Fortify SCA 简介

Fortify SCA 是一个静态的、白盒的软件源代码安全测试工具。它通过内置的五大主要分析引擎：数据流、语义、结构、控制流、配置流等对应用软件的源代码进行静态的分析，分析的过程中与它特有的软件安全漏洞规则集进行全面地匹配、查找，从而将源代码中存在的安全漏洞扫描出来，并给予整理报告。扫描的结果中不但包括详细的安全漏洞的信息，还会有相关的安全知识的说明，以及修复意见的提供。

1．Fortify SCA 扫描引擎介绍：

Foritfy SCA 主要包含的五大分析引擎：

- 数据流引擎：跟踪,记录并分析程序中的数据传递过程所产生的安全问题。
- 语义引擎：分析程序中不安全的函数,方法的使用的安全问题。
- 结构引擎：分析程序上下文环境,结构中的安全问题。
- 控制流引擎：分析程序特定时间,状态下执行操作指令的安全问题。
- 配置引擎：分析项目配置文件中的敏感信息和配置缺失的安全问题。
- 特有的 X-Tier™跟踪器：跨跃项目的上下层次,贯穿程序来综合分析问题

2. Fortify SCA 的工作原理：

Foritfy SCA 首先通过调用语言的编译器或者解释器把前端的语言代码（如 JAVA，C/C++源代码）转换成一种中间媒体文件 NST（Normal Syntax Tree）将其源代码之间的调用关系，执行环境，上下文等分析清楚。然后再通过上述的五大分析引擎从五个切面来分析这个 NST，匹配所有规则库中的漏洞特征，一旦发现漏洞就抓取出来。最后形成包含详细漏洞信息的 FPR 结果文件，用 AWB 打开查看。

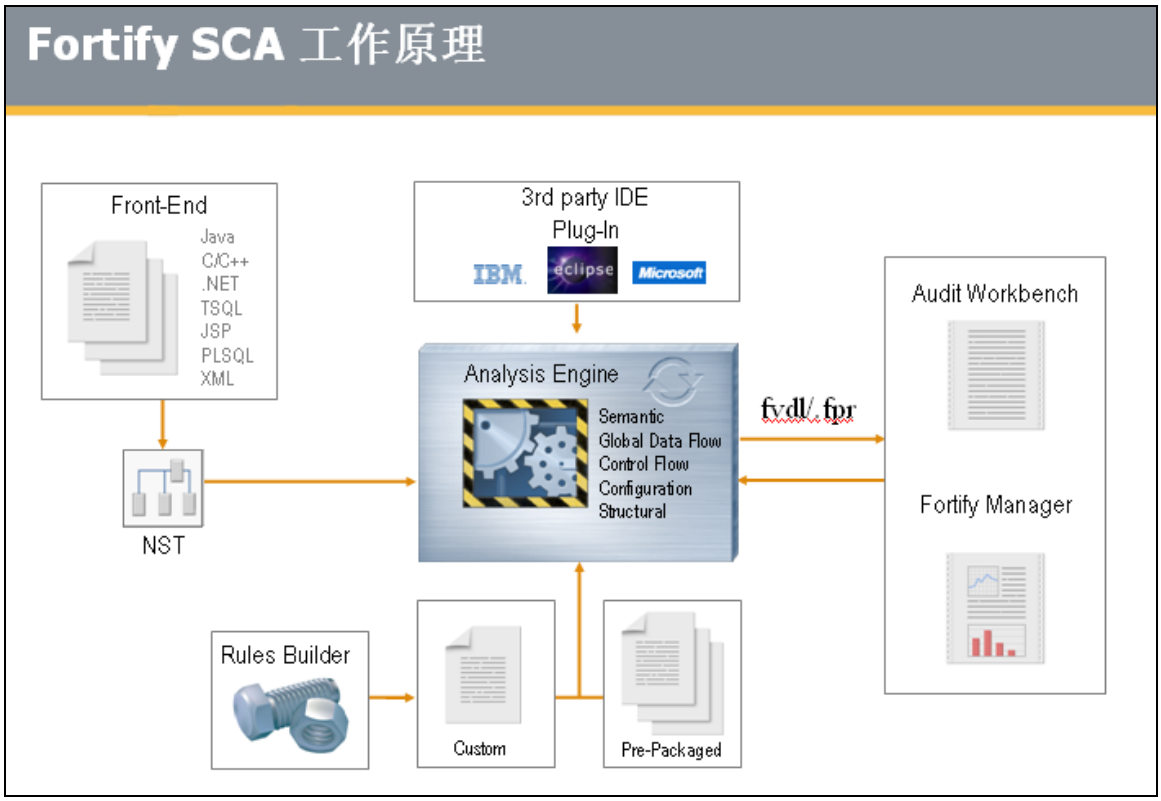

图 1：Fortify SCA 工作原理图

2．Fortify SCA 扫描的结果如下：

Fortify SCA 的结果文件为.FPR 文件，包括详细的漏洞信息：漏洞分类，漏洞产生的全路径，漏洞所在的源代码行，漏洞的详细说明及修复建议等。如下图：

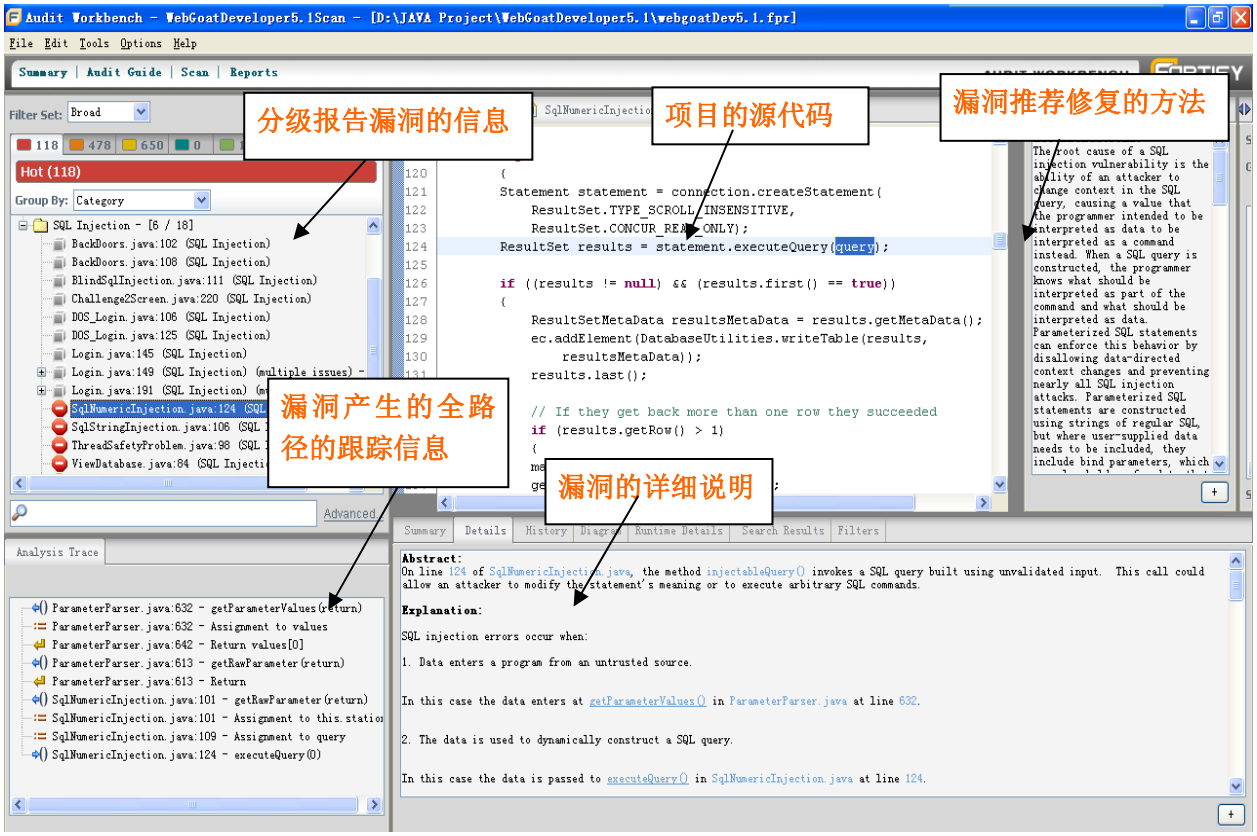

图 2：Foritfy AWB 查看结果图

3. Fortify SCA 支持的平台：

Fortify Source Code Analysis 支持以下平台和体系结构：

操作系统	版本	体系结构
HP-UX	11v1	
IBM AIX	5.2	
Linux	Fedora Core 7 Red Hat ES 4 和 5 Novelle SUSE 10	x86 和 x64
Mac OS X	10.4 和 10.5	PPC 和 x86
Sun™ Solaris™	8、9 和 10	SPARC
Windows	2000 2003 XP Vista	x86 x86 和 x64 x86 和 x64 x86

4. Fortify SCA 支持的编程语言：

Fortify Source Code Analysis 支持以下编程语言：

语言	版本
Adobe ColdFusion	5
.NET	1.1 和 2.0
C/C++	请参见“编译器”
Classic ASP	
Java	1.3、1.4、1.5 和 1.6
JavaScript	
PHP	
PL/SQL	
T-SQL	
VB for Applications	6
VB Script	

5．Fortify SCA plug-In 支持的有:

操作系统	IDE
Linux	Eclipse 3.2, 3.3 RAD 7 RSA 7
Windows	Eclipse 3.2, 3.3 RAD 6, 7 RSA 7 Visual Studio 2003, 2005
Mac OS X	Eclipse 3.2, 3.3 RAD 7 RSA 7

6．Fortify SCA 目前能够扫描的安全漏洞种类有：

目前Fortify SCA可以扫描出约 300 种漏洞，Fortify将所有安全漏洞整理分类，根据开发语言分项目，再细分为 8 个大类，约 300 个子类，具体详细信息可登录Fortify 官方网站 http://www.fortify.com/vulncat/ 进行查询：

Code Correctness: Call to GC.Collect()
Missing Check against Null
Object Model Violation: Just One of Equals() and GetHashCode() Defined
Often Misused: Authentication
Unchecked Return Value
Code Correctness: Class Implements ICloneable
Code Correctness: Missing [Serializable] Attribute

Code Correctness: Misspelled Method Name
Code Correctness: null Argument to Equals()
Dead Code: Unused Field
Dead Code: Unused Method
Null Dereference
Obsolete
Unreleased Resource
JavaScript Hijacking: Vulnerable Framework
Poor Logging Practice: Use of a System Output Stream
System Information Leak
Trust Boundary Violation
ASP.NET Misconfiguration: Request Validation Disabled
ASP.NET Misconfiguration: Trace Output
Poor Error Handling: Empty Catch Block
Poor Error Handling: Overly Broad Catch
Poor Error Handling: Program Catches NullReferenceException
Command Injection
Cross-Site Scripting
Denial of Service
HTTP Response Splitting
Log Forging
Path Manipulation
Resource Injection
SQL Injection
SQL Injection: NHibernate
Setting Manipulation
ASP.NET Bad Practices: Use of Impersonation Context
ASP.NET Misconfiguration: Persistent Authentication
Access Control: Database
Insecure Randomness
Password Management
Password Management: Hardcoded Password
Password Management: Weak Cryptography
Privacy Violation
ASP.NET Bad Practices: Non-Serializable Object Stored in Session
Code Correctness: Call to System.gc()
Code Correctness: Erroneous finalize() Method
EJB Bad Practices: Use of AWT/Swing
EJB Bad Practices: Use of Class Loader
EJB Bad Practices: Use of Sockets
EJB Bad Practices: Use of Synchronization Primitives
EJB Bad Practices: Use of java.io

J2EE Bad Practices: Sockets
J2EE Bad Practices: getConnection
Missing Check against Null
Missing Check for Null Parameter
Object Model Violation: Erroneous clone() Method
Object Model Violation: Just one of equals() and hashCode() Defined
Often Misused: Authentication
Poor Style: Explicit Call to finalize()
Statistical: Checked Return Value
Statistical: Function Return Unused
Statistical: Unassigned Return Value
Unchecked Return Value
Code Correctness: Call to Thread.run()
Code Correctness: Class Does Not Implement Cloneable
Code Correctness: Erroneous Class Compare
Code Correctness: Erroneous String Compare
Code Correctness: Misspelled Method Name
Code Correctness: null Argument to equals()
Dead Code: Expression is Always false
Dead Code: Expression is Always true
Dead Code: Unused Field
Dead Code: Unused Method
Null Dereference
Obsolete
Poor Style: Confusing Naming
Poor Style: Empty Synchronized Block
Poor Style: Identifier Contains Dollar Symbol ($)
Poor Style: Redundant Initialization
Poor Style: Value Never Read
Unreleased Resource: Database
Unreleased Resource: Streams
J2EE Bad Practices: Leftover Debug Code
JavaScript Hijacking: Ad Hoc Ajax
JavaScript Hijacking: Vulnerable Framework
Poor Logging Practice: Logger Not Declared Static Final
Poor Logging Practice: Multiple Loggers
Poor Logging Practice: Use of a System Output Stream
System Information Leak
System Information Leak: Missing Catch Block
Trust Boundary Violation
Unsafe Mobile Code: Access Violation
Unsafe Mobile Code: Inner Class

Unsafe Mobile Code: Public finalize() Method
Unsafe Mobile Code: Unsafe Array Declaration
Unsafe Mobile Code: Unsafe Public Field
Poor Error Handling: Empty Catch Block
Poor Error Handling: Overly Broad Catch
Poor Error Handling: Overly Broad Throws
Poor Error Handling: Program Catches NullPointerException
Poor Error Handling: Return inside Finally
Poor Error Handling: Unhandled SSL Exception
Command Injection
Cross-Site Scripting
Denial of Service
HTTP Response Splitting
Log Forging (debug)
Log Forging
Missing XML Validation
NUMBER Taint Sources
Path Manipulation
Process Control
Resource Injection
SQL Injection
SQL Injection: Hibernate
Setting Manipulation
Struts: Erroneous validate() Method
Unsafe JNI
Unsafe Reflection
Access Control: Database
Insecure Randomness
Password Management
Password Management: Hardcoded Password
Password Management: Password in Redirect
Password Management: Weak Cryptography
Privacy Violation
Code Correctness: Double-Checked Locking
J2EE Bad Practices: Non-Serializable Object Stored in Session
J2EE Bad Practices: System.exit
J2EE Bad Practices: Threads
Race Condition: Singleton Member Field
Race Condition: Static Database Connection
Session Fixation