

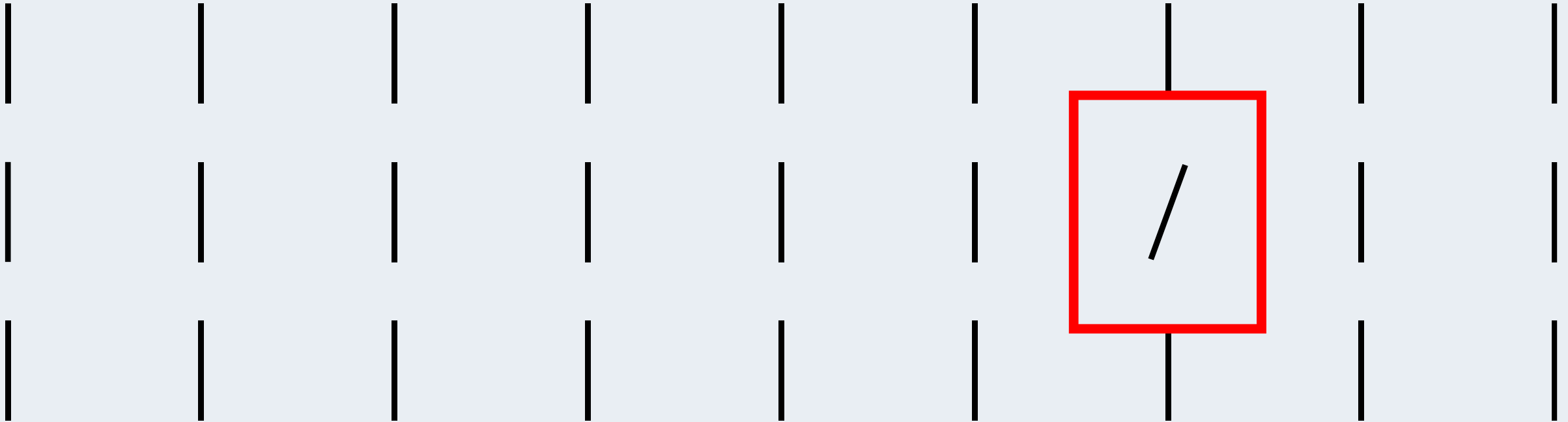
# PT Application Inspector 4



Universal tool for locating  
vulnerabilities  
in applications



# The importance of analyzing source code



# Web attacks are the cause of most data leaks

## In 98% of cases

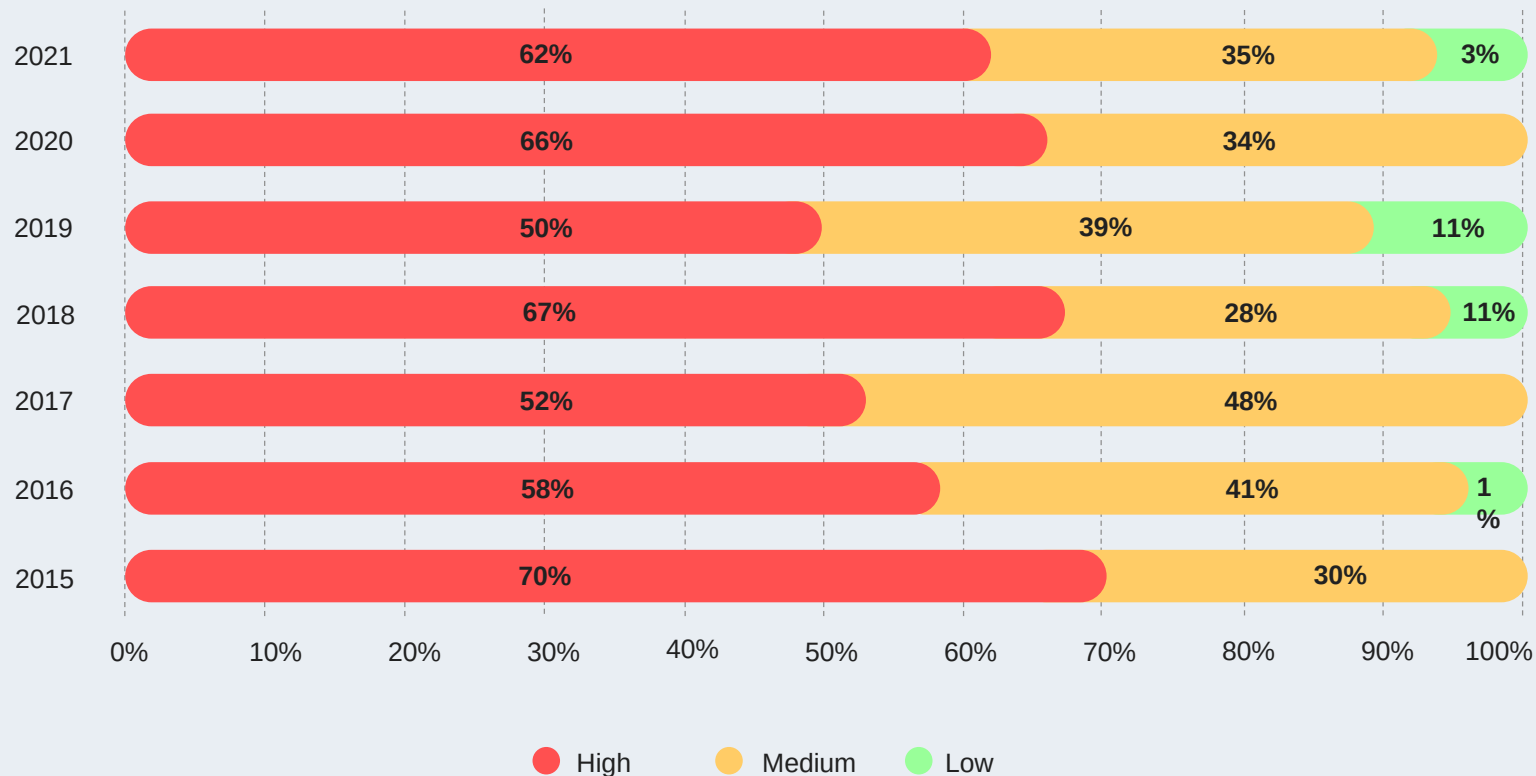
attackers are able to  
attack users

## 91% of web applications

leak important data

## 72% of vulnerabilities

were associated with errors  
in web application code

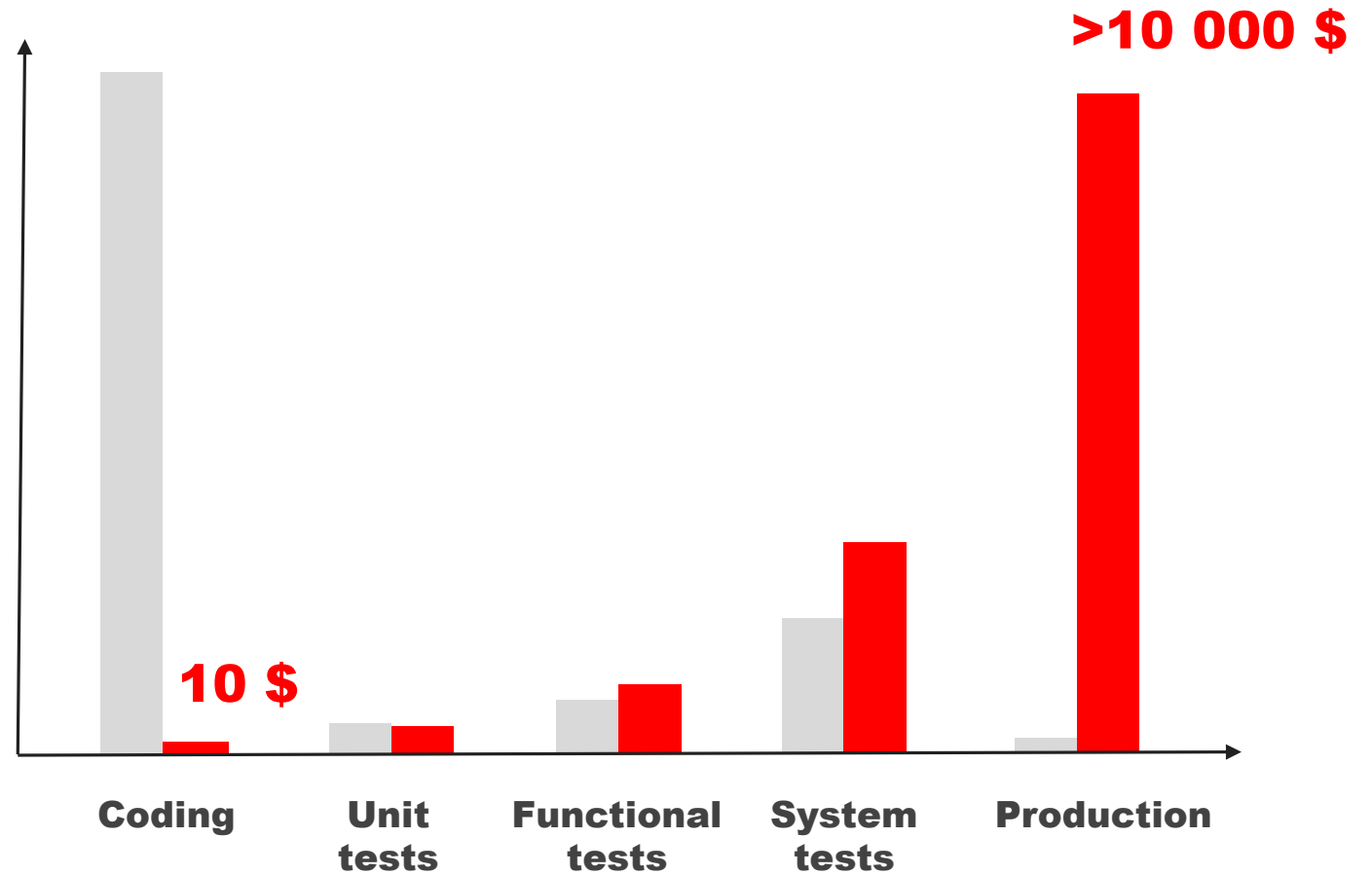


\* Web application vulnerabilities and threats in 2021–2022.

# Economics

**The cost of a bug and why you need shift-left**

Applied Software  
Measurement, Capers Jones



# How to find a vulnerability

```
user.bid = basket.id // keep track of original basket for challenge
solution check      insecurity.authenticatedUsers.put(token, user)
res.json({c authentication: { token, bid: basket.id, uemail: user.data.email } })
}).catch(error => {      next(error)      }) }
```

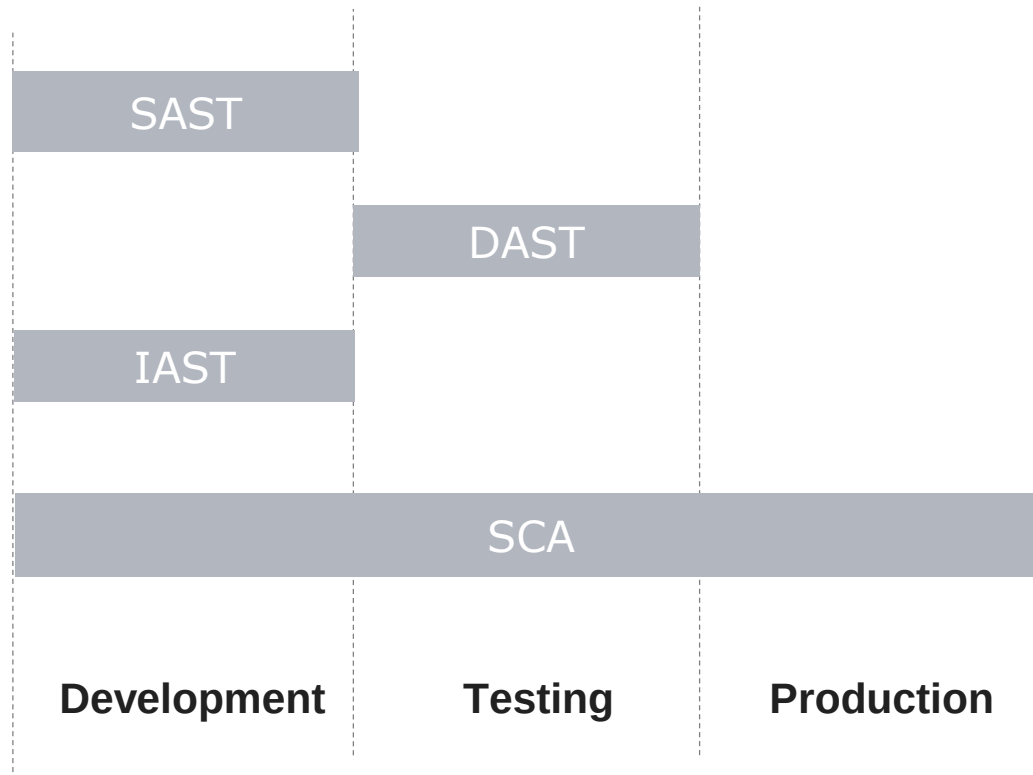
```
return (req, res, next) => {
  verifyPreLoginChallenges(req)
  models.sequelize.query('SELECT * FROM Users WHERE
email = '${req.body.email} || '''
  AND password = '${insecurity.hash(req.body.password
  || '')}',
```

```
{ model: models.User, plain: true }).then((authenticatedUser) => {
  let user = utils.queryResultToJson(authenticatedUser)
  const rememberedEmail = insecurity.userEmailFrom(req)
  if (rememberedEmail && req.body.oauth) {
    models.User.findOne(
      { where:
        { email: rememberedEmail } }).then(rememberedUser => {
      user = utils.queryResultToJson(rememberedUser)
      utils.solveIf(challenges.loginCisoChallenge, () => {
```

## Manual analysis

can be used for small code fragments, but is not fit for the conditions of modern industrial development

# Application security testing tools



## Static application security testing (SAST)

locates vulnerabilities in the source code

## Dynamic application security testing (DAST)

checks the running application

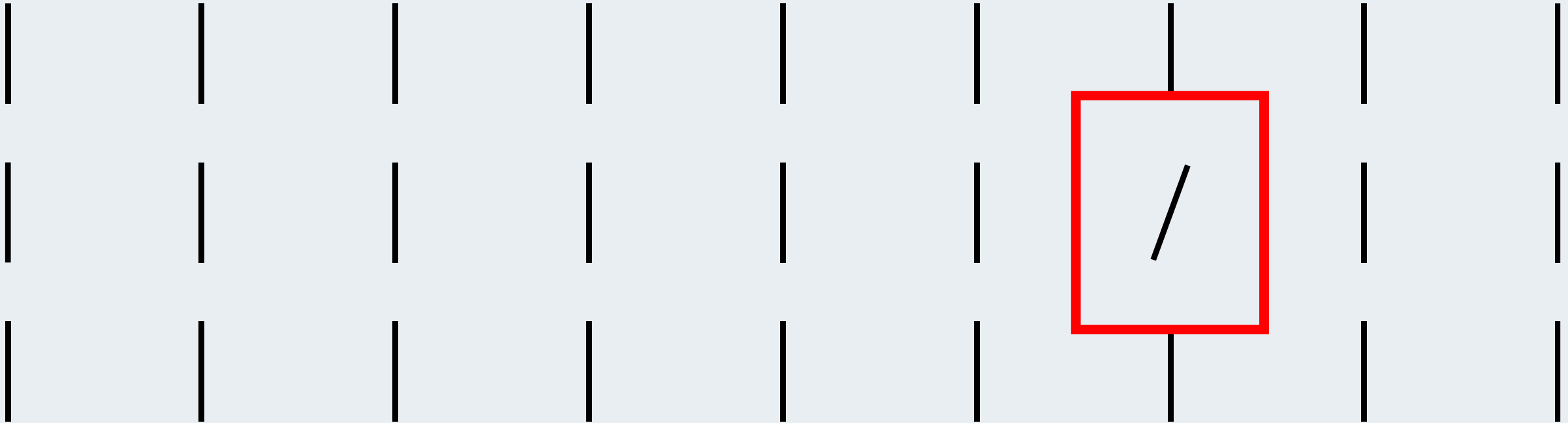
## Interactive application security testing (IAST)

provides instrumentation of code and searches for vulnerabilities as it runs

## Software composition analysis (SCA)

detects vulnerabilities in 3rd party code

# Application Inspector

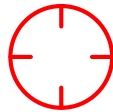


# PT Application Inspector features



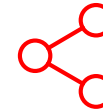
## Thoroughly identifies and prioritizes vulnerabilities

PT Application Inspector not only detects vulnerabilities, but also identifies the conditions that allow their exploitation. This is accomplished using a combination of detection mechanisms, and by the expertise of the team at Positive Technologies



## Easy to use

Analysis can be started without configuring the application or obtaining access to the environment. Simply indicate the folder that contains the source code



## Integrates into existing Development lifecycle

PT Application Inspector carefully integrates itself into existing software development lifecycle using plugins for connection to application set-up and delivery systems. This makes it possible to ensure safe code development



# Benefits



## Expertise: From praxis to product

Specialists from the application security analysis division make regular updates to the product's knowledge base, adding information about particular vulnerabilities discovered in real applications.



## Flexible scaling

The power and speed of analysis conducted by PT AI can be horizontally scaled as needed. The solution's architecture maximally streamlines this process and minimizes time and effort required.



## A minimum of false positives

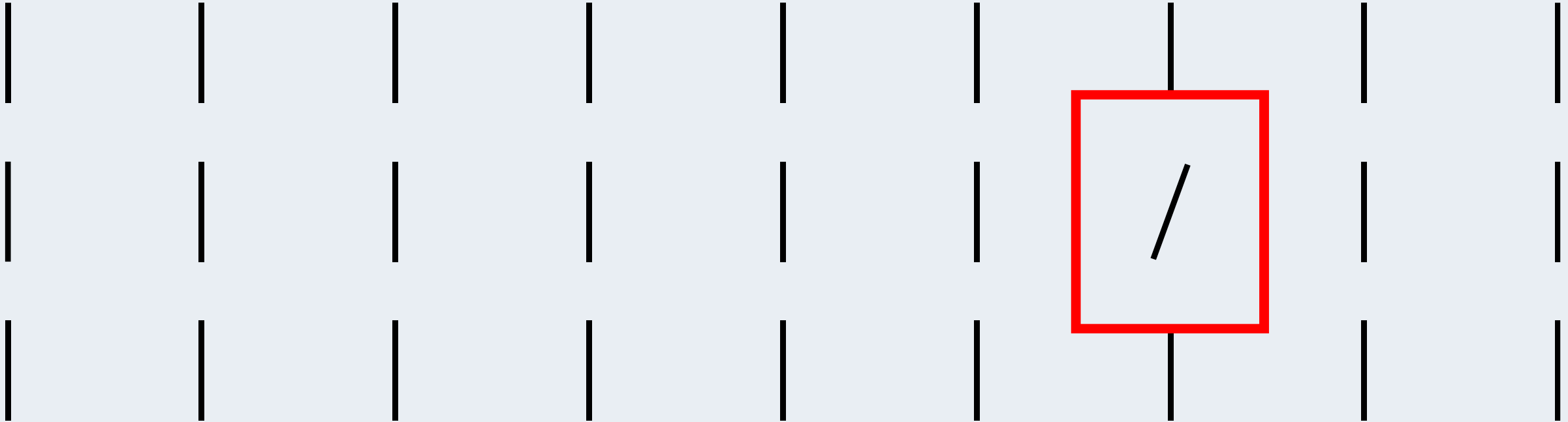
PT Application Inspector identifies the exploit conditions for located vulnerabilities and conducts a triage to prioritize their elimination. This helps developers concentrate on the most important problems and lowers expenses incurred by having experts check results manually.



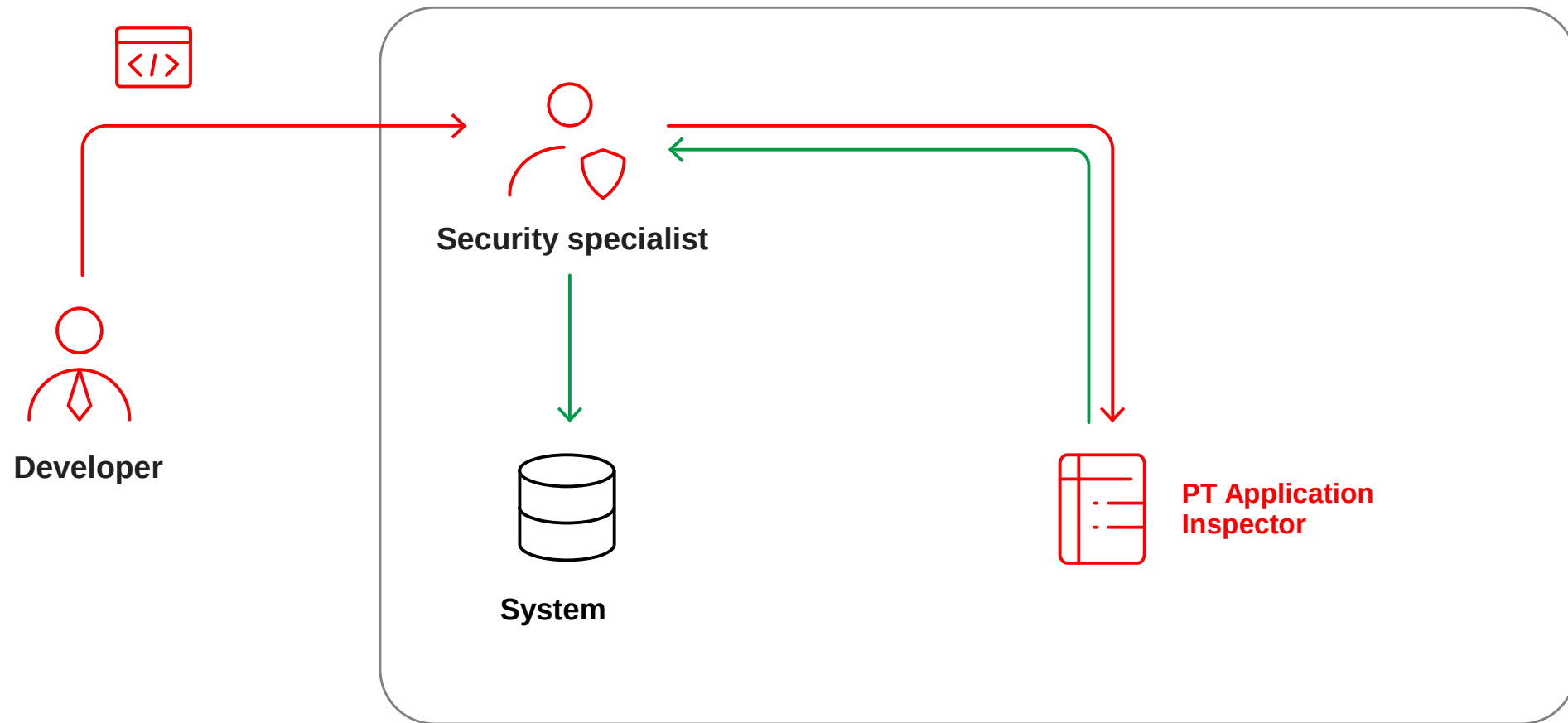
## Rapid vulnerability elimination

Specialized virtual patches for use in PT Application Firewall can be created based on the results of PT AI analyses. This helps prevent vulnerability exploits in applications.

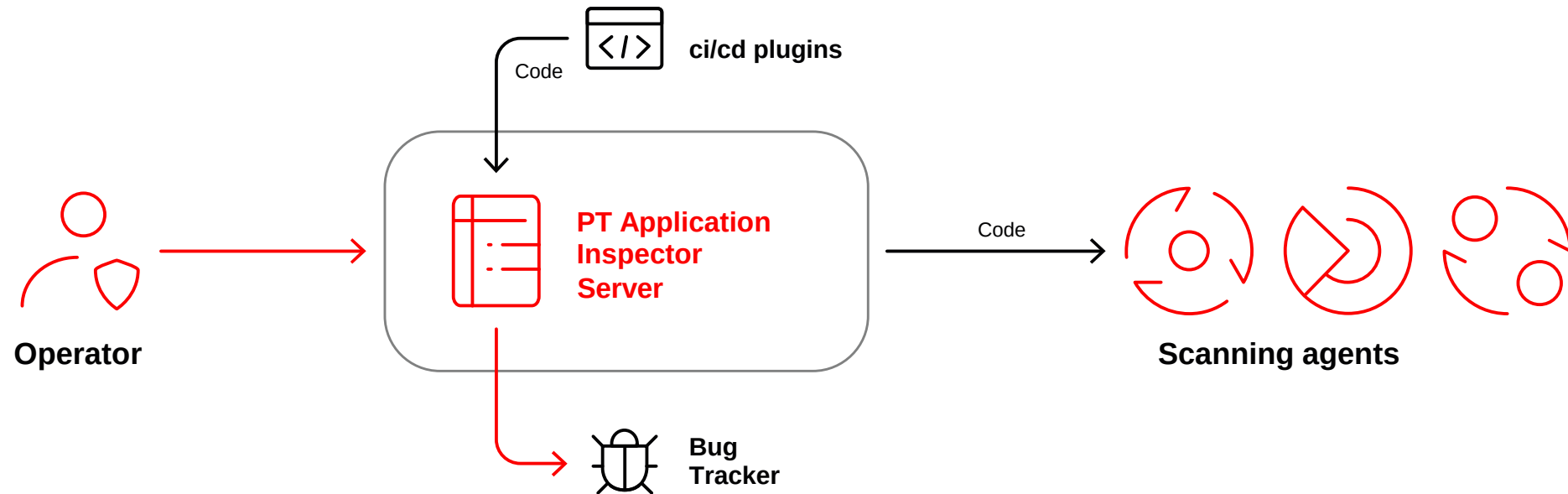
# Use cases



# Basic. Code acceptance



# Integration into the development lifecycle



# Characteristics

**A convenient tool for searching for application vulnerabilities**

## Languages:

Java, PHP, C#, VB.NET, JavaScript, Python, Kotlin, Go, C/C++, Objective-C, Swift, SQL (T-SQL, PL/SQL, MySQL) + TypeScript

## Development environment:

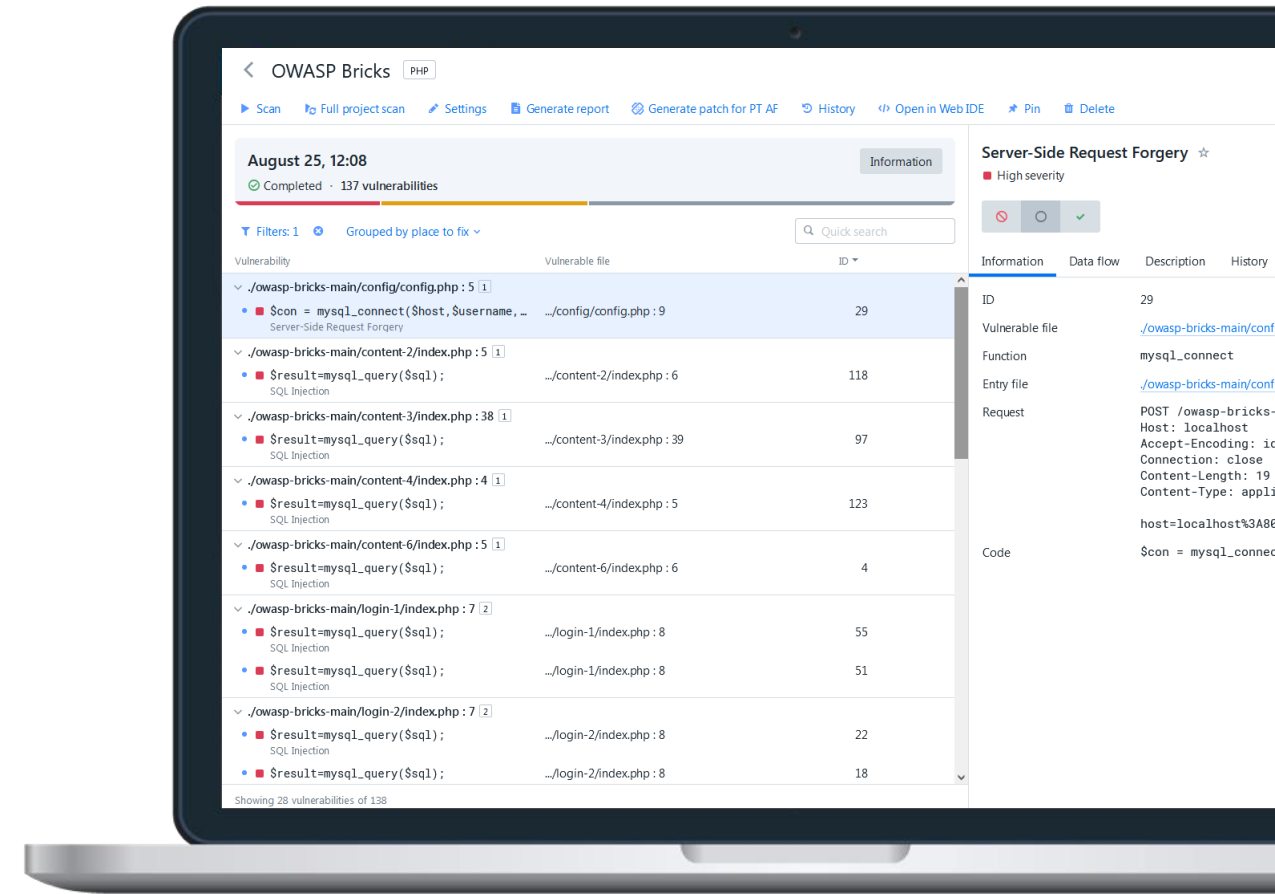
Jenkins, TeamCity, GitLab CI, Azure

## IDE Integrations:

JetBrains, Visual Studio Code

## Deployment:

Linux + Docker + SSO



# Licensing

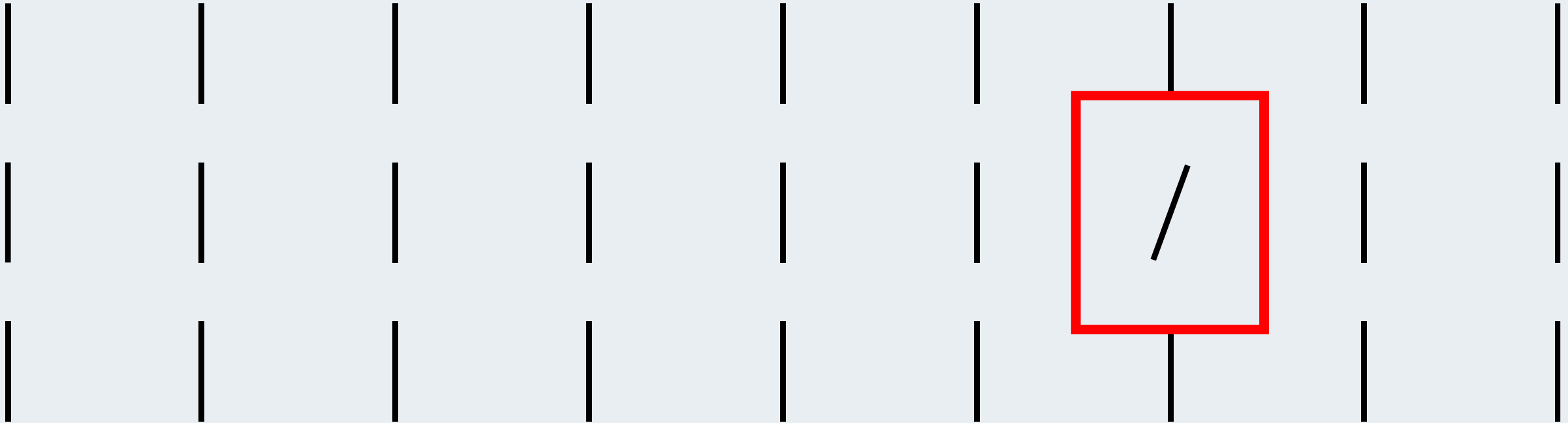
**By number of programming languages**

**By number of scanning agents**

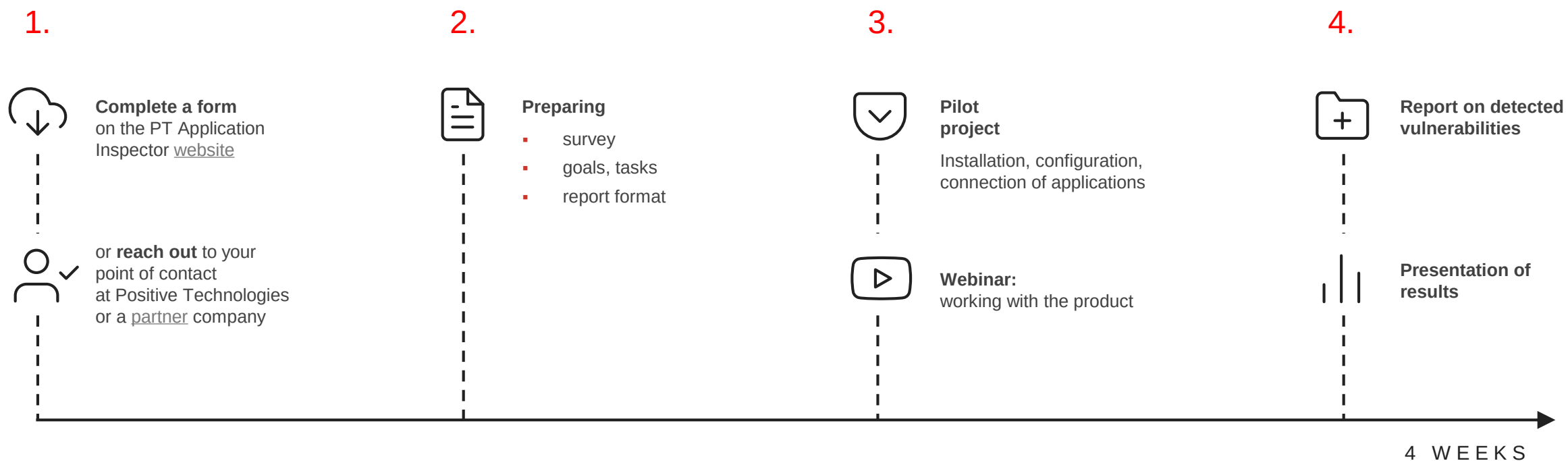
**No restrictions:**

- on the number of project
- on lines of code
- on the number of users

# How to get started with PT Application Inspector



# How to conduct a pilot test





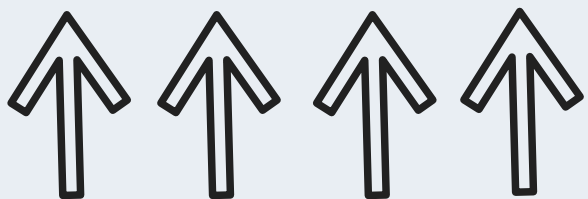
Plugins:  
IntelliJ  
Platform IDE



Plugins:  
Visual Studio  
Code

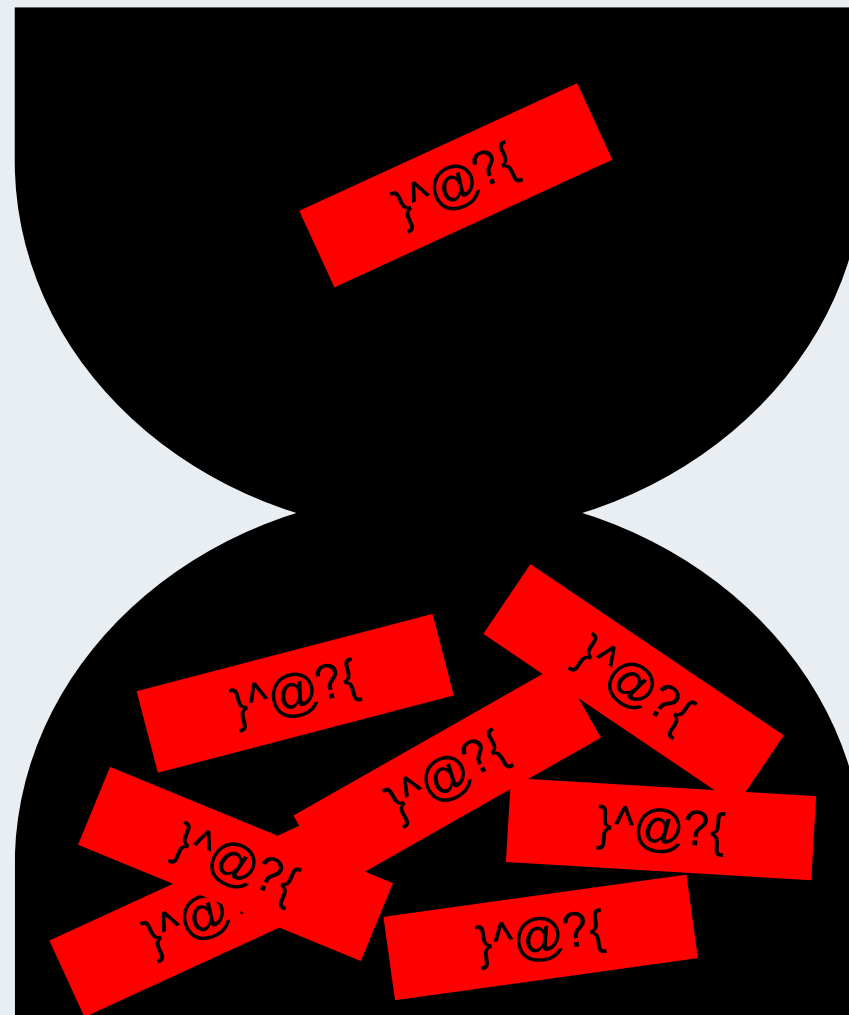


Code for assessing  
the quality of work  
SAST



# Open source

Positive Technologies



[ptsecurity.com](https://ptsecurity.com)