Secure defaults

developer-friendly security

Claudio Merloni
claudio@semgrep.com
https://linkedin.com/in/claudiomerloni

Semgrep

Secure defaults is NOT just...

Secure defaults is NOT just...

...having developers fix all security bugs

Secure defaults is NOT just...

...having developers fix all security bugs

...only fixing high priority issues

Secure defaults

make it easy to write secure code
make it hard to write insecure code

Security
team

Security
team
devs

Security researchers at Semgrep

semgrep

Pieter De Cremer

0xDC0DE

Claudio Merloni

p4p3r

Semgrep

Early adopters are doing this already

Netflix

https://www.youtube.com/watch?v=HIdexRqjpWc

Meta / Facebook

https://about.fb.com/news/2019/01/designing-security-for-billions/

Microsoft

https://www.acsac.org/2007/workshop/Howard.pdf

Google

https://sre.google/books/building-secure-reliable-systems/

Snowflake

https://semgrep.dev/blog/2021/appsec-development-keeping-it-all-together-at-scale

Semgrep

https://semgrep.dev/blog/2020/fixing-leaky-logs-how-to-find-a-bug-and-ensure-it-never-returns

And many more

Secure defaults

WHY	Security must scale
WHAT	The secure way, the easy way
WHO	Success stories
HOW	Think long term, high impact

Secure defaults

WHY	Security must scale
WHAT	The secure way, the easy way
WHO	Success stories
HOW	Think long term, high impact

Despite security automations, vulnerabilities are still prevalent

Every application

suffers from security issues throughout its lifetime

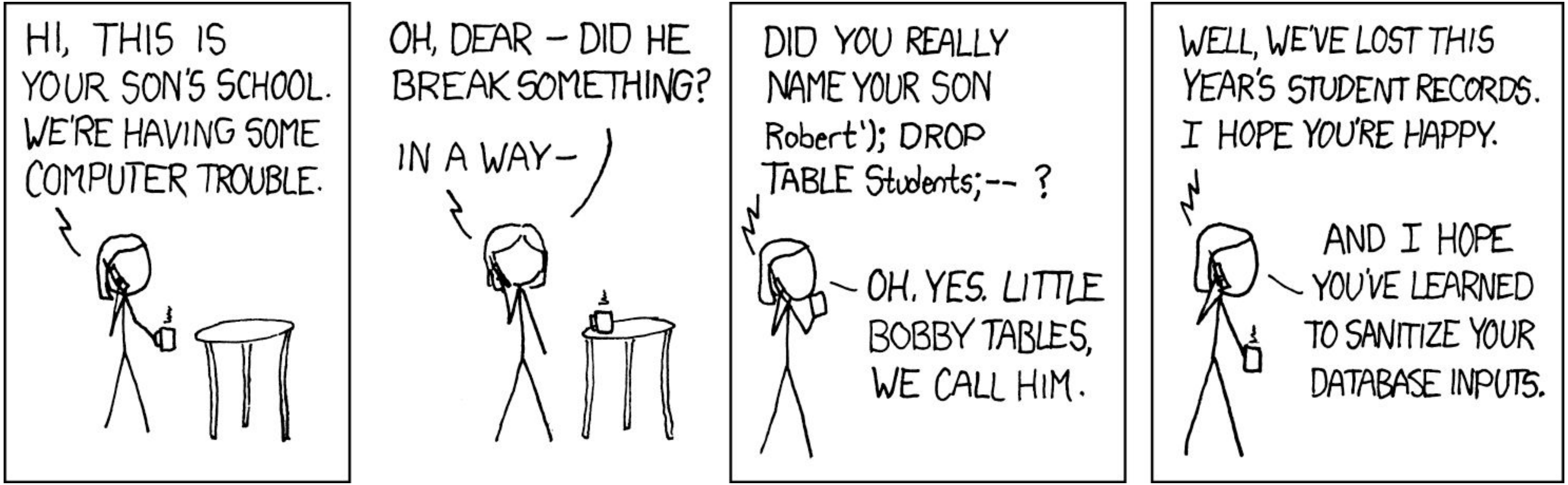
Problems in the underlying code

caused by developers

Not new problems

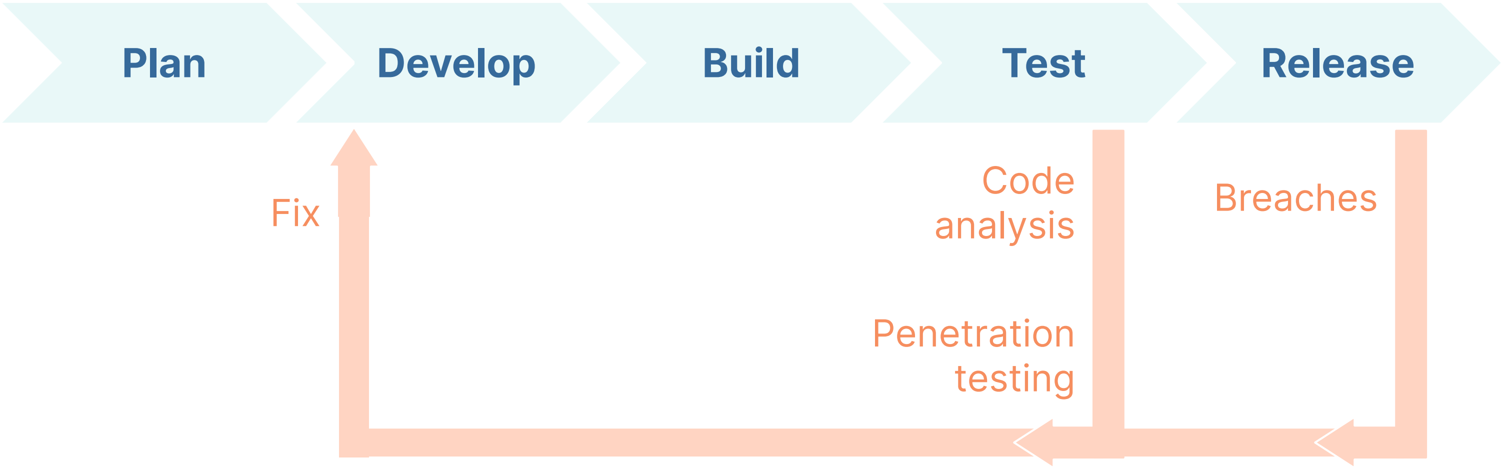
the same vulnerabilities exist for decades

Despite security automations, vulnerabilities are still prevalent


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

https://xkcd.com/327/

Traditional security tools were designed to be part of software testing



The development team and security team historically had an adversarial relationship

Different responsibilities

with sometimes conflicting goals

Friction

caused by lack of time
caused by lack of skill

AppSecCali 2019 - A Pragmatic Approach for Internal Security Partnerships - Netflix

Modern development practices require security teams to adapt

Automation benefits security

deploy fixed vulnerabilities faster

But makes our life more difficult

code changes constantly

Security testing is often too slow

biggest inhibitor to developer productivity

Balance complexity with speed

regex-based
linters

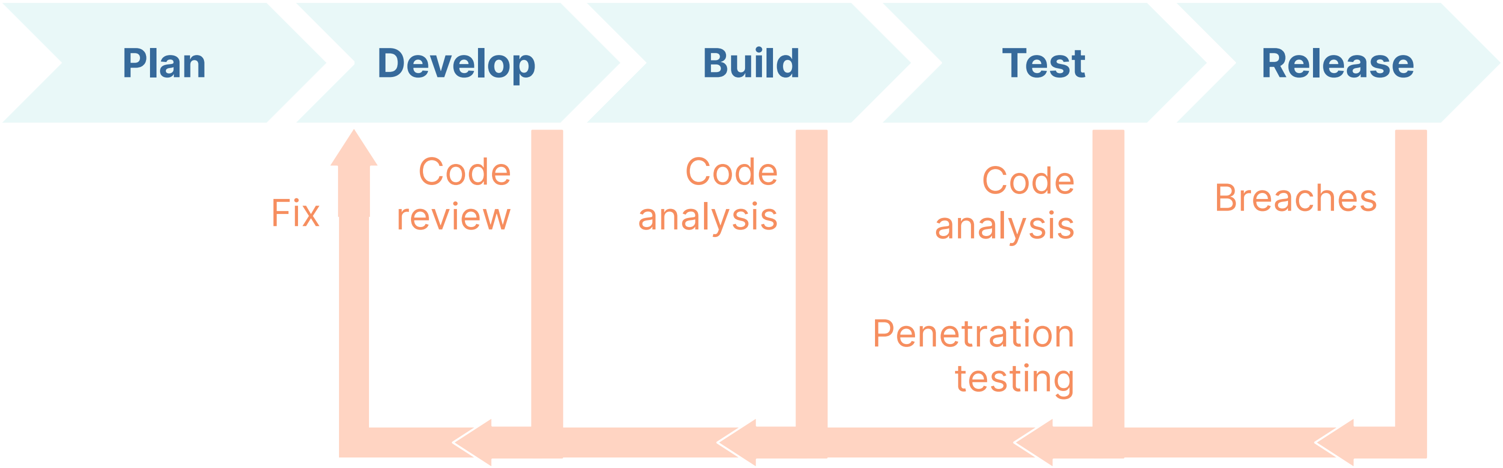
whole-program
static analysis

fast and easy
but dumb

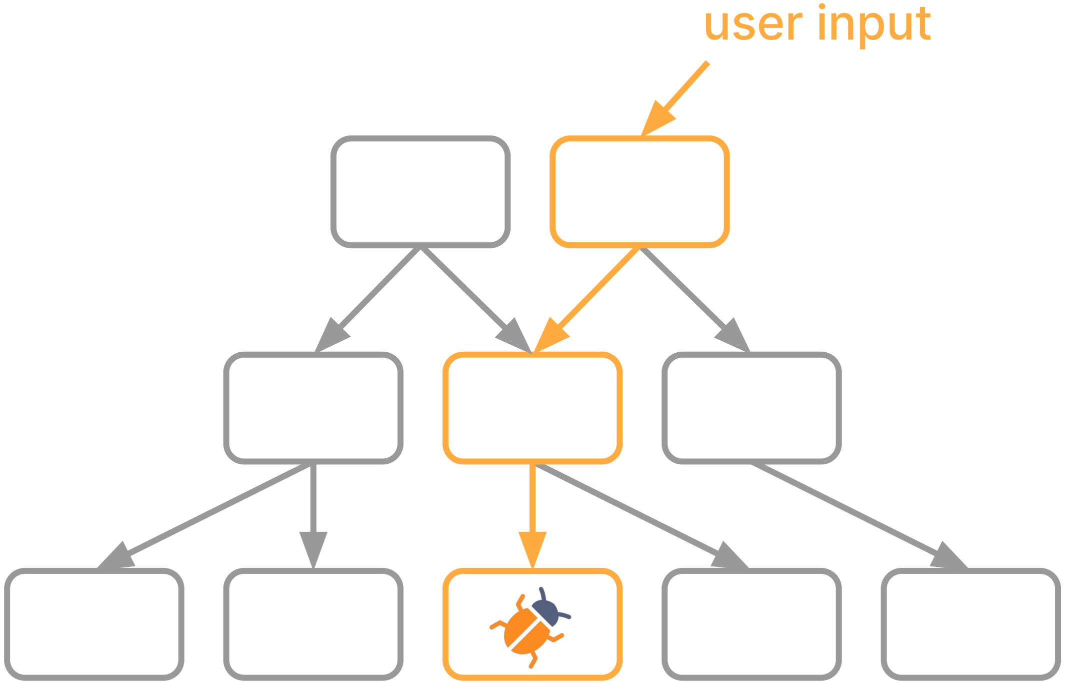
powerful and complex
but slow

https://instagram-engineering.com/static-analysis-at-scale-an-instagram-story-8f498ab71a0c

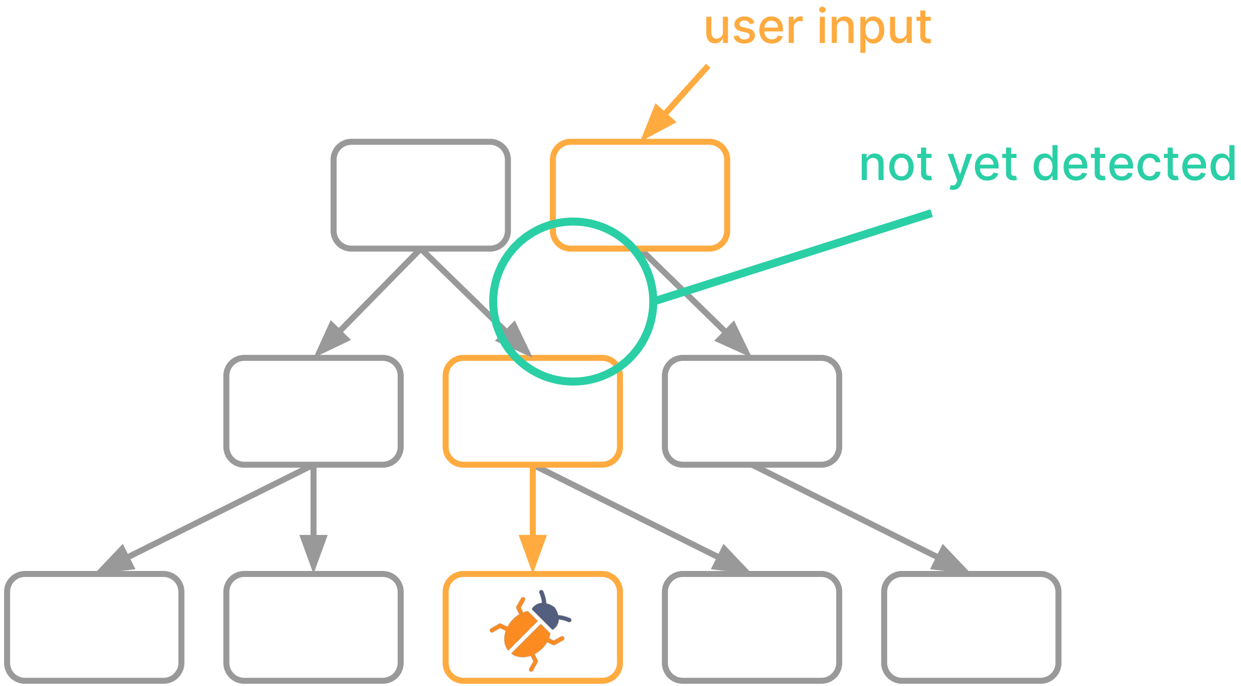
A shift-left movement is ongoing to address security earlier in development



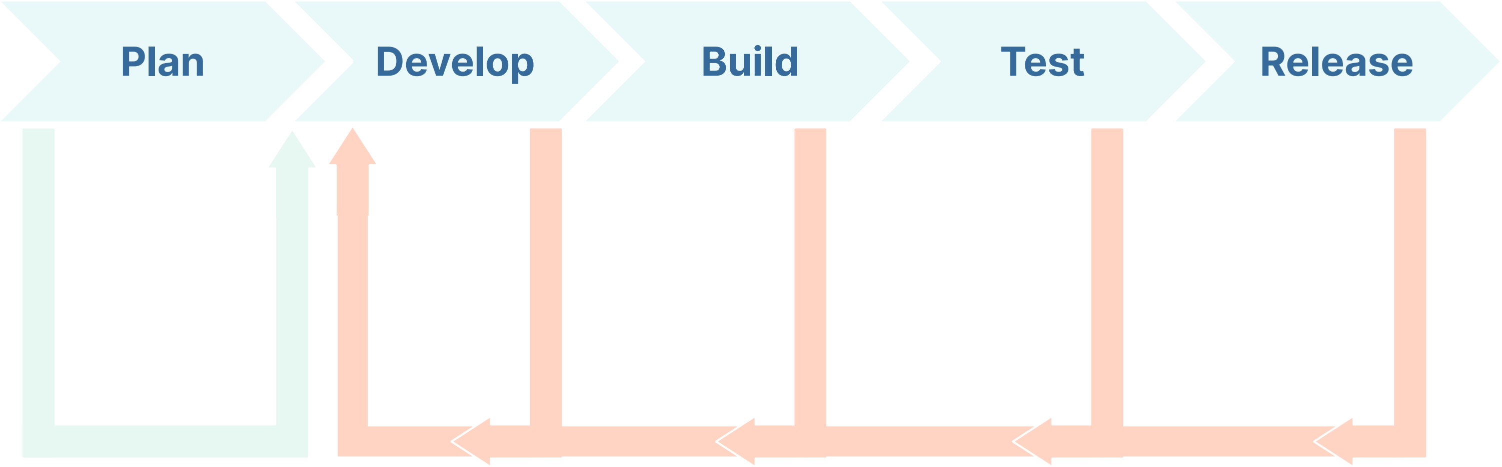
Traditional security tools use a reactive approach



Traditional security tools use a reactive approach



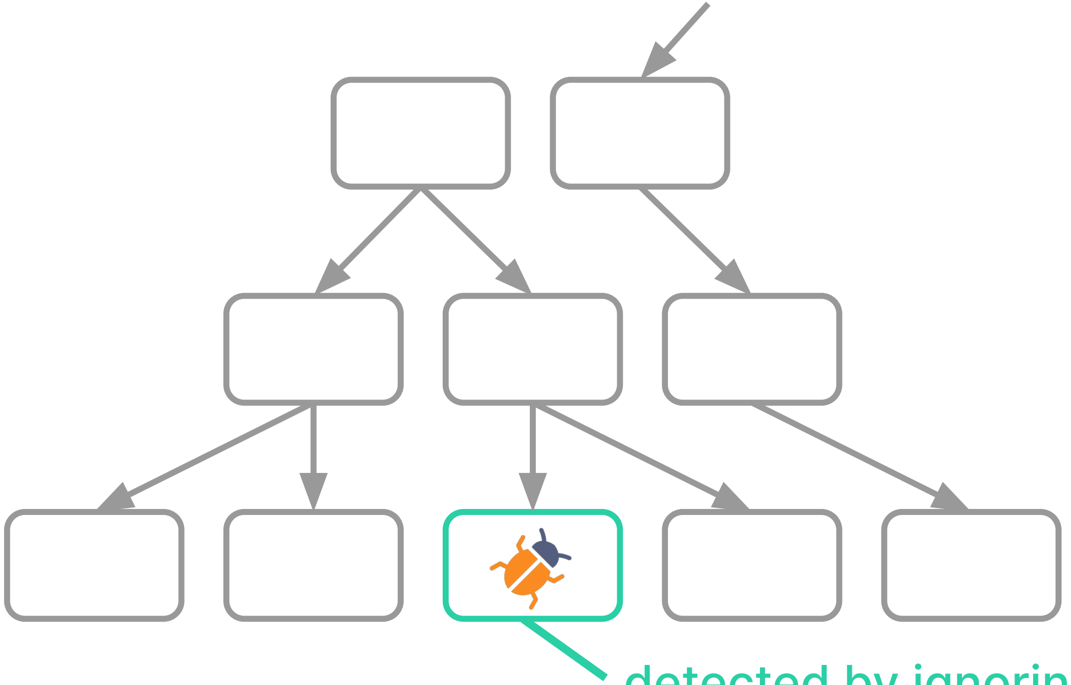
Security teams should be enablers



Prevent & fix vulnerabilities

Detect vulnerabilities

With secure defaults we can be more proactive



detected by ignoring context

They should provide developers with role-specific tools

Relevant
to the developer’s work

Efficient
in meeting the developer’s needs

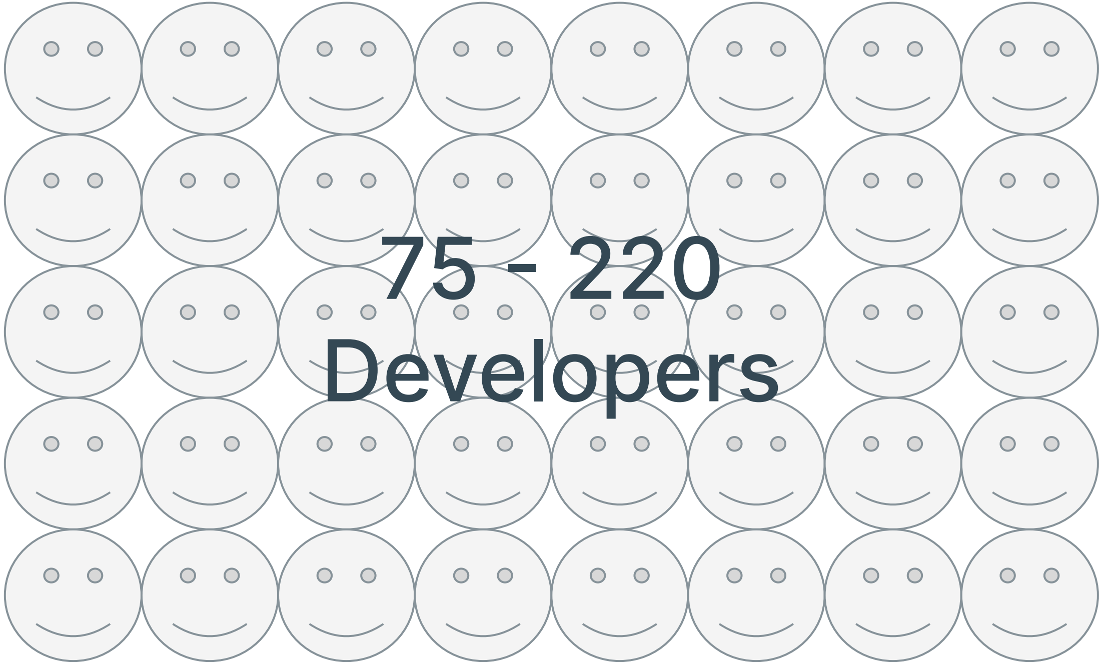
Usable
and well-integrated into the developer’s workflow

Secure defaults

WHY	Security must scale
WHAT	The secure way, the easy way
WHO	Success stories
HOW	Think long term, high impact

The security team is responsible for finding vulnerabilities in the software

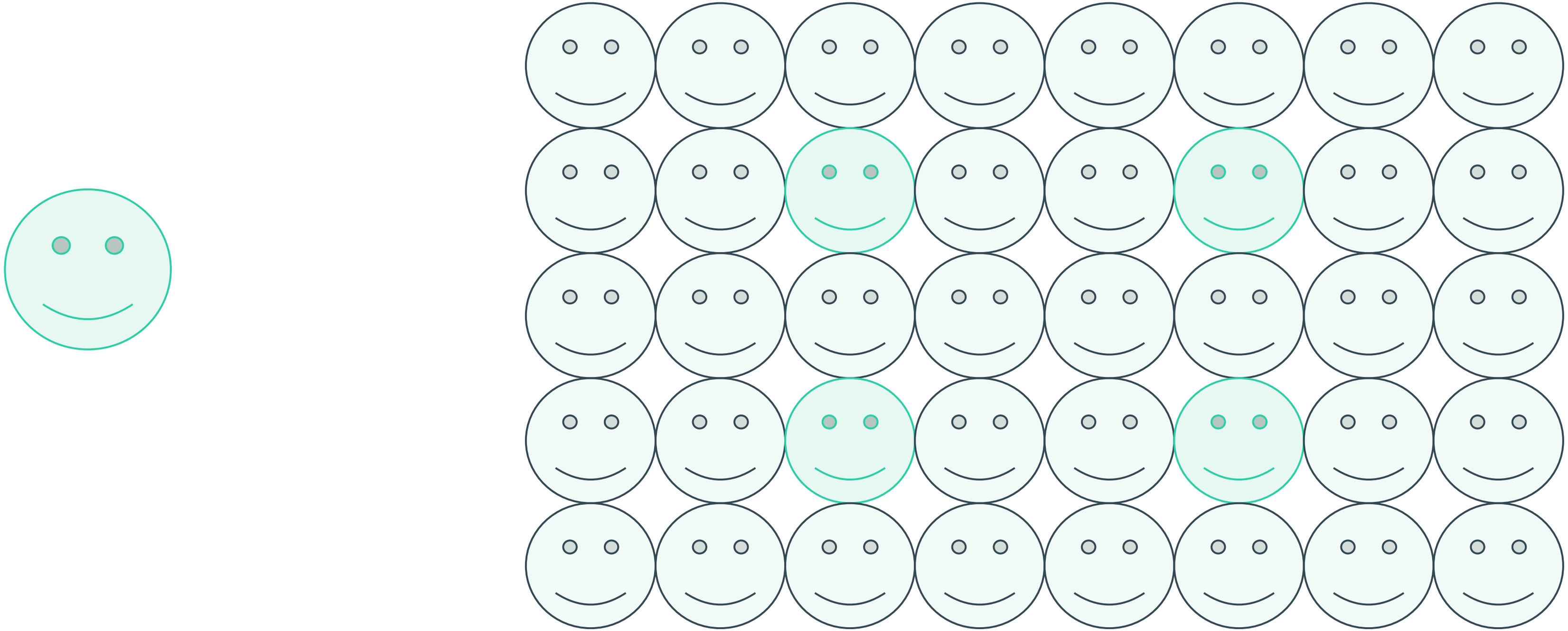
1
AppSec



75 - 220
Developers

[APPSEC Cali 2018 - We Come Bearing Gifts: Enabling Product Security](#)

Security should become a shared responsibility



Shared responsibility means shared goals

Ship features fast
what developers care about

Prevent and fix vulnerabilities
what security people care about

Improving one at the detriment of the other
is not real improvement

Security is not special
Plan and scope it with the rest of the work

To make secure code more scalable, we can learn from the DevOps movement

Before: Operators responsible

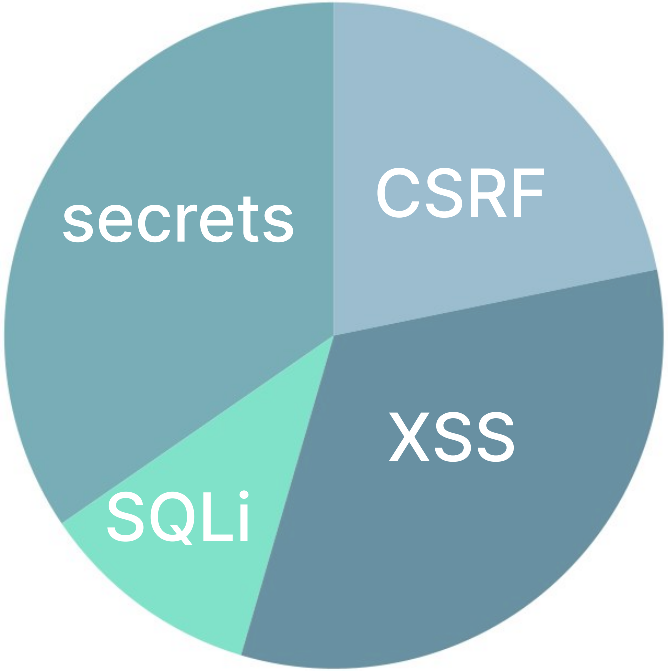
developers throw finished code over the wall

After: Self-service deployment

with CICD pipeline and infrastructure as code

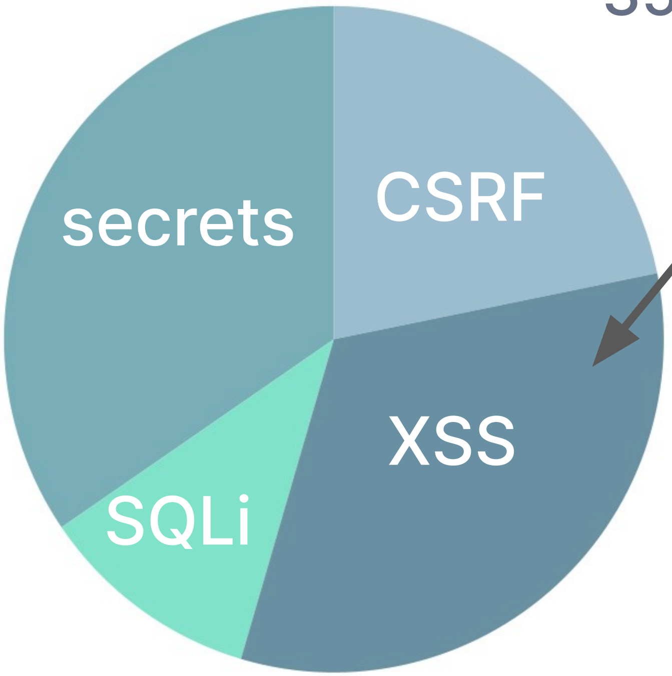
Eliminate bug classes one at a time

AppSec time spent



Eliminate bug classes one at a time

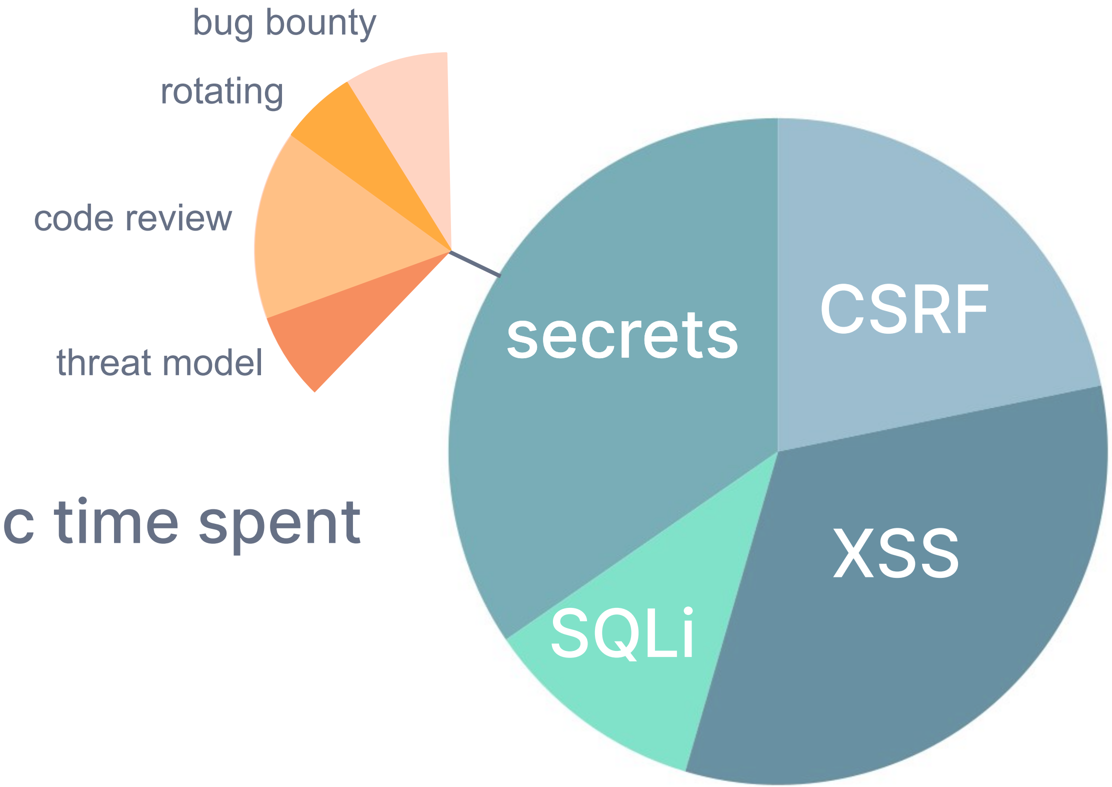
AppSec time spent



35% at Google!

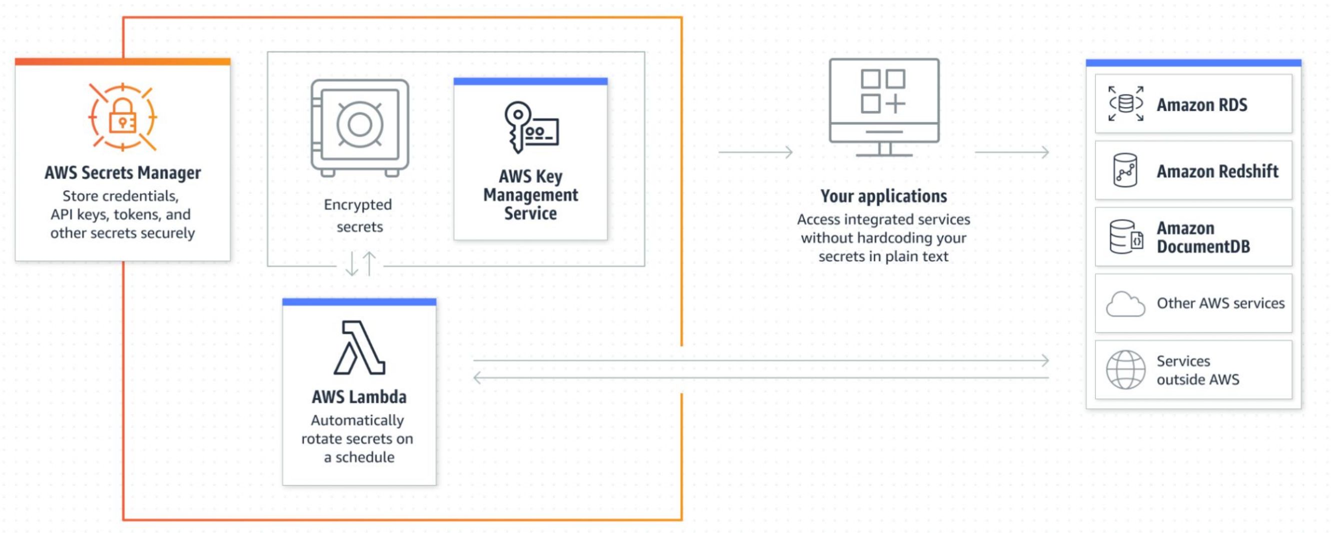
https://handouts.secappdev.org/handouts/2023/lukasweichselbaum_modern-security-features-for-web-apps.pdf

Eliminate bug classes one at a time



AppSec time spent

Example 1: secrets must be stored in AWS


AWS Secrets Manager
Store credentials, API keys, tokens, and other secrets securely
Encrypted secrets
AWS Key Management Service
AWS Lambda
Automatically rotate secrets on a schedule
Your applications
Access integrated services without hardcoding your secrets in plain text
Amazon RDS
Amazon Redshift
Amazon DocumentDB
Other AWS services
Services outside AWS

Example 1: secrets must be stored in AWS

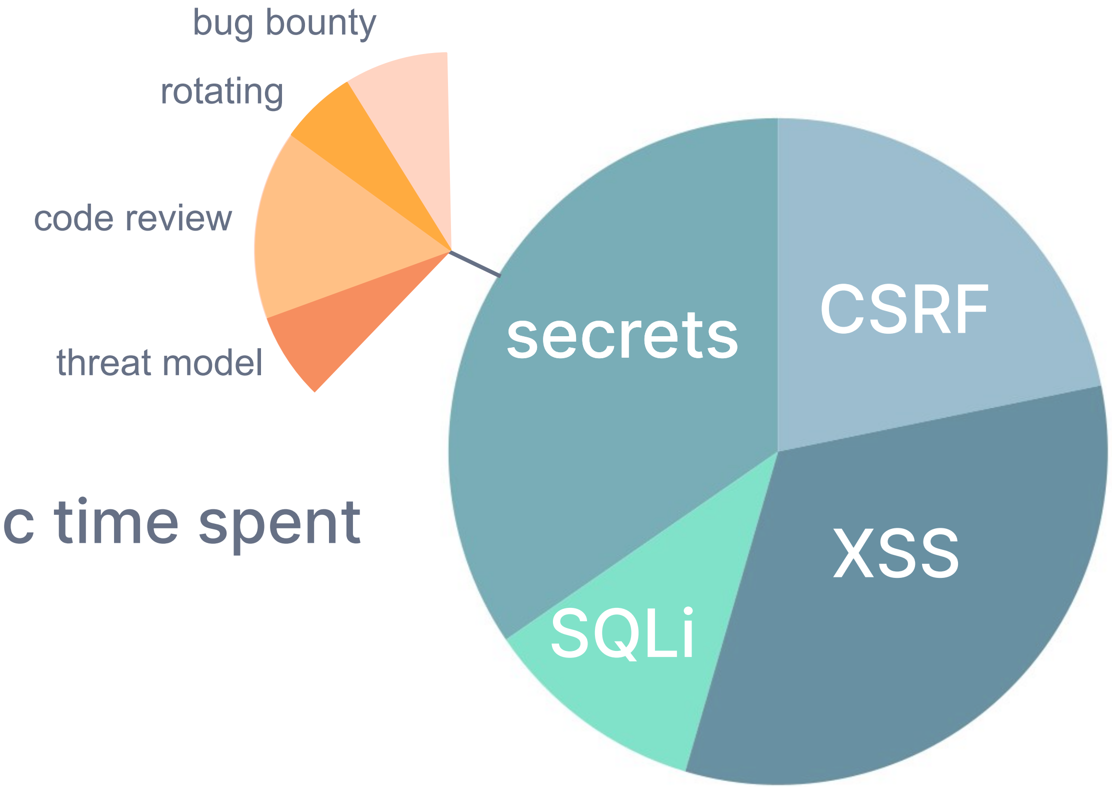
Python

```python
response = client.get_secret_value(
    SecretId='MyTestDatabaseSecret',
)
print(response)
```

Java

```java
private final SecretCache cache  = new SecretCache();

@Override public String handleRequest(String secretId,  Context c) {
final String secret  = cache.getSecretString(secretId);
    System.out.println(secret);
}
```

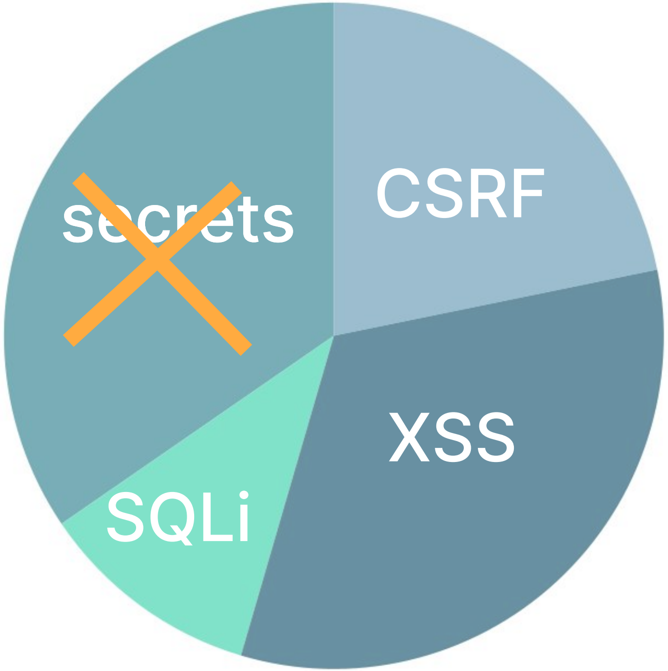
Eliminate bug classes one at a time



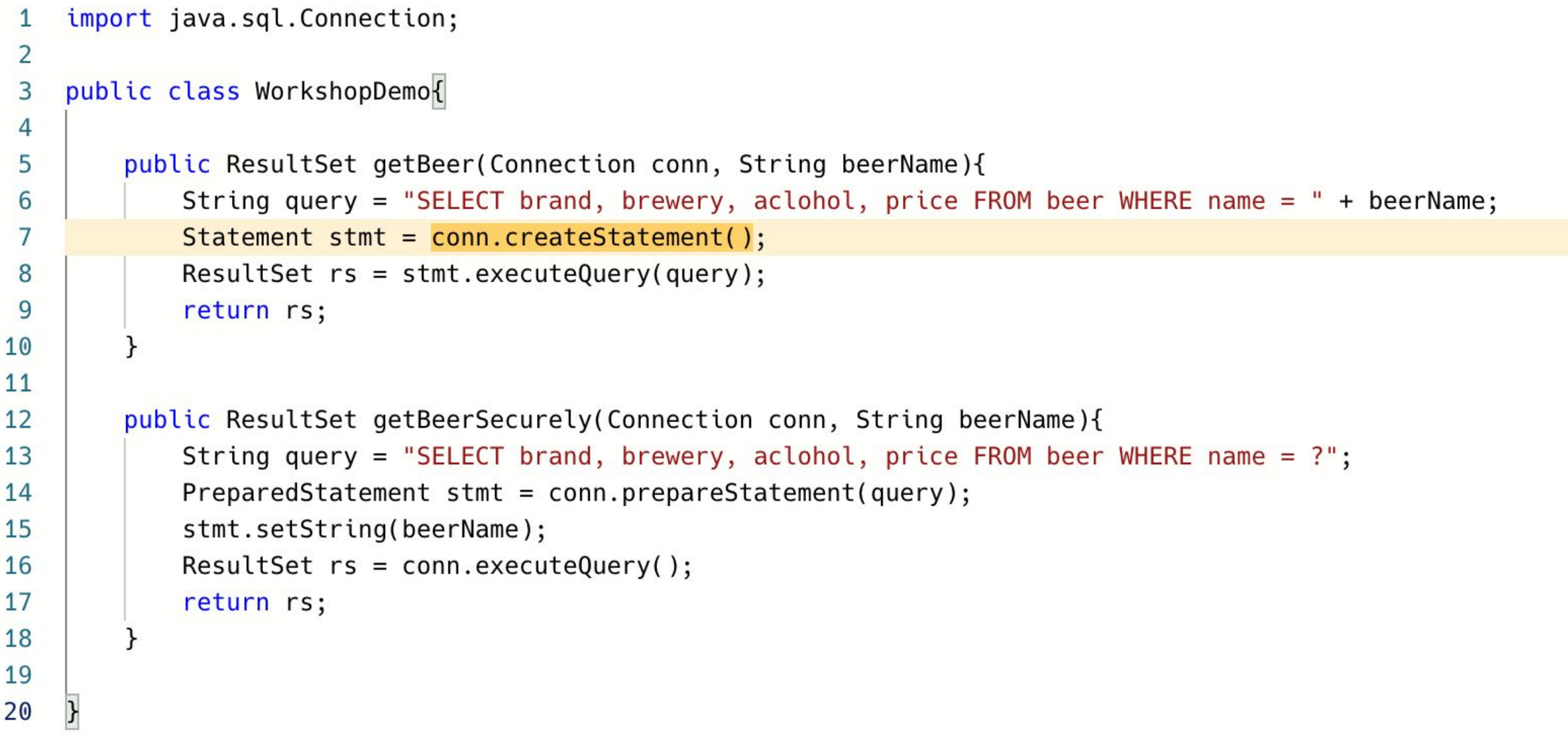
AppSec time spent

Killing bug classes leads to compounding effects to leverage your time better

AppSec time spent



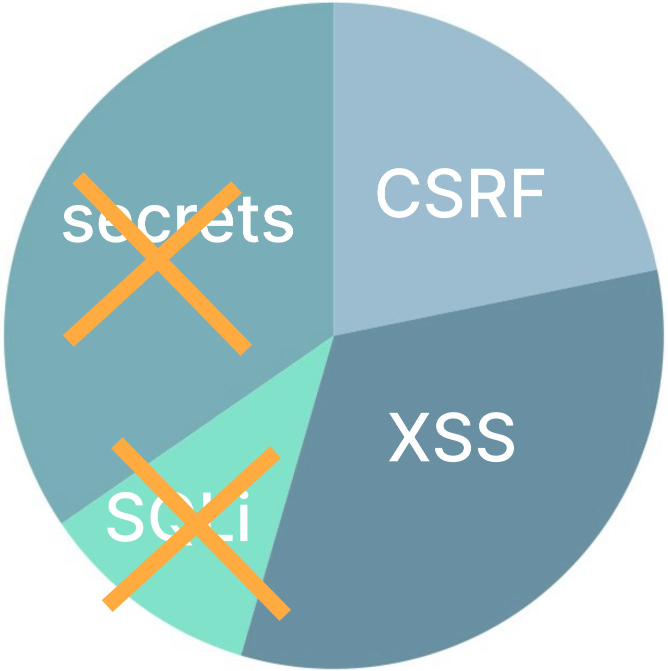
Example 2: queries must be parameterized



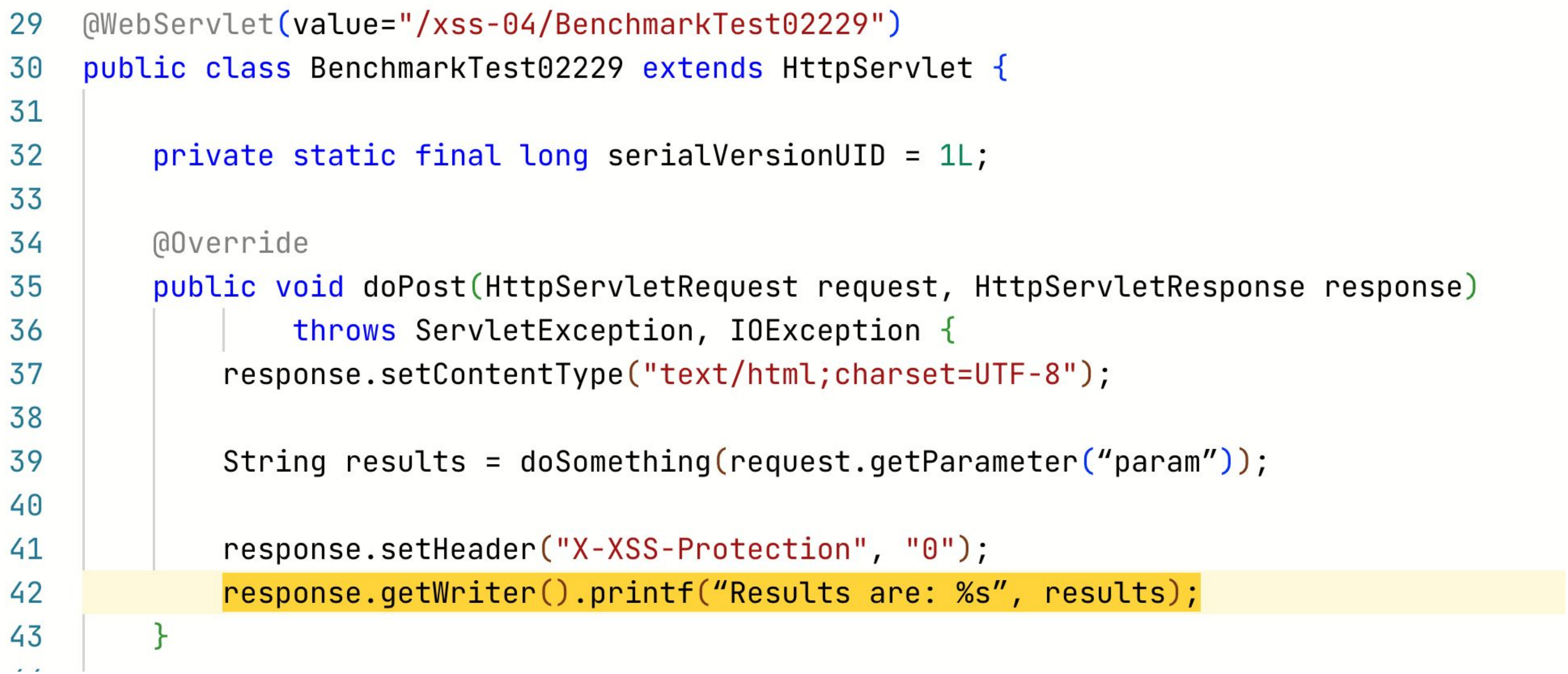
```java
import java.sql.Connection;

public class WorkshopDemo{

    public ResultSet getBeer(Connection conn, String beerName){
        String query = "SELECT brand, brewery, aclohol, price FROM beer WHERE name = " + beerName;
        Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery(query);
        return rs;
    }

    public ResultSet getBeerSecurely(Connection conn, String beerName){
        String query = "SELECT brand, brewery, aclohol, price FROM beer WHERE name = ?";
        PreparedStatement stmt = conn.prepareStatement(query);
        stmt.setString(beerName);
        ResultSet rs = conn.executeQuery();
        return rs;
    }

}
```

Killing bug classes leads to compounding effects to leverage your time better

AppSec time spent



Example 3: no direct response writer

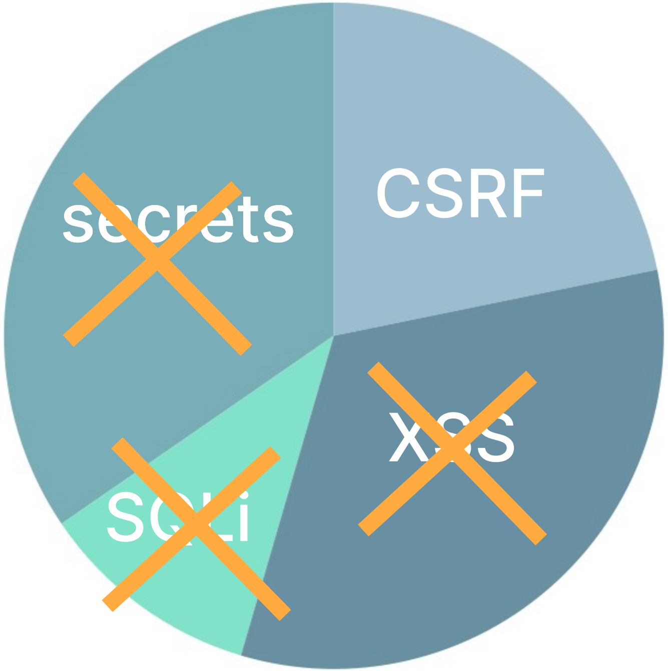


```java
@WebServlet(value="/xss-04/BenchmarkTest02229")
public class BenchmarkTest02229 extends HttpServlet {

    private static final long serialVersionUID = 1L;

    @Override
    public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");

        String results = doSomething(request.getParameter(“param”));

        response.setHeader("X-XSS-Protection", "0");
        response.getWriter().printf(“Results are: %s”, results);
    }
```

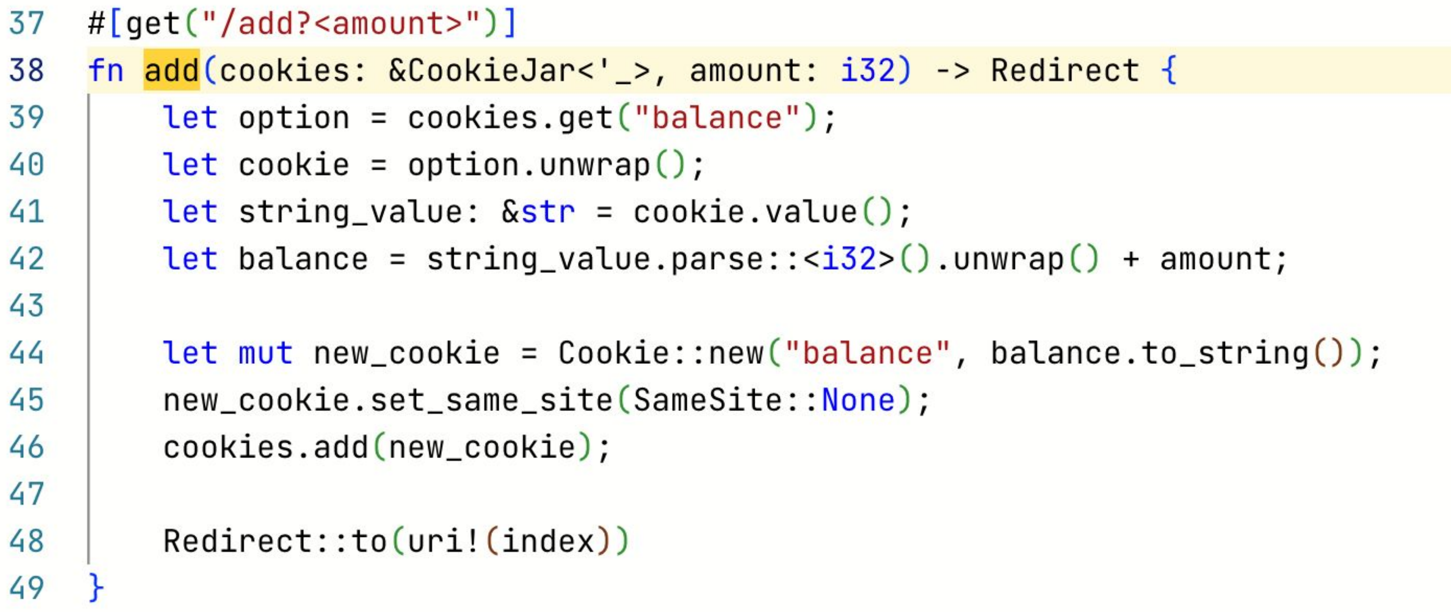
Solution: Use framework like JavaServer Faces (JSF) instead

Killing bug classes leads to compounding effects to leverage your time better

AppSec time spent



Example 4: Prevent interaction with cross-origin endpoints



```rust
#[get("/add?<amount>")]
fn add(cookies: &CookieJar<'_>, amount: i32) -> Redirect {
    let option = cookies.get("balance");
    let cookie = option.unwrap();
    let string_value: &str = cookie.value();
    let balance = string_value.parse::<i32>().unwrap() + amount;

    let mut new_cookie = Cookie::new("balance", balance.to_string());
    new_cookie.set_same_site(SameSite::None);
    cookies.add(new_cookie);

    Redirect::to(uri!(index))
}
```

Solution: Use Sec Fetch headers

https://www.youtube.com/watch?v=vxz2eK9y0I4&t=1s

Secure defaults

WHY	Security must scale
WHAT	The secure way, the easy way
WHO	Success stories
HOW	Think long term, high impact

What does success look like?

Classes of security risk eliminated

Average time to find and fix reduced

Average severity reduced

Bug bounty costs reduced

How Netflix does secure defaults

Netflix Culture Meets Product Security | by Bryan D. Payne | Medium
The Paved Road at Netflix
APPSEC Cali 2018 - We Come Bearing Gifts: Enabling Product Security
Scaling Appsec at Netflix. By Astha Singhal
AppSecCali 2019 - A Pragmatic Approach for Internal Security Partnerships
The Show Must Go On: Securing Netflix Studios At Scale
Scaling Appsec at Netflix (Part 2) | by Netflix Technology Blog

How Netflix does secure defaults

In-house consulting

no long-term relationships, no clear priorities

Per-app assessment does not scale

actionable self-service is important

Scaling Appsec at Netflix. By Astha Singhal

How Netflix does secure defaults

Context, not control

not required, recommended

Partnerships

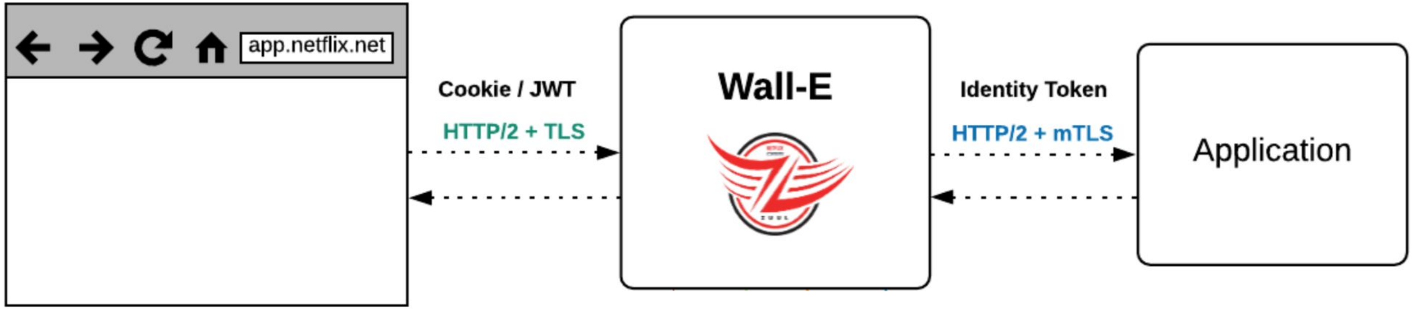
invest in paved road together with the consuming team

APPSEC Cali 2018 - We Come Bearing Gifts: Enabling Product Security with Culture and Cloud

How Netflix does secure defaults

Missing or incomplete authentication

most critical type of issue they regularly faced

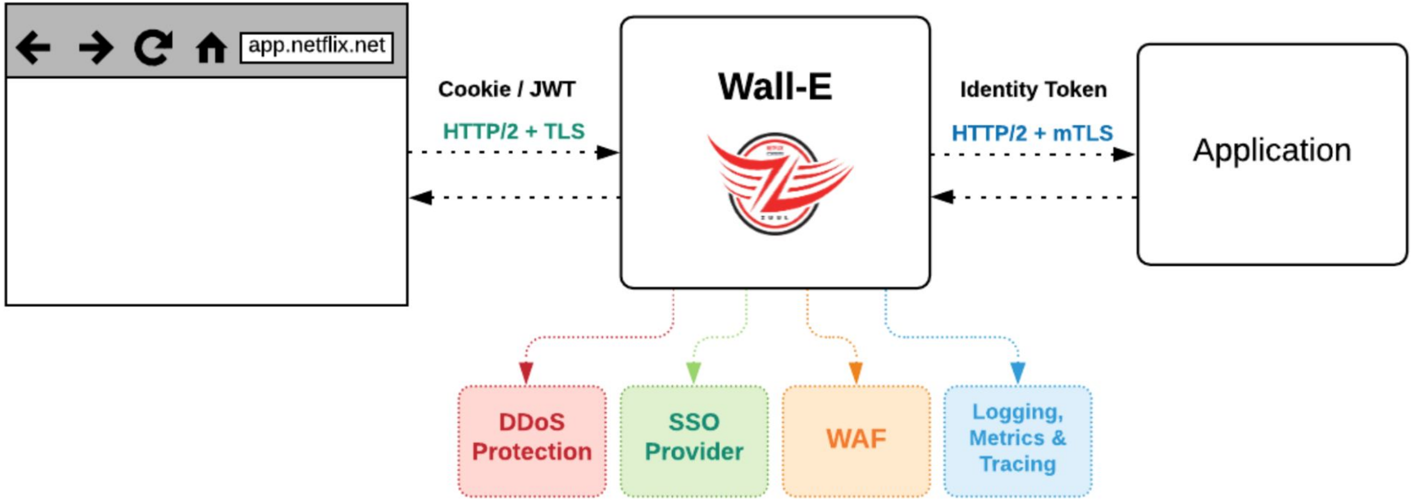


The Show Must Go On: Securing Netflix Studios At Scale

How Netflix does secure defaults

No organic adoption

until other features were added



The Show Must Go On: Securing Netflix Studios At Scale

How Netflix does secure defaults

Paved road simplifies reviews

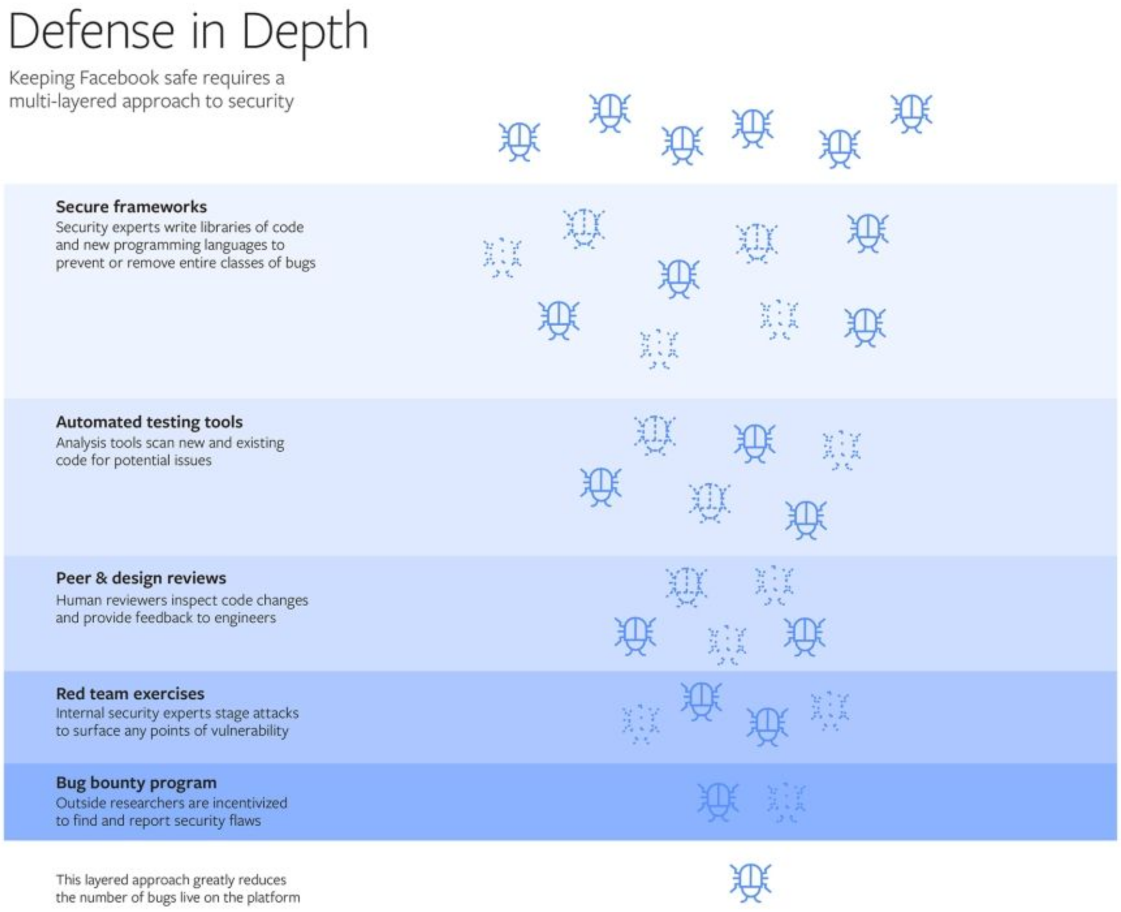
are you using it or not?

Security was not the main motivation

the secure default allowed developers to move faster

The Show Must Go On: Securing Netflix Studios At Scale

How Meta / Facebook does secure defaults



Defense in Depth

Keeping Facebook safe requires a multi-layered approach to security

Secure frameworks
Security experts write libraries of code and new programming languages to prevent or remove entire classes of bugs

Automated testing tools
Analysis tools scan new and existing code for potential issues

Peer & design reviews
Human reviewers inspect code changes and provide feedback to engineers

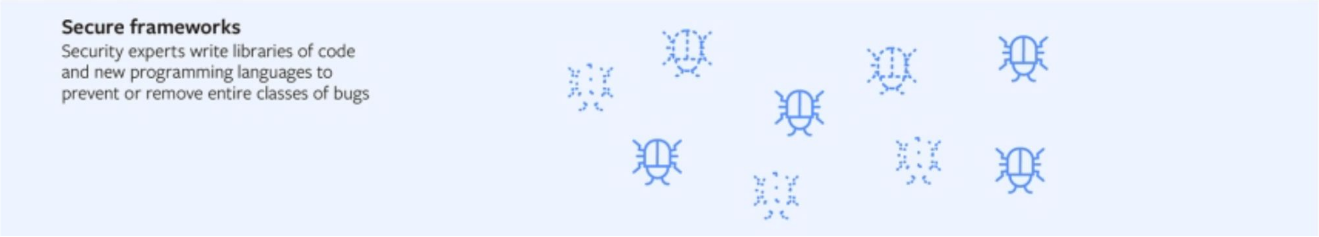
Red team exercises
Internal security experts stage attacks to surface any points of vulnerability

Bug bounty program
Outside researchers are incentivized to find and report security flaws

This layered approach greatly reduces the number of bugs live on the platform

[Designing Security for Billions - Facebook]()

How Meta / Facebook does secure defaults



Secure frameworks
Security experts write libraries of code and new programming languages to prevent or remove entire classes of bugs

Hack: an update of PHP to add typing information

allows more and better static analysis

XHP: augmentation to PHP/Hack to integrate HTML

very effective at preventing XSS

Designing Security for Billions - Facebook

How Semgrep does secure defaults

Self-service DevSec
without security team

Faster resolution
solved in minutes

Security can focus on high-impact work
not fixing devs latest XSS mistake

Fixing leaky logs: how to find a bug and ensure it never returns - Nathan Brahms

How Semgrep does secure defaults

Found tokens being logged

1. **Mitigate**
   Revert logging change
2. **The secure default**
   Replace `str` param with `ObfuscatedStr`
3. **Enforcement**

Fixing leaky logs: how to find a bug and ensure it never returns - Nathan Brahms

How Semgrep does secure defaults

3. Enforcement

Block commits to SQLAlchemy models for security review

Yearly training on the pitfalls of logging sensitive data

Audit logs weekly

File an issue with your SAST provider, demanding they add checks to catch sensitively logged data!

Fixing leaky logs: how to find a bug and ensure it never returns - Nathan Brahms

How Semgrep does secure defaults

3. Enforcement

~~Block commits to SQLAlchemy models for security review~~

~~Yearly training on the pitfalls of logging sensitive data~~

~~Audit logs weekly~~

~~File an issue with your SAST provider, demanding they add checks to catch sensitively logged data!~~

Fixing leaky logs: how to find a bug and ensure it never returns - Nathan Brahms

Secure defaults

WHY	Security must scale
WHAT	The secure way, the easy way
WHO	Success stories
HOW	Think long term, high impact

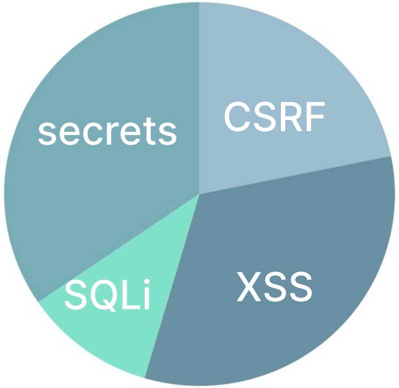
Think long term, think high impact

Think long term, high impact

1. Select vulnerability class
2. Build a scalable solution and make it the default
3. Measure adoption
4. Drive organic adoption

1. Select vulnerability class

AppSec time spent



Focus on best ROI

maximize impact, minimize ongoing time requirements

Reduce risk, ensure a baseline

don't try to find and fix every bug

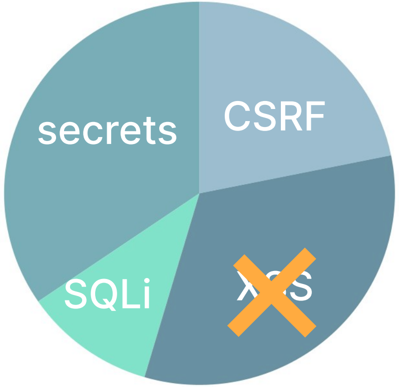
Eliminate bug classes

find and prevent at scale for compound effect

[BSidesSF 2020 - How to 10X Your Company's Security (Without a Series D) - Clint Gibler]

1. Select vulnerability class

AppSec time spent



Focus on best ROI

maximize impact, minimize ongoing time requirements

Reduce risk, ensure a baseline

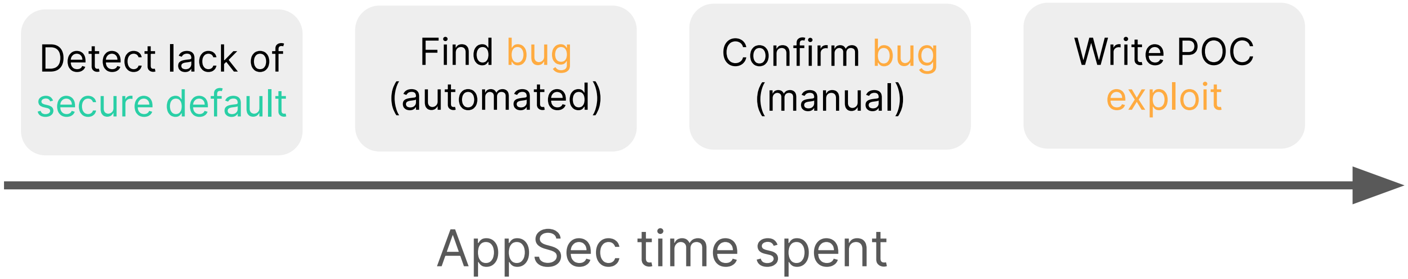
don’t try to find and fix every bug

Eliminate bug classes

find and prevent at scale for compound effect

BSidesSF 2020 - How to 10X Your Company’s Security (Without a Series D) - Clint Gibler

2. Build a scalable solution and make it the default



Detect lack of secure default

Find bug (automated)

Confirm bug (manual)

Write POC exploit

AppSec time spent

[Scaling AppSec - sec4dev - Clint Gibler]

3. Measure adoption

Team	Score
1	🟩
2	🟩
3	🟧

Track costs and fix time

per team and per bug class

Track adoption of secure defaults

speak to your “customers”

also provides friendly peer pressure

4. Drive organic adoption by productizing your secure defaults

Integrate into existing features

make the secure way, the easy way

Add non-security features

make it attractive to use

4. Drive organic adoption

Integrate into existing features

make the secure way, the easy way

Add non-security features

make it attractive to use

Automate checks

to observe, and to enforce adoption

An effective false positive is a marking where the developer chooses not to take action

False positive (FP)

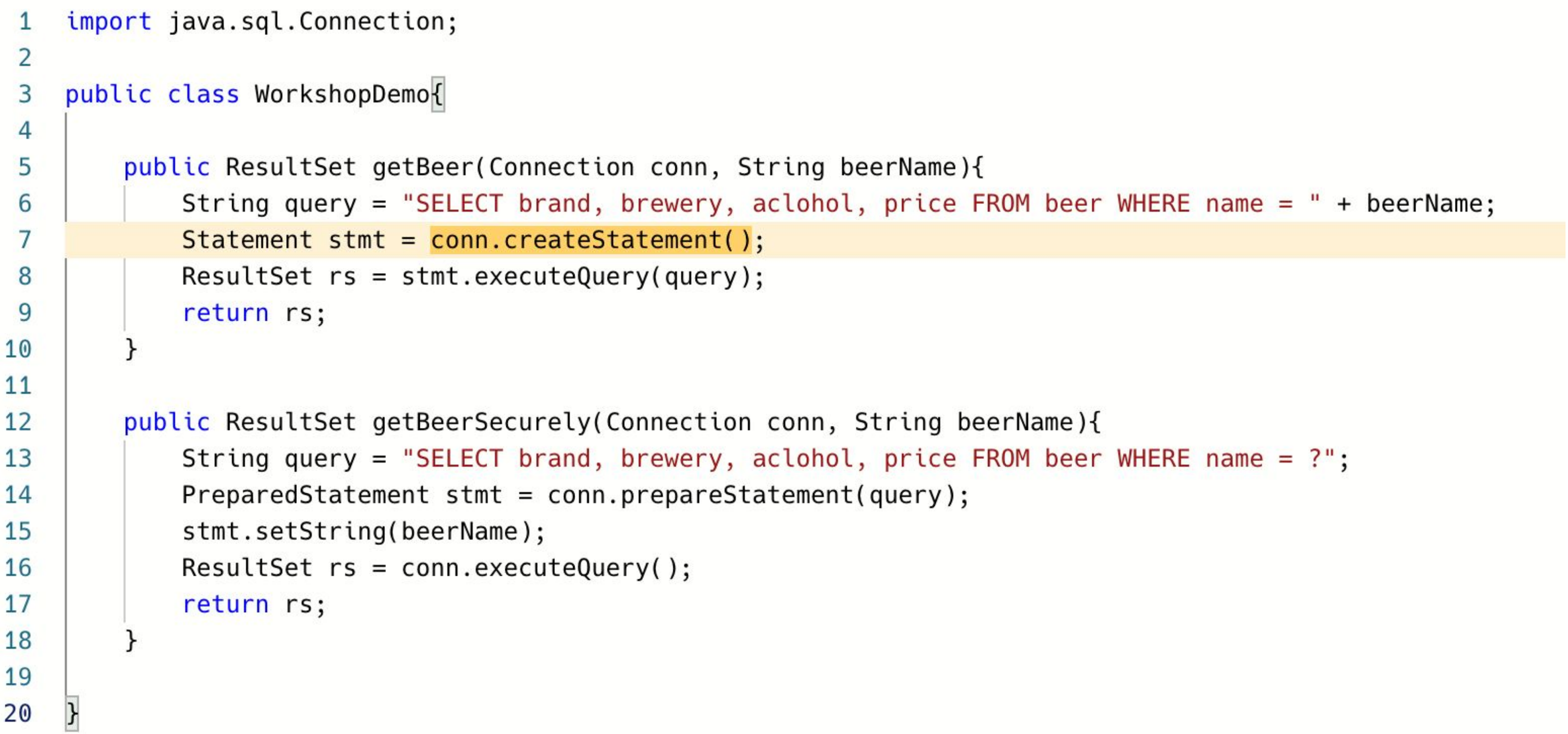
security perspective
secure code marked as insecure

Effective False Positive (EFP)

developer perspective
any marking a developer won’t fix

Tricorder: Building a Program Analysis Ecosystem, Sadowski et. al, Google

An effective false positive is a marking where the developer chooses not to take action



```java
import java.sql.Connection;

public class WorkshopDemo{

    public ResultSet getBeer(Connection conn, String beerName){
        String query = "SELECT brand, brewery, aclohol, price FROM beer WHERE name = " + beerName;
        Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery(query);
        return rs;
    }

    public ResultSet getBeerSecurely(Connection conn, String beerName){
        String query = "SELECT brand, brewery, aclohol, price FROM beer WHERE name = ?";
        PreparedStatement stmt = conn.prepareStatement(query);
        stmt.setString(beerName);
        ResultSet rs = conn.executeQuery();
        return rs;
    }

}
```

Drive adoption with better tools

Relevant
project-specific guidelines

Efficient
fast scan times, well-integrated

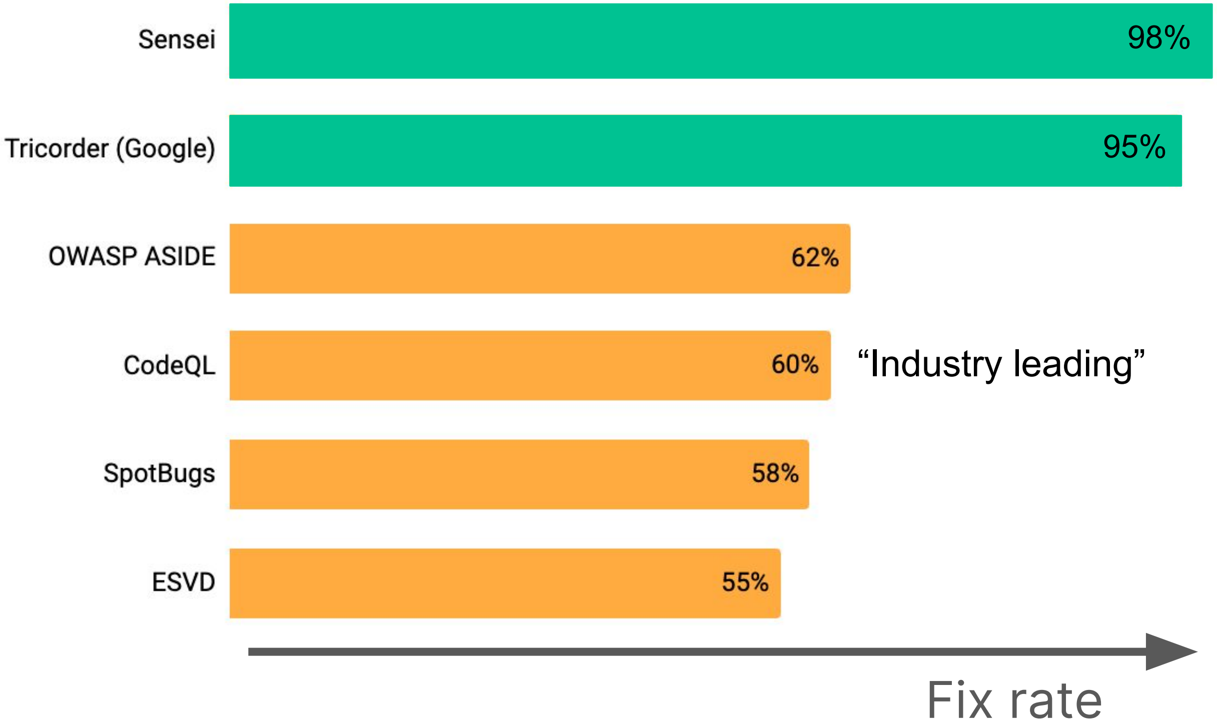
Usable
not just detect mistakes, but help with fixing

[The Paved Path Methodology, Pieter De Cremer, OWASP BeNeLux Days]()

A relevant tool allows for customized rules

With
customized rules

Without
customized rules



The Paved Path Methodology, Pieter De Cremer, OWASP BeNeLux Days
Find critical vulnerabilities and eradicate them, forever - CodeQL

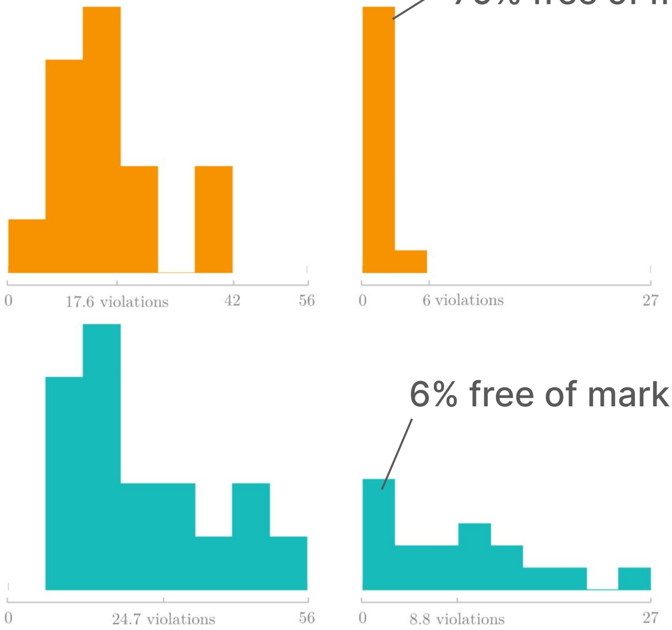
A usable tool provides remediation guidance

during assignment

when finished

Remediation guidance

No remediation guidance



79% free of markings

6% free of markings

The Paved Path Methodology, Pieter De Cremer, OWASP BeNeLux Days

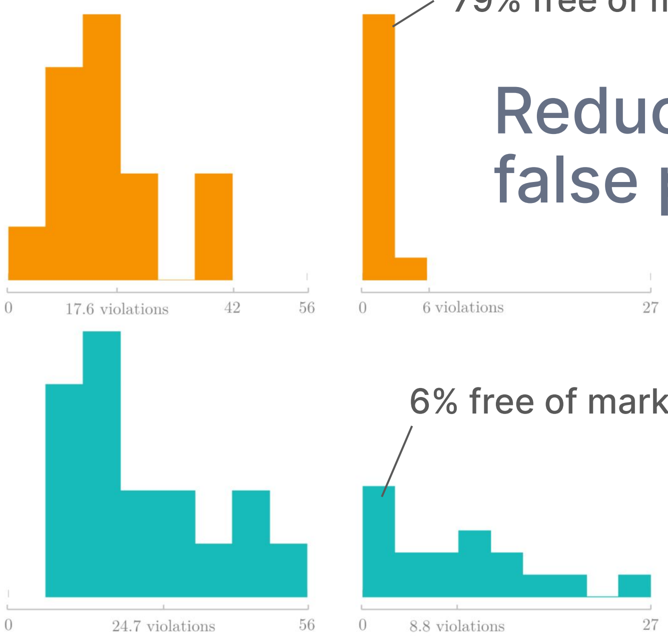
A usable tool provides remediation guidance

during assignment

when finished

Remediation guidance

No remediation guidance



79% free of markings

6% free of markings

Reduced effective false positives!

The Paved Path Methodology, Pieter De Cremer, OWASP BeNeLux Days

Providing remediation guidance greatly reduces effective false positives

Rules with autofix have 50% higher fix rate

compared to rules without autofix

Secure defaults

WHY	Security must scale
WHAT	The secure way, the easy way
WHO	Success stories
HOW	Think long term, high impact

Secure defaults is NOT just...

...having developers fix all security bugs

...only fixing high priority issues

Secure defaults is NOT just...

...having developers fix all security bugs
but building scalable self-service solutions

...only fixing high priority issues

Secure defaults is NOT just...

...having developers fix all security bugs
but building scalable self-service solutions

...only fixing high priority issues
but killing high-impact bug classes

TL;DR secure defaults

WHY Security must scale

speed of development has increased
security experts are understaffed

WHAT The secure way, the easy way
systematic fundamental solutions
productizing those solutions

WHO Early adopters have been successful
Netflix, Meta, Google, Snowflake, Semgrep, and more

HOW Think long term, high impact
leverage your time most effectively now
to have big wins in the future
automate smart with role-specific tools

Secure defaults
developer-friendly security

Claudio Merloni
claudio@semgrep.com
https://linkedin.com/in/claudiomerloni

Semgrep