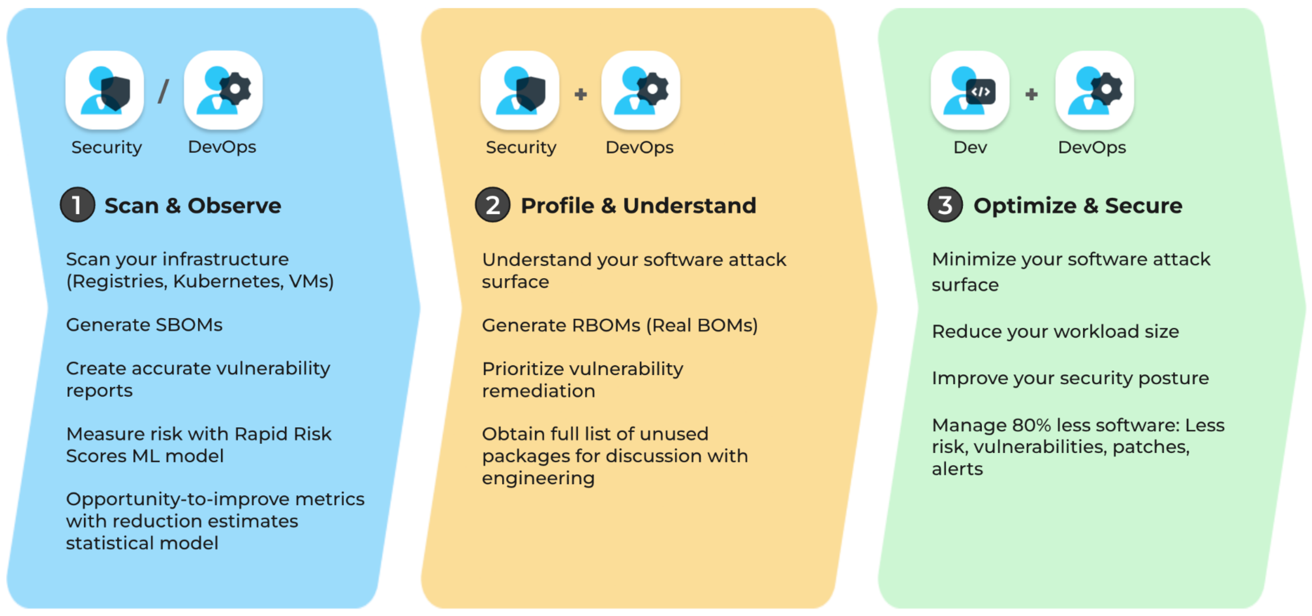
Security
DevOps
1 Scan & Observe
Scan your infrastructure (Registries, Kubernetes, VMs)
Generate SBOMs
Create accurate vulnerability reports
Measure risk with Rapid Risk Scores ML model
Opportunity-to-improve metrics with reduction estimates statistical model
Security
DevOps
2 Profile & Understand
Understand your software attack surface
Generate RBOMs (Real BOMs)
Prioritize vulnerability remediation
Obtain full list of unused packages for discussion with engineering
Dev
DevOps
3 Optimize & Secure
Minimize your software attack surface
Reduce your workload size
Improve your security posture
Manage 80% less software: Less risk, vulnerabilities, patches, alerts

Figure 1. RapidFort's SASM platform and product suites

Coverage scripts facilitate a reliable method for automating RapidFort's workflow and integrating it into your CI pipelines. The following diagram represents where coverage scripts fit in the SDLC:

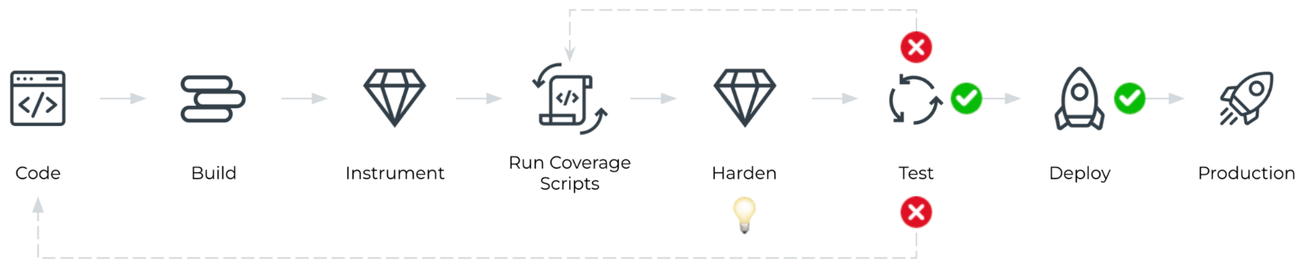

Figure 2. Optimization workflow using Coverage Scripts

💡 At this point, you gain complete visibility into which components are in use and which vulnerabilities are "hot," corresponding to software components in the execution path. Even if you don't deploy a hardened image, you can use this information to simplify interactions with your security teams for remediation requests.

You can then instruct RapidFort to use the workload profile to create an optimized container using the rfharden command. Automated, manual, regression and all other tests are run on the optimized/hardened container.

By creating and maintaining coverage scripts, you can integrate workload optimization and hardening into your CI pipeline, automatically removing “***zombie code***” from your workloads on every build. The benefits are as follows:

- Hardening bugs become part of the standard SDLC, are taken care of by the dev team, and are addressed by updating the coverage scripts.
- Regression tests and QA are performed once on the hardened image.

Recap and Points To Consider

- Developers of coverage scripts identify the executables, shared libraries, and scripts needed by the workload in different configurations.
- Coverage scripts ensure these components are invoked by the payloads and configurations that the script exercises.
- Ensure to invoke different configurations of the application, where applicable.
- Ensure that periodic and cron jobs are triggered, including screen sessions and shell scripts that might run periodically and not as a cron job.
- If your workload has scripts to update SSL certificates periodically, ensure they are invoked.
- Try some error payloads to invoke error modules.
- If you are unsure which data, configuration, and UI files (HTML, jpeg, etc.) are needed, use hardening profiles and the --keep-data-files option to the **rfharden** command.
- Refer to the **rfharden** manual page for further control over the optimization process: https://docs.rapidfort.com/using-rapidfort/cli/rfharden, or use “**rfharden –help**”
- Don’t overthink it! Let your test processes catch your initial coverage script errors and iterate quickly.