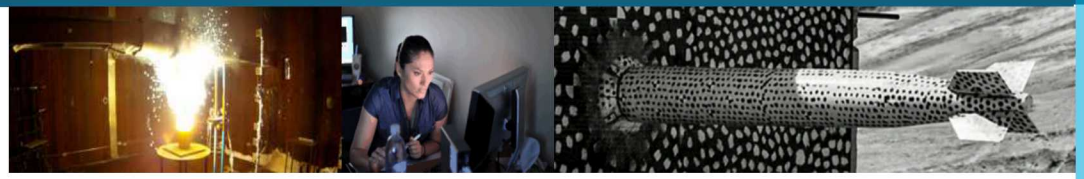
SAND2019-5748C

Improving the Software Delivery Life Cycle Through Automation and Analytics



PRESENTED BY

Vince Urias, Jeff Hsia, Ang Rivas

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Overview

Motivation

"75% of security breaches happen at the application"

"Over 70 percent of security vulnerabilities exist at the application layer, not the network layer"

"If only 50 percent of software vulnerabilities were removed prior to production ... costs would be reduced by 75 percent"

92% of reported vulnerabilities are in applications not in networks"

"The cost of fixing a bug in the field is $30,000 vs. $5,000 during coding"

Mean Time to Fix

A 2017 industry study from White Hat Security revealed that the "Mean Time to Fix" for web application flaws categorized as "serious" averaged **193 days** across all industries.

In the same study, for one industry (Education) the figure jumped to **342 days** of exposure.

In a similar study from Veracode, 70% of 22,430 applications submitted to their testing platform in 2015 contained exploitable security vulnerabilities.

Think about it…

How would you report to your management that a "serious" and likely exploitable vulnerability was present on your primary public facing web site or a 3rd party hosted portal for more than six months?

What compensating control or controls do you think you could explain to placate management that a serious vulnerability could not be exploited?

Verizon's 2018 Breach Report says 90% of attacks last year were perpetrated by outsiders and 52% used some form of hacking. How does this help you explain application risk?

So what about training?

Why don't we have a higher investment in software developers training ?

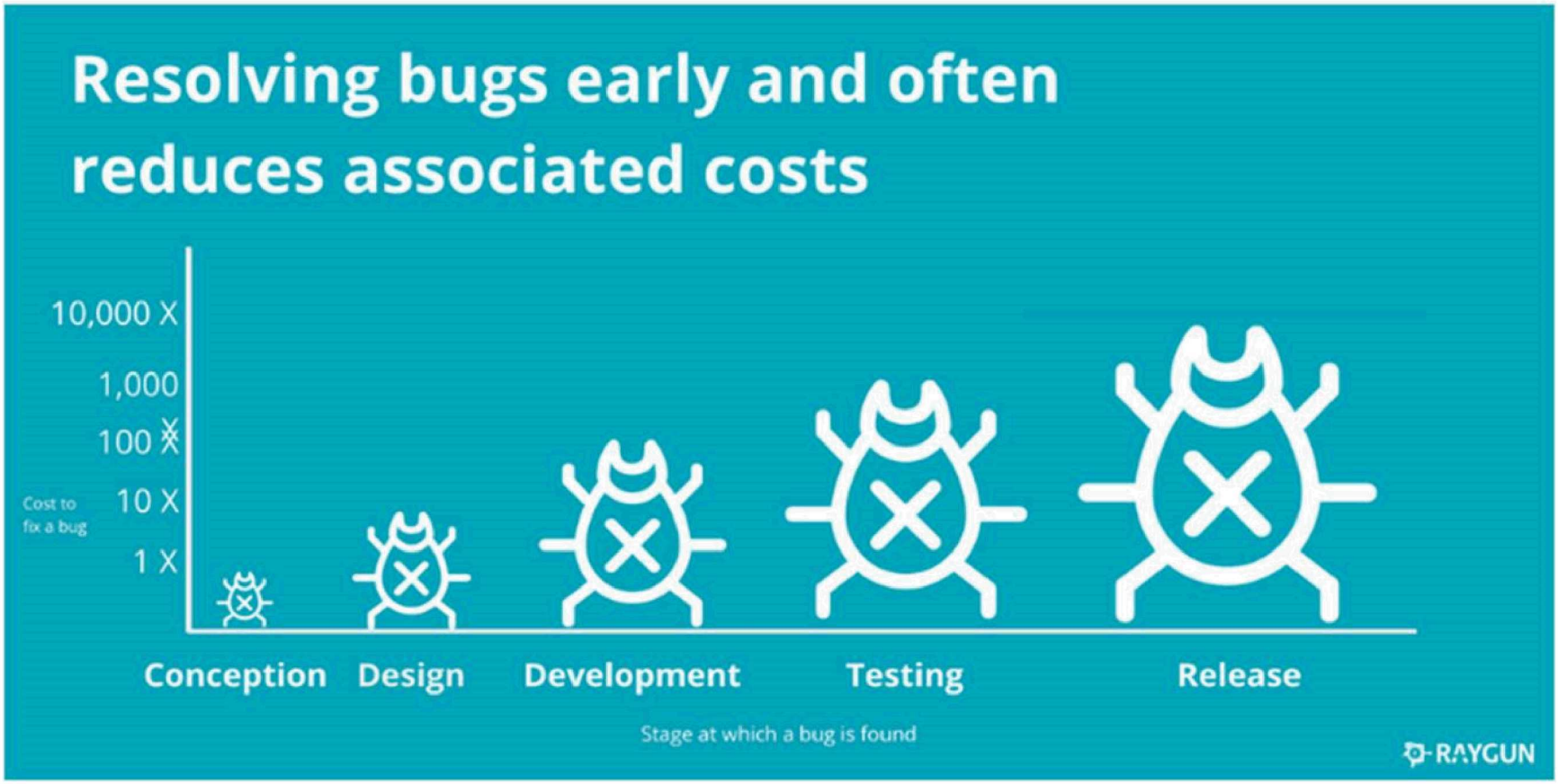
What should they focus on ?

Is the training effective?

Is it another sunk cost? Cost of doing business?

Is it worth the resources (cost of the developers time, cost to the cooperation, cost to the timeline of the programs)?

Early is better



Let's frame the problem

Software complexity is increasing

- And so are bugs

Software security education is not uniformly or even frequently taught at universities

- Concepts like buffer overflows, xxs , … are not well understood
- Their impacts to the software development lifecycle are also not well understand
- Early to mid career developers have a ranging set of background

Security for software development education is seen as a tax and does not provide a measurable way of ensuring progress

As the old saying goes:
It's harder to read code than to write it.

Can we measure any of it?

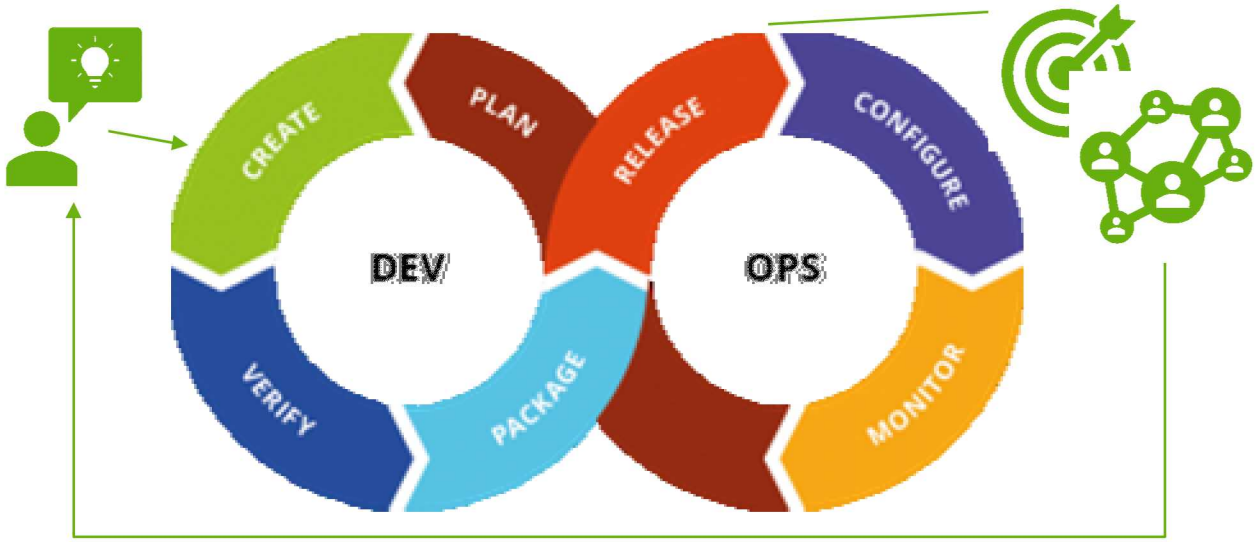
- There are numerous building blocks available:
  - Secure developing practices (though education)
  - Static Source Code Analysis
  - A common lexicon (CWEs) to describe vulnerably in source
  - Automation (through the use of REST APIs
- Can we create a POC off of these ?

Can we make software development education measurable and targeted through the use of several primitives?

If training is an investment, how do we measure the returns?

…we should be spending money on security training for developers. These are people who can be taught expertise in a fast-changing environment, and this is a situation where raising the average behavior increases the security of the overall system."

Bruce Schneier - Writer, cryptographer



Methodology

We looked at the building blocks and looked for tools

- Some of our requirements , not bound to one tool
- Normalize data inputs and outputs
- Create workflows that are automated and don't impact the users or their development life cycle
- Use tools they already have access to

Develop a process and a reference architecture

- gitlab
- Static Source Code Analysis (checkmarx CxSAST)
- Software Education (checkmarx)
- Splunk (Data normalization)

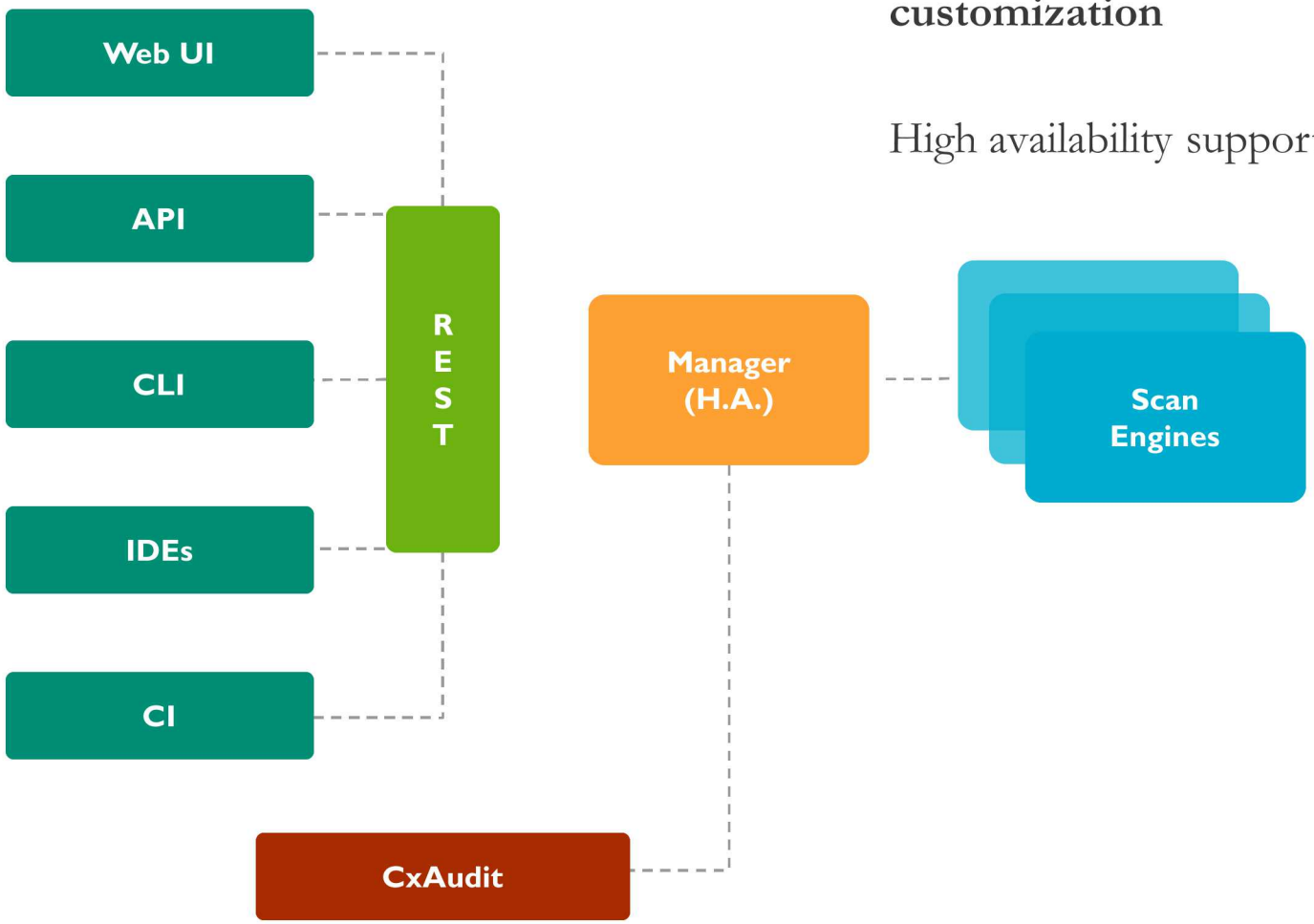
Workflow Overview

Checkmarx SAST Architecture

Highly scalable architecture

REST APIs for automation and customization

High availability support



Hands-On, Gamified AppSec Training for Enterprise Developers

The hands-on interactive training platform built by developers for developers.

Application Security Training for Major Programming Languages

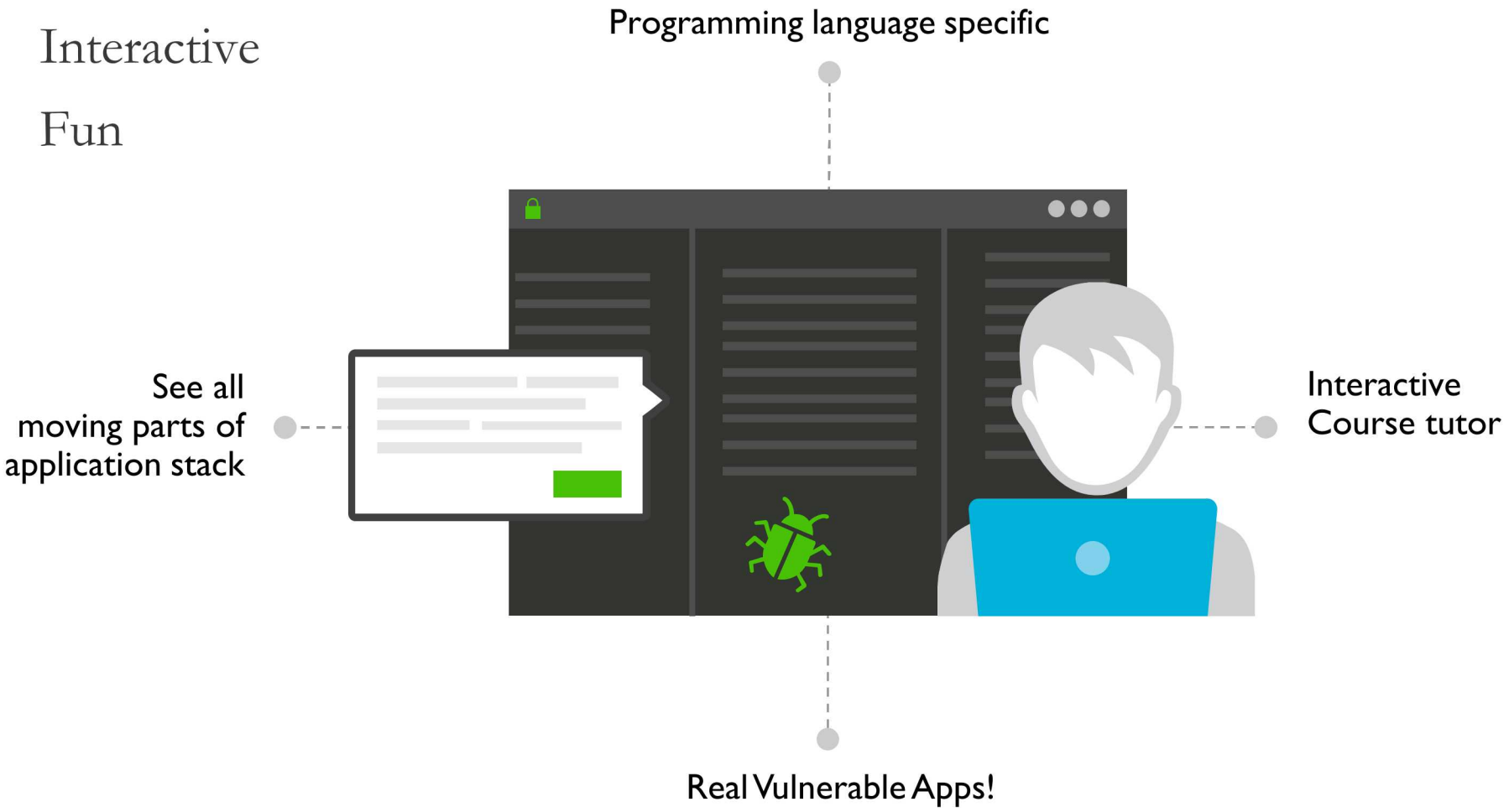
Codebashing in a Nutshell

Game-like

Hands-on

Interactive

Fun



What are some of the analytics we're tracking?

Discussion and walkthrough

Demo