CONTRAST SECURITY

SECURITY OBSERVABILITY 101: THINKING INSIDE THE BOX!

JEFF WILLIAMS, COFOUNDER AND CTO
@PLANETLEVEL

OWASP CHARLOTTE – OCT 2021

REAL WORLD APPSEC FACTS FROM LAST 12 MONTHS…

VULNERABILITY FACTS

- 96% of applications have at least one vulnerability
- The average software project introduces 2-3 new vulnerabilities every month
- The average application has 30+ vulnerabilities and 2+ high or critical flaws in open source libraries
- Average application codebase:
  - – 20% is custom code
  - – 6% is OSS that actually runs
  - – 74% is never used
- Only 14% of libraries are the latest version

CONTRAST SECURITY

2021 APPLICATION SECURITY OBSERVABILITY REPORT

Understanding Application and API Risk by Connecting Real-world Vulnerabilities With Actual Attack Data

2021 REPORT

Figure 11. Percent of vulnerabilities remaining by days since first discovered.

% OF VULNERABILITIES REMAINING
DAYS SINCE DISCOVERY
VULNERABILITIES
SERIOUS VULNERABILITIES
30 DAYS: 61% OF VULNERABILITIES ARE REMEDIATED
90 DAYS: 69% OF VULNERABILITIES ARE REMEDIATED
30 DAYS: 79% OF SERIOUS VULNERABILITIES ARE REMEDIATED
90 DAYS: 87% OF SERIOUS VULNERABILITIES ARE REMEDIATED

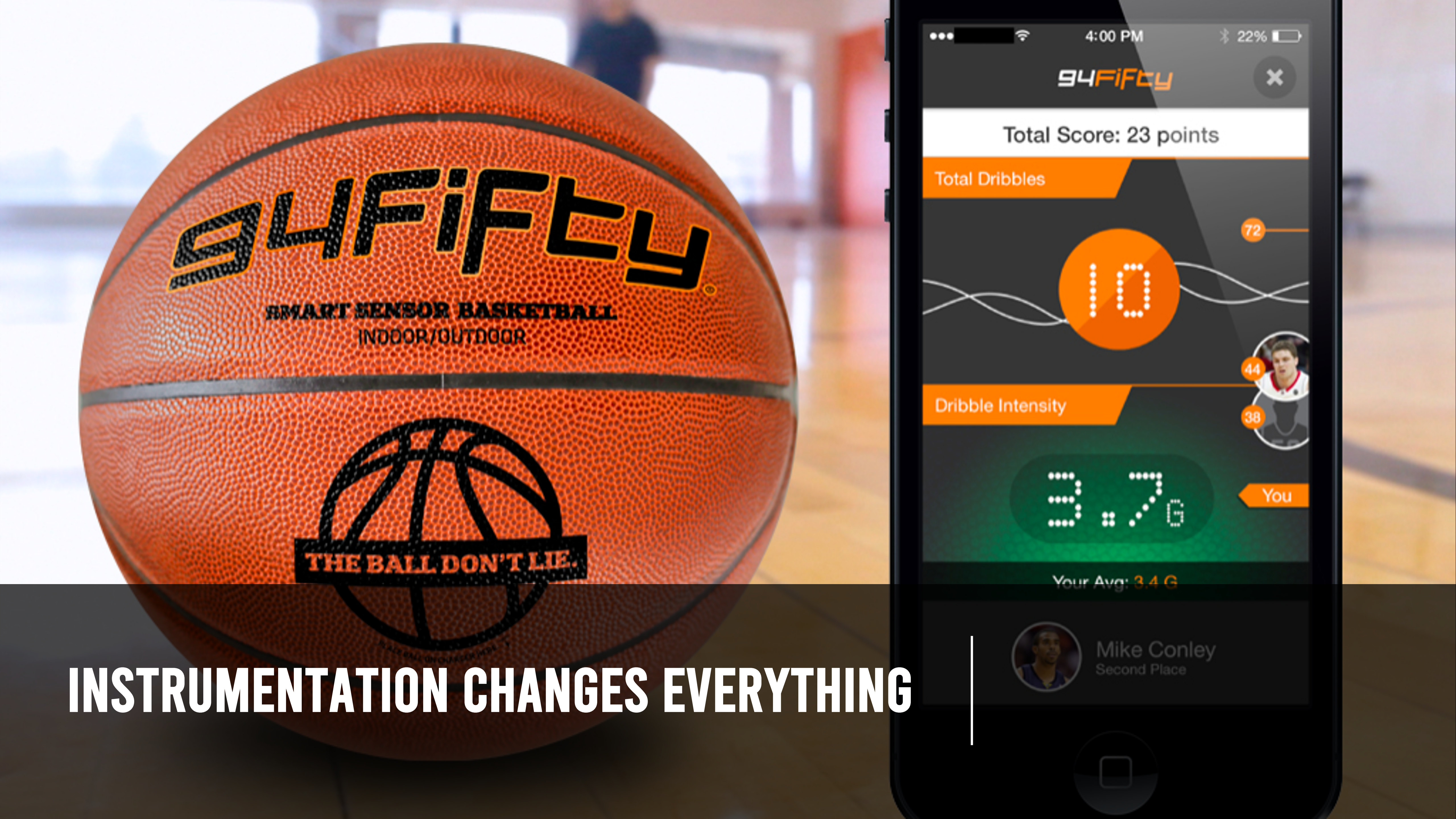
the MTTR varies widely according to the type of
ple, are resolved in an average of 10 days,
entication categories take more than
ged by organizations' success in
ious vulnerabilities is just over

22

Contrast 2021 AppSec Observability Report

ATTACK FACTS

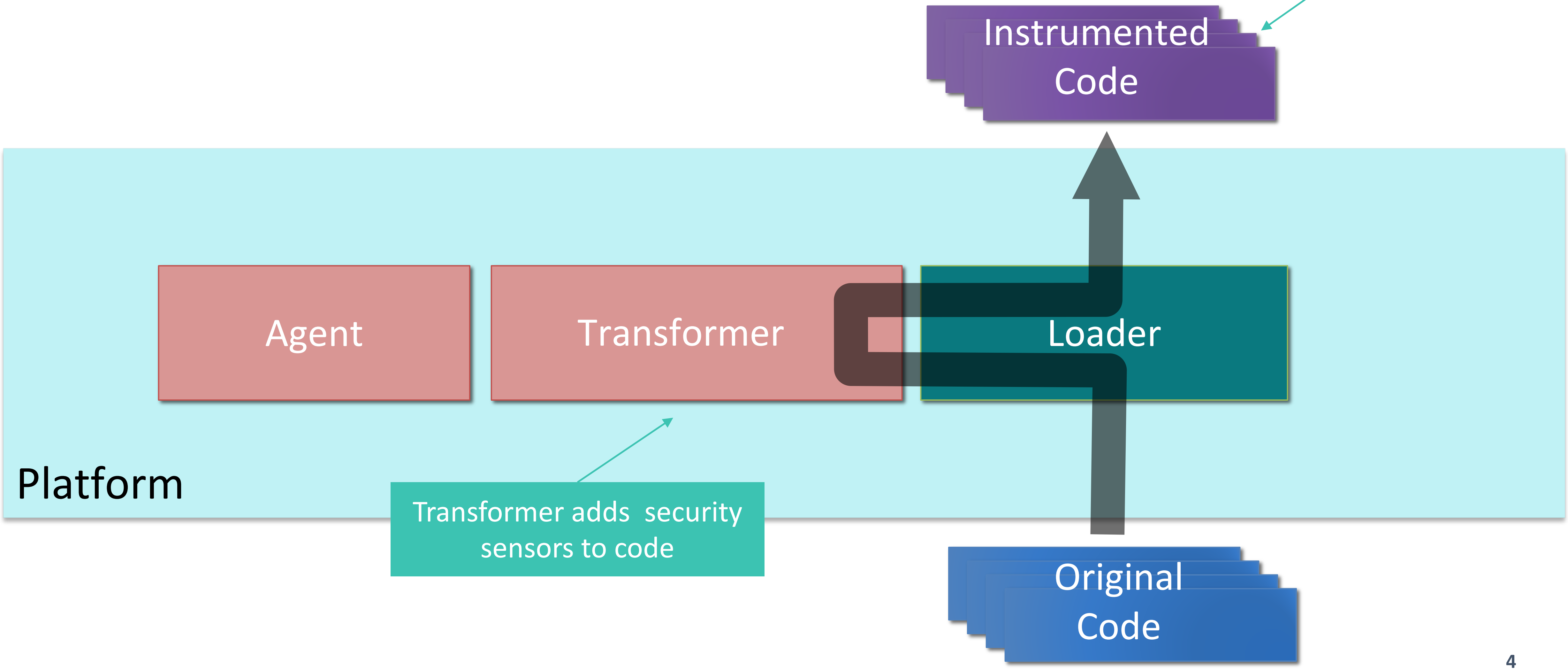
- The average application is attacked over 13,000 times a month
- Zero applications were not attacked every single month
- 99% of attacks do not connect with their intended vulnerability
- Attacks on all vulnerabilities are trending up over the last 12 months



INSTRUMENTATION CHANGES EVERYTHING

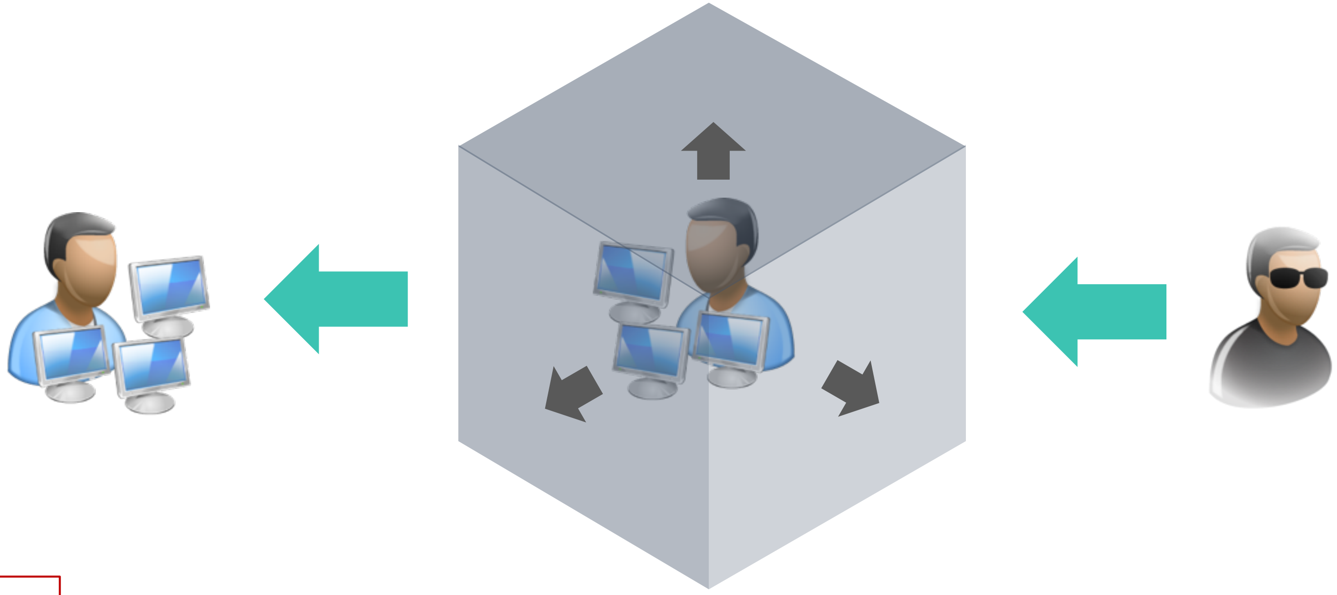
INSTRUMENTATION IS EASY

Run instrumented code!



Transformer adds security sensors to code

ADD IN SENSORS TO REVEAL SECURITY



WIN!

USE INSTRUMENTATION
TO DEFINE SENSORS
THAT MAKE SOFTWARE
BEHAVIOR OBSERVABLE

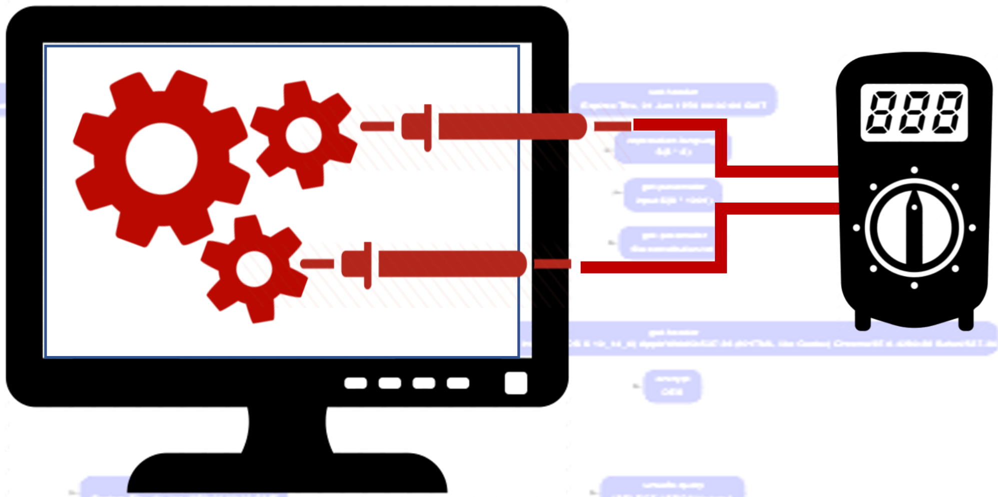
INSTRUMENTATION
PROVIDES CONTINUOUS
ACCURATE REALTIME
TELEMETRY

"RUNTIME REALITY"
FULLY ASSEMBLED AND
RUNNING APPS/APIS
ARE THE ONLY SOURCE
OF TRUTH

THE JAVA OBSERVABILITY TOOLKIT (JOT)

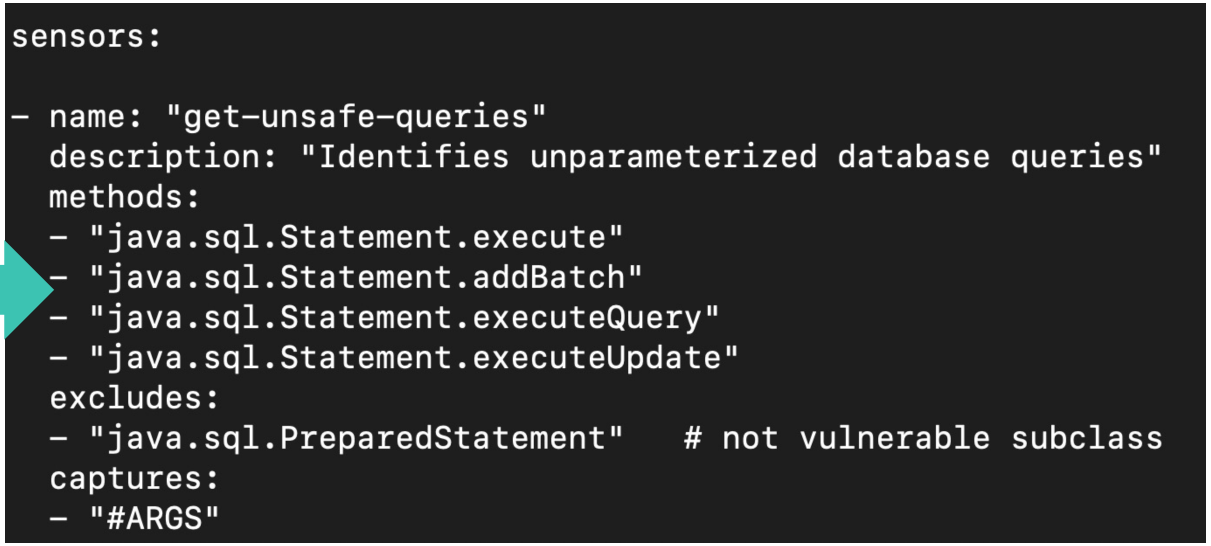
FREE OPEN SOURCE INSTRUMENTATION

${JOT}

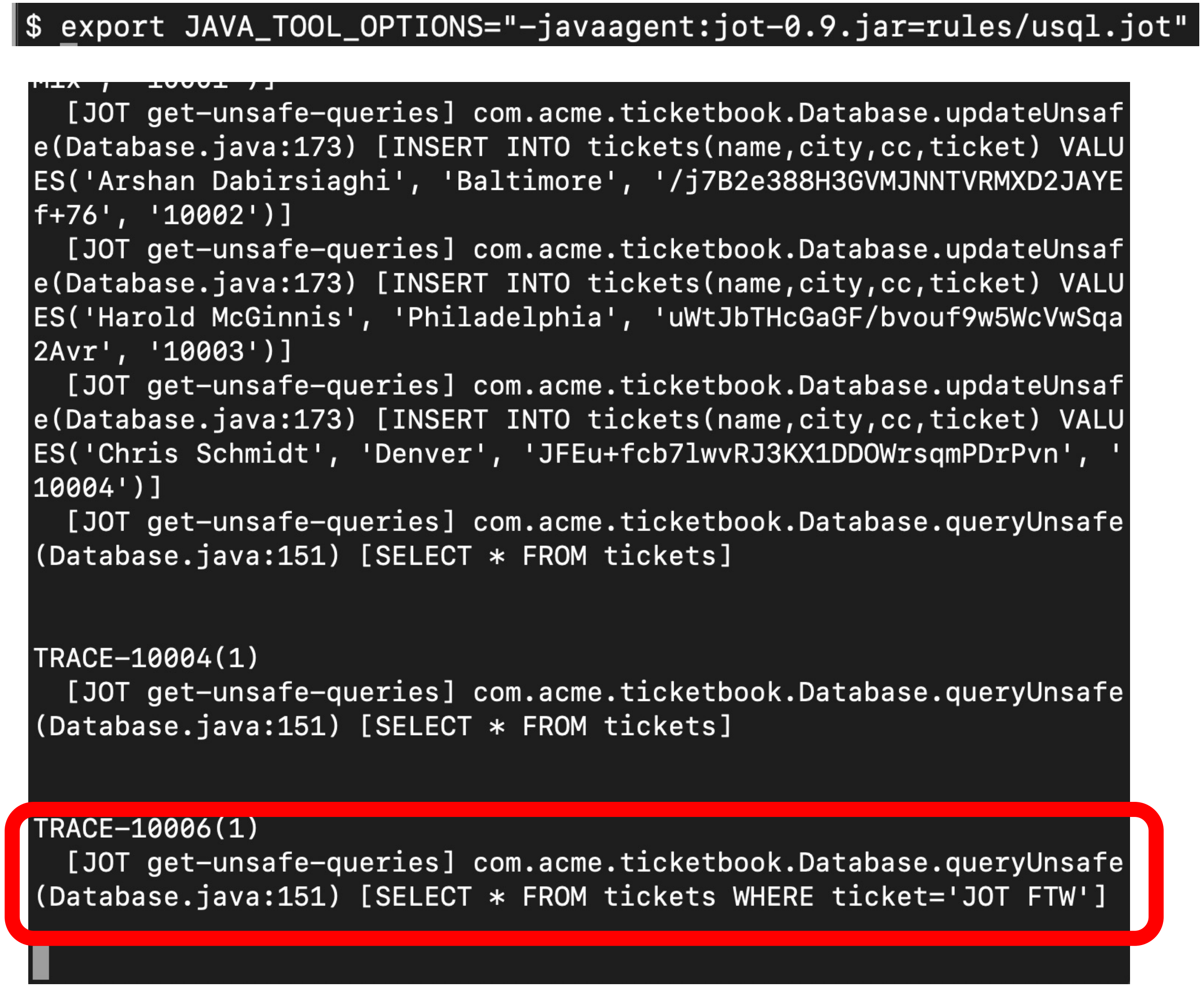


https://github.com/planetlevel/jot

RIDICULOUSLY SIMPLE EXAMPLE: WEAK SQL QUERIES



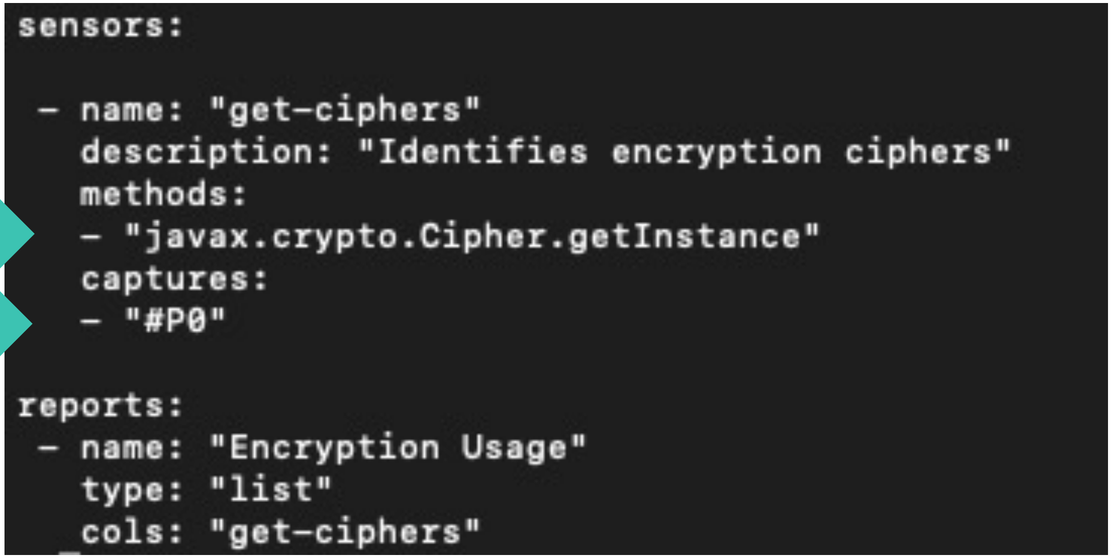
```
sensors:

- name: "get-unsafe-queries"
  description: "Identifies unparameterized database queries"
  methods:
  - "java.sql.Statement.execute"
  - "java.sql.Statement.addBatch"
  - "java.sql.Statement.executeQuery"
  - "java.sql.Statement.executeUpdate"
  excludes:
  - "java.sql.PreparedStatement"   # not vulnerable subclass
  captures:
  - "#ARGS"
```



```
$ export JAVA_TOOL_OPTIONS="-javaagent:jot-0.9.jar=rules/usql.jot"
```

```
  [JOT get-unsafe-queries] com.acme.ticketbook.Database.updateUnsaf
e(Database.java:173) [INSERT INTO tickets(name,city,cc,ticket) VALU
ES('Arshan Dabirsiaghi', 'Baltimore', '/j7B2e388H3GVMJNNTVRMXD2JAYE
f+76', '10002')]
  [JOT get-unsafe-queries] com.acme.ticketbook.Database.updateUnsaf
e(Database.java:173) [INSERT INTO tickets(name,city,cc,ticket) VALU
ES('Harold McGinnis', 'Philadelphia', 'uWtJbTHcGaGF/bvouf9w5WcVwSqa
2Avr', '10003')]
  [JOT get-unsafe-queries] com.acme.ticketbook.Database.updateUnsaf
e(Database.java:173) [INSERT INTO tickets(name,city,cc,ticket) VALU
ES('Chris Schmidt', 'Denver', 'JFEu+fcb7lwvRJ3KX1DDOWrsqmPDrPvn', '
10004')]
  [JOT get-unsafe-queries] com.acme.ticketbook.Database.queryUnsafe
(Database.java:151) [SELECT * FROM tickets]


TRACE-10004(1)
  [JOT get-unsafe-queries] com.acme.ticketbook.Database.queryUnsafe
(Database.java:151) [SELECT * FROM tickets]


TRACE-10006(1)
  [JOT get-unsafe-queries] com.acme.ticketbook.Database.queryUnsafe
(Database.java:151) [SELECT * FROM tickets WHERE ticket='JOT FTW']
```

WHAT ENCRYPTION IS HAPPENING?

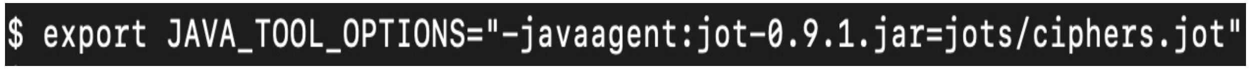


```
sensors:

  - name: "get-ciphers"
    description: "Identifies encryption ciphers"
    methods:
    - "javax.crypto.Cipher.getInstance"
    captures:
    - "#P0"

reports:
  - name: "Encryption Usage"
    type: "list"
    cols: "get-ciphers"
```

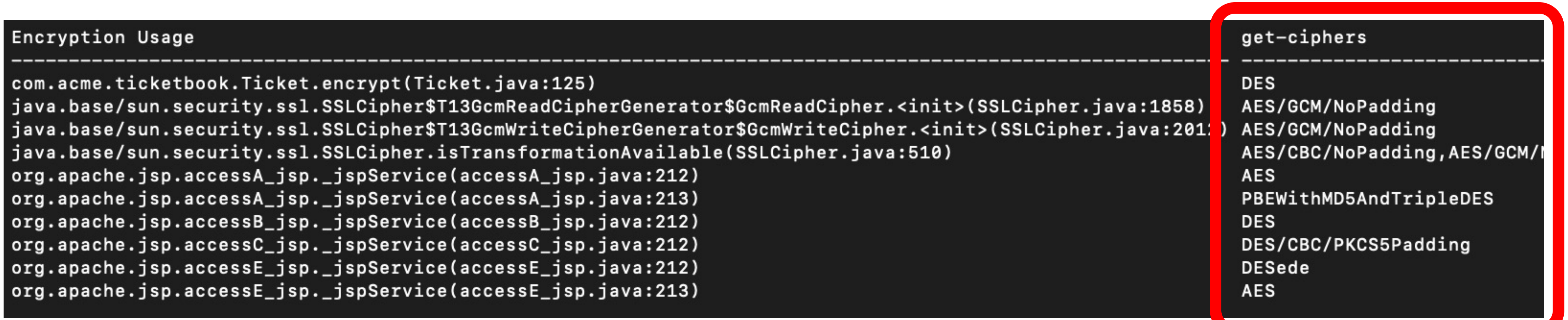
In JOT, a "capture" is a "spring expression" (SPEL) that allows you to extract data using references to objects in the running app/API.

- #P0 is the first parameter to the method
- #OBJ is the object itself
- #RET is the return value from the method

You can call methods on these references!!!

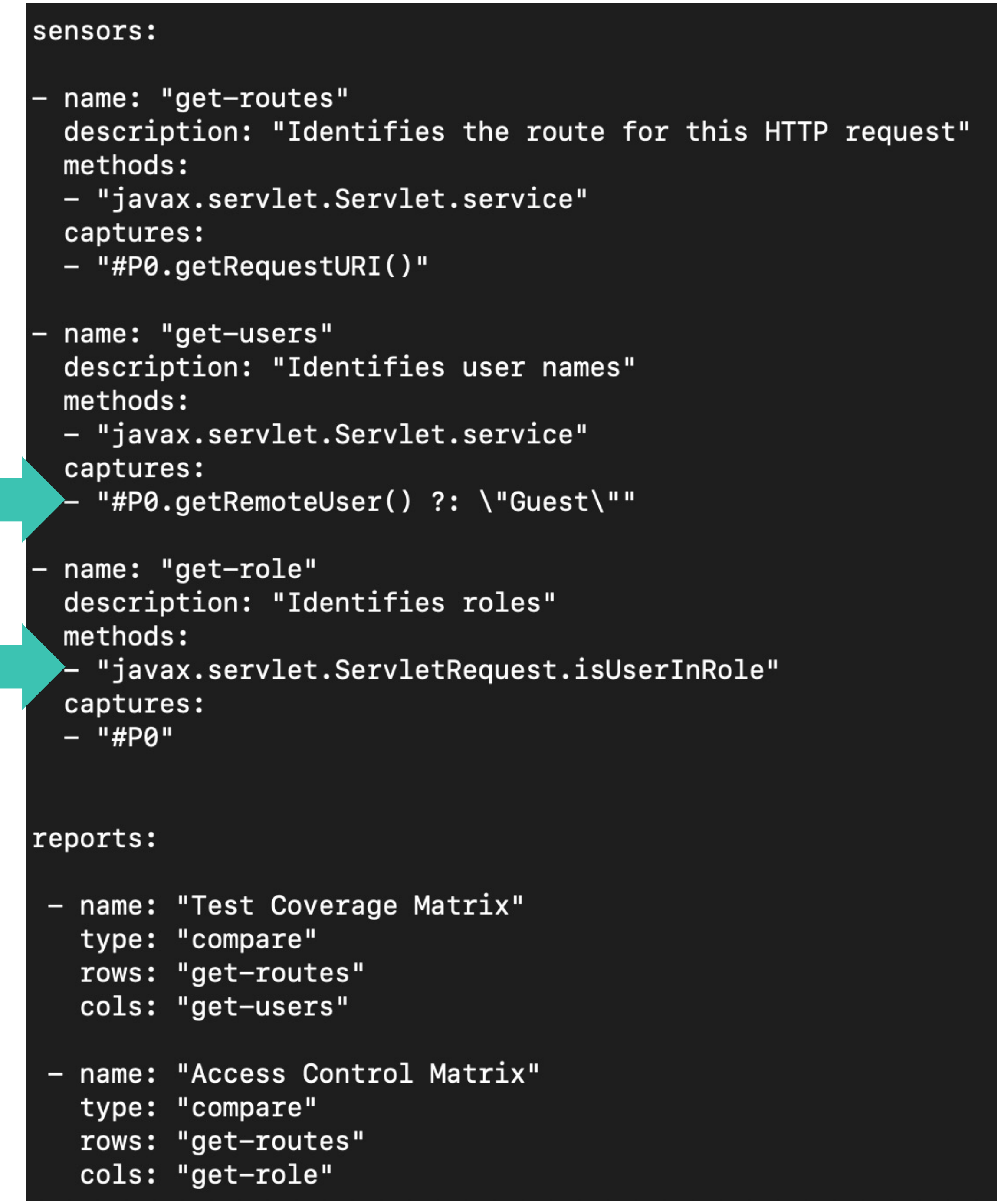


```
$ export JAVA_TOOL_OPTIONS="-javaagent:jot-0.9.1.jar=jots/ciphers.jot"
```

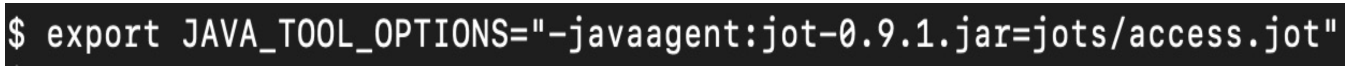


```
Encryption Usage                                                                                           get-ciphers
-------------------------------------------------------------------------------------------------------   --------------------------
com.acme.ticketbook.Ticket.encrypt(Ticket.java:125)                                                        DES
java.base/sun.security.ssl.SSLCipher$T13GcmReadCipherGenerator$GcmReadCipher.<init>(SSLCipher.java:1858)   AES/GCM/NoPadding
java.base/sun.security.ssl.SSLCipher$T13GcmWriteCipherGenerator$GcmWriteCipher.<init>(SSLCipher.java:2012) AES/GCM/NoPadding
java.base/sun.security.ssl.SSLCipher.isTransformationAvailable(SSLCipher.java:510)                         AES/CBC/NoPadding,AES/GCM/N
org.apache.jsp.accessA_jsp._jspService(accessA_jsp.java:212)                                               AES
org.apache.jsp.accessA_jsp._jspService(accessA_jsp.java:213)                                               PBEWithMD5AndTripleDES
org.apache.jsp.accessB_jsp._jspService(accessB_jsp.java:212)                                               DES
org.apache.jsp.accessC_jsp._jspService(accessC_jsp.java:212)                                               DES/CBC/PKCS5Padding
org.apache.jsp.accessE_jsp._jspService(accessE_jsp.java:212)                                               DESede
org.apache.jsp.accessE_jsp._jspService(accessE_jsp.java:213)                                               AES
```

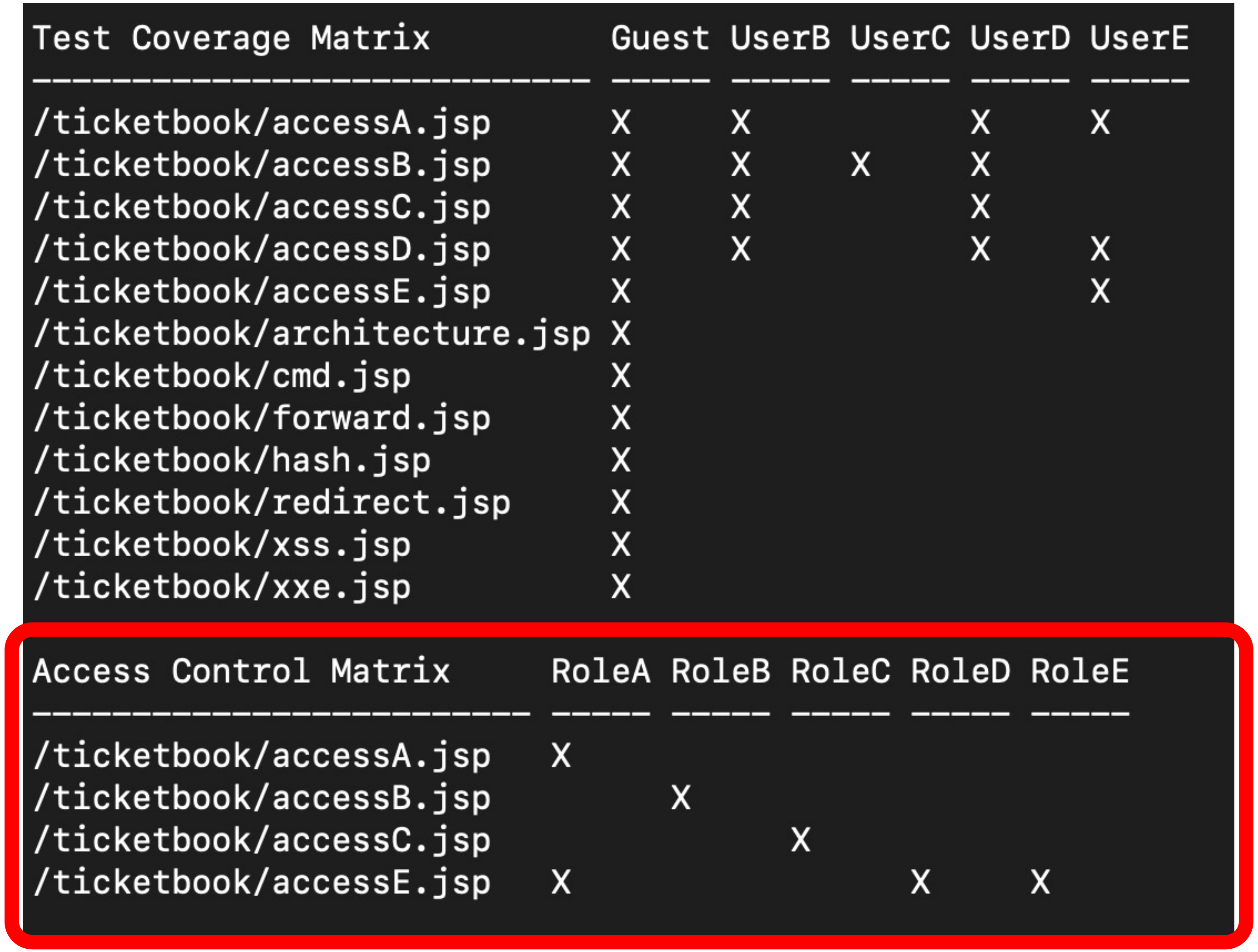
VERIFYING ACCESS CONTROL?



```
sensors:

- name: "get-routes"
  description: "Identifies the route for this HTTP request"
  methods:
  - "javax.servlet.Servlet.service"
  captures:
  - "#P0.getRequestURI()"

- name: "get-users"
  description: "Identifies user names"
  methods:
  - "javax.servlet.Servlet.service"
  captures:
  - "#P0.getRemoteUser() ?: \"Guest\""

- name: "get-role"
  description: "Identifies roles"
  methods:
  - "javax.servlet.ServletRequest.isUserInRole"
  captures:
  - "#P0"


reports:

 - name: "Test Coverage Matrix"
   type: "compare"
   rows: "get-routes"
   cols: "get-users"

 - name: "Access Control Matrix"
   type: "compare"
   rows: "get-routes"
   cols: "get-role"
```

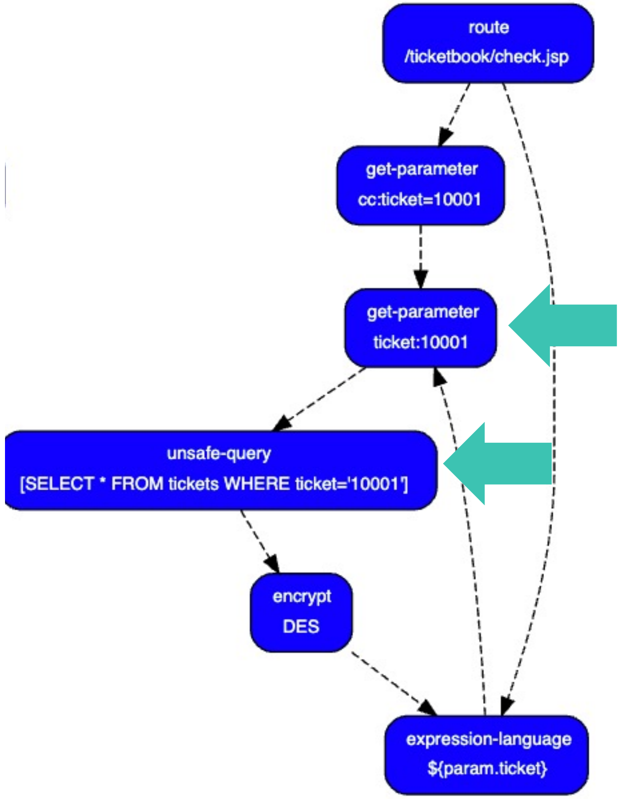


```
$ export JAVA_TOOL_OPTIONS="-javaagent:jot-0.9.1.jar=jots/access.jot"
```

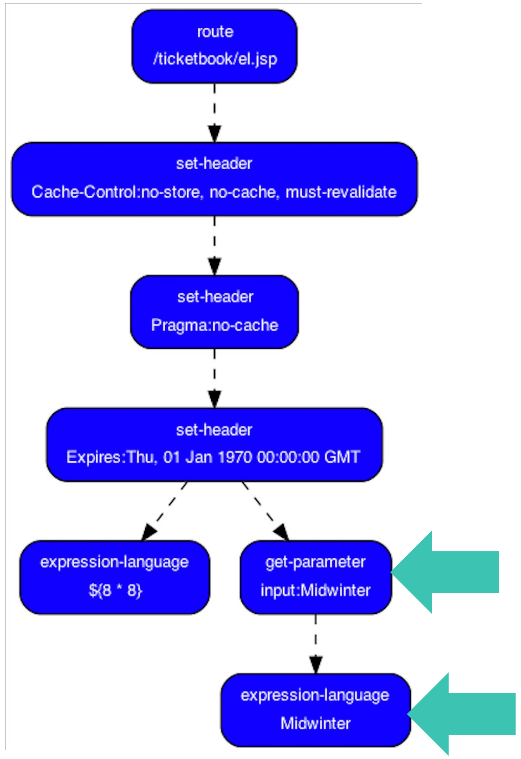


```
Test Coverage Matrix         Guest UserB UserC UserD UserE
---------------------------- ----- ----- ----- ----- -----
/ticketbook/accessA.jsp      X     X           X     X
/ticketbook/accessB.jsp      X     X     X     X
/ticketbook/accessC.jsp      X     X           X
/ticketbook/accessD.jsp      X     X           X     X
/ticketbook/accessE.jsp      X                       X
/ticketbook/architecture.jsp X
/ticketbook/cmd.jsp          X
/ticketbook/forward.jsp      X
/ticketbook/hash.jsp         X
/ticketbook/redirect.jsp     X
/ticketbook/xss.jsp          X
/ticketbook/xxe.jsp          X

Access Control Matrix     RoleA RoleB RoleC RoleD RoleE
------------------------- ----- ----- ----- ----- -----
/ticketbook/accessA.jsp   X
/ticketbook/accessB.jsp         X
/ticketbook/accessC.jsp               X
/ticketbook/accessE.jsp   X                 X     X
```

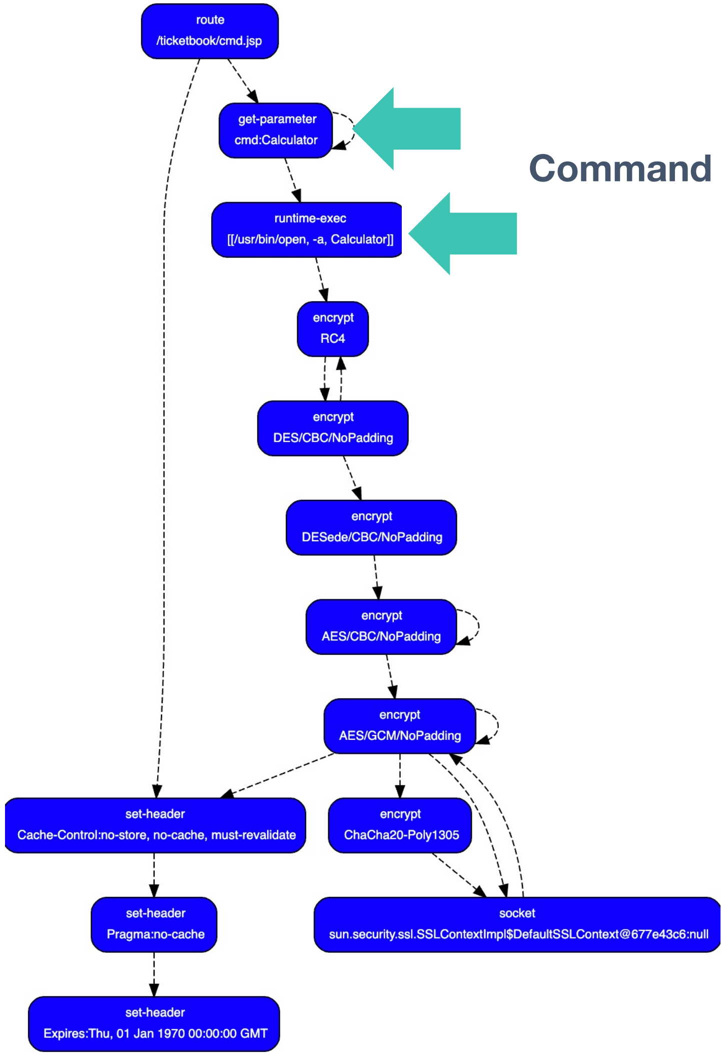
CHEATING AT PENTESTS FOR FUN AND PROFIT



SQL Injection

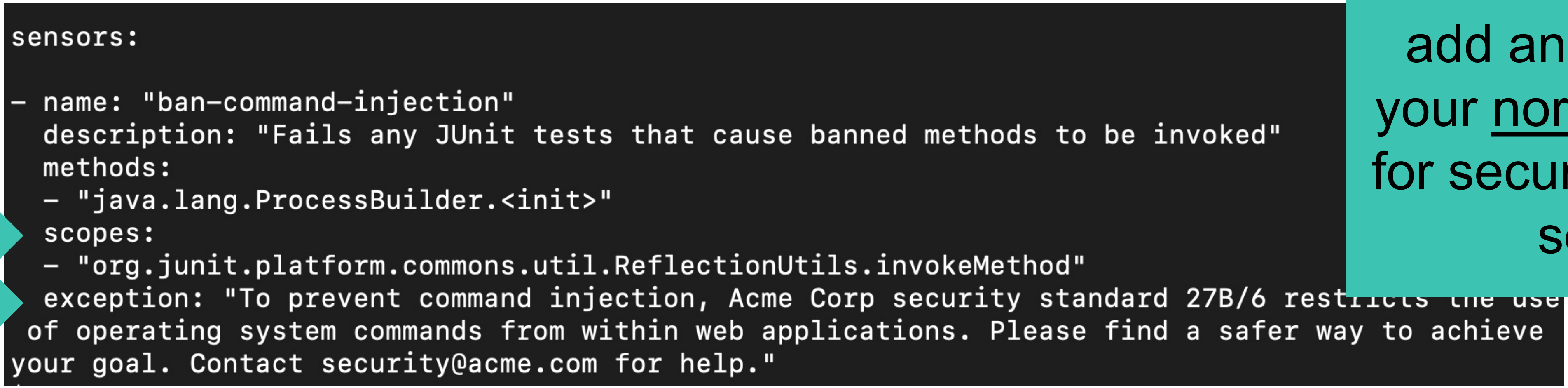


EL Injection



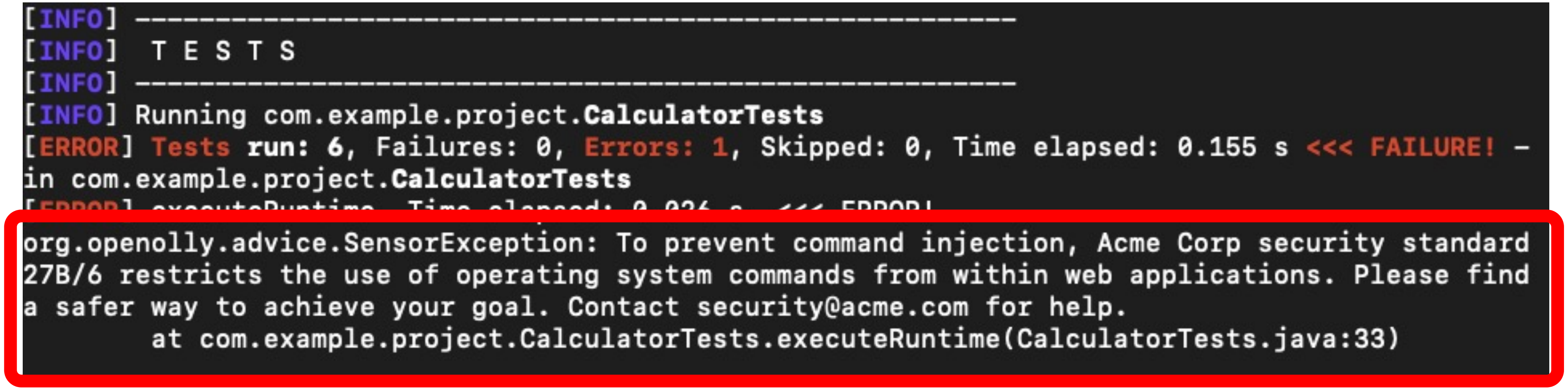
Command Injection

COMMUNICATING SECURITY VIA TEST FAILURES



```
sensors:

- name: "ban-command-injection"
  description: "Fails any JUnit tests that cause banned methods to be invoked"
  methods:
  - "java.lang.ProcessBuilder.<init>"
  scopes:
  - "org.junit.platform.commons.util.ReflectionUtils.invokeMethod"
  exception: "To prevent command injection, Acme Corp security standard 27B/6 restricts the use
 of operating system commands from within web applications. Please find a safer way to achieve
your goal. Contact security@acme.com for help."
```

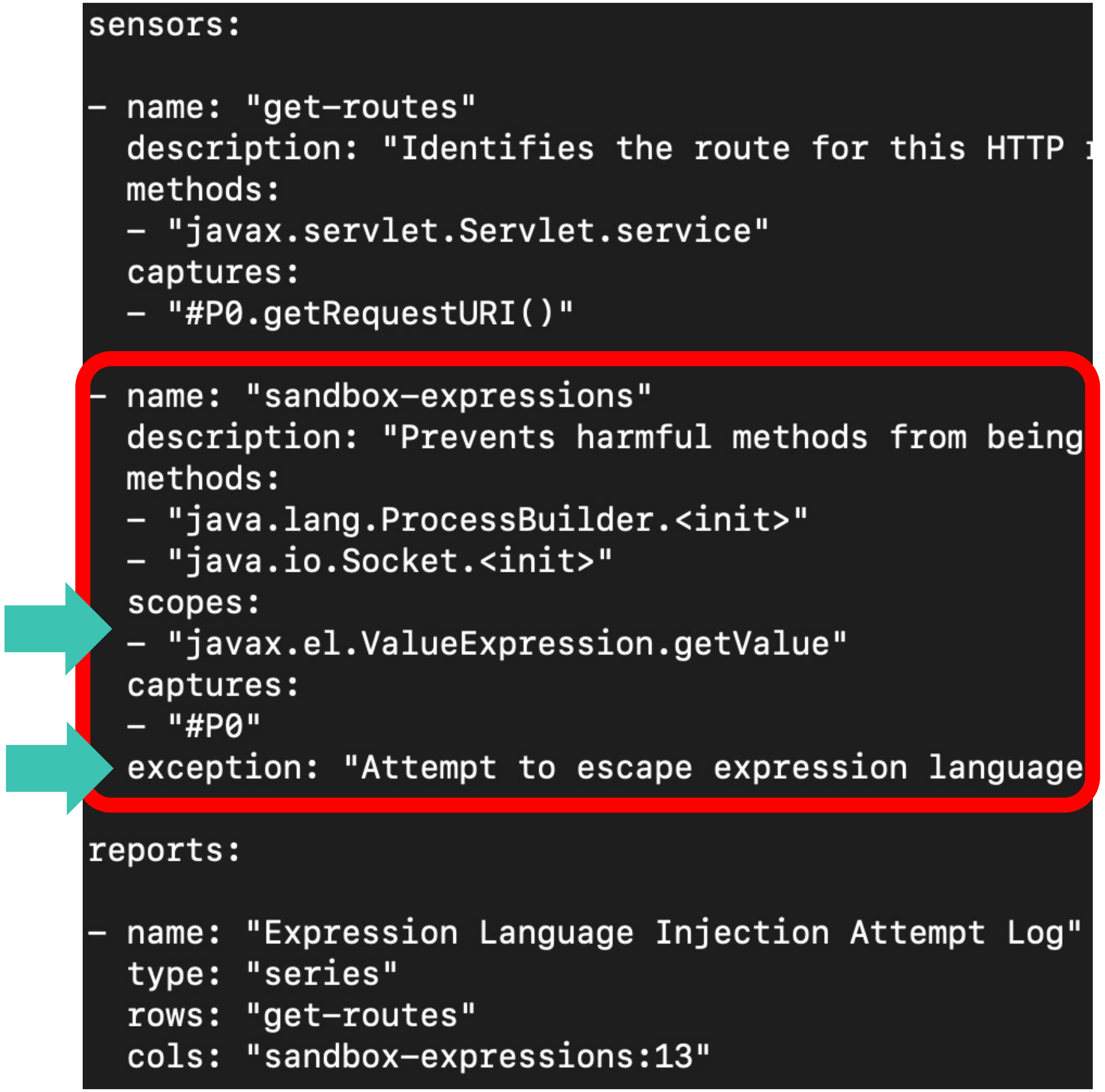
Define this "scope" and add an exception. Now your normal test cases fail for security reasons if your sensor fires!



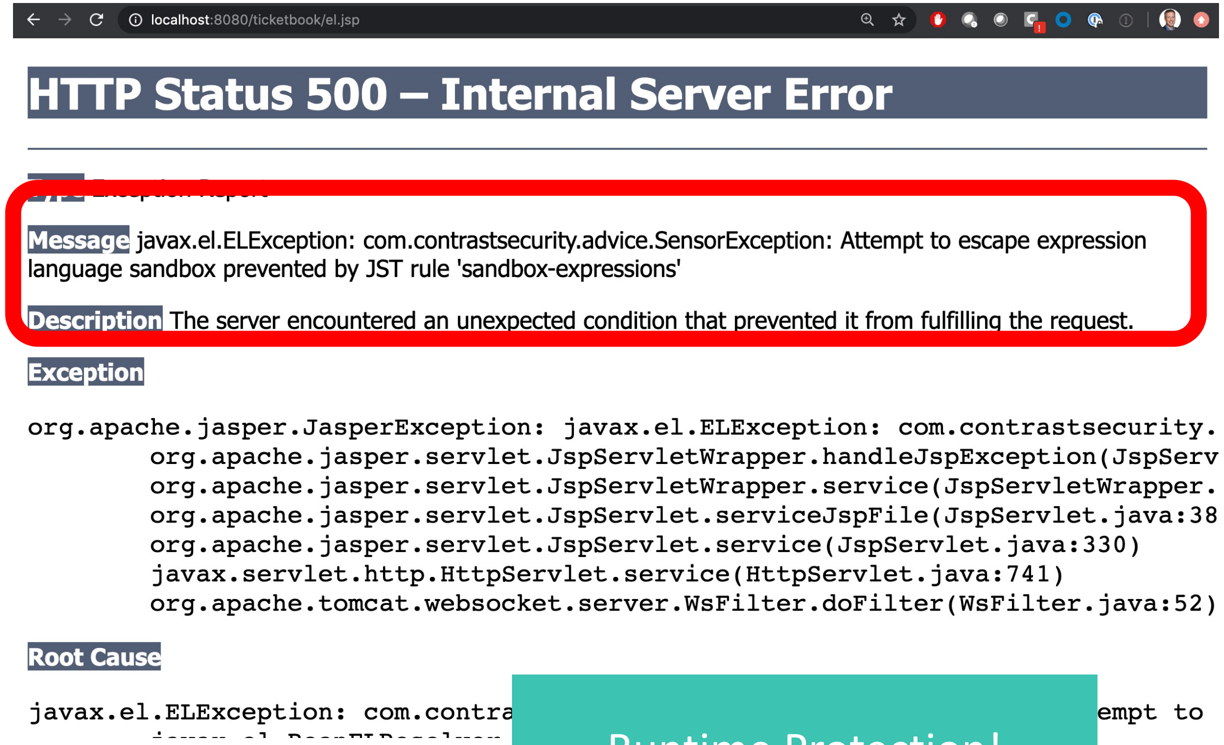
```
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running com.example.project.CalculatorTests
[ERROR] Tests run: 6, Failures: 0, Errors: 1, Skipped: 0, Time elapsed: 0.155 s <<< FAILURE! -
in com.example.project.CalculatorTests
[ERROR] executeRuntime  Time elapsed: 0.026 s  <<< ERROR!
org.openolly.advice.SensorException: To prevent command injection, Acme Corp security standard
27B/6 restricts the use of operating system commands from within web applications. Please find
a safer way to achieve your goal. Contact security@acme.com for help.
        at com.example.project.CalculatorTests.executeRuntime(CalculatorTests.java:33)
```

https://www.linkedin.com/pulse/developer-friendly-security-reporting-jeff-williams

HOW CAN I PREVENT EXPRESSION LANGUAGE INJECTION FROM BEING EXPLOITED?



```
sensors:

- name: "get-routes"
  description: "Identifies the route for this HTTP r
  methods:
  - "javax.servlet.Servlet.service"
  captures:
  - "#P0.getRequestURI()"

- name: "sandbox-expressions"
  description: "Prevents harmful methods from being
  methods:
  - "java.lang.ProcessBuilder.<init>"
  - "java.io.Socket.<init>"
  scopes:
  - "javax.el.ValueExpression.getValue"
  captures:
  - "#P0"
  exception: "Attempt to escape expression language

reports:

- name: "Expression Language Injection Attempt Log"
  type: "series"
  rows: "get-routes"
  cols: "sandbox-expressions:13"
```



HTTP Status 500 – Internal Server Error

Type Exception Report

Message javax.el.ELException: com.contrastsecurity.advice.SensorException: Attempt to escape expression language sandbox prevented by JST rule 'sandbox-expressions'

Description The server encountered an unexpected condition that prevented it from fulfilling the request.

Exception

```
org.apache.jasper.JasperException: javax.el.ELException: com.contrastsecurity.
        org.apache.jasper.servlet.JspServletWrapper.handleJspException(JspServ
        org.apache.jasper.servlet.JspServletWrapper.service(JspServletWrapper.
        org.apache.jasper.servlet.JspServlet.serviceJspFile(JspServlet.java:38
        org.apache.jasper.servlet.JspServlet.service(JspServlet.java:330)
        javax.servlet.http.HttpServlet.service(HttpServlet.java:741)
        org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:52)
```

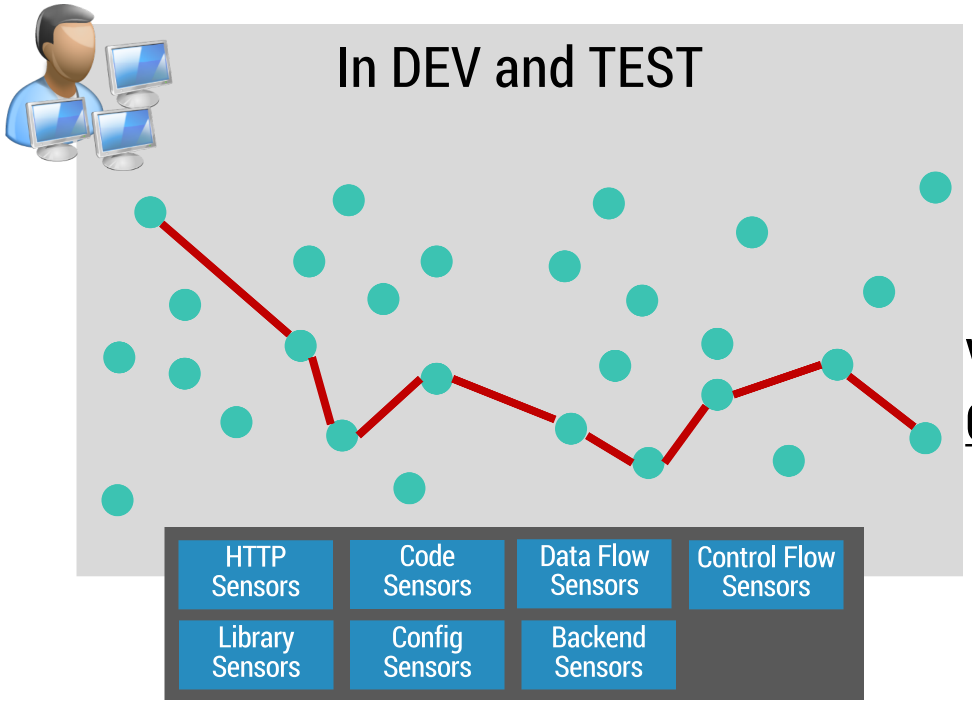
Root Cause

```
javax.el.ELException: com.contra                                  empt to
```

Runtime Protection!

IAST

Interactive Application Security Testing is simply using instrumentation to detect vulnerabilities.
USE IT IN DEVELOPMENT.



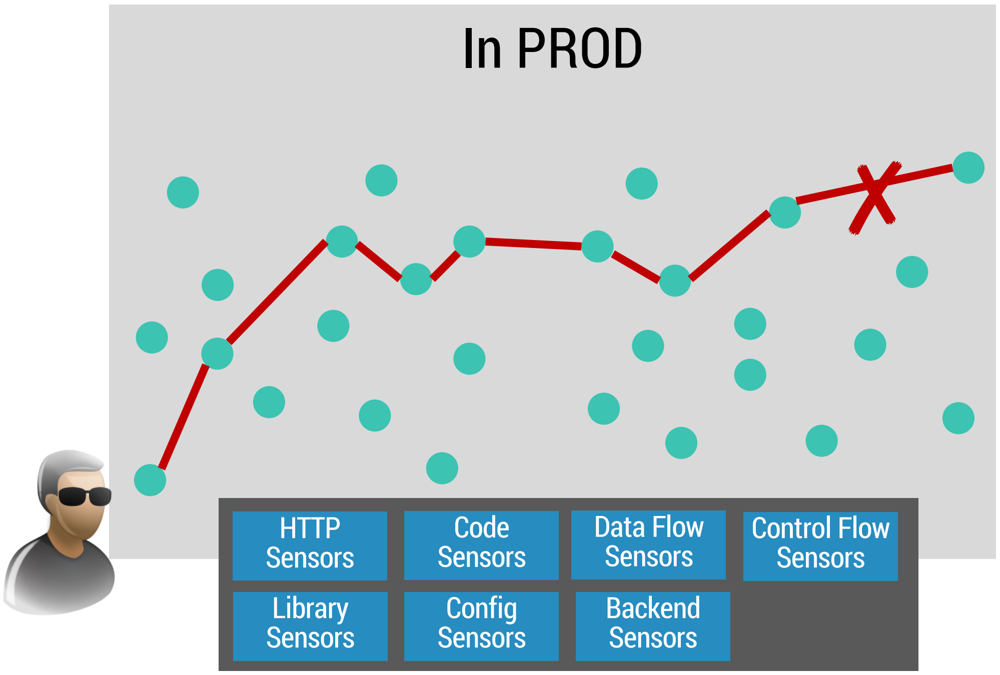
In DEV and TEST

HTTP Sensors
Code Sensors
Data Flow Sensors
Control Flow Sensors
Library Sensors
Config Sensors
Backend Sensors

Vulnerability Confirmed

RASP

Runtime Application Self-Protection is simply using instrumentation to detect attacks and prevent exploits.
USE IT IN PRODUCTION



In PROD

HTTP Sensors
Code Sensors
Data Flow Sensors
Control Flow Sensors
Library Sensors
Config Sensors
Backend Sensors

Exploit Prevented

THE MOVE TO MODERN

SOFTWARE SECURITY

LEGACY

SCAN AND FIREWALL MODEL

- Disruptive, bottleneck
- Can’t keep up, even with army of experts
- After the fact, inaccurate
- Snapshot in time
- Tool soup, security silos

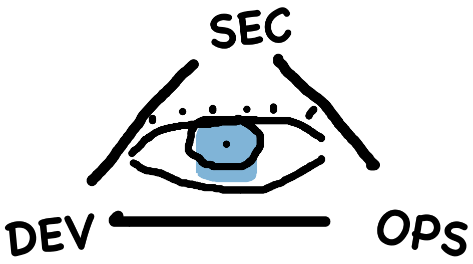
MODERN

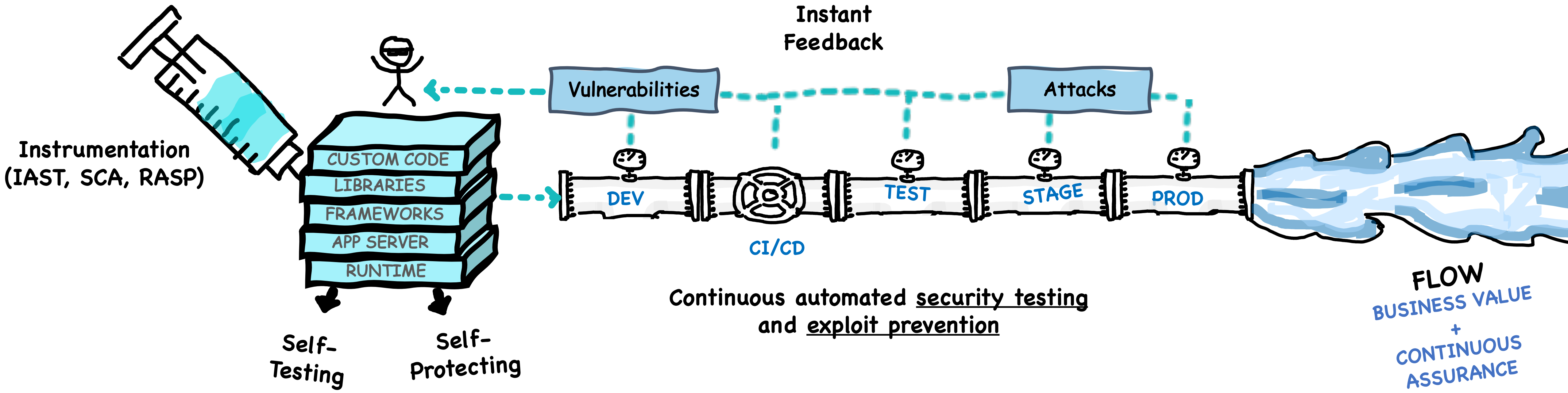
EMBEDDED MODEL

- Embedded, works in flow, frictionless
- Force multiplier, no experts required
- Direct observation, instant feedback
- Continuous, always-on
- One platform across dev, sec, ops

SECURITY OBSERVABILITY ACCELERATES INNOVATION

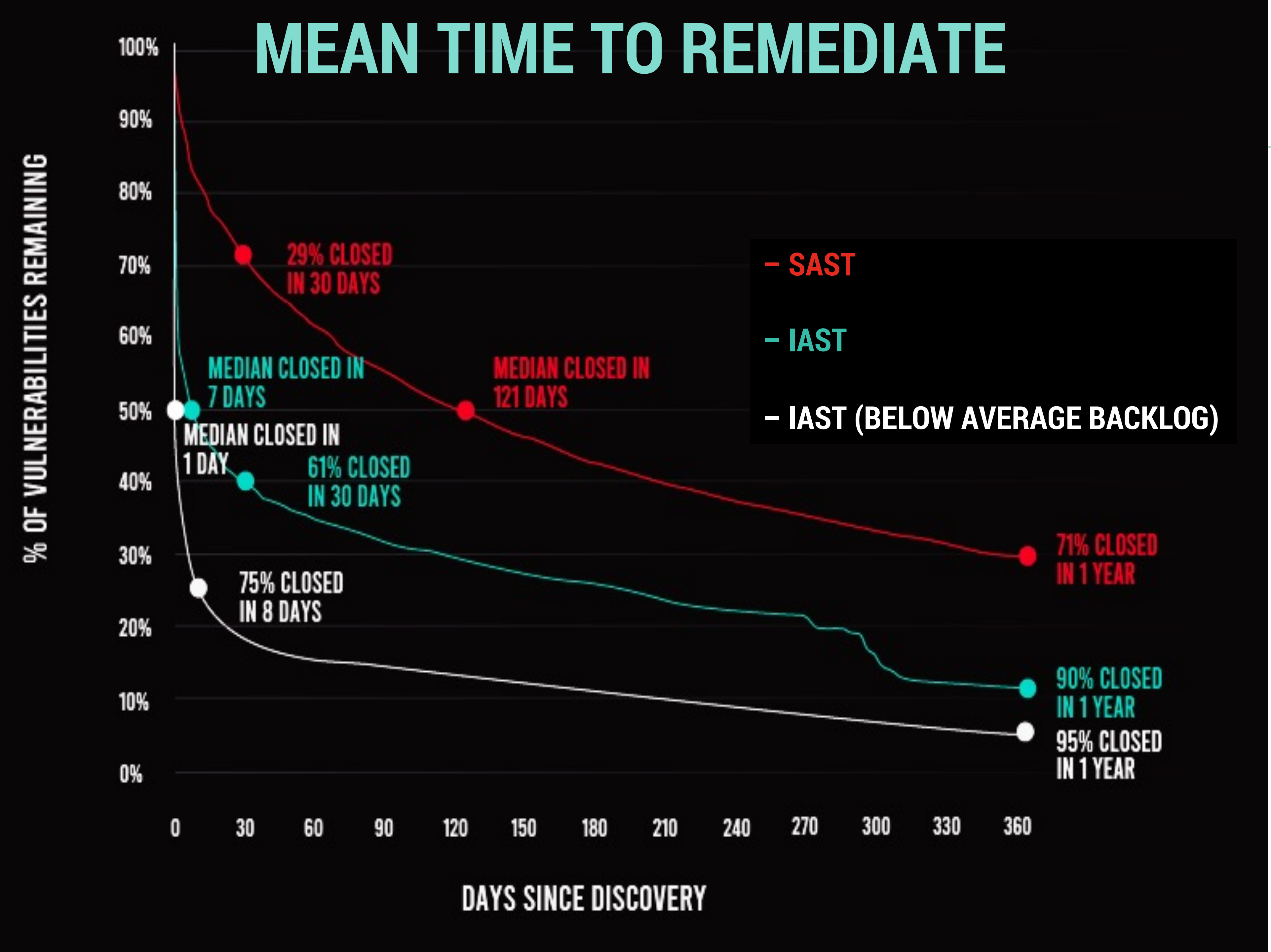
SECURITY OBSERVABILITY
(INSIDE - OUT)



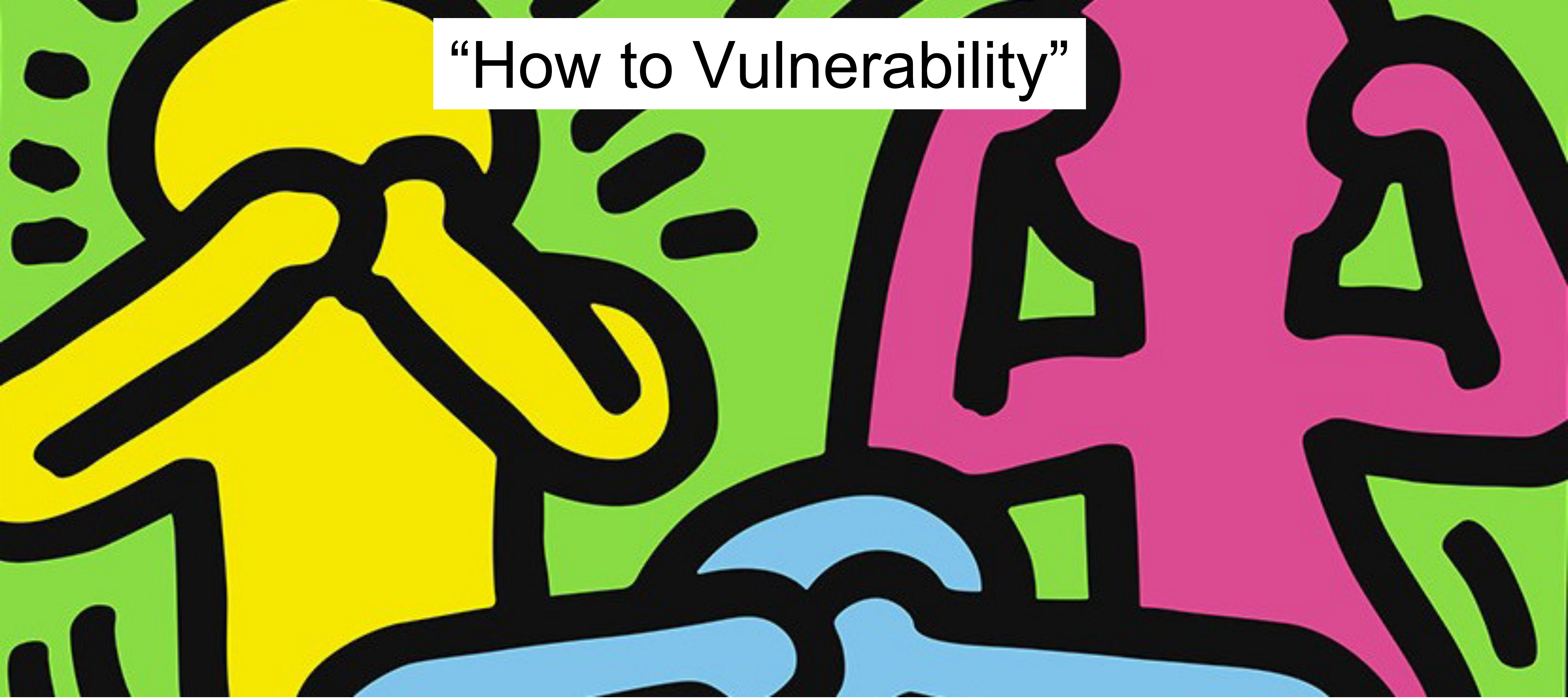


Continuous automated security testing and exploit prevention

MEAN TIME TO REMEDIATE



OBSERVABILITY YIELDS A **17X** IMPROVEMENT IN MTTR OVER SCANNING



“How to Vulnerability”

https://www.linkedin.com/pulse/how-vulnerability-jeff-williams

“Making Security in a Software Factory”

https://www.linkedin.com/pulse/making-security-software-factory-jeff-williams/

CONTRAST SECURITY

ASK ME ANYTHING

JEFF WILLIAMS, COFOUNDER AND CTO

@PLANETLEVEL

OWASP CHARLOTTE – OCT 2021