Simone Riccetti

La sicurezza nello sviluppo delle applicazioni Web

Security Day 2010

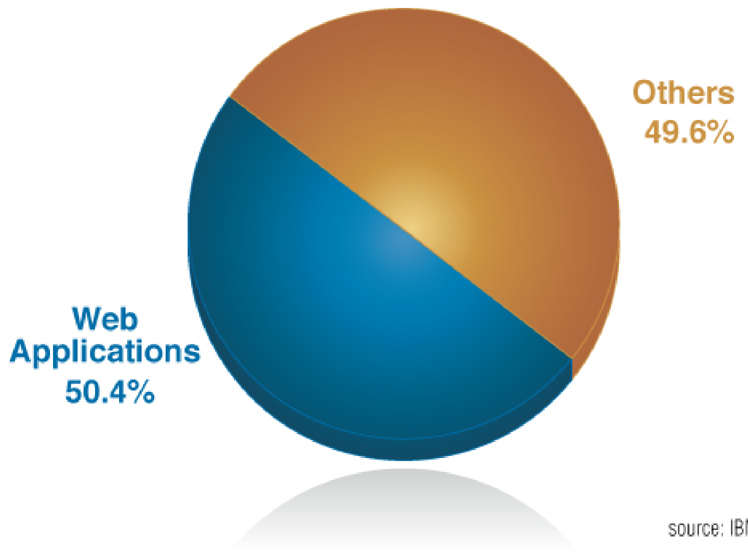
Agenda

- Security Landscape
- Common Vulnerabilities
- Analysis Techniques
- IBM Secure Software Engineering

Web App Vulnerabilities Continue to Dominate

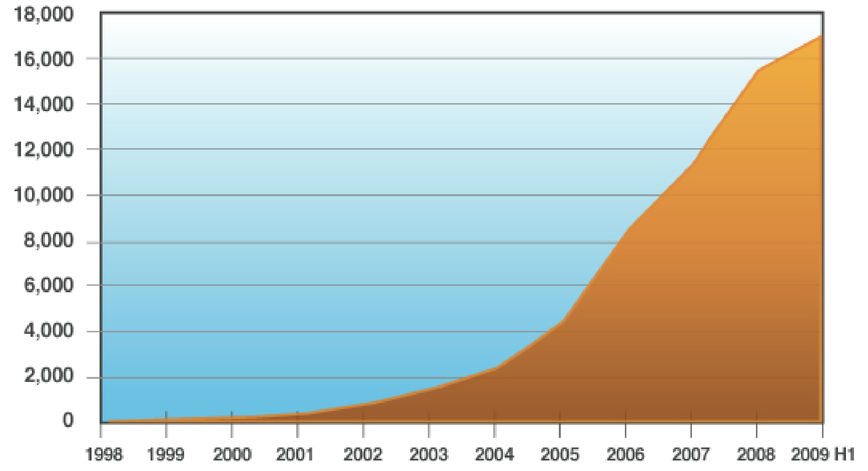
- **50.4%** of all vulnerabilities are Web application vulnerabilities
- SQL injection and Cross-Site Scripting are neck and neck in a race for the top spot

Web Application Vulnerabilities
as a Percentage of All Disclosures in 2009 H1



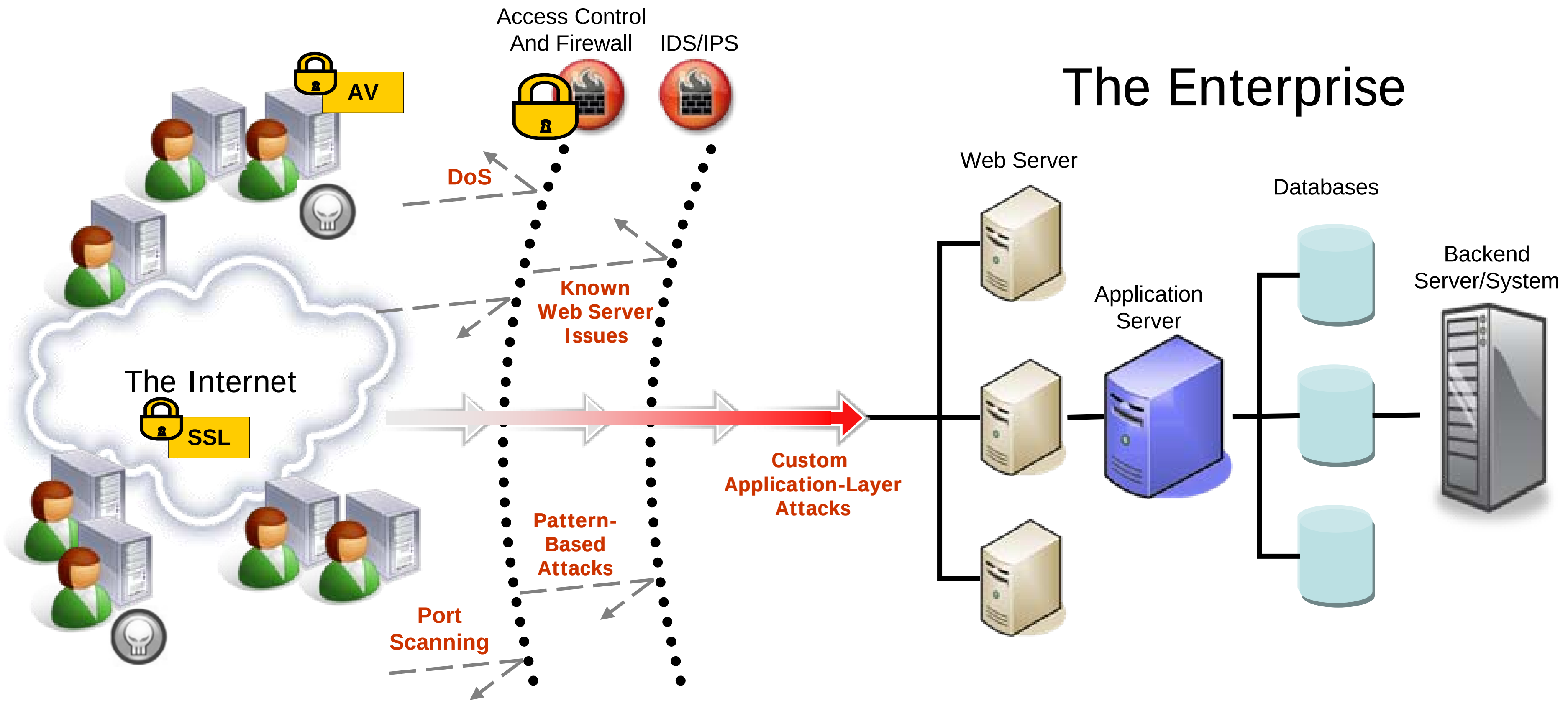
source: IBM X-Force®

Vulnerability Disclosures Affecting Web Applications
Cumulative, year over year



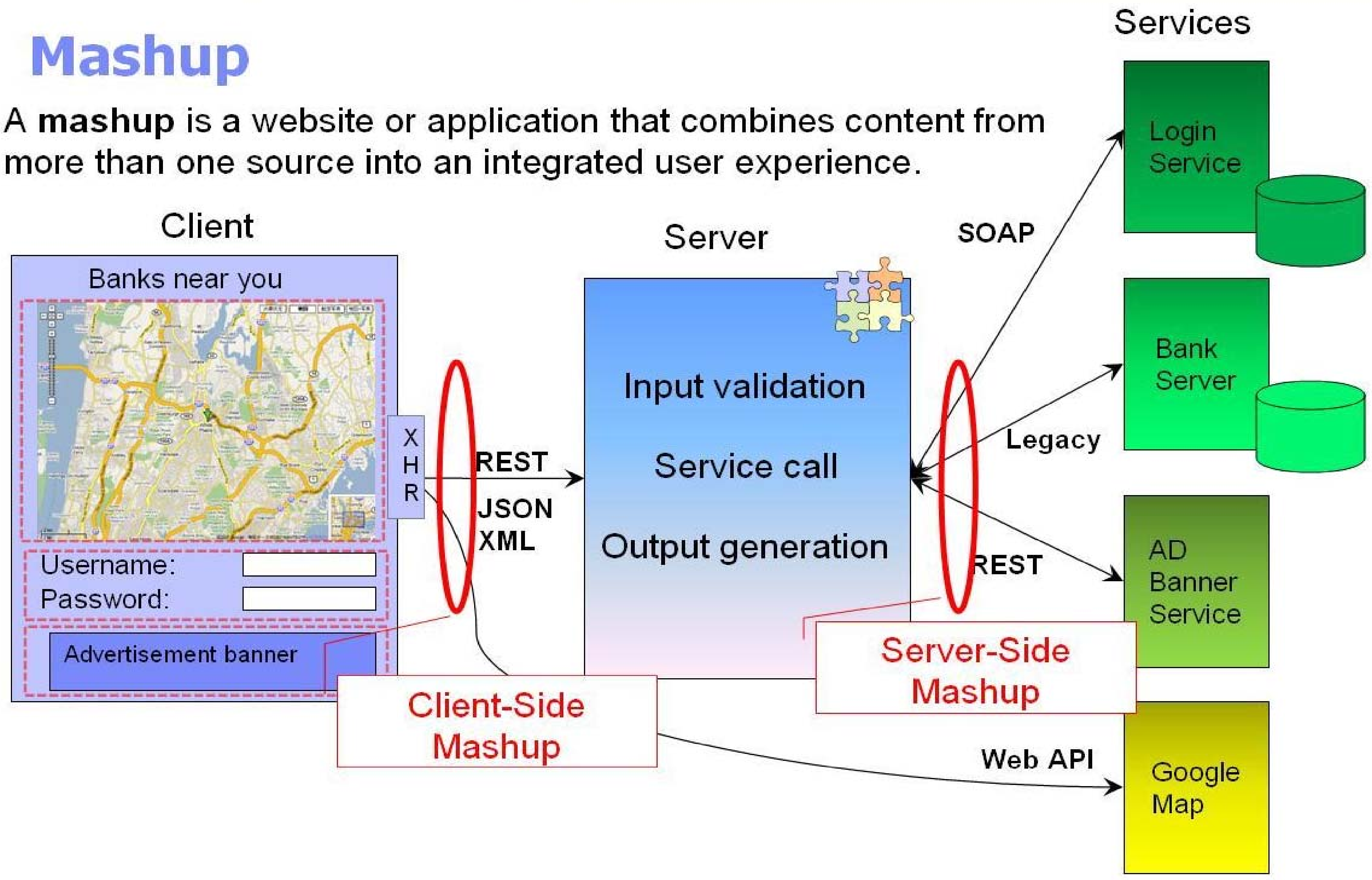
source: IBM X-Force®

Simple Security Landscape

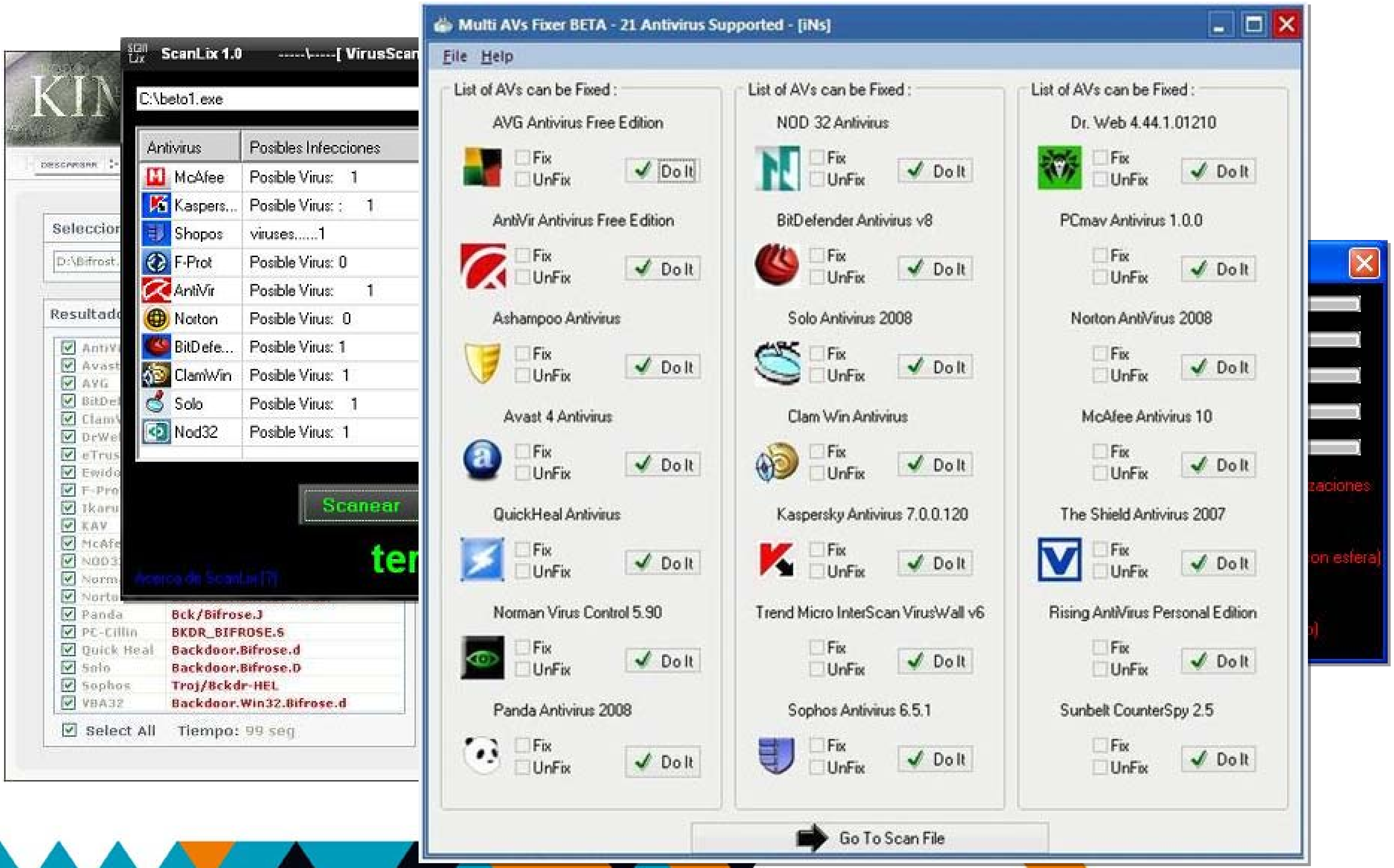


Mashup

A **mashup** is a website or application that combines content from more than one source into an integrated user experience.



Security Day 2010



ScanLix 1.0 -----\-----[VirusScan

C:\beto1.exe

Antivirus	Posibles Infecciones
McAfee	Posible Virus: 1
Kaspers...	Posible Virus: : 1
Shopos	viruses......1
F-Prot	Posible Virus: 0
AntiVir	Posible Virus: 1
Norton	Posible Virus: 0
BitDefe...	Posible Virus: 1
ClamWin	Posible Virus: 1
Solo	Posible Virus: 1
Nod32	Posible Virus: 1

D:\Bifrost.

✓ Panda	Bck/Bifrose.J
✓ PC-Cillin	BKDR_BIFROSE.S
✓ Quick Heal	Backdoor.Bifrose.d
✓ Solo	Backdoor.Bifrose.D
✓ Sophos	Troj/Bckdr-HEL
✓ VBA32	Backdoor.Win32.Bifrose.d

✓ Select All   Tiempo: 99 seg

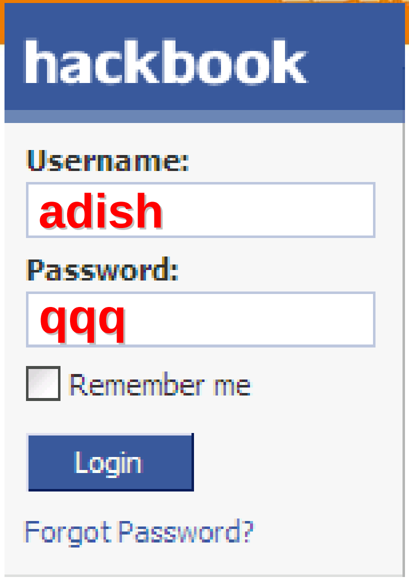
Multi AVs Fixer BETA - 21 Antivirus Supported - [iNs]

List of AVs can be Fixed :	List of AVs can be Fixed :	List of AVs can be Fixed :
AVG Antivirus Free Edition	NOD 32 Antivirus	Dr. Web 4.44.1.01210
AntiVir Antivirus Free Edition	BitDefender Antivirus v8	PCmav Antivirus 1.0.0
Ashampoo Antivirus	Solo Antivirus 2008	Norton AntiVirus 2008
Avast 4 Antivirus	Clam Win Antivirus	McAfee Antivirus 10
QuickHeal Antivirus	Kaspersky Antivirus 7.0.0.120	The Shield Antivirus 2007
Norman Virus Control 5.90	Trend Micro InterScan VirusWall v6	Rising AntiVirus Personal Edition
Panda Antivirus 2008	Sophos Antivirus 6.5.1	Sunbelt CounterSpy 2.5

Agenda

- Security Landscape
- Common Vulnerabilities
- Analysis Techniques
- IBM Secure Software Engineering

es. SQL Injection



Username: adish

Password: qqq

- User input is embedded as-is in predefined SQL statements:

```
Stmt = "SELECT * from tUsers where
            userid='" + iUserID + "' AND
            password='" + iPassword + "'";
```

UserID	Username	Password	Name
1824	adish	qqq	Adi Sharabani

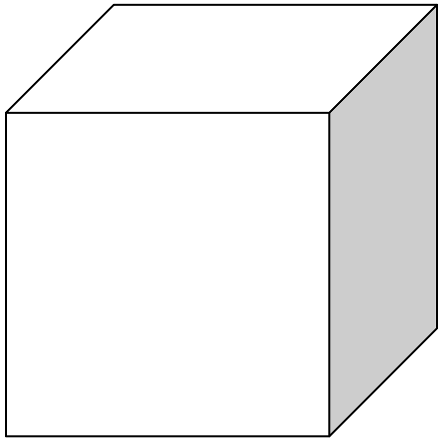
- Hacker supplies input that modifies the original SQL statement, for example:
  - `iUserID = ' or 1=1 --`

UserID	Username	Password	Name
1	Admin	$#kaoeFor56	Administrator

Agenda

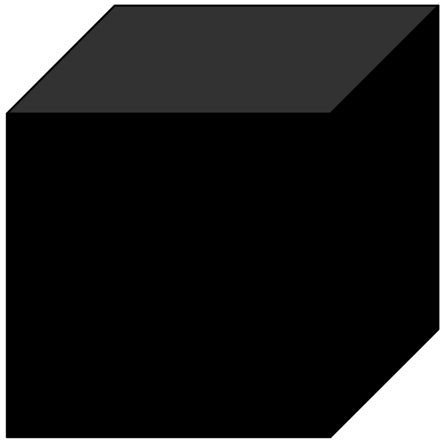
- Security Landscape
- Common Vulnerabilities
- Analysis Techniques
- IBM Secure Software Engineering

General testing techniques

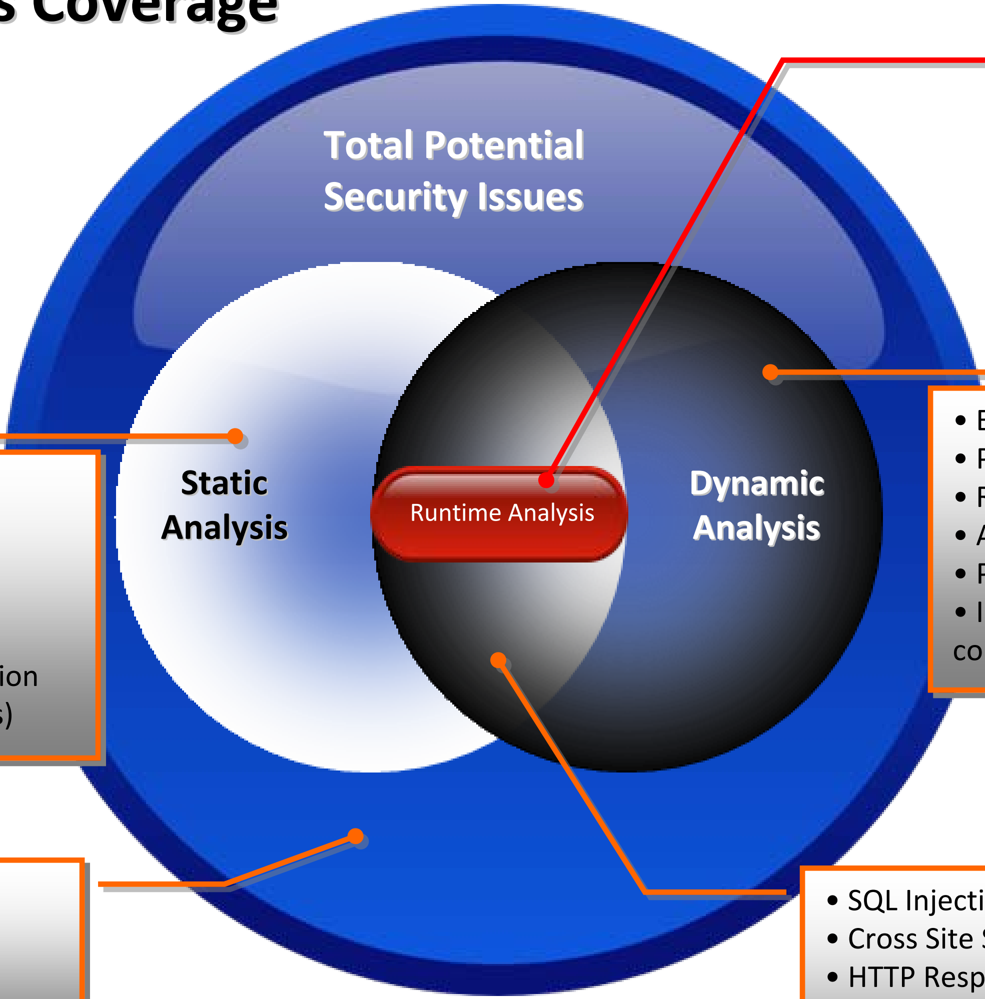


White-Box Analysis

Black-Box Analysis



Security Issues Coverage



Runtime Analysis serves as the "glue" between Static & Dynamic Analysis. It helps correlate results and improve overall accuracy & actionability

- Null Pointer Dereference
- Threading Issues
- Code Quality Issues
- Issues in Dead Code
- Insecure Crypto Functions
- Issues in Back-End Application Code (Multi-Tier Applications)

- Environment Configuration Issues
- Patch Level Issues
- Runtime Privileges Issues
- Authentication Issues
- Protocol Parser/Serializer Issues
- Issues in external 3rd party components

- Application Logic Issues

- SQL Injection
- Cross Site Scripting
- HTTP Response Splitting
- OS Commanding
- LDAP Injection
- …

What is static analysis?

- The study of things that are not changing.
- Evaluating code without executing it.
- Algorithms for analyzing algorithms.
- Process of building theoretical models of how an application works, from its code and binaries, and looking for weaknesses from these models

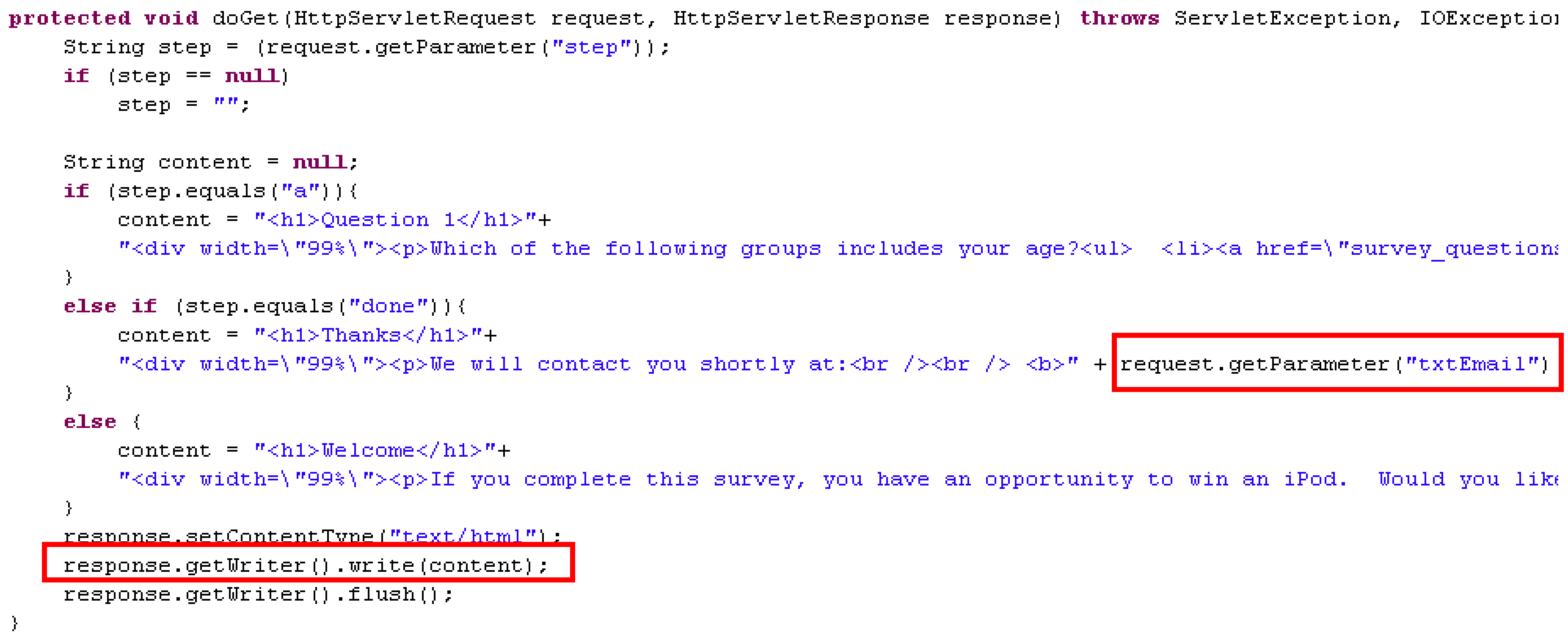
IBM and Static Analysis

- IBM has been researching static analysis since the 1970’s
- IBM has dozens of publications, patents in the static analysis field

IBM T.J. Watson Research Center, NY

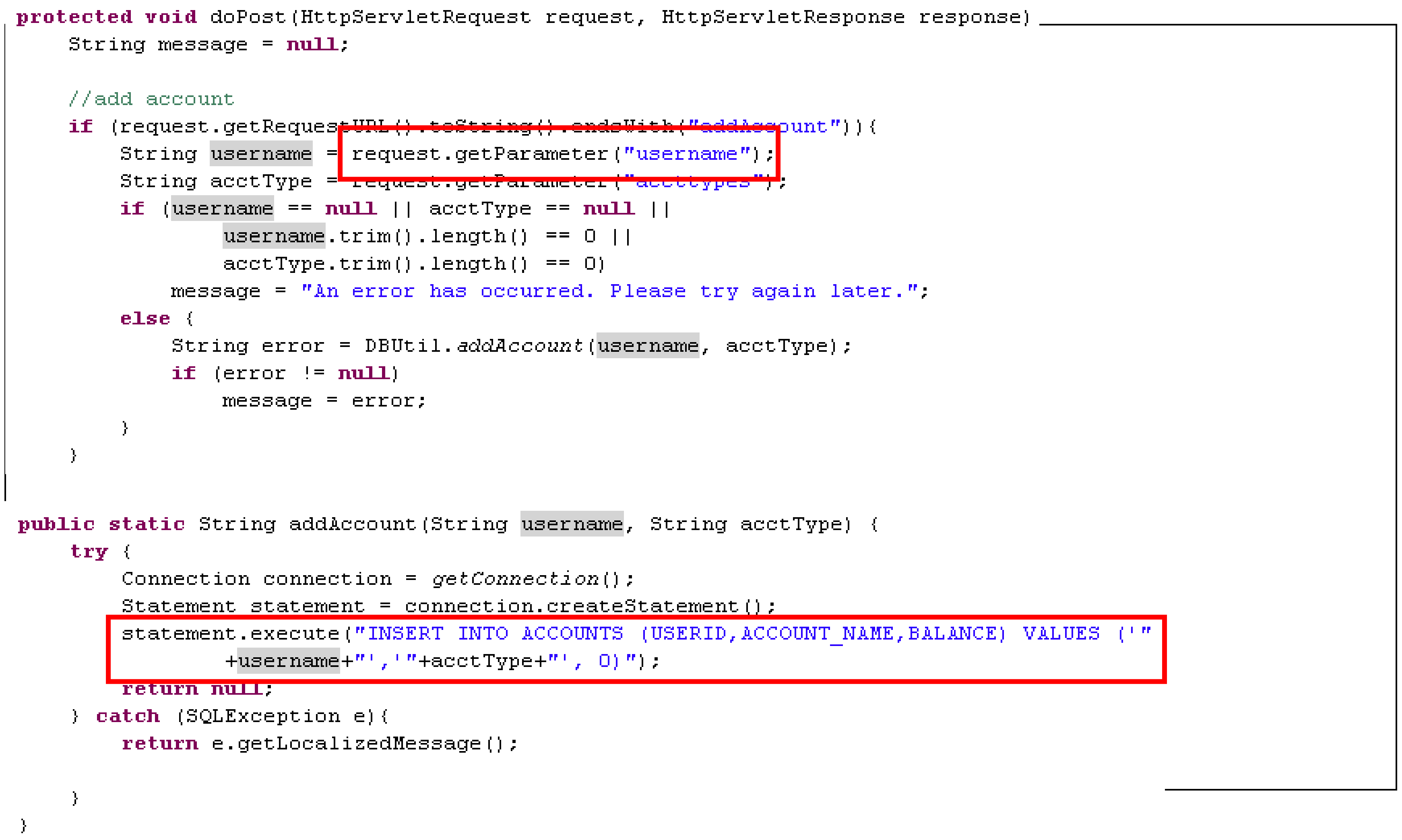
Mark Wegman, IBM Fellow
Invented SSA (Static Single Assignment) form back in the 1980’s;
This form is used by virtually all compilers and static analyzers today.

How is this code vulnerable?



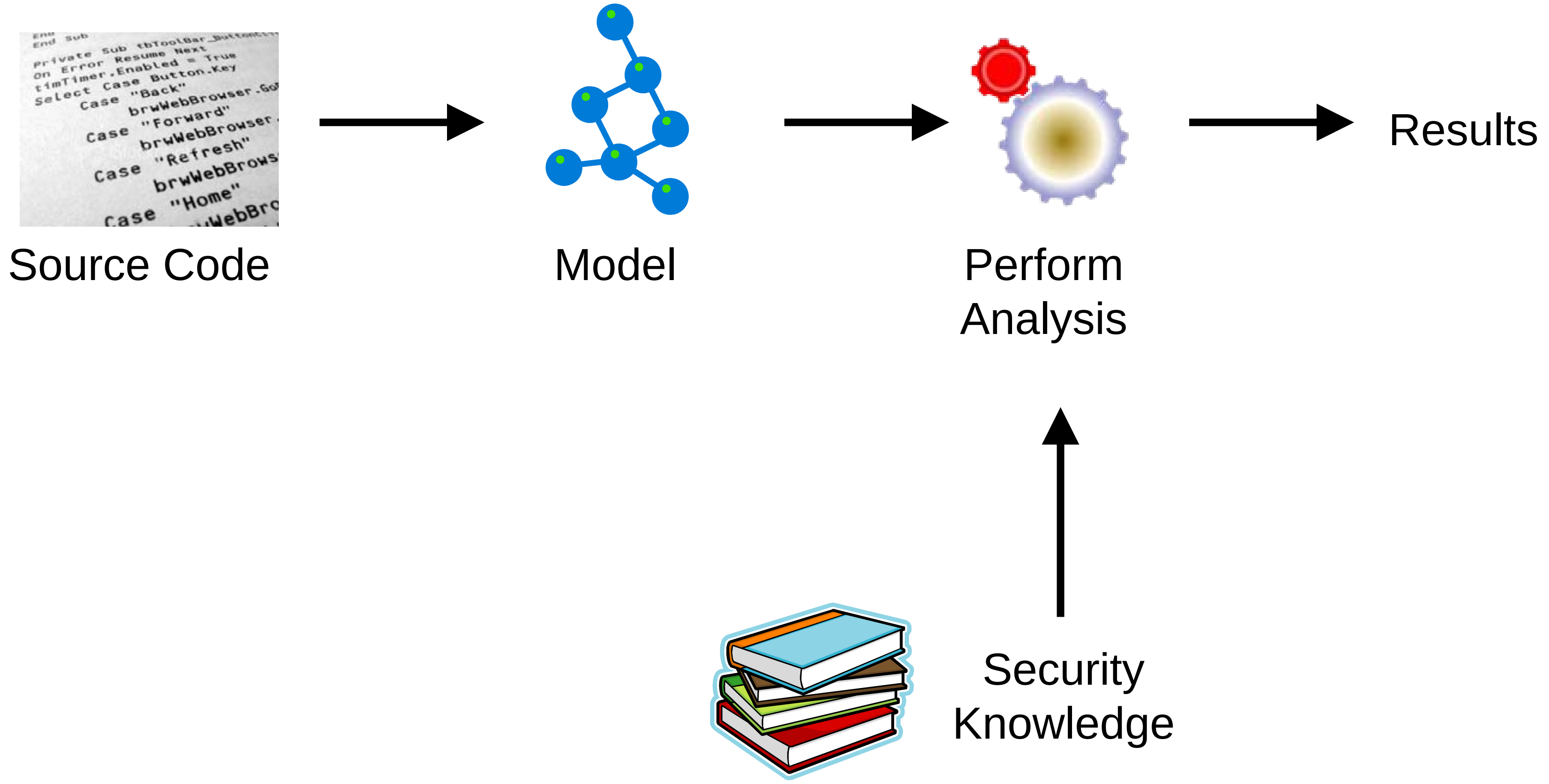
```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOExceptio
    String step = (request.getParameter("step"));
    if (step == null)
        step = "";

    String content = null;
    if (step.equals("a")){
        content = "<h1>Question 1</h1>"+
        "<div width=\"99%\"><p>Which of the following groups includes your age?<ul>  <li><a href=\"survey_question
    }
    else if (step.equals("done")){
        content = "<h1>Thanks</h1>"+
        "<div width=\"99%\"><p>We will contact you shortly at:<br /><br /> <b>" + request.getParameter("txtEmail")
    }
    else {
        content = "<h1>Welcome</h1>"+
        "<div width=\"99%\"><p>If you complete this survey, you have an opportunity to win an iPod.  Would you lik
    }
    response.setContentType("text/html");
    response.getWriter().write(content);
    response.getWriter().flush();
}
```

How is this code vulnerable? (2)



```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    String message = null;

    //add account
    if (request.getRequestURL().toString().endsWith("addAccount")){
        String username = request.getParameter("username");
        String acctType = request.getParameter("accttypes");
        if (username == null || acctType == null ||
                username.trim().length() == 0 ||
                acctType.trim().length() == 0)
            message = "An error has occurred. Please try again later.";
        else {
            String error = DBUtil.addAccount(username, acctType);
            if (error != null)
                message = error;
        }
    }

public static String addAccount(String username, String acctType) {
    try {
        Connection connection = getConnection();
        Statement statement = connection.createStatement();
        statement.execute("INSERT INTO ACCOUNTS (USERID,ACCOUNT_NAME,BALANCE) VALUES ('"
                +username+"','"+acctType+"', 0)");
        return null;
    } catch (SQLException e){
        return e.getLocalizedMessage();

    }
}
```

Generic static analysis process



Taint Analysis – How It Works

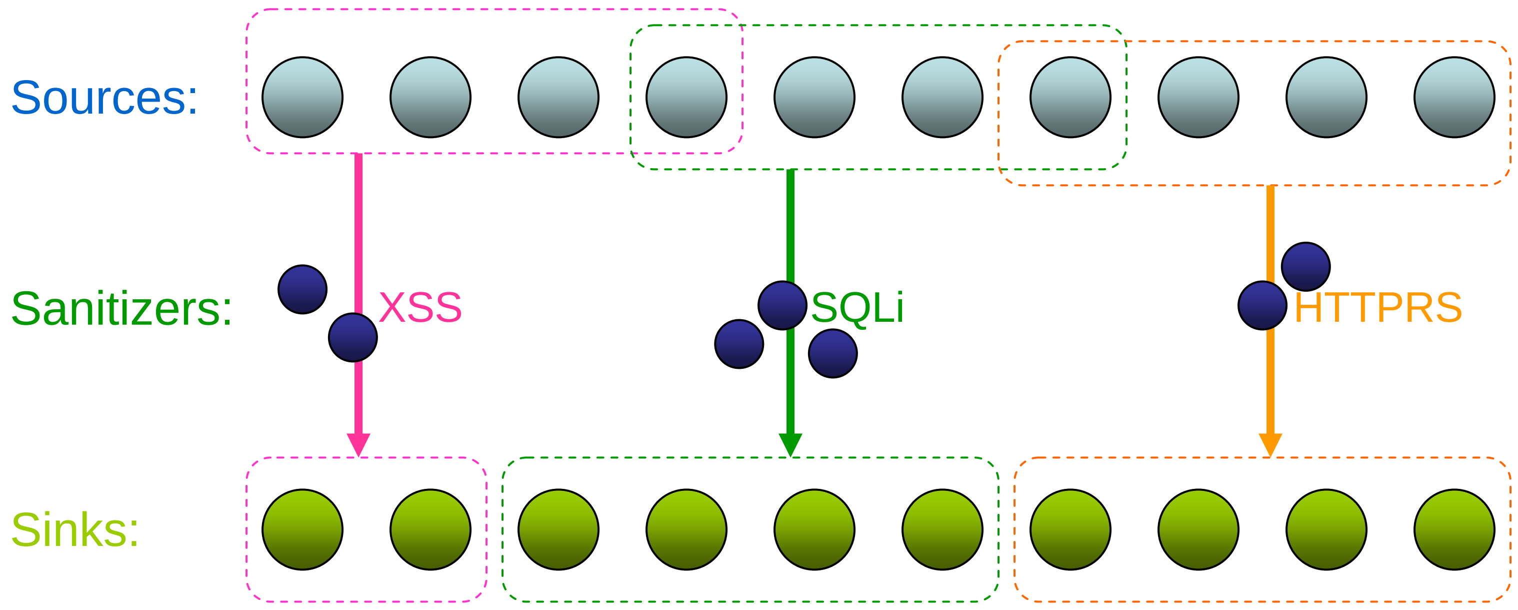
- **Build model**
  - Graph representing all data-flow and control-flow
- **Find ENTRY POINTS**
  - All public web-interfaces
- **Start search from SOURCES**
  - Find where data can flow into
- **Find if data can flow into SINKS**
- **Cut-off data-flow at SANITIZERS**

Models:
#1: Call Graph (CG)
#2: System Dependence Graph (SDG)

Search is done using a technique called *Program Slicing*

Taint Analysis Rules

- Groups of sources, sinks, sanitizers determine issue types



How White Box Scanners Work

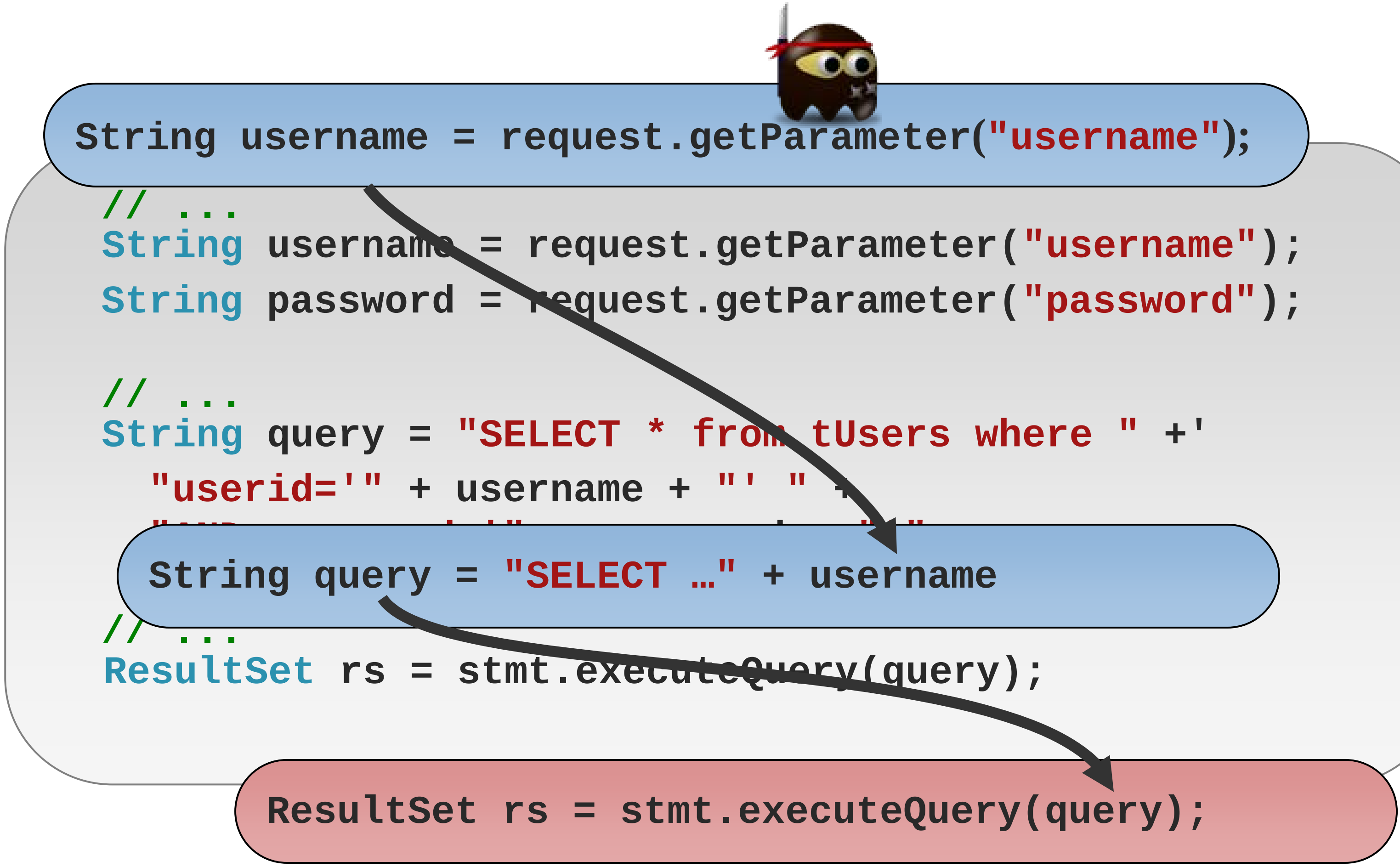
Source – a method returning tainted string

```
// ...
String username = request.getParameter("username");
String password = request.getParameter("password");

// ...
String query = "SELECT * from tUsers where " +
  "userid='" + username + "' " +
  "AND password='" + password + "'";

// ...
ResultSet rs = stmt.executeQuery(query);
```

User can change executed SQL commands

Sink - a potentially dangerous method

How White Box Scanners Work



```
String username = request.getParameter("username");
```

```
// ...
String username = request.getParameter("username");
String password = request.getParameter("password");

// ...
String query = "SELECT * from tUsers where " +'
  "userid='" + username + "' " +
```

```
String query = "SELECT …" + username
```

```
// ...
ResultSet rs = stmt.executeQuery(query);
```

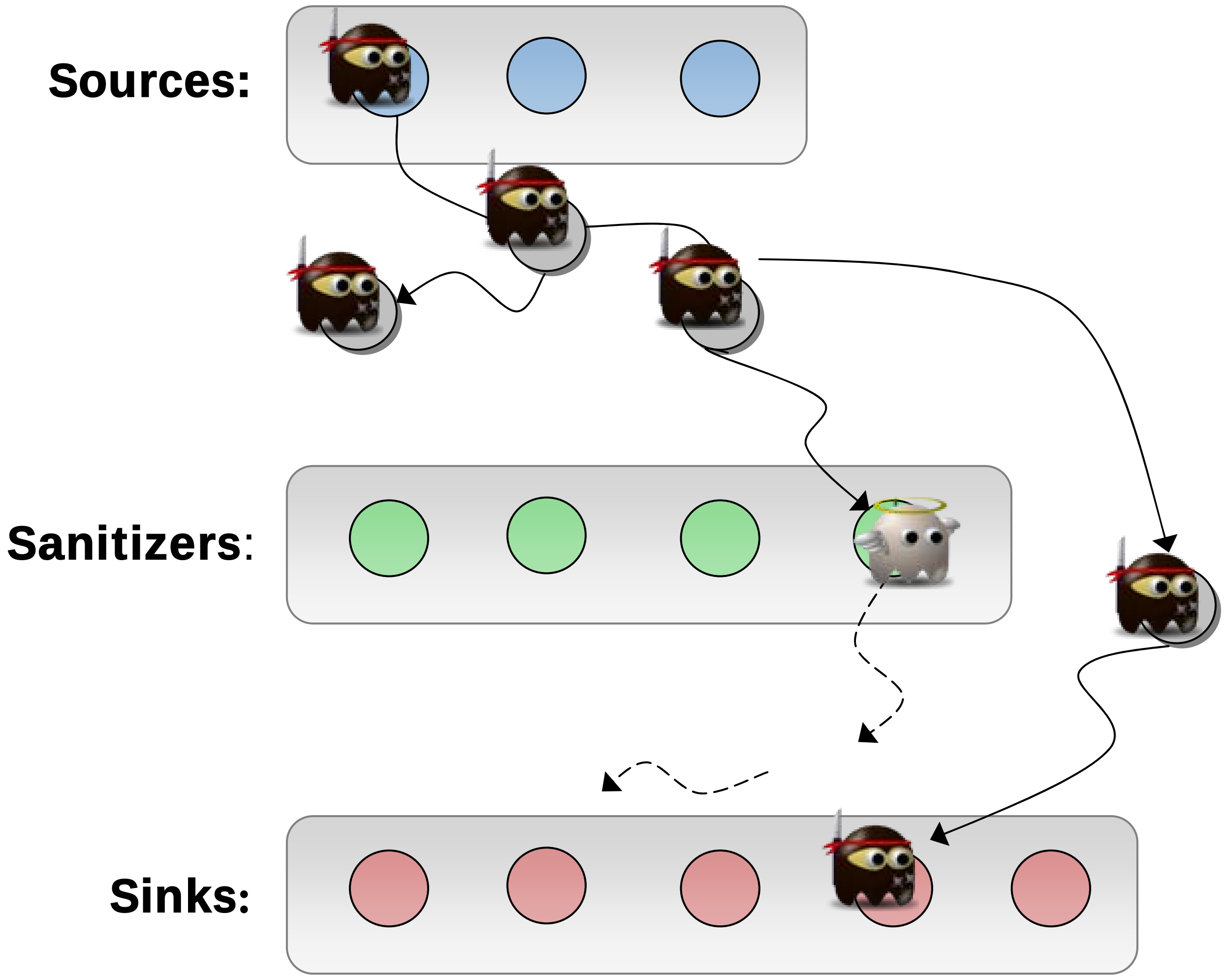
```
ResultSet rs = stmt.executeQuery(query);
```

A Common Fix (not the best one...)

```
// ...
String username = request.getParameter("username");
String password = request.getParameter("password");

// ...
String query = "SELECT * from tUsers where " +
   "userid='" + Encode(username) + "' " +
   "AND password='" + Encode(password) + "'";

// ...
ResultSet rs = s        ecuteQuery(query);
```

Sanitizer:
a method returning
a non-tainted string

How White Box Scanners Work



IBM’s String Analysis Technology

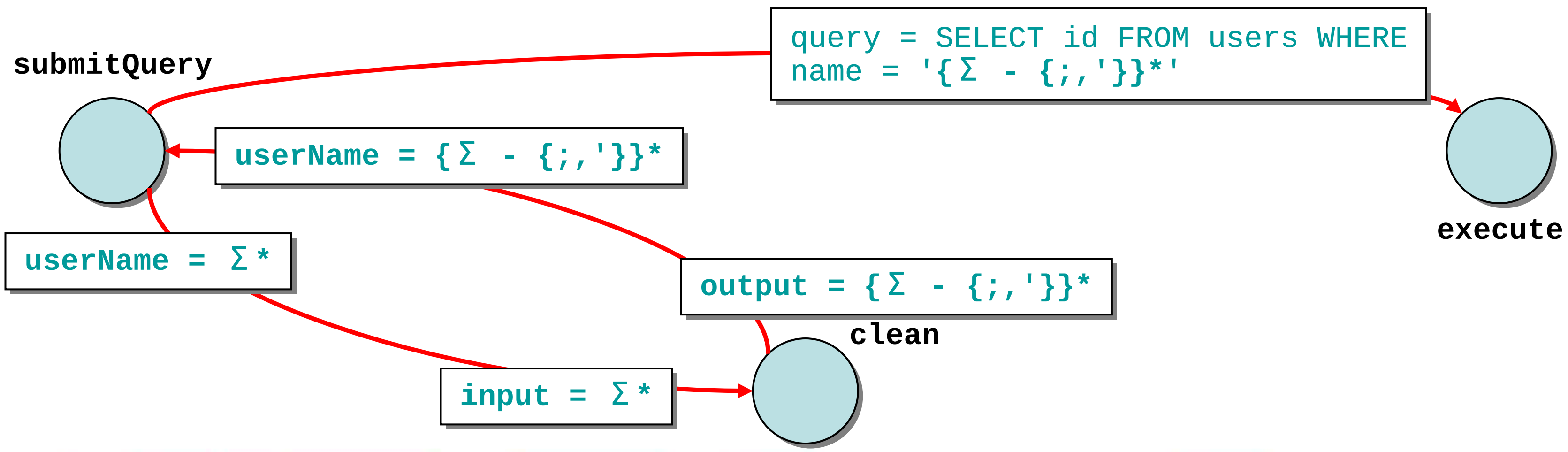
- The next generation of static analyzer technology
- Addresses High False Positive rate of Traditional Static Analyzers and their configuration requirements
- Automatically and statically detects the grammar of a string at the point of use

```
public void submitQuery(String userName) {
    userName = clean(userName);
    String query = "SELECT id FROM users WHERE name = '" +
          userName + "'";
    execute(query);
}
public String clean(String input) {
    return input.replaceAll(";","").replaceAll("'","");
}
```

```
input → .*

output → [~;’]*
```

How It Works

```
public void submitQuery(String userName) {
    userName = clean(userName);
    String query = "SELECT id FROM users WHERE name = '" +
          userName + "'";
    execute(query);
}
public String clean(String input) {
    String output = input.replaceAll(";","").replaceAll("'","");
    return output;
}
```



Advantages of String Analysis

IBM Tokyo Research Lab

- World’s smartest static analyzer
  - ✓No need to define what the sanitizers are
  - ✓Understands inline sanitization
  - ✓Understands validators
  - ✓Can verify your sanitizers really do what they’re supposed to

- What this means for you
  - Greater accuracy out-of-the-box
  - Less configuration
  - More reliable results
  - Easier to use

What is Dynamic Analysis?

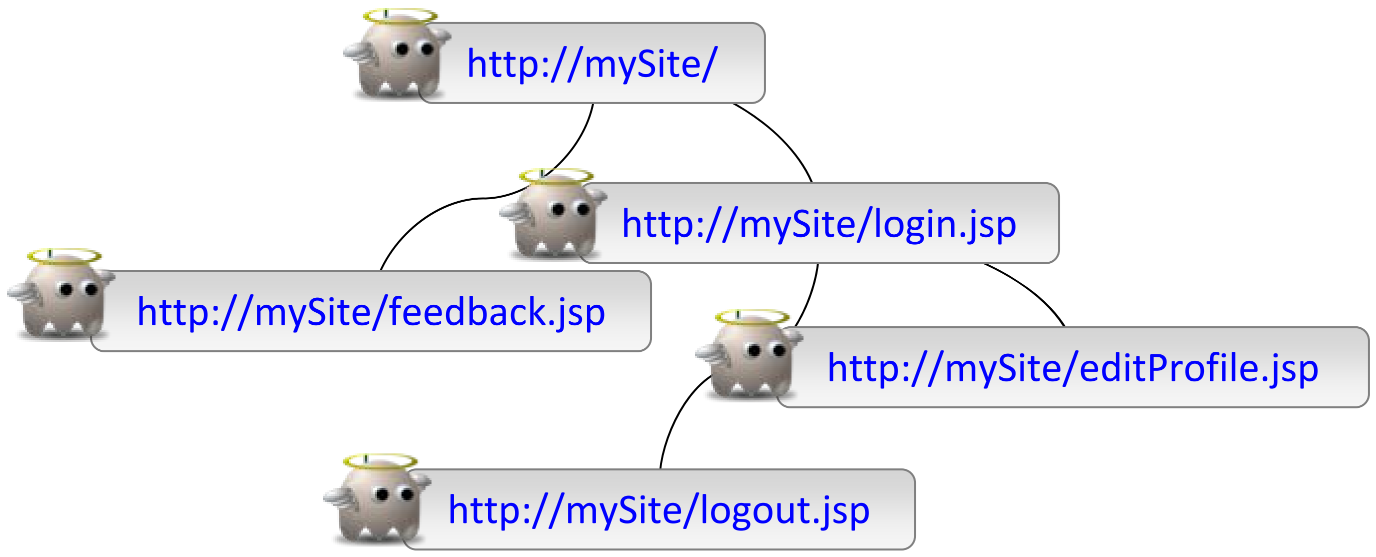
- Tests the web application while it is running
- Also know as Black Box testing, since it doesn’t know how the internals of the application works
- It works by first exploring the application to build its site model and determine the attack vectors
- It then tests the application by injecting calculated faults into HTTP(S) requests and analyzing the response for vulnerabilities

Dynamic Analysis

- The following values would be appended to the username parameter original value in order to test it for SQL injection
  - username=jsmith**’**
  - username=jsmith**\’**
  - username=jsmith**;**
  - username=jsmith **having 1=1--**
  - etc
- The test is validated if it causes a database exception in the response

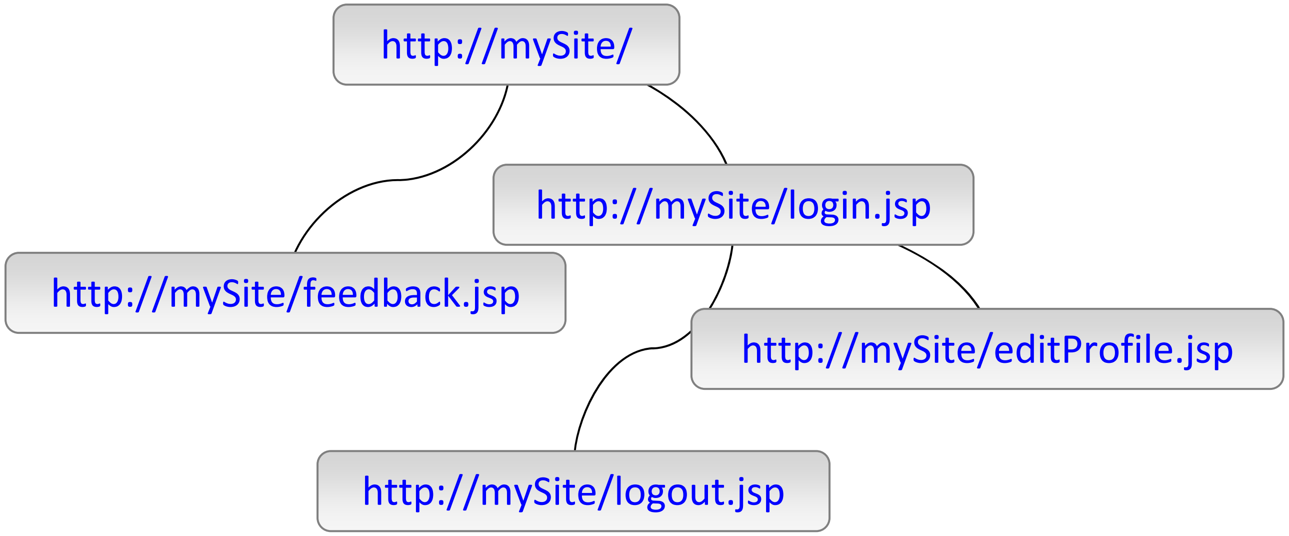
How Black-Box Scanners Work

- Stage 1: Crawling as an honest user



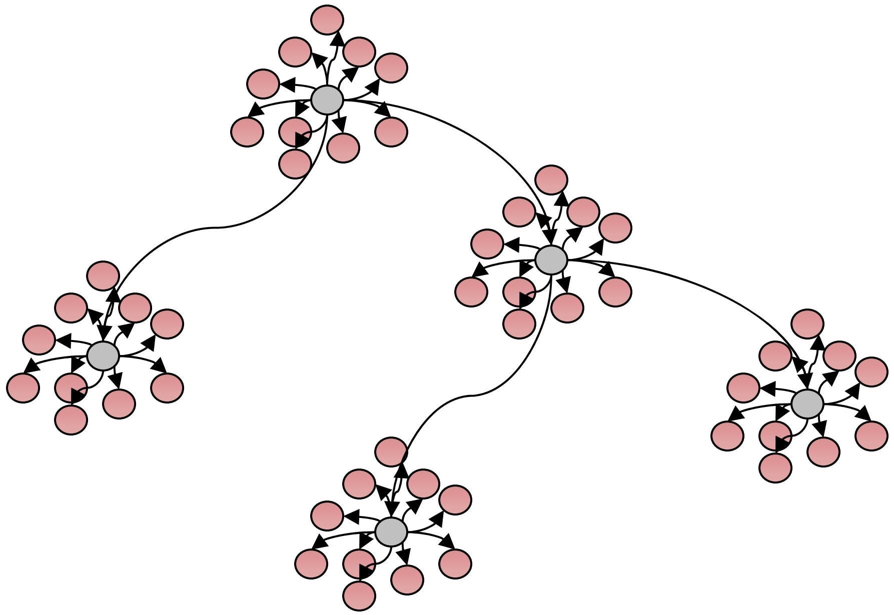
How Black-Box Scanners Work

- Stage 1: Crawling as an honest user



How Black-Box Scanners Work

- Stage 1: Crawling as an honest user
- Stage 2: Testing by tampering requests (ex. HTTP Request)



- Stage 3: Analyze response of system (ex. HTTP Response)
- Stage 4: Categorization

What is Run Time Analysis

- Run Time Analysis gives visibility into the internal working of an application while Dynamic Analysis is being performed
- Allows pin pointing of problem source code while performing Dynamic Analysis
- Works by monitoring the method invocation during black box testing

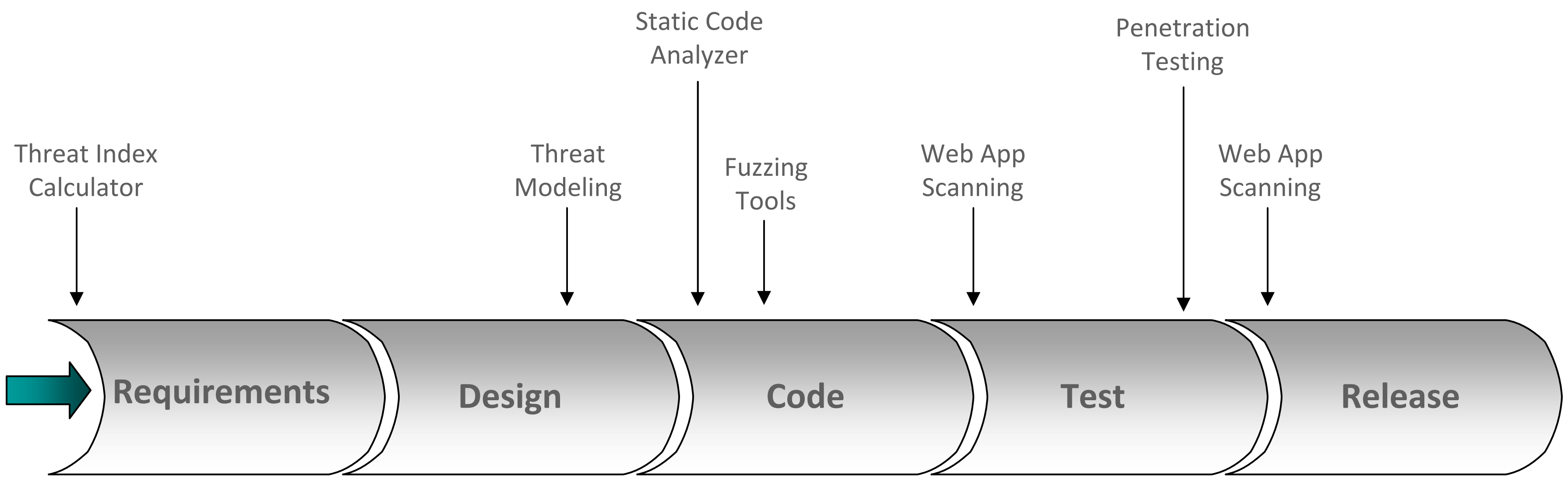
Agenda

- Security Landscape
- Common Vulnerabilities
- Analysis Techniques
- IBM Secure Software Engineering

What is Secure Engineering?

- SE goes beyond writing secure code. SE permeates the entire development process. We are ‘sprinkling’ security into:
  - Requirements
  - Design
  - Code/coding
  - Test/testing
  - Documentation
  - Serviceability
    - Specifically, education to Service and Support teams

Software Development Lifecycle



Threat Modeling

Microsoft TM Tool

Fuzzing Tools

Open Source

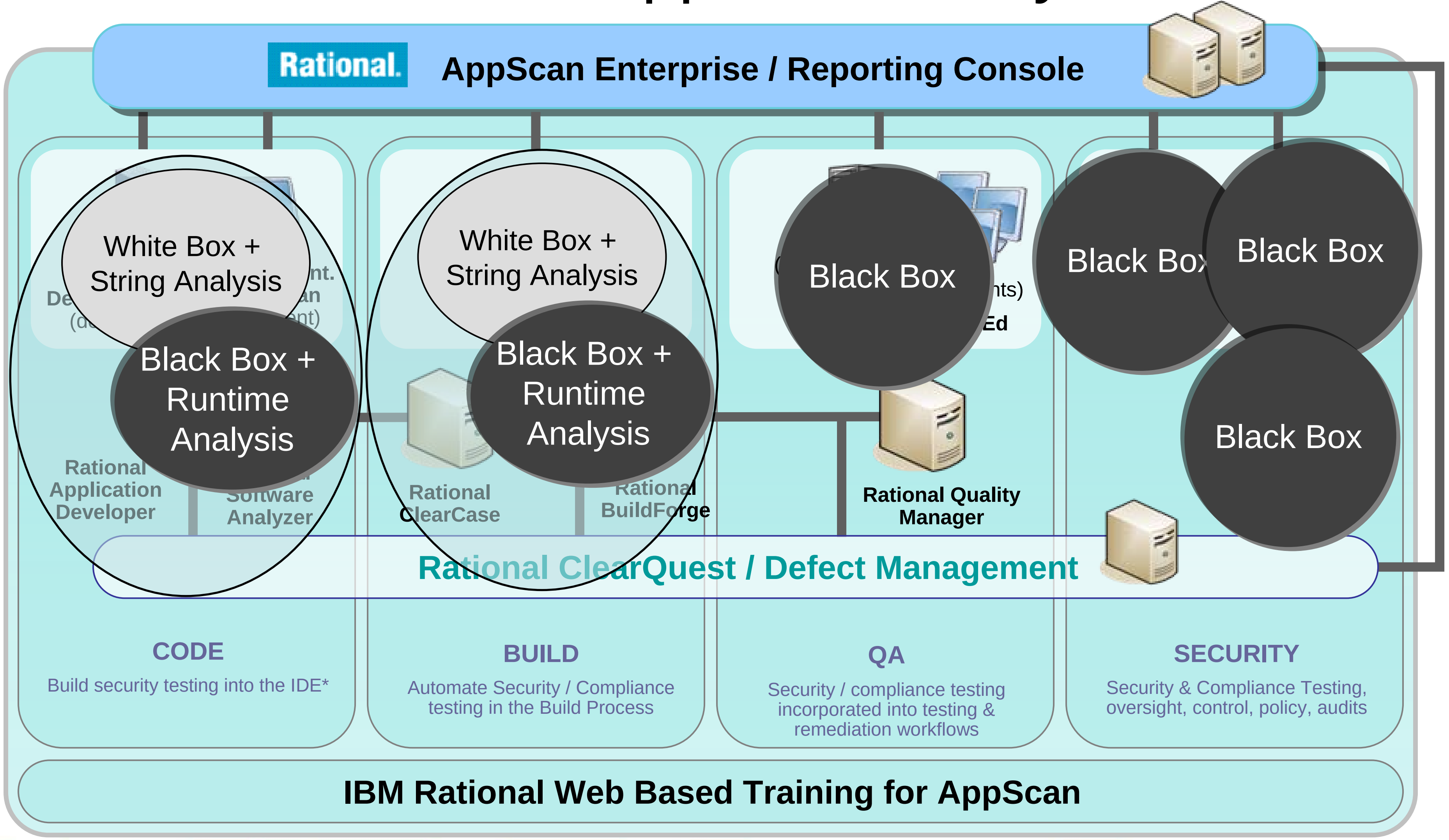
Static Code Analyzers

Rational AppScan Source

Web App Scanning

Rational AppScan Standard

IBM Rational AppScan Ecosystem


Rational.
AppScan Enterprise / Reporting Console
White Box + String Analysis
Black Box + Runtime Analysis
Rational Application Developer
Software Analyzer
White Box + String Analysis
Black Box + Runtime Analysis
Rational ClearCase
Rational BuildForge
Black Box
Rational Quality Manager
Black Box
Black Box
Black Box
Rational ClearQuest / Defect Management
CODE
Build security testing into the IDE*
BUILD
Automate Security / Compliance testing in the Build Process
QA
Security / compliance testing incorporated into testing & remediation workflows
SECURITY
Security & Compliance Testing, oversight, control, policy, audits
IBM Rational Web Based Training for AppScan

Grazie!