Starting Left rather than Shifting Left ?

Geneva Chapter - March 25 2021

@SecuringApps

Personal thoughts based on

- Past experience as
  - application security engineer in some tech companies
  - application security consultant in various industries
- Application security books, e.g.
  - Agile application security
  - Alice & Bob learn Application security
- Job descriptions for application security positions
  - Reading between the lines to understand what those organisations are (not) doing
  - Applying to some of them to learn more during the interviews about the real application security activities carried out

Assumptions

- Organization produces **high quality** software **efficiently**, e.g. has deployed successfully agile/DevOps processes

- Organization **truly cares about security**, i.e.
  - Compliance is not the highest priority goal (but rather a by-product of a successful security assurance program)
  - **Accepts to spend**
    - **Money**, i.e. to hire security engineers and buy tooling
    - **And time**, i.e. other employees have time allocated for security matters

Statement

- **Shifting Left tries to fix more efficiently the symptoms** of an insecure development pipeline
- **Starting Left** aims to **make development pipeline less insecure**
- **A bottom-up approach is more likely to make security an emergent property** (rather than a traditional top down approach)
  - Similar to quality assurance and the proven efficiency of unit testing or test driven development

Constraints

- Developers
  - Based on the assumptions, their yield is already maximized
  - Any extra security work will lower this yield (thus not every developer will care a lot)
  - And even more if it requires context switching

- Application security engineers
  - Limited number available on the market
  - Hardly proficient in the various languages/frameworks used
  - Very often overwhelmed by
    - tracking new projects
    - very limited time available for each of those new projects
      (except for SaaS companies with a main product that can get a 100% focus)
    - purchasing and plumbing security tools
    - triaging results from those tools
    - various security initiatives

- NB: Shifting Left is transferring this application security burden to developers

Proposal

In the short term: maximize efficiency rather than minimizing risk.
This should provide a better risk reduction in the mid term.

Action plan:

1. **Focus first on new features**
   Handle legacy later with existing QA processes

2. **Only teach the secure coding basics that are easy to unit test**

3. Do much more only with developers that have been committed to 1. and 2.

NB: The scope is bootstrapping a useful application security program. After successful take-off, you could for example consider the most efficient activities from OWASP SAMM

Proposal details 1.

- Propose a Rapid Risk Assessment to be filled-in by project team
  - e.g. from Slack https://github.com/slackhq/goSDL
- Avoid frightening project teams about this process
  - Otherwise it will be bypassed
  - Simple self-service form
  - Only require workshop with applications security engineer for Critical or High risks
- Output of this workshop
  - Clear security requirements
    - Easy to implement: in project scope
    - Otherwise: documented in the backlog
  - Identify developers interested (or not) by implementing those security requirements
- NB: Scaling this process (e.g. via an existing list of security requirements or reference projects) is not covered by this bootstrapping action plan

Proposal details 2.

- 2 hour hands-on training session for each of those 3 topics
  - Input validation
  - Access control
  - Authentication
- Training session content
  - Show business impact by exploiting real-world vulnerabilities (ideally from codebase of the organization)
  - Target only the languages/frameworks used by those developers
  - Gives very specific guidance
    - On how to implement the protection
    - On how to unit test this protection
    - Documented via a “secure-coding” reference project in the code repository
  - Identify developers asking questions: they are likely to be interested by those security topics

Proposal details 3.

- Make a short list of developers interested by security topics from
  - RRA workshops
  - Secure coding trainings
  - Looking at those re-using the “secure-coding” project in their code and their unit tests
    - e.g. by scanning code repo
- Brainstorm with them how to
  - change this “secure-coding” project into helper libraries
  - enforce and monitor those helper libraries are used
  - make sure one of those developers is always included in any project team
- Organize with them advanced secure coding sessions, e.g.
  - SSRF
  - XXE
  - Hands-on crypto
  - Any other vulnerability making the news

Measuring success

- **Low level indicators**
  - Their goal is to **monitor a global trend**
    - Initial success expected, i.e. trend improving in the first few months
    - **What matters is sustainable improvement**, i.e. avoiding a decrease later on
  - Examples
    - Ratio RRA forms vs all new features (1.)
    - Ratio security requirements implemented vs backlog (1.)
    - Ratio developers using “secure-coding” project in code (2.)
    - Ratio security unit tests vs other unit tests (2.)
    - Number of identified developer (3.)
    - Ratio projects using secure helper libraries (3.)
- **High level indicator: Trust**
  - Ratio of developers considering application security engineers useful

Conclusion

- You don’t need to plug yet another security tool
- Proposal
  - Step 1: Rapid Risk Assessment + identify security enthusiasts
  - Step 2: Very specific trainings + identify security enthusiasts
  - Step 3: Do more with those security enthusiasts
- Measuring success
  - Monitor global trend by combining low level indicators from steps 1, 2 and 3
  - Trust: is collaboration with appsec team useful to developers ?