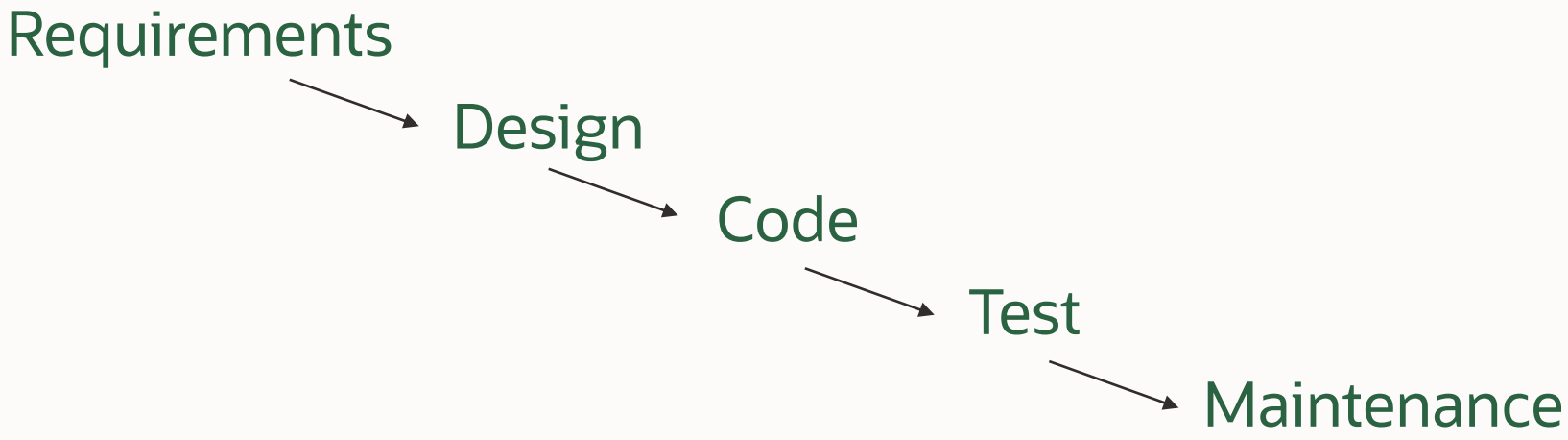
ORACLE

Towards Intelligent Application Security

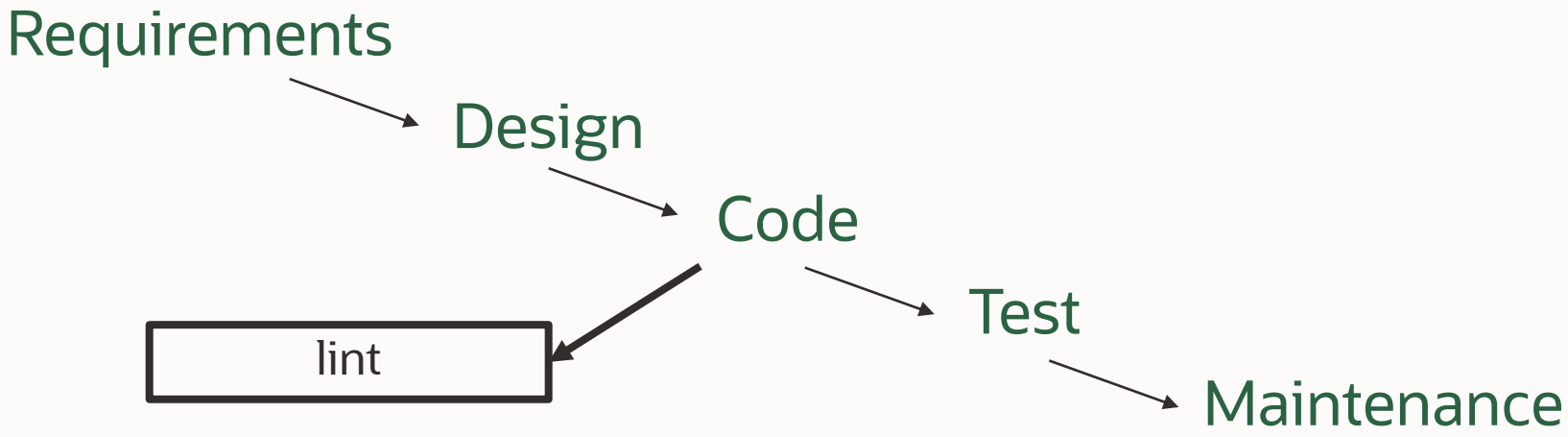
Cristina Cifuentes
Oracle Labs, Australia
June 2021

Yesteryear – Waterfall Model



Yesteryear – A Debugging Tool

Introduced in 1978 by Stephen Johnson, Bell Labs



Notice that bugs are different to vulnerabilities.
Vulnerabilities are security bugs

1970s-1990s
Security was an after thought not tied to the waterfall model, it was a non-functional requirement at most

Yesteryear – Application Security Testing in the 2000s

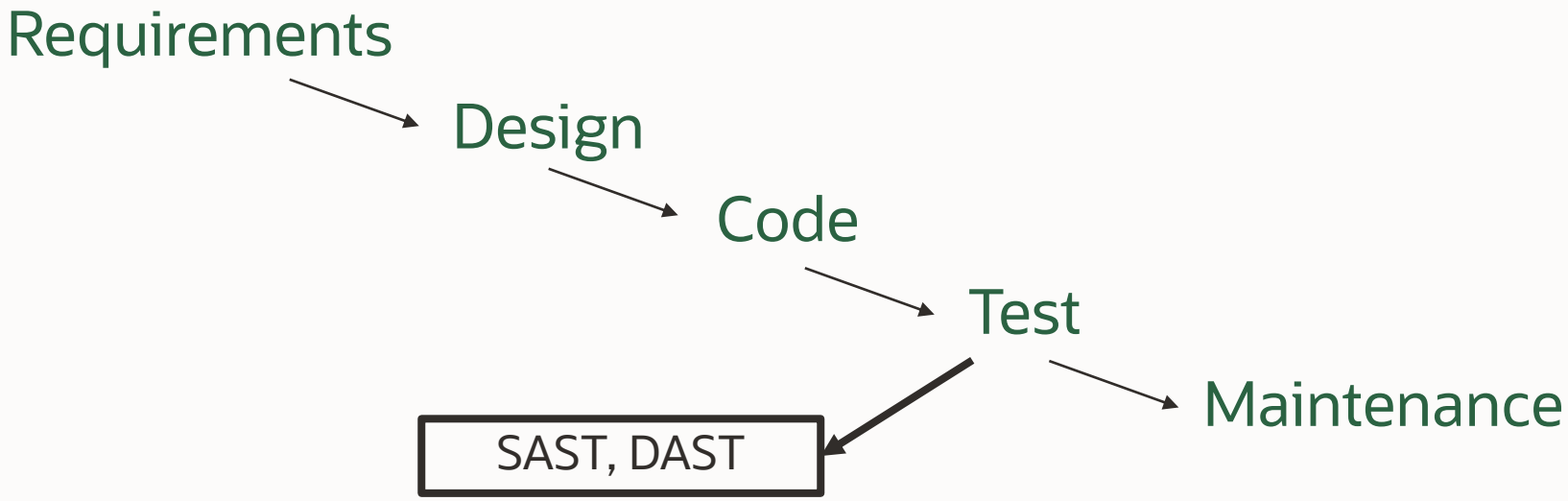
Static application security testing (SAST) is a set of technologies designed to **analyze** application **source code, byte code and binaries** for coding and design conditions that are indicative of **security vulnerabilities**. SAST solutions analyze an application from the "inside out" **in a nonrunning state**.

Dynamic application security testing (DAST) technologies are designed to detect conditions indicative of a **security vulnerability in an application in its running state**. Most DAST solutions test only the exposed HTTP and HTML interfaces of Web-enables applications. […]

https://www.gartner.com/en/information-technology/glossary/

Yesteryear – Application Security Testing

Application Security Testing in the Late 2000s

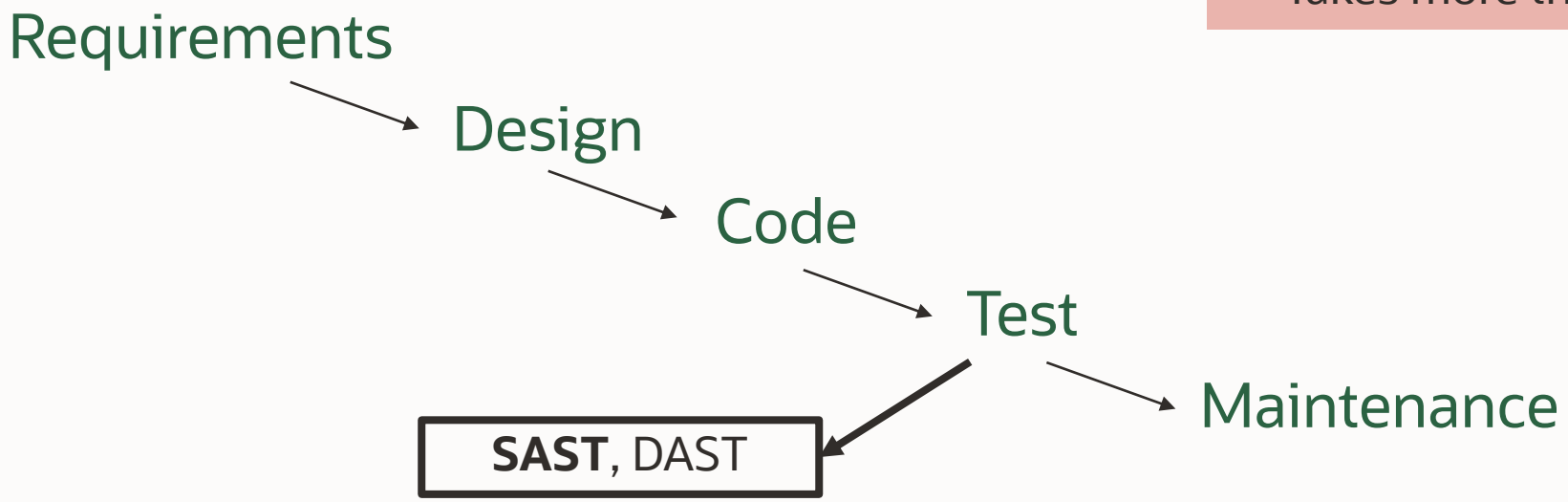


Yesteryear – Application Security Testing

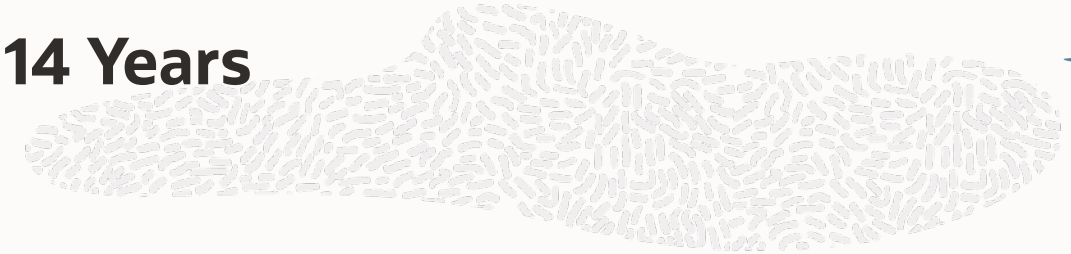
Static Application Security Testing in the Late 2000s

Challenges with SAST:

- Too many false positives
- Takes more than a nightly to run



During the Past 14 Years



Build ⟵ Test

- Efficient analysis of full codebase
  - Used to be nightly runs
  - Now part of Continuous Integration

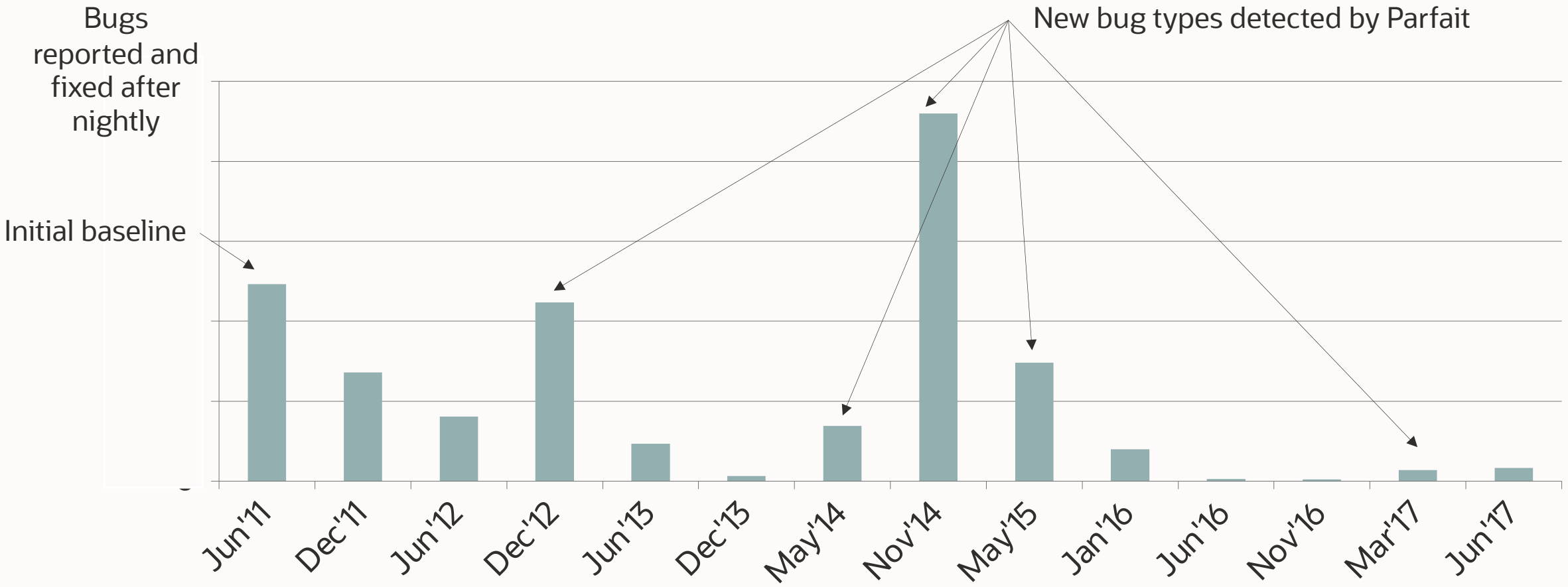
Parfait – Scalable, Deep Static Code Analysis

Codebase	Non Commented Lines of Code	Number of Bugs and Vulnerability Types	Analysis runtime	Runtime in KLOC/min
Oracle Linux Kernel 5	16,586,325 C	34	19 m 20s	858 KLOC/min
Cloud service	1,216,168 Java	5	7 m 2 s	173 KLOC/min

Parfait – Designing a scalable bug checker, Static Program Analysis workshop, June 2008.
Boosting the performance of flow-sensitive points-to analysis using value flow, FSE 2011.

Parfait – Precise, Deep Static Code Analysis

Bugs fixed by developers once **baseline** had been established



Path-sensitive data flow analysis simplified, ICFEM 2013.
Internal deployment of the Parfait static code analysis tool at Oracle, Invited talk, APLAS 2013.
The Parfait static code analysis framework – Lessons learnt. DECAF workshop 2016.

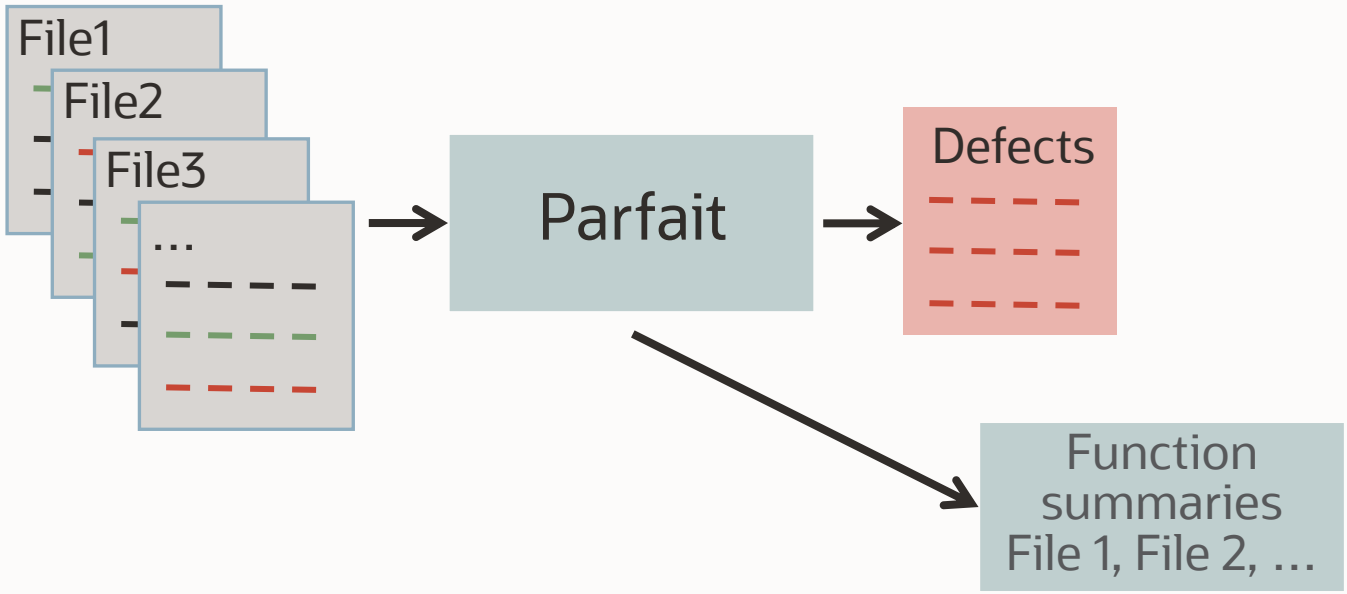
During the Past 14 Years

- Efficient analysis of full codebase
  - Used to be nightly runs
  - Now part of Continuous Integration

- Efficient analysis of changeset
  - Prevent bugs from being introduced into the codebase
  - Can be hooked into the commit, push, pull request or merge request

Analysis of Full Codebase vs Analysis of Commit/Push/Pull/Merge Request (Incremental Analysis)

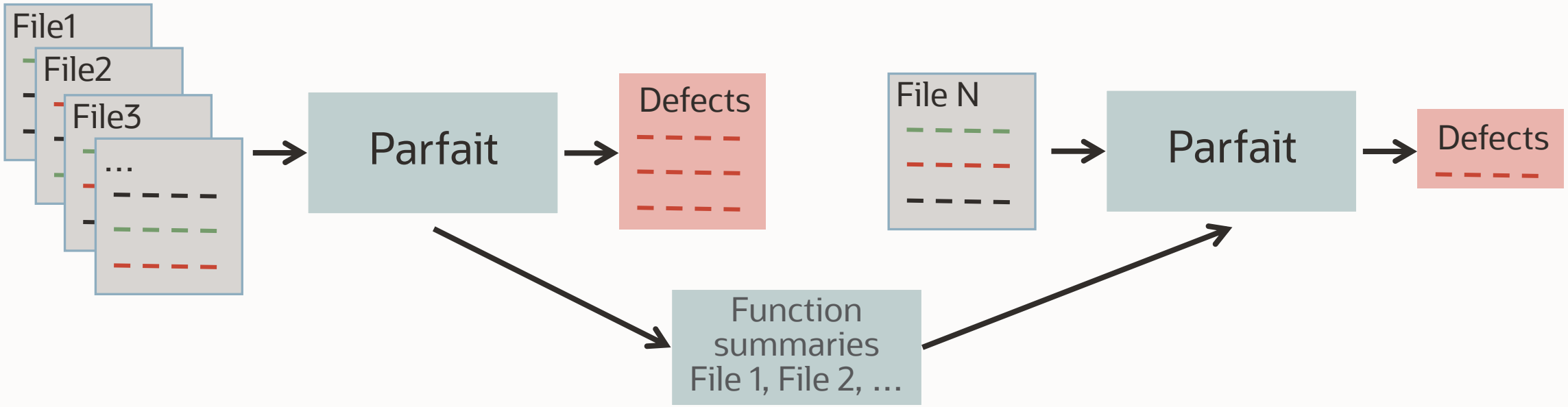
Analysis of full codebase



Analysis of Full Codebase vs Analysis of Commit/Push/Pull/Merge Request (Incremental Analysis)

Analysis of full codebase

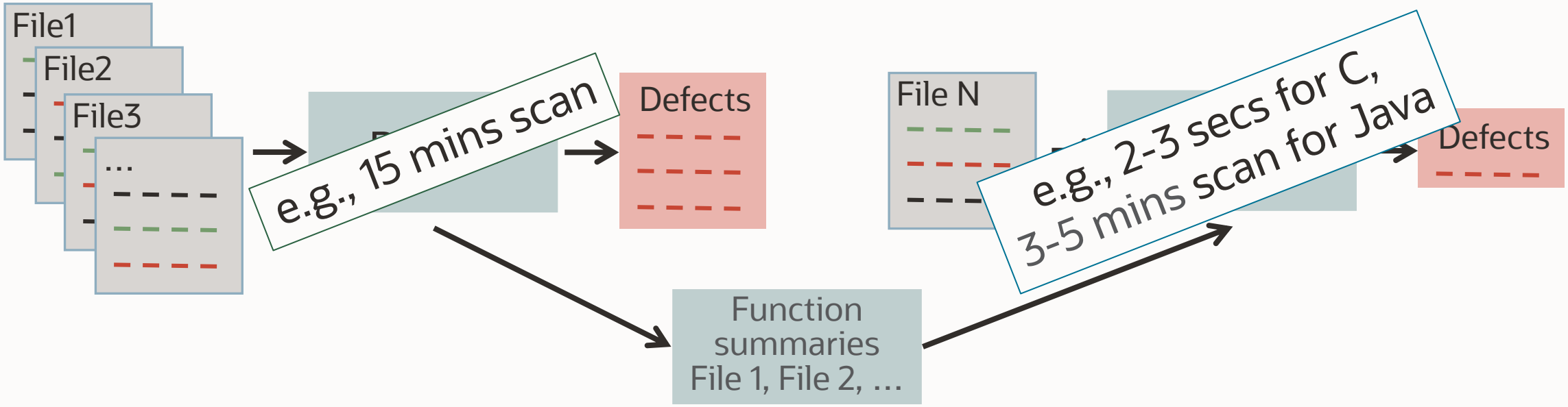
Analysis of changeset



Analysis of Full Codebase vs Analysis of Commit/Push/Pull/Merge Request (Incremental Analysis)

Analysis of full codebase

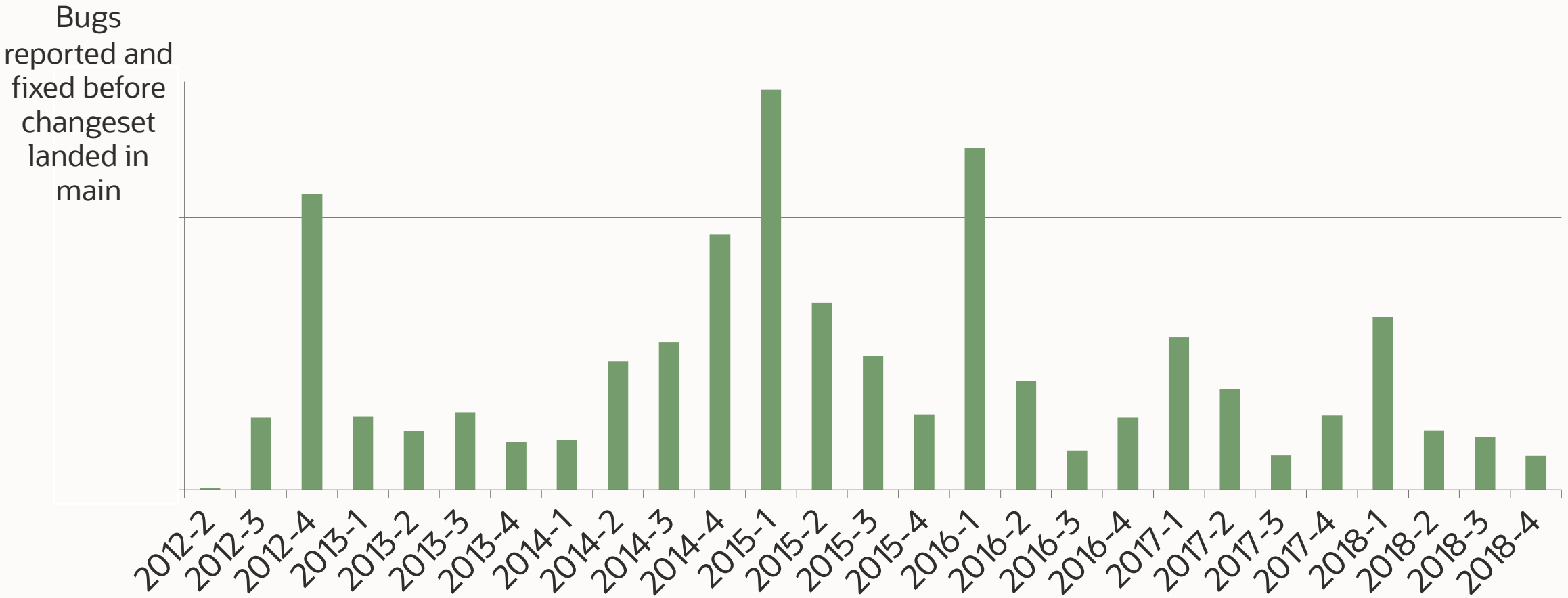
Analysis of changeset



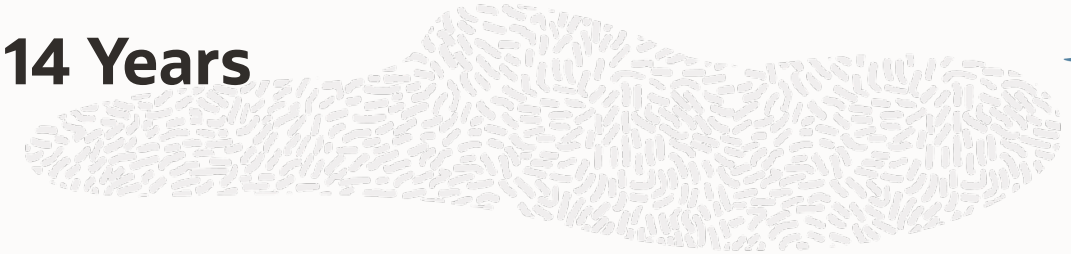
Transitioning Parfait into a development tool, IEEE Security & Privacy, 2012.
In writing 2021: Modular taint analysis for Java.

Bugs Prevented from Being Introduced into the Codebase

Changeset analysis prevents **80%** of new bugs (compared to baseline)



During the Past 14 Years



Code ← Build ← Test

- Efficient analysis of full codebase
  - Used to be nightly runs
  - Now part of Continuous Integration
- Efficient analysis of changeset
  - Prevent bugs from being introduced into the codebase
  - Can be hooked into the commit, push, pull request or merge request

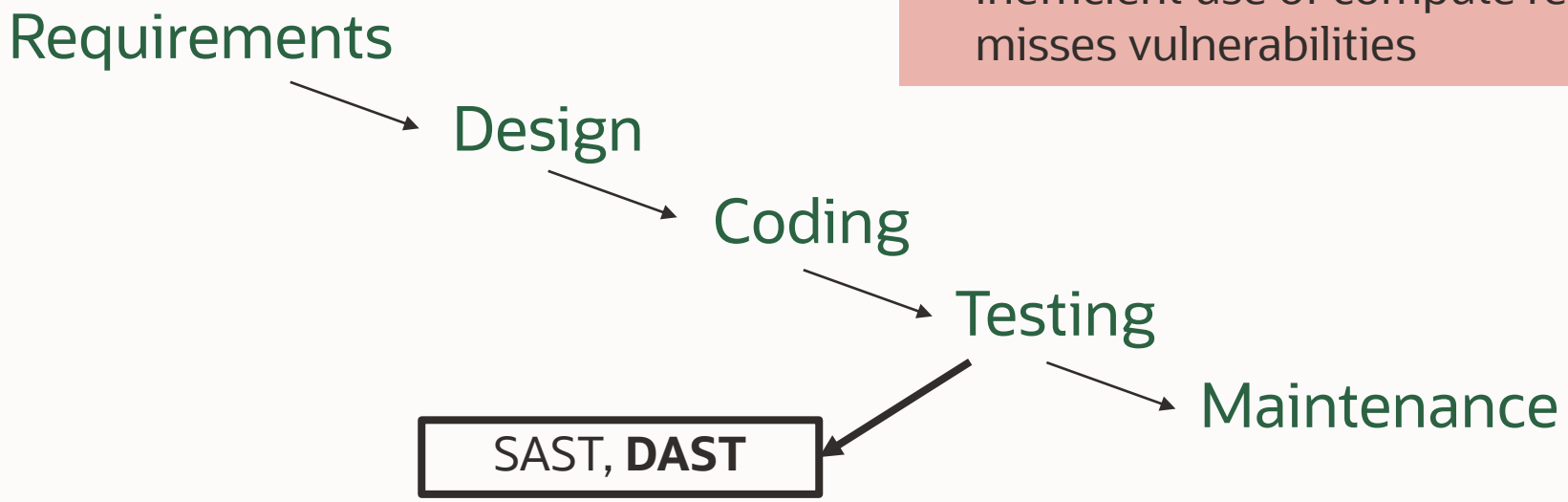
Innovations for SAST via Parfait:
- Precise results
- Scalable, can integrate early in the development cycle

Yesteryear – Application Security Testing in the 2000s

Dynamic Web Application Security Testing in 2015

Challenges with DAST for Web apps:

- Automation lacking for effective use of the tool
- Inefficient use of compute resources; misses vulnerabilities

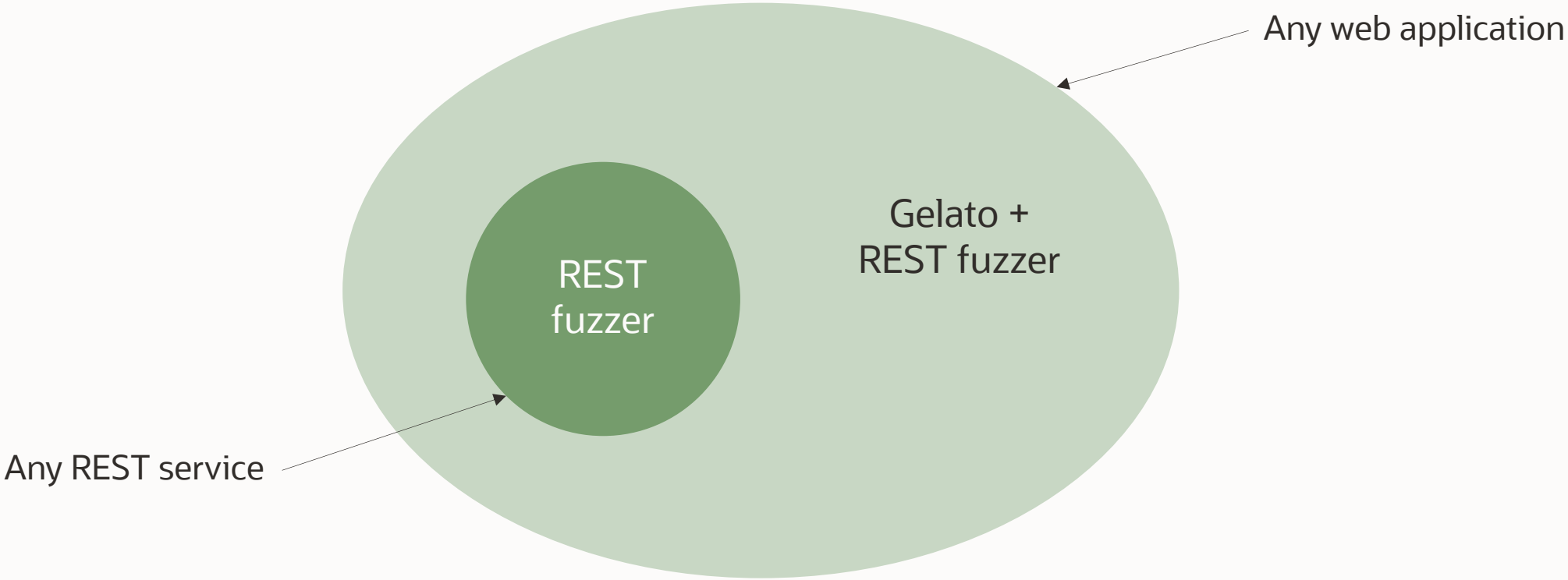


During the Past 4 Years

Test

- Automatic, high coverage detection of endpoints
  - Automated – no input from pentester or developer needed
  - Generation of Swagger/Open API spec to drive inputs into existing blackbox REST fuzzer

Enabling Fuzzing of Any Web Application Automatically



Gelato: Feedback-driven and guided security analysis of client-side applications, arXiv 2020.

During the Past 4 Years

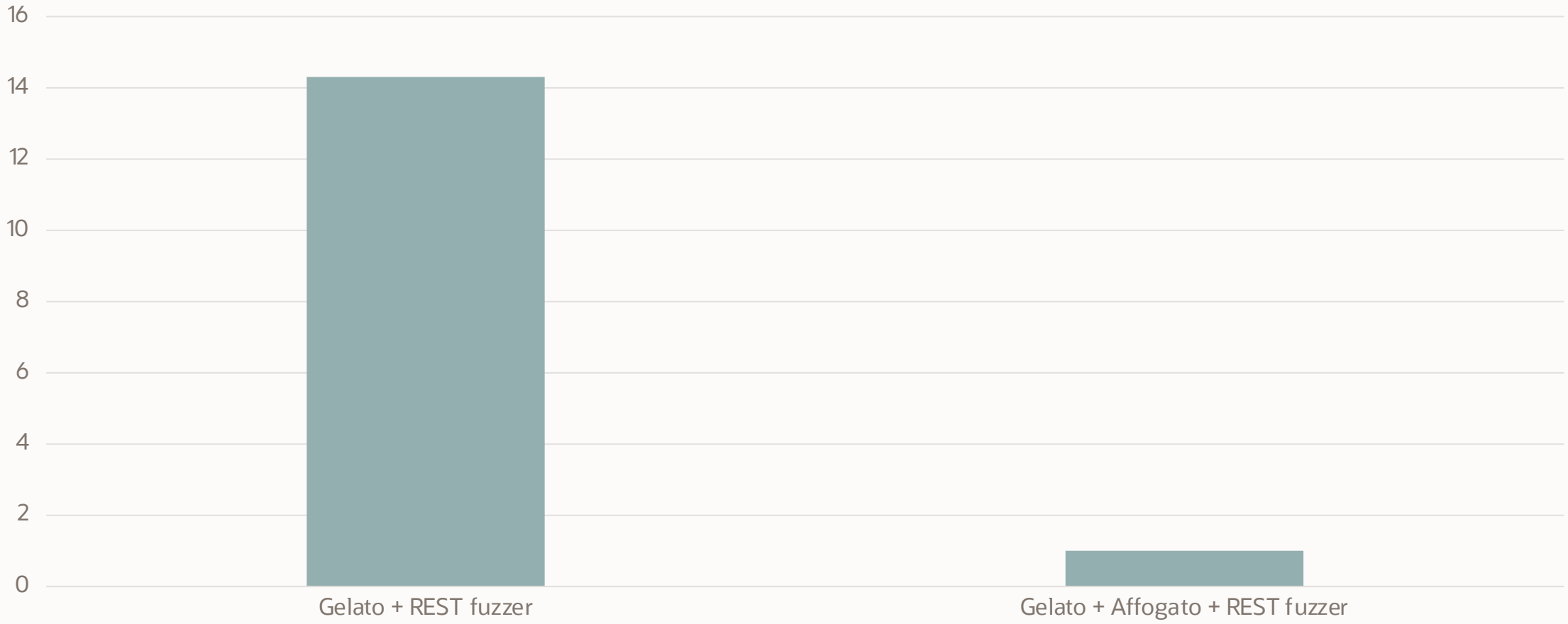
Test

- Automatic, high coverage detection of endpoints
  - Automated – no input from pentester or developer needed
  - Generation of Swagger/Open API spec to drive inputs into existing blackbox REST fuzzer

- Integrate greybox solution
  - Greybox approach provides context to drive efficiency into existing blackbox solutions

Efficient Use of Compute Resources

Comparative runtime to find 0-days in web applications



AFFOGATO: runtime detection of injection attacks for Node.js, ISSTA/ECOOP workshops (SOAP), 2018.
Under review: BackREST: a model-based feedback-driven greybox fuzzer for web applications, 2021.

During the Past 4 Years

Test

Innovations for DAST via Gelato and Affogato
- Automated attack surface detection
- Efficient use of compute resources finds more 0-days

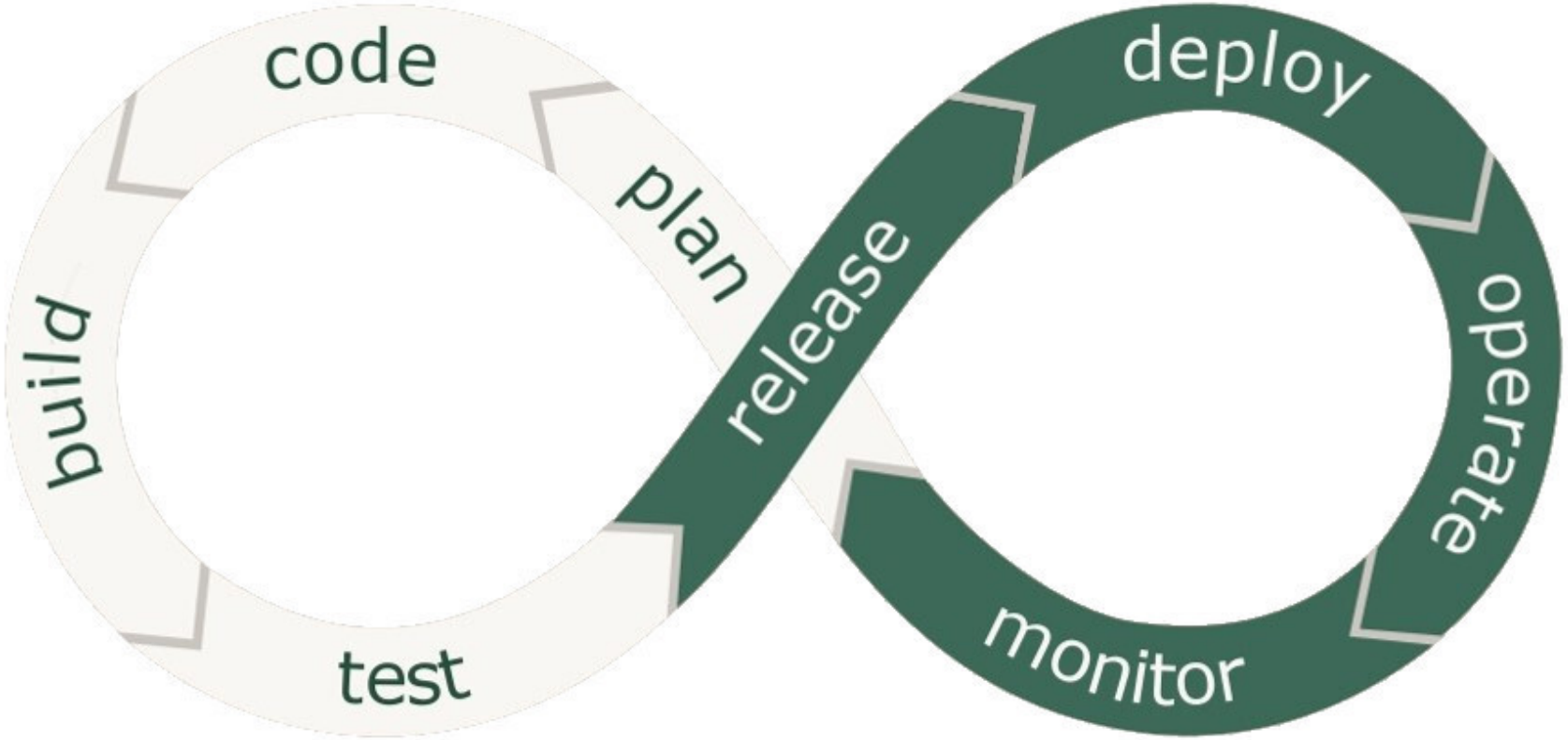
- Automatic, high coverage detection of endpoints
  - Automated – no input from pentester or developer
  - Generation of Swagger spec to drive inputs into existing blackbox REST fuzzer

- Integrate greybox solution
  - Greybox approach provides context to drive efficiency into existing blackbox solutions

What Else Has Changed Over the Past 10 Years?

The waterfall model of yesteryear is dying,
long live DevOps

DevOps



Today – DevSecOps – Integrating Security Into DevOps

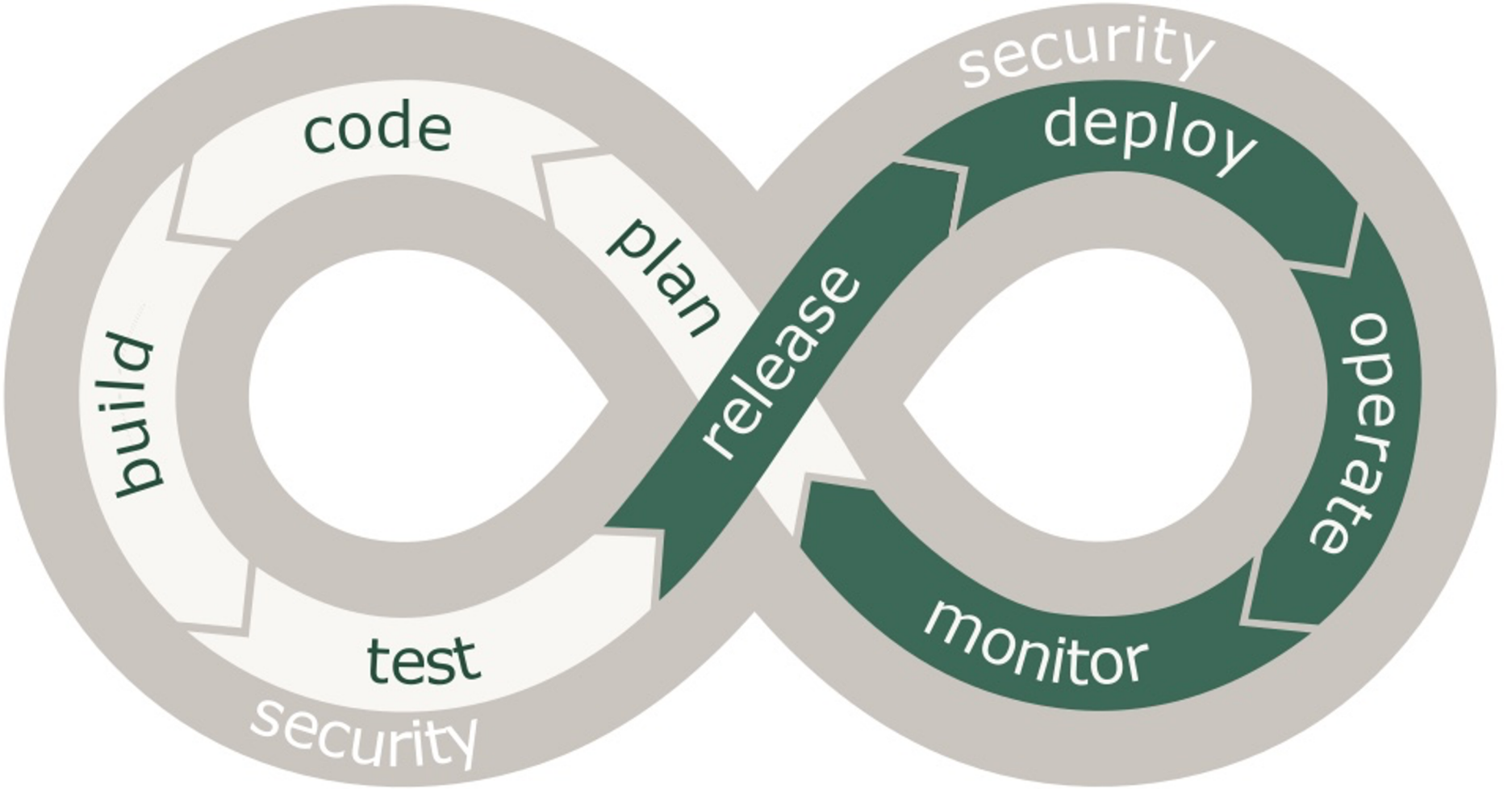


Diagram with thanks to Patrick McLaughlin

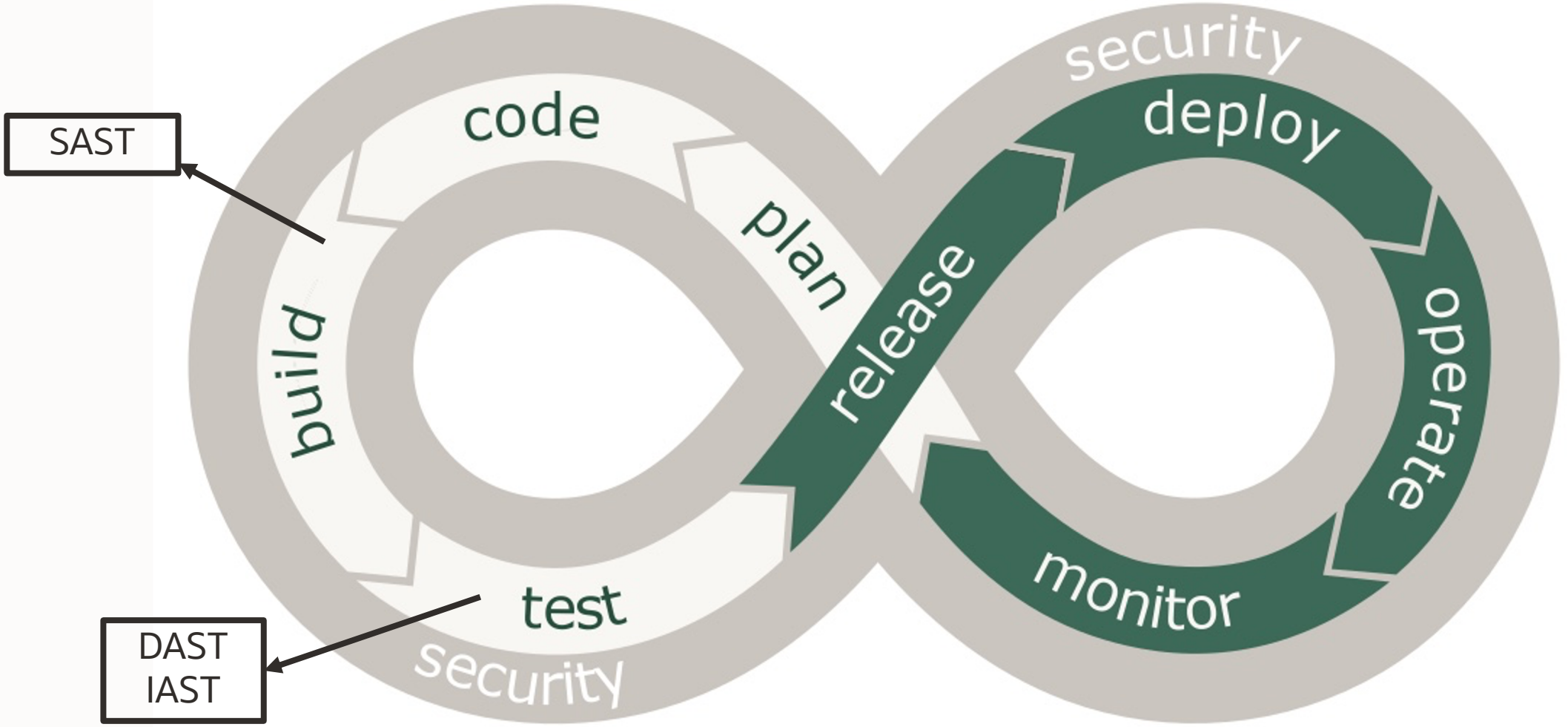
"DevSecOps is an organizational software engineering culture and practice that aims at **unifying software development** (Dev), **security** (Sec) and **operations** (Ops). The main characteristic of DevSecOps is **to automate, monitor, and apply security at all phases** of the software lifecycle: plan, develop, build, test, release, deliver, deploy, operate, and monitor. In DevSecOps, **testing and security are shifted to the left** through automated unit, functional, integration, and security testing - this is a key DevSecOps differentiator since security and functional capabilities are tested and built simultaneously."

DoD Enterprise DevSecOps Reference Design

12 August 2019

https://dodcio.defense.gov/Portals/0/Documents/DoD%20Enterprise%20DevSecOps%20Reference%20Design%20v1.0_Public%20Release.pdf?ver=2019-09-26-115824-583

Application Security in DoD's DevSecOps Reference Design

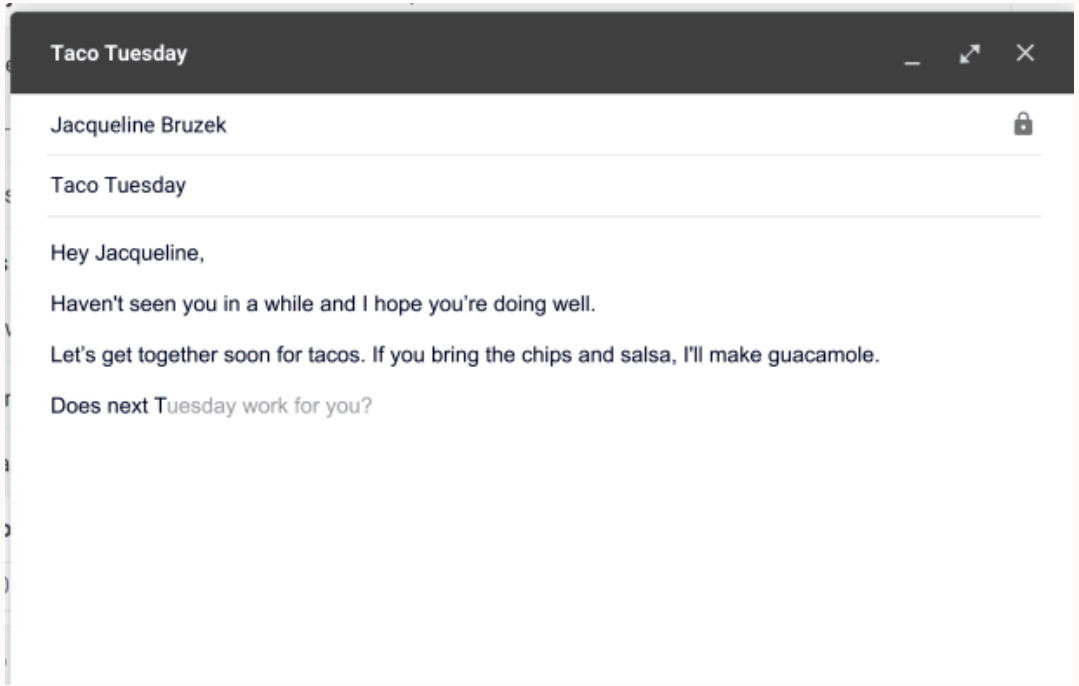


https://dodcio.defense.gov/Portals/0/Documents/DoD%20Enterprise%20DevSecOps%20Reference%20Design%20v1.0_Public%20Release.pdf?ver=2019-09-26-115824-583

Learning-based techniques have ripen

We Are Living With Intelligent Applications

Gmail's Smart Compose



iOS's Predictive Text

https://www.theverge.com/2018/5/10/17340224/google-gmail-how-to-use-smart-compose-io-2018
https://support.apple.com/en-au/HT207525

Microsoft's Visual Studio IntelliCode

Code completion suggestions based on 1,000s of open source projects

https://visualstudio.microsoft.com/services/intellicode/

Facebook's Aroma

Code-to-code search and recommendation tool

```
Bitmap bitmap = BitmapFactory.decodeStream(input);
```

```
final BitmapFactory.Options options = new BitmapFactory.Options();
options.inSampleSize = 2;
// ...
Bitmap bmp = BitmapFactory.decodeStream(is, null, options);
```

```
try {
  InputStream is = am.open(fileName);
  image = BitmapFactory.decodeStream(is);
  is.close();
} catch (IOException e) {
  // ...
}
```

https://ai.facebook.com/blog/aroma-ml-for-code-recommendation/

Amazon's CodeGuru Reviewer

Identifies critical issues and hard-to-find performance bugs and suggests ways to fix them

HelloWorldFunction/src/main/java/helloworld/App.java

```
56 +        item_values.put("location", new AttributeValue(ipv4));
57 +        item_values.put("date", new AttributeValue(now));
58 +
59 +        final AmazonDynamoDB ddb = AmazonDynamoDBClientBuilder.defaultClient();
```

danilop 3 minutes ago Author Owner

Recommendation generated by Amazon CodeGuru Reviewer. Leave feedback on this recommendation by replying to the comment or by reacting to the comment using emoji.

This code is written so that the client cannot be reused across invocations of the Lambda function.
To improve the performance of the Lambda function, consider using static initialization/constructor, global/static variables and singletons. It allows to keep alive and reuse HTTP connections that were established during a previous invocation.
Learn more about best practices for working with AWS Lambda functions.

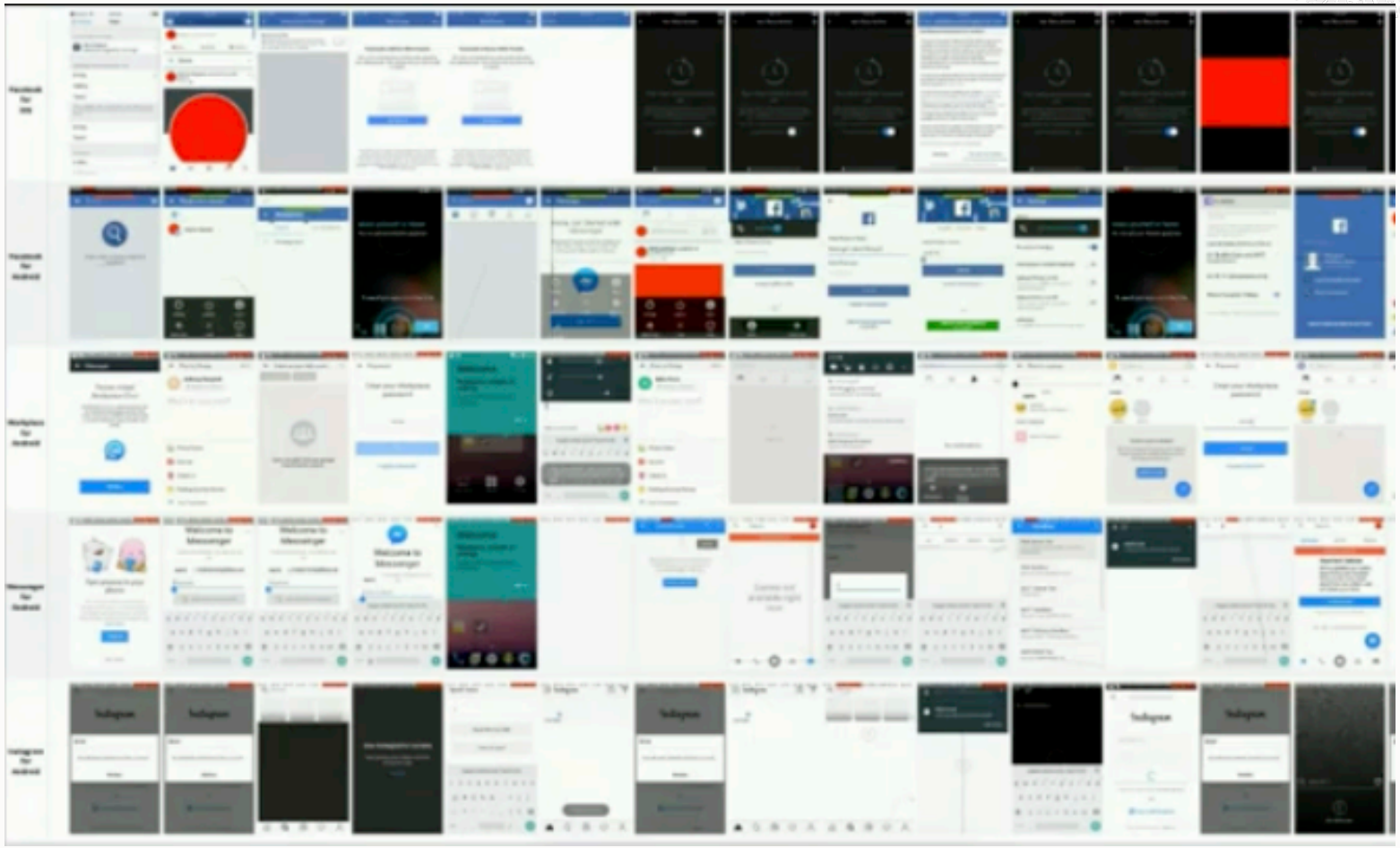
Reply...

Resolve conversation

https://aws.amazon.com/codeguru/

Facebook's Sapienz

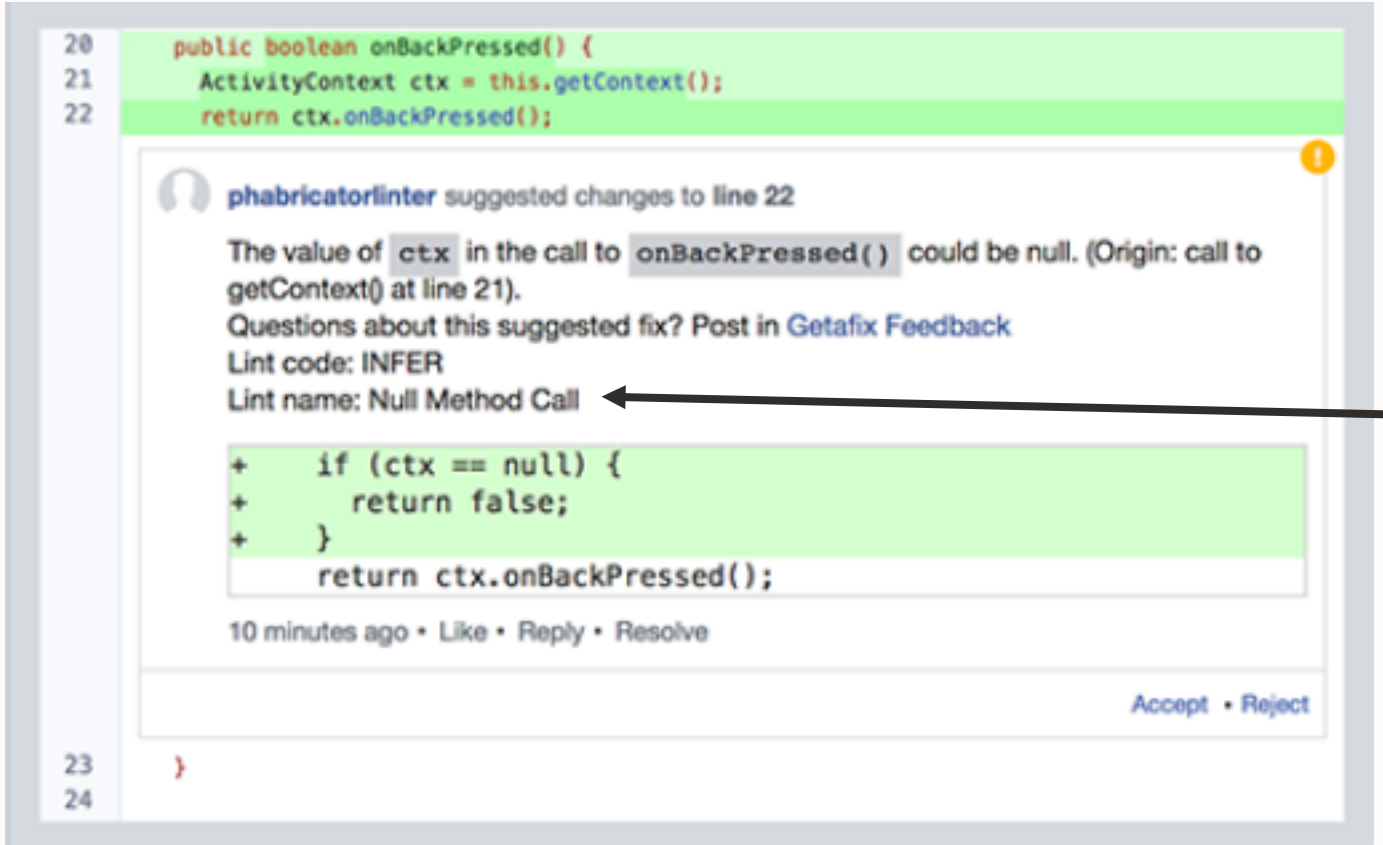
Automatic generation of tests for Android applications based on system testing



"Automated Fault-Finding and Fixing at Facebook", Mark Harman, POPL 2019 Keynote

Facebook's GetAFix

Finds fixes for bugs and offers them to engineers

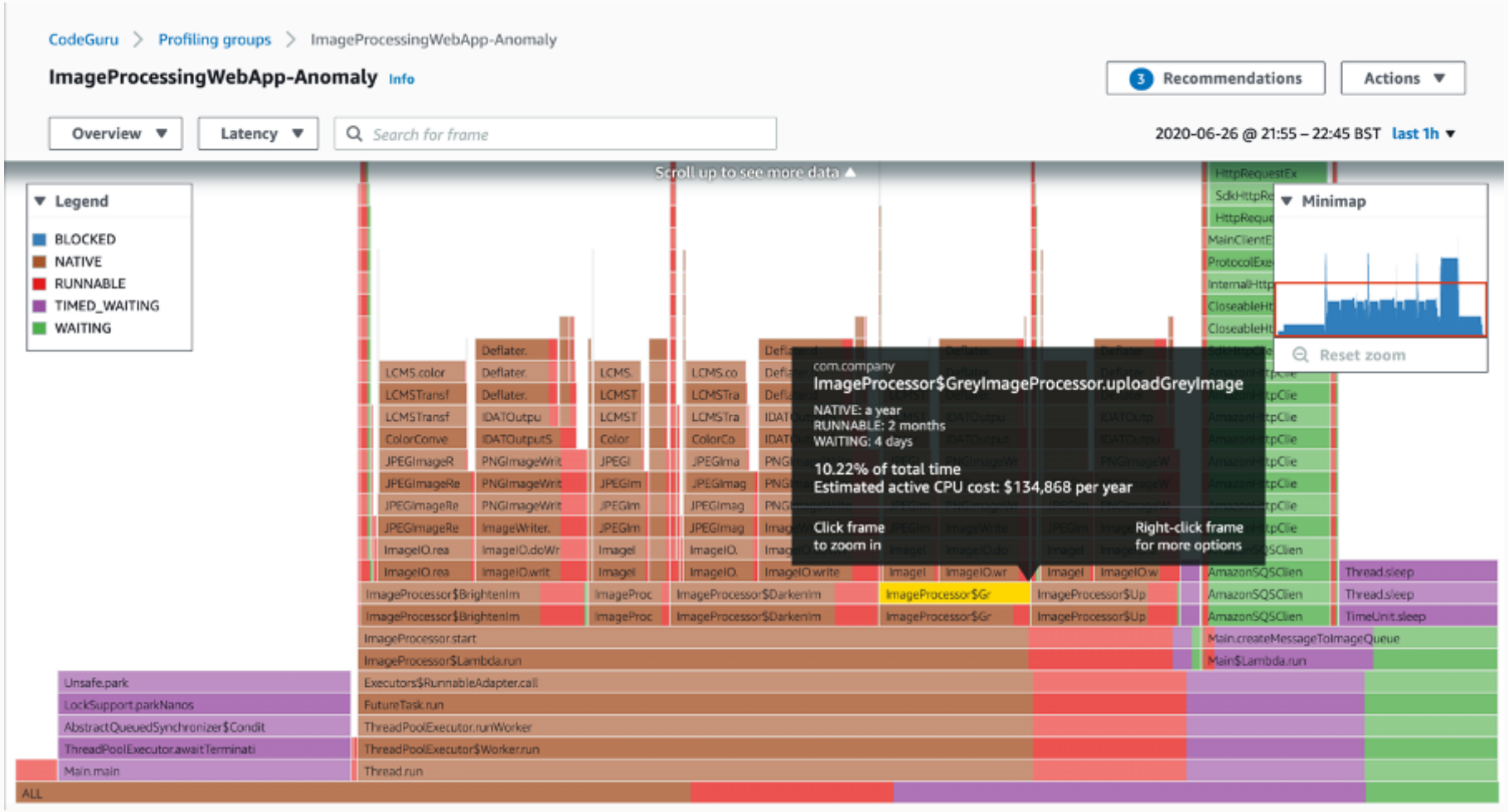


A code fix to a **lint** error

https://engineering.fb.com/developer-tools/getafix-how-facebook-tools-learn-to-fix-bugs-automatically/

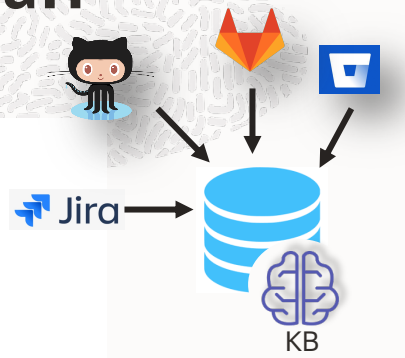
Amazon's CodeGuru Profiler

Finds most expensive lines of code and recommends improvements



https://aws.amazon.com/codeguru/

These Examples Help Developers, Testers and Operations Staff



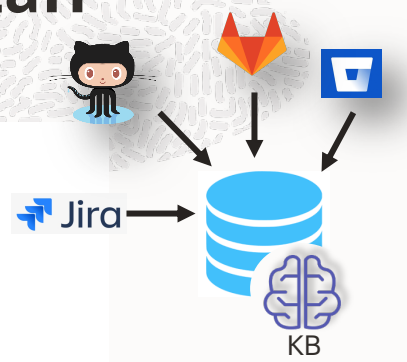
Intelligent coding
e.g., VS IntelliCode, Aroma, CodeGuru Reviewer, GetAFix

Proposed fixes

Intelligent testing
e.g., Sapienz, GetAFix

Intelligent monitoring
e.g., CodeGuru Profiler

These Examples Help Developers, Testers and Operations Staff



Intelligent coding
e.g., VS IntelliCode, Aroma, CodeGuru Reviewer, GetAFix

Proposed fixes

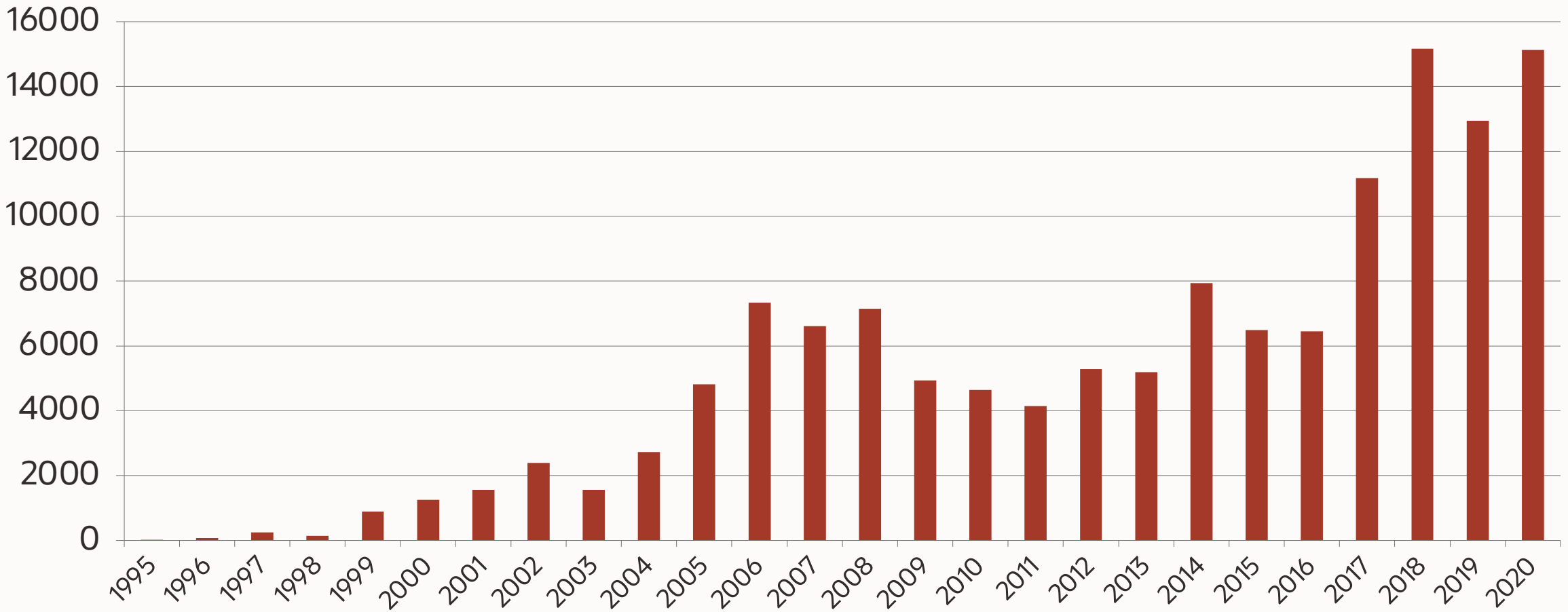
Some are still at the "lint" level

Intelligent testing
e.g., Sapienz, GetAFix, Bug2Commit

Intelligent monitoring
e.g., CodeGuru Profiler

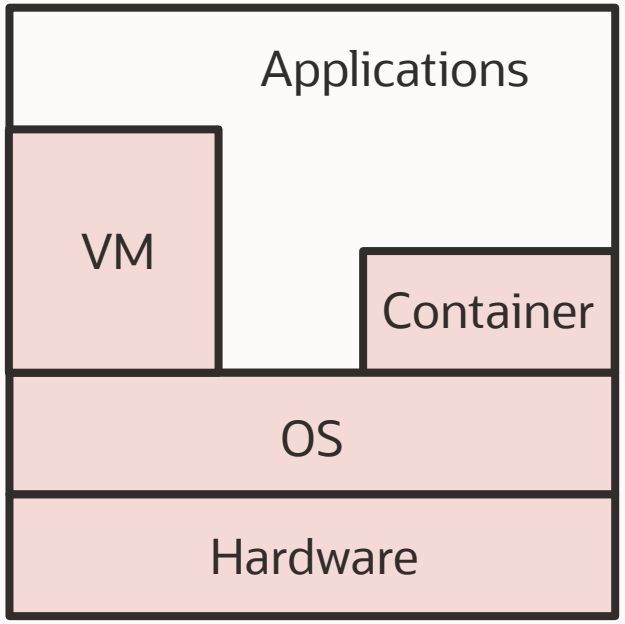
What is the Future of Application Security?

Exploited CVE Vulnerabilities Per Year



https://nvd.nist.gov

Vulnerabilities in the Systems and Applications Stack



SQL injection, XSS, Unsafe deserialisation, …

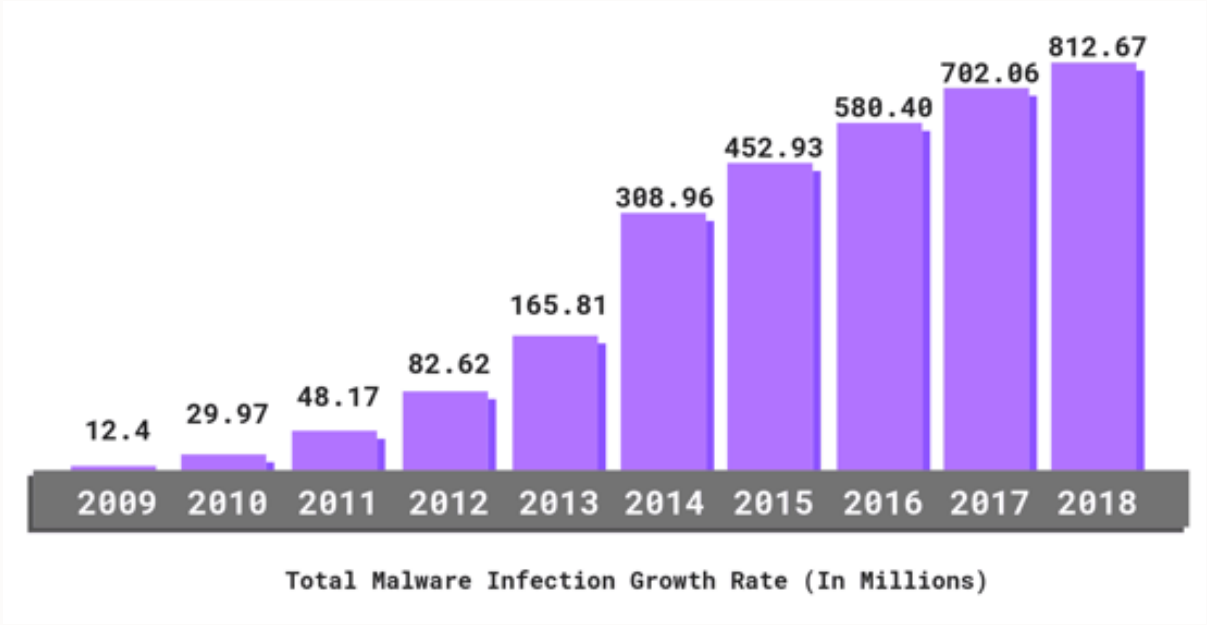
Unguarded Caller-Sensitive Method call, Unsafe deserialisation, …

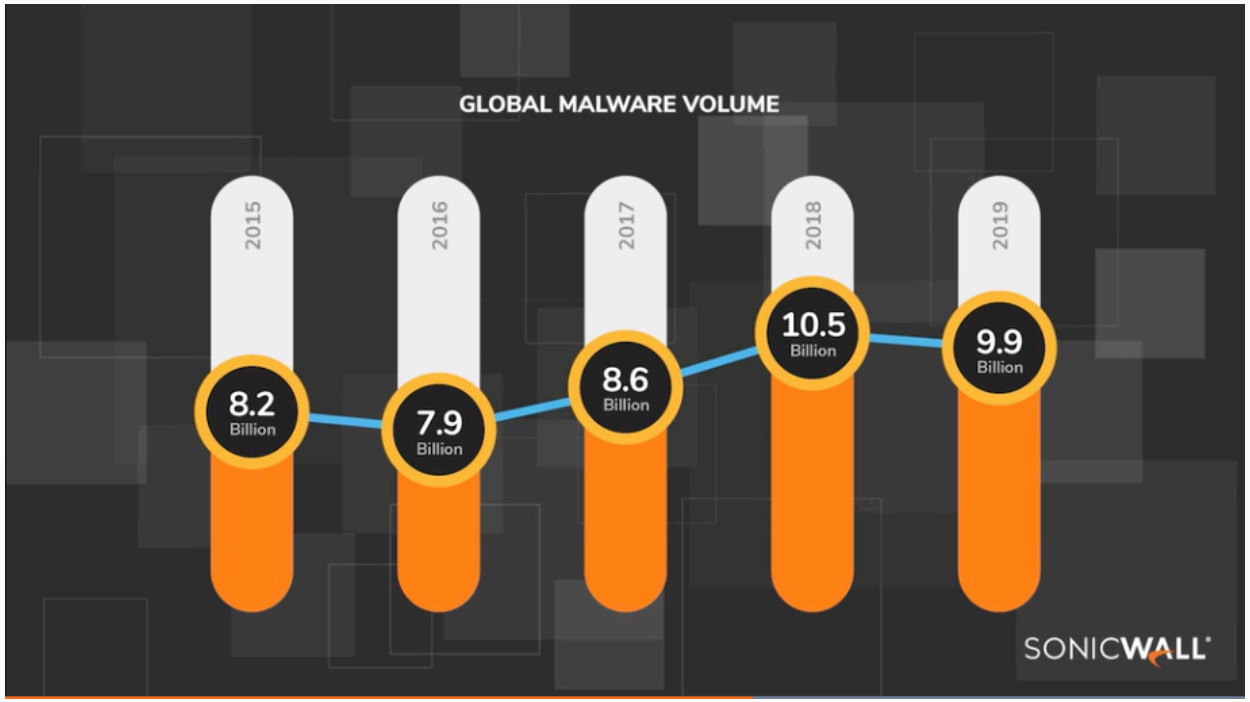
OS + VM + Application vulnerabilities

Buffer overflow, use after free, …

Spectre, Meltdown, L1TF, …

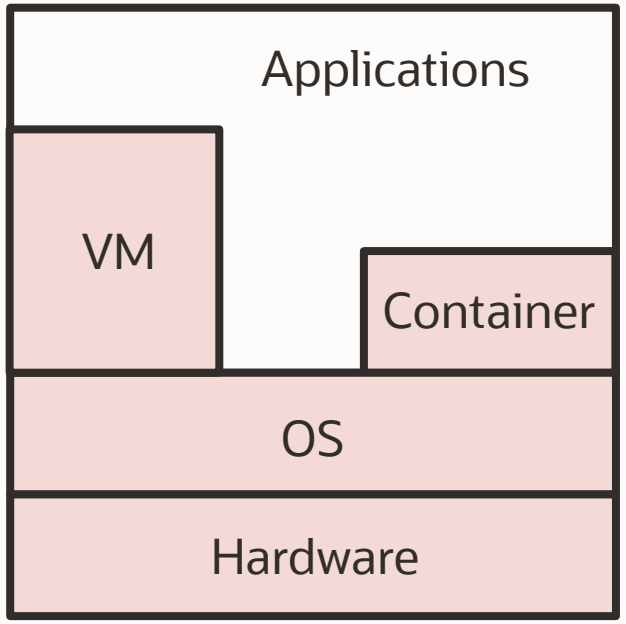
The Rise of Malware





https://purplesec.us/resources/cyber-security-statistics/
https://www.sonicwall.com/medialibrary/en/infographic/infographic-2020-sonicwall-cyber-threat-report.pdf

Sample Systems and Applications Software Affected by Malware



SolarWinds Orion Platform, Dec 2020

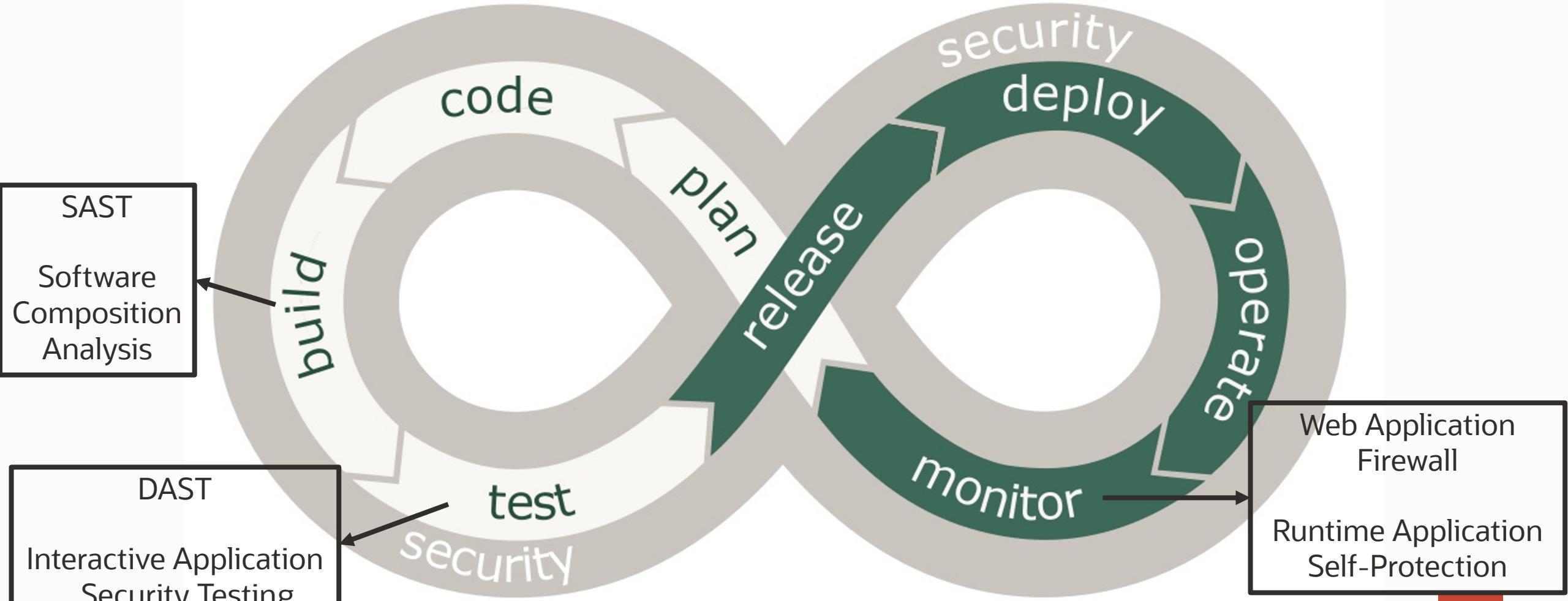
ESLint Scope, Jul 2018

Cryptominer in Community Amazon Machines image, Aug 2020

XCodeGhost (iOS), Sep 2015

Apple's M1 chip, Feb 2021

Today – Application Security Testing in the DevSecOps Model



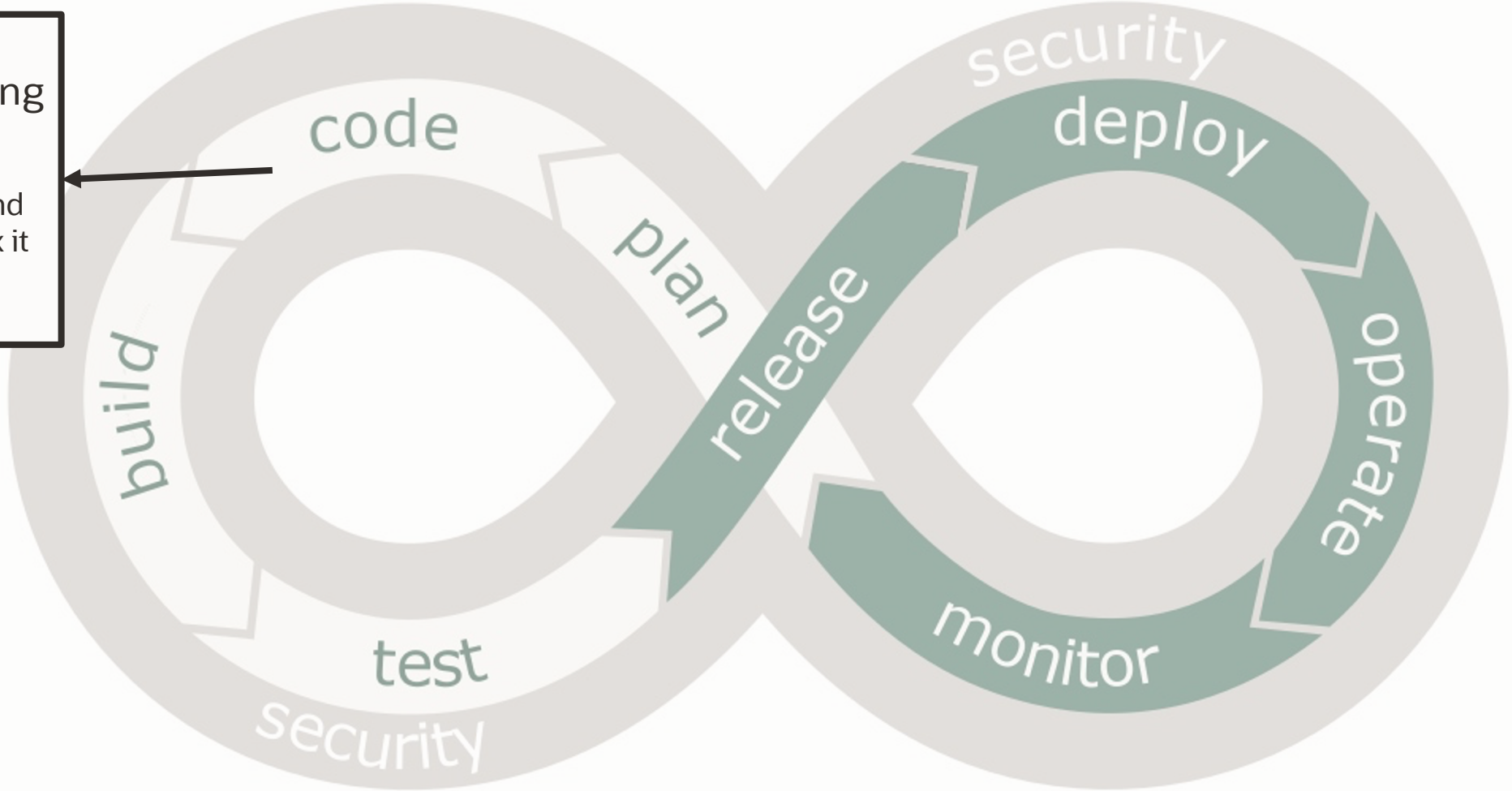
We Can Build an Intelligent Application Security Future

#ias

Intelligent Application Security

Intelligent security coding

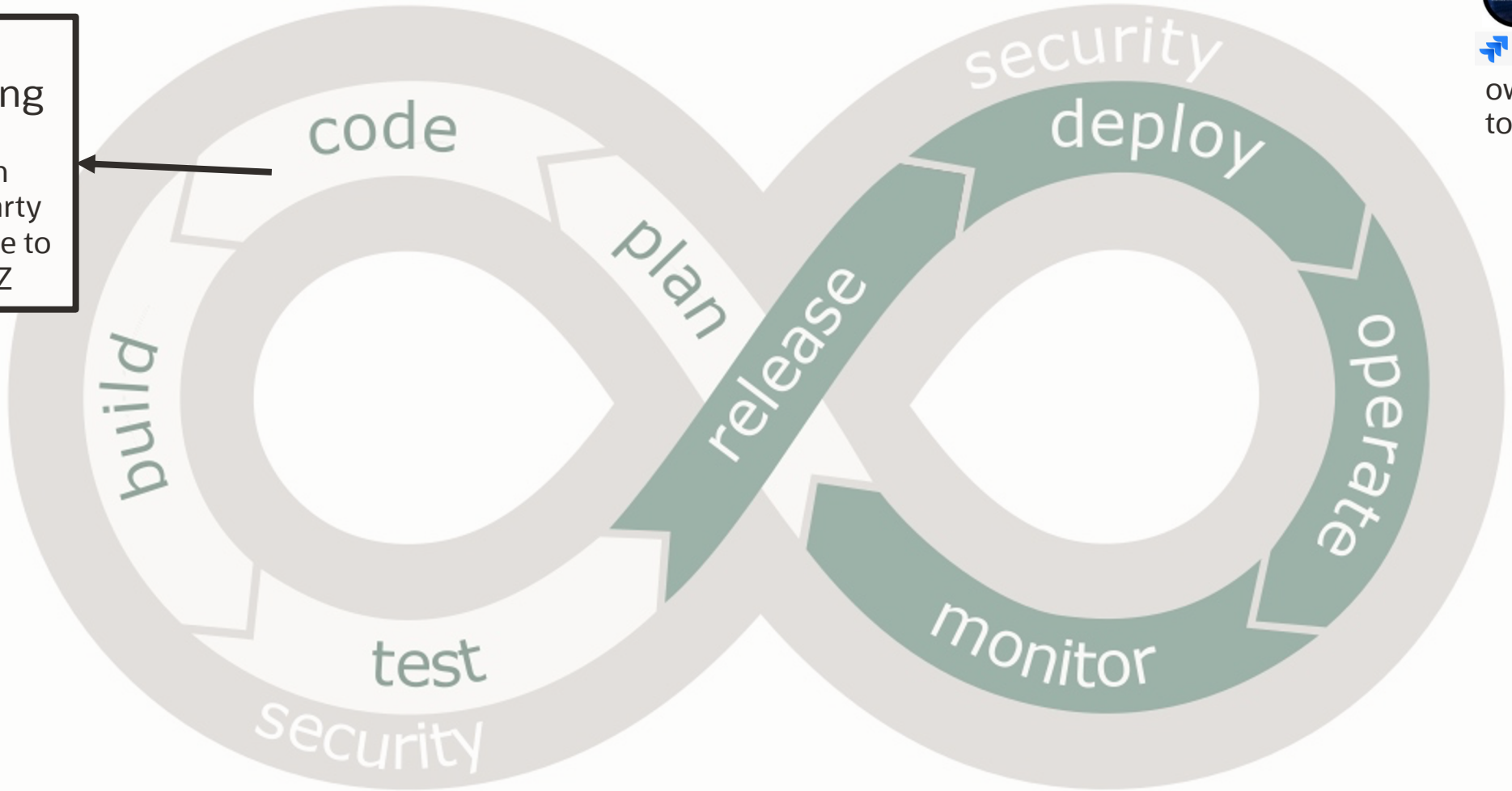
You have a bug at line X and here's how to fix it [before you commit]

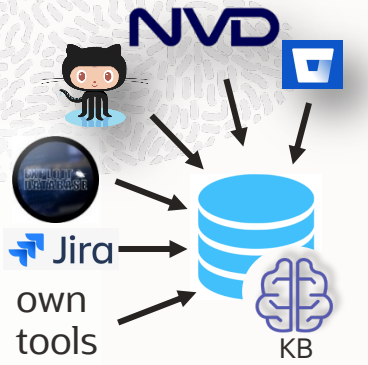


Intelligent Application Security

Intelligent security coding

You depend on vulnerable 3rd party library Y – upgrade to clean version Z

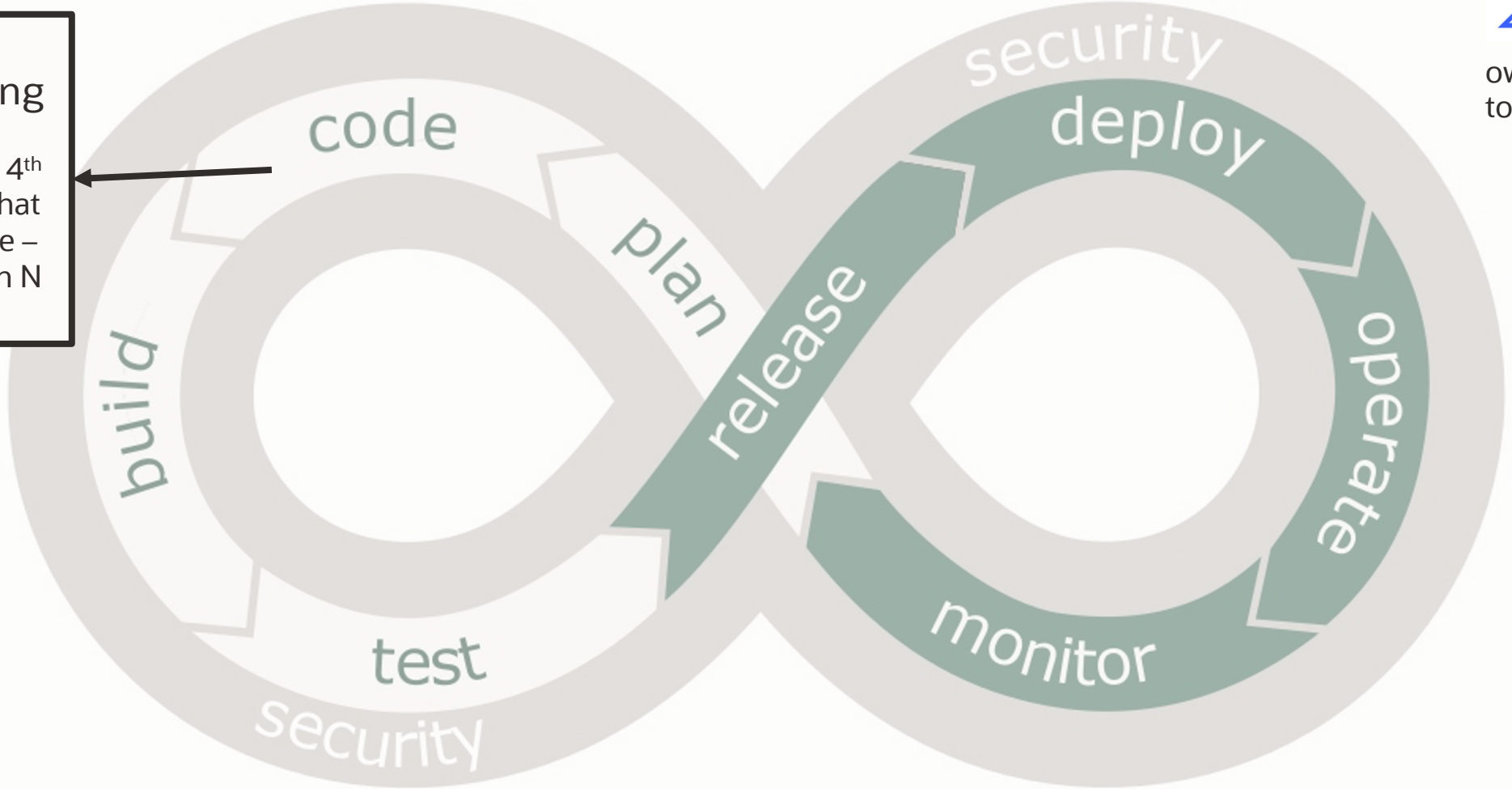


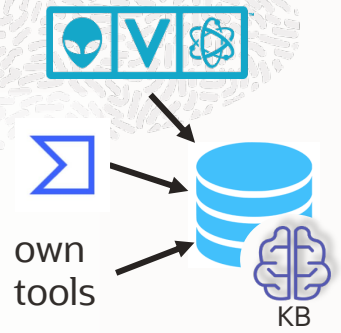


Intelligent Application Security

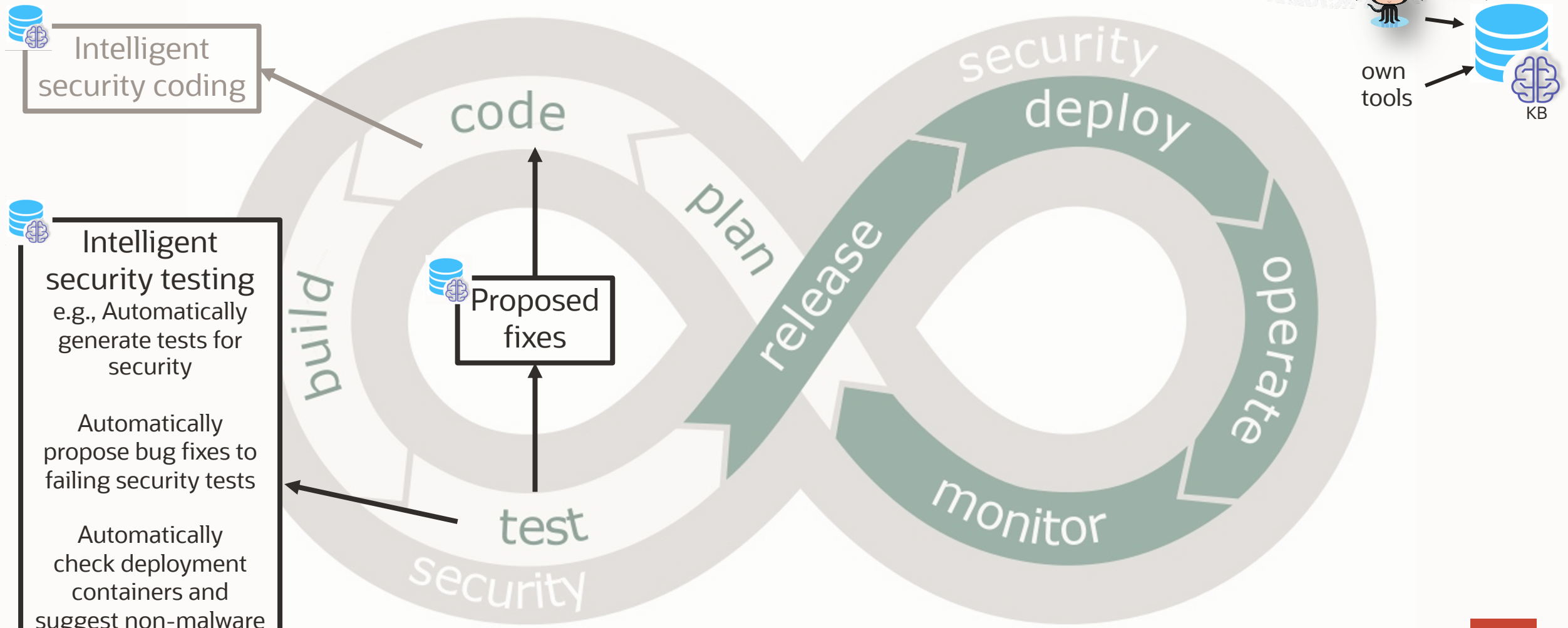
Intelligent security coding

You depend on a 4th party library M that contains malware – use clean version N instead





Intelligent Application Security



How do we check for malware imported via a 3rd party dependency or contained in a container?

“A decompiler is a program that reads a program written in a machine language – the source language – and translates it into an equivalent program in a high-level language – the target language.”

Cristina Cifuentes

“Reverse Compilation Techniques”, PhD Thesis,
Queensland University of Technology, July 1994

Decompilation to Reverse Engineer Malware

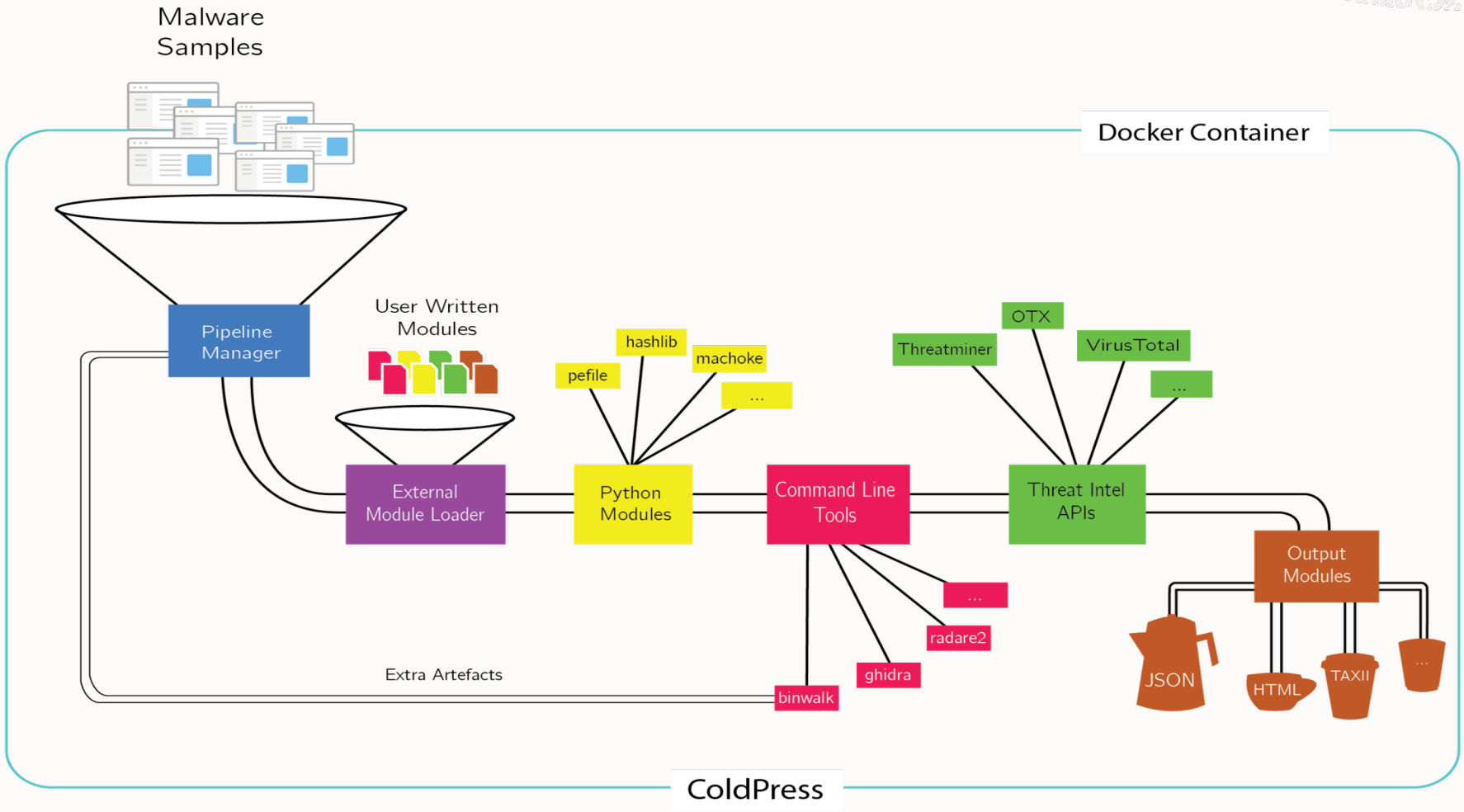
Ghidra's decompilation of WannaCry

```
nt entry(undefined4 param_1,int param_2,undefined4 param_3)


 int iVar1;
 int iVar2;
 int iVar3;
 undefined4 uVar4;

 iVar1 = param_2;
 iVar2 = _DAT_10003140;
 if (param_2 != 0) {
   if ((param_2 != 1) && (param_2 != 2)) goto LAB_10001231;
   if ((DAT_10003150 != (code *)0x0) &&
      (iVar2 = (*DAT_10003150)(param_1,param_2,param_3), iVar2 == 0)) {
     return 0;
   }
   iVar2 = FUN_1000113e(param_1,param_2,param_3);
 }
 if (iVar2 == 0) {
   return 0;
 }
AB_10001231:
 iVar2 = FUN_10001000(param_1,param_2,param_3);
 if (param_2 == 1) {
```

https://www.nsa.gov/resources/everyone/ghidra/

ColdPress

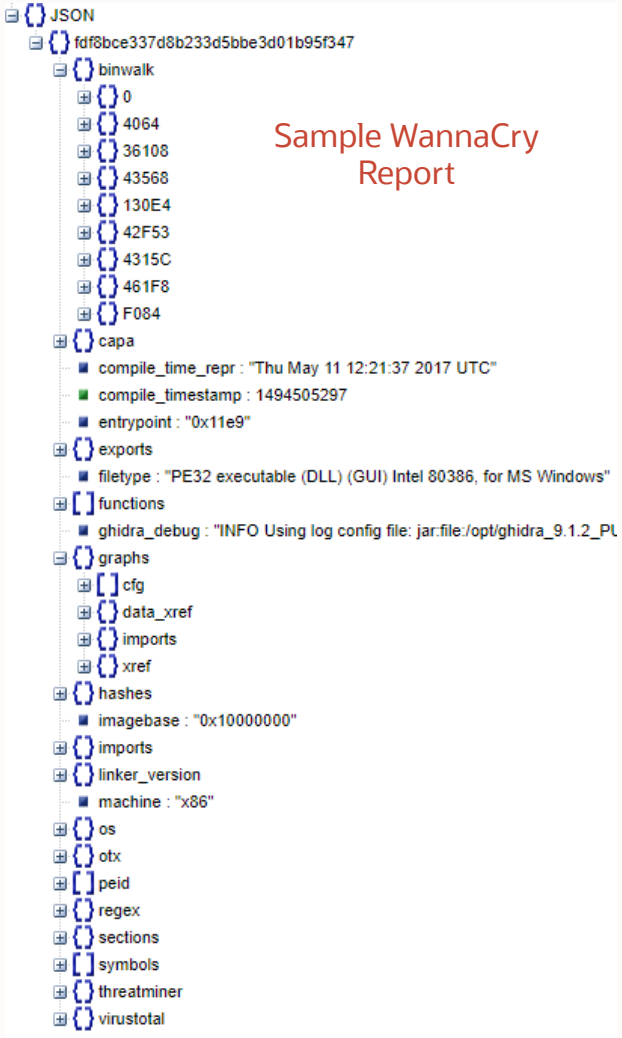
Fully automated extraction of Indicators of Compromise from Windows binaries using static malware analysis open source tools



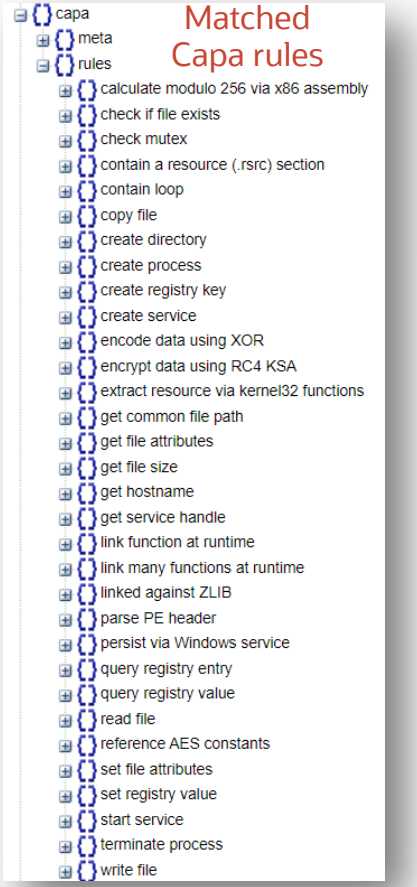
https://github.com/uqcyber/ColdPress

Decompilation to Extract Indicators of Compromise

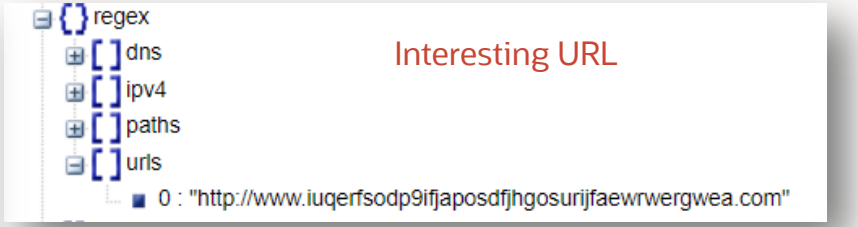
Sample WannaCry Report



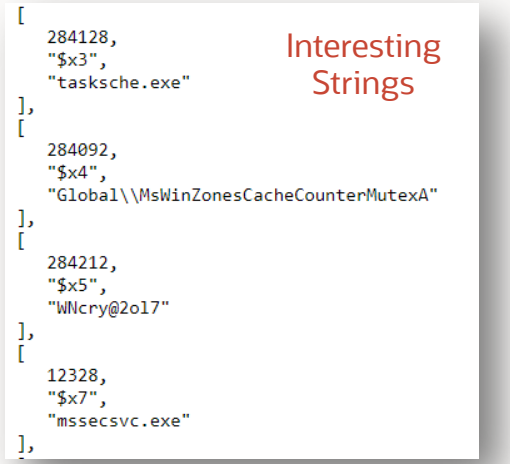
Matched Capa rules



Interesting URL

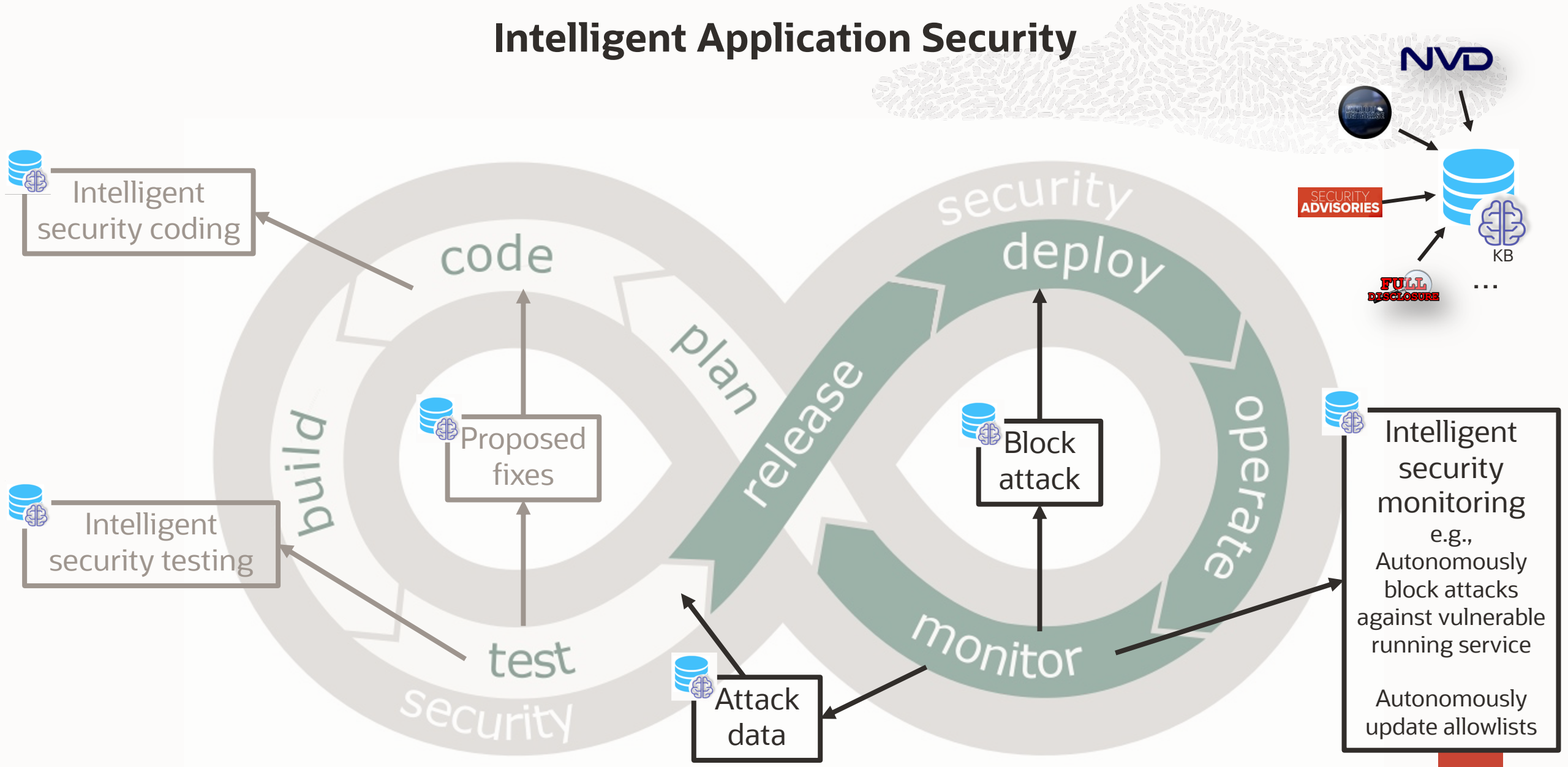


Interesting Strings

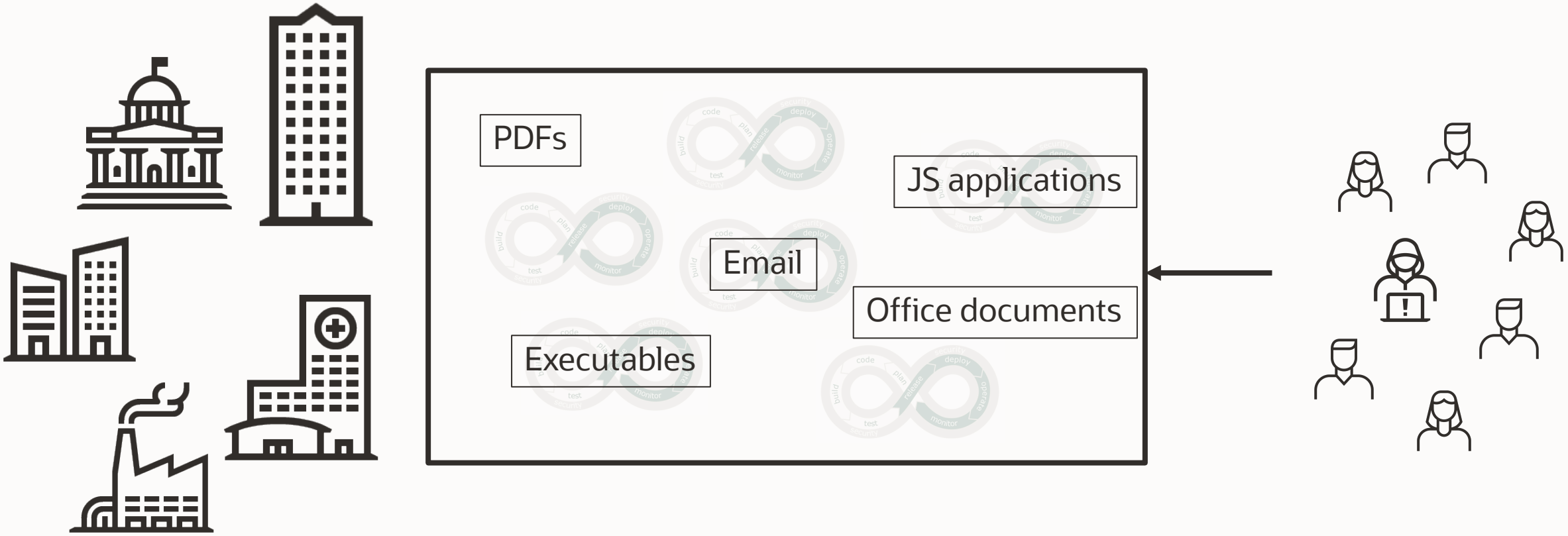


https://github.com/uqcyber/ColdPress

Intelligent Application Security



An Organisation Runs Many Applications and Consumes Many Data Formats



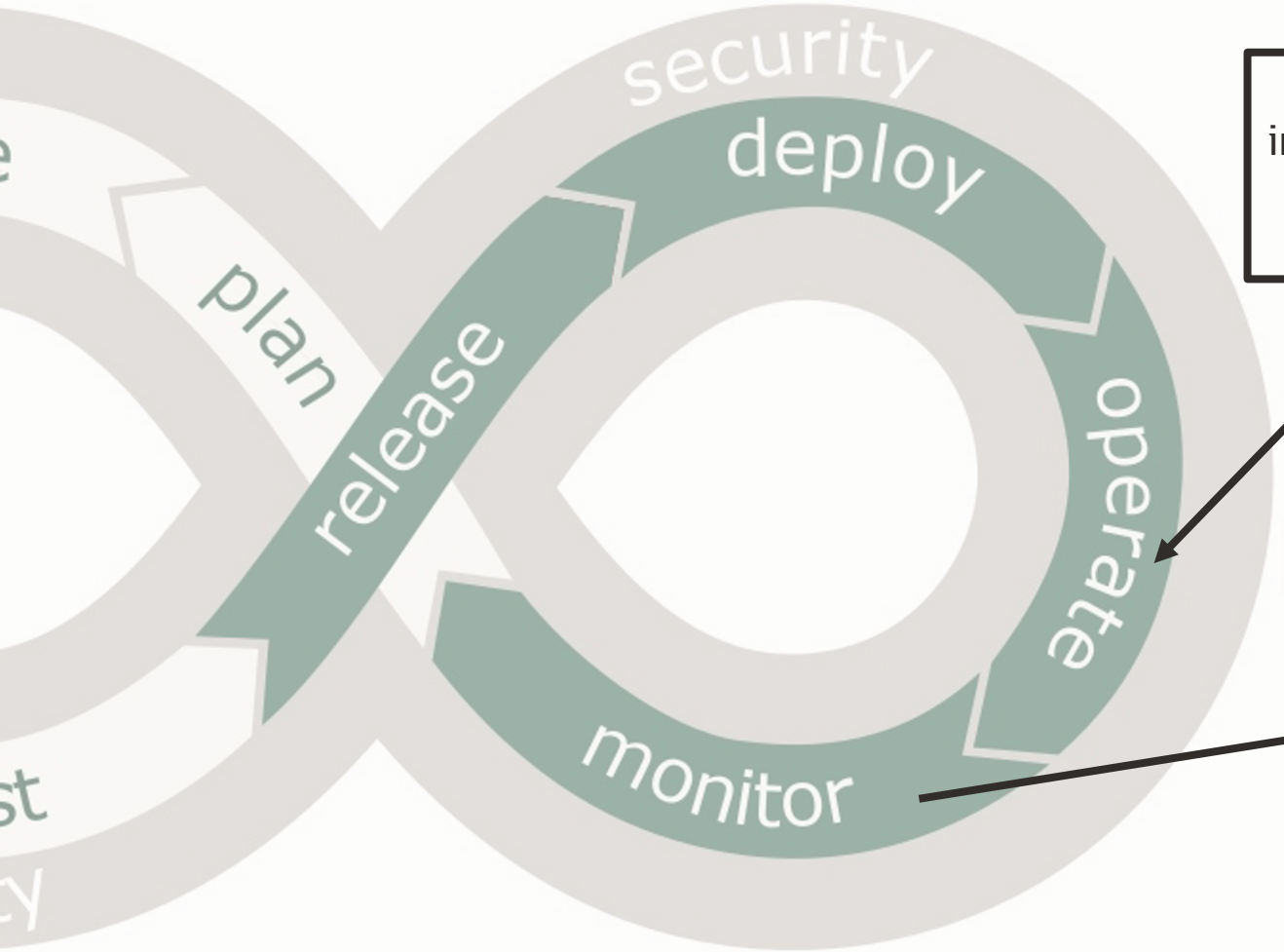
Office and PDF malicious files account for 33% of new malware detection. Exe accounts for 15.8%.

Mid-Year Update – 2020 SonicWall Cyber Threat Report

July 2020

https://www.sonicwall.com/resources/2020-cyber-threat-report-mid-year-update-pdf/

Ingesting Data and Accessing Websites



Malware injection through 'normal use' of the application

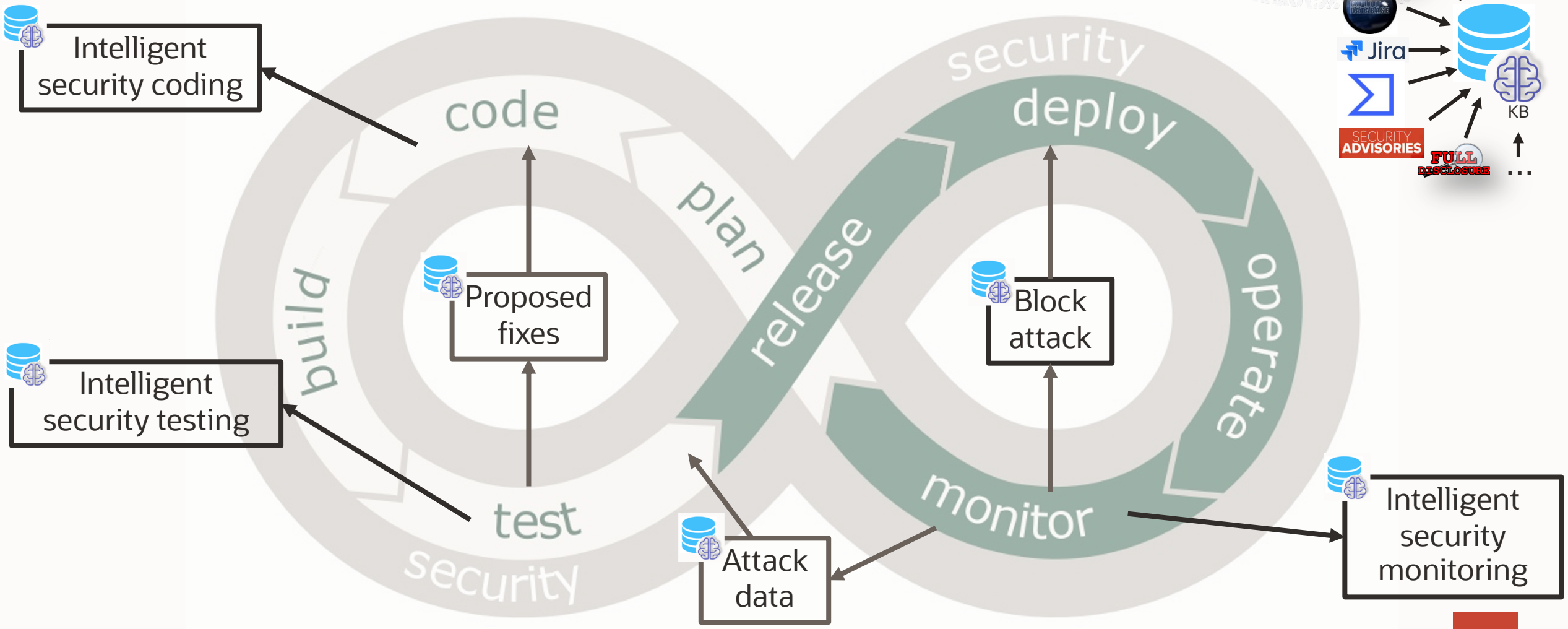
Intelligent security monitoring, including malware detection

SIEM

KB

...

Tomorrow – Intelligent Application Security



ORACLE

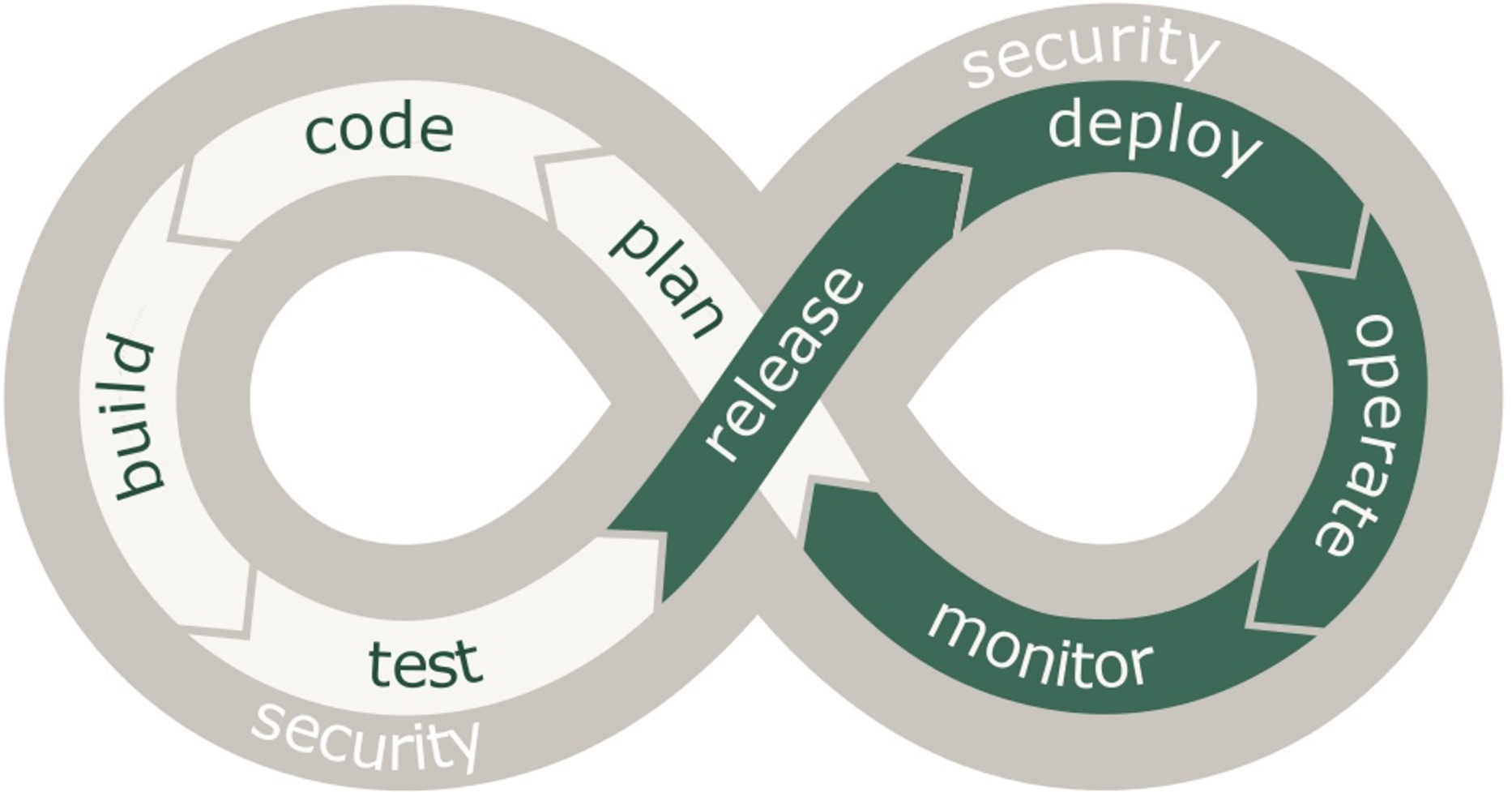
“Intelligent Application Security aims to provide an automated approach to integrate security into all aspects of application development and operations, at scale, using learning techniques that incorporate signals from the code and beyond, to provide actionable intelligence to developers, security analysts, operations staff, and autonomous systems.”

Cristina Cifuentes

October 2020

Recap

Today – DevSecOps – Integrating Security Into DevOps



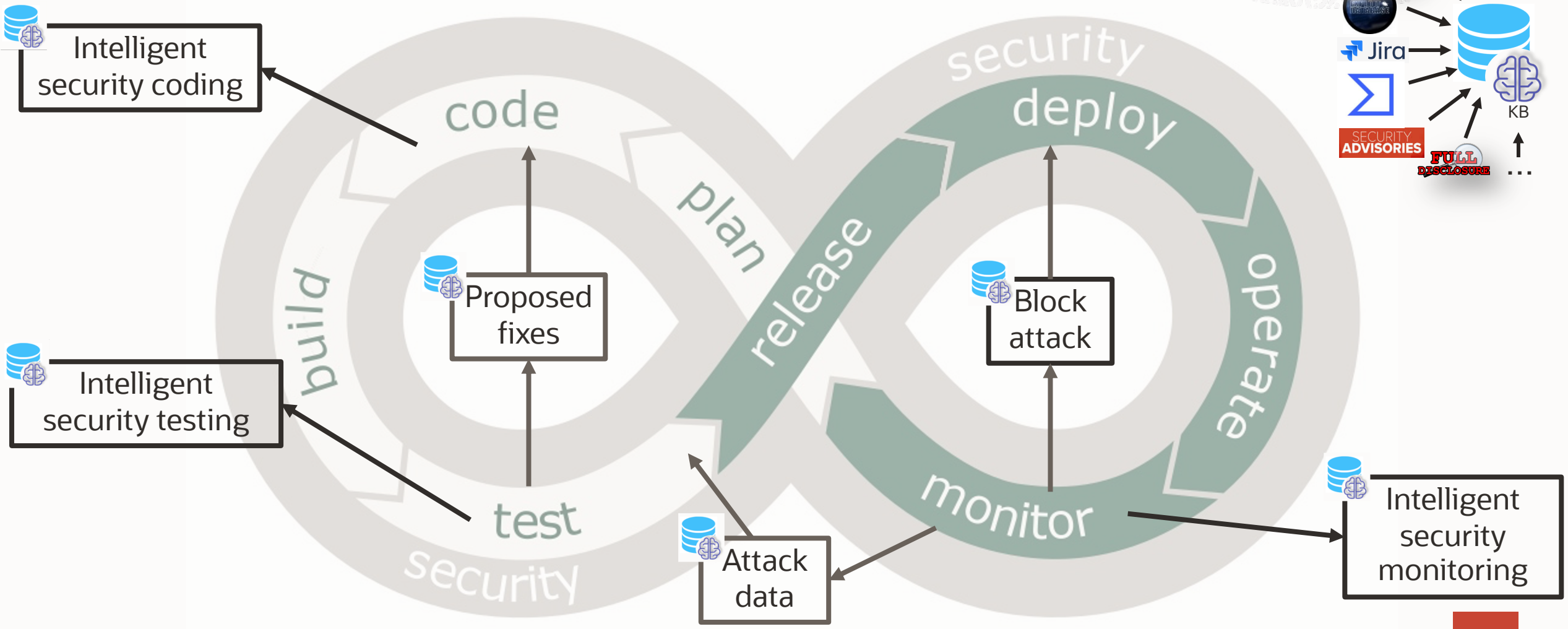
Learning-based technologies have ripen over the past 10 years

Security is not just for expert developers

Automation is key

It's time to combine program analysis, learning-based techniques and data analytics to make Intelligent Application Security, at scale, a reality

Tomorrow – Intelligent Application Security



ORACLE

#ias

cristina.cifuentes@oracle.com

http://labs.oracle.com

Twitter: @criscifuentes

LinkedIn: drcristinacifuentes


ORACLE