# 代码审计

我司为XXXXXX提供信息系统所有代码进行整体的安全审计。发现（源）代码存在的安全漏洞，并对导致安全漏洞的错误代码进行定位和验证，提供修复方案。语言方面可以支持：Java，JSP，C，C++,.NET（C#）,XML，ASP，PHP，JS，VB等。运行环境支持：Windows，Red Hat Linux，Ubuntu，Centos，麒麟Linux等主流系统。

服务期内对：

* xxxxxx

提供1次代码审计，并提交相应次数的《（源）代码审计报告》。

## 1.1 代码审计服务内容

代码审计服务的范围包括使用Java，JSP，C，C++,.NET（C#）,XML，ASP，PHP，JS，VB等主流语言开发的B/S、C/S应用系统，以及使用XML语言编写的文件、SQL语言和数据库存储过程等，运行环境支持Windows，Red Hat Linux，Ubuntu，Centos，麒麟Linux等主流系统。

源代码安全审计服务从数据流分析、控制流分析、语义分析、配置分析、结构分析等五个方面全面分析软件源代码安全问题。

借助源代码分析工具，针对信息系统源代码扫描、分析，语言方面可以支持：Java/JSP C/C++, .NET平台,TSQL/PLSQL, Cold Fusion，XML，CFML，ASP，PHP，JS，VB等。操作系统方面支持：Windows, Solaris, Red Hat Linux, Mac OS X, HP-UX, IBM AIX等并对导致安全漏洞的错误代码进行定位和验证，提供修复方案。

## 1.2 代码审计服务参考标准

* CVE(Common Vulnerabilities & Exposures) 公共漏洞字典表
* OWASP（Open Web Application Security Project公共漏洞字典表
* 《软件安全开发标准》（ISO/IEC 27034）
* 《独立审计准则第20号-计算机信息系统环境下的审计》
* 《审计署关于印发信息系统审计指南的通知》（审计发【2012】11号）

## 1.3 审计分类

* **整体代码审计**

整体代码审计是指代码审计服务人员对被审计系统的所有源代码进行整体的安全审计，代码覆盖率为100%，整体代码审计采用源代码扫描和人工分析确认相结合的方式进行分析，发现源代码存在的安全漏洞。但整体代码审计属于白盒静态分析，仅能发现代码编写存在的安全漏洞，无法发现业务功能存在的缺陷。

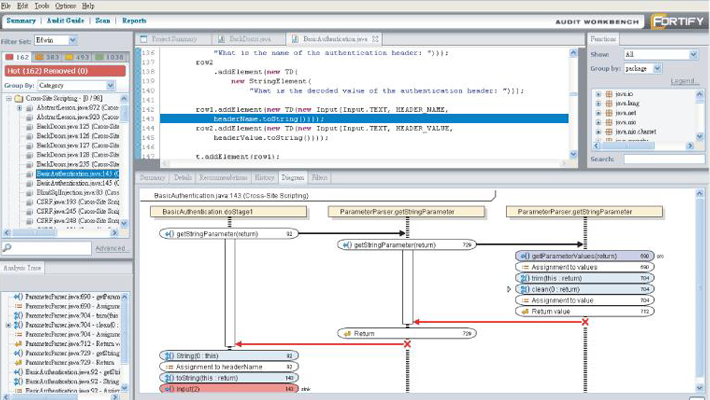
* **功能点人工代码审计**

功能点人工代码审计是对某个或某几个重要的功能点的源代码进行人工代码审计，发现功能点存在的代码安全问题。功能点人工代码审计需要收集系统的设计文档、系统开发说明书等技术资料，以便代码审计服务人员能够更好的了解系统业务功能。由于人工代码审计工作量极大，所以需要分析并选择重要的功能点，有针对性的进行人工代码审计。

## 1.4 审计工具

**Fortify SCA**

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。



## 1.5 代码审计实施流程

源代码审计服务主要分为四个阶段，包括代码审计前期准备阶段、代码审计阶段实施、复查阶段实施以及成果汇报阶段：

* **前期准备阶段**

在实施代码审计工作前，技术人员会和客户对代码审计服务相关的技术细节进行详细沟通。由此确认代码审计的方案，方案内容主要包括确认的代码审计范围、最终对象、审计方式、审计要求和时间等内容。

* **代码审计阶段实施**

在源代码审计实施过程中，技术人员首先使用代码审计的扫描工具对源代码进行扫描，完成初步的信息收集，然后由人工的方式对源代码扫描结果进行人工的分析和确认。

根据收集的各类信息对客户要求的重要功能点进行人工代码审计。

结合自动化源代码扫描和人工代码审计两方的结果，代码审计服务人员需整理代码审计服务的输出结果并编制代码审计报告，最终提交客户和对报告内容进行沟通。

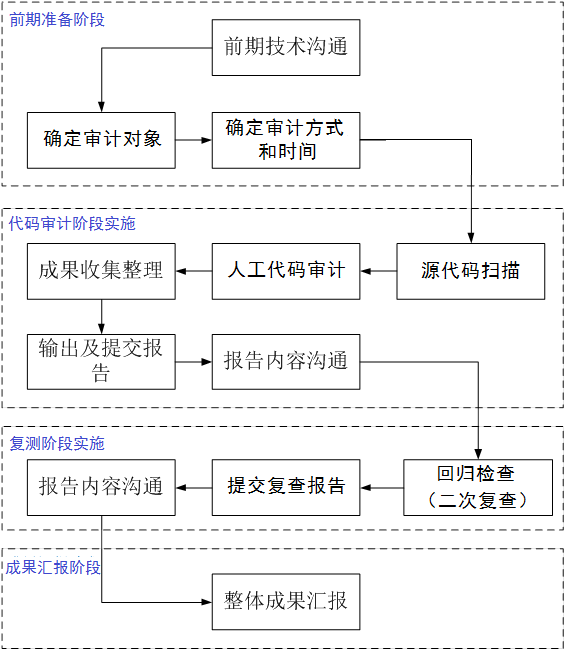
* **复测阶段实施**

经过第一次代码审计报告提交和沟通后，等待客户针对代码审计发现的问题整改或加固。经整改或加固后，代码审计服务人员进行回归检查，即二次检查。检查结束后提交给客户复查报告和对复查结果进行沟通。

* **成果汇报阶段**

根据一次代码审计和二次复查结果，整理代码审计服务输出成果，最后汇总形成《信息系统代码审计报告》。

图 代码审计服务流程



## 1.6 风险控制及输出成果

为避免风险的产生，源代码审计工作通常不会在生产或测试服务器上进行。XXXXXX信息中心需要提供源代码或存储源代码的计算机载体。代码审计服务人员会将一些代码审计工具安装在存储源代码的计算机载体中，在完成代码审计后卸载这些工具，以保护业务资产不受损害。

在代码审计过程中，确定代码审计服务人员和配合人员的联系方式，便于及时沟通并解决服务过程中的各类问题。

## 1.7 源代码审计重点

* **跨站请求伪装漏洞**

漏洞：提交表单中没有用户特有的标识。

影响：攻击者可利用跨站请求伪装 (CSRF) 漏洞假冒另一用户发出未经授权的请求，即恶意用户盗用其他用户的身份使用特定资源。

* **注入漏洞**

漏洞：对访问数据库的SQL语句没有进行任何过滤，可能导致SQL注入。

影响：如果SQL注入成功，攻击者可以获取网站数据库的信息，可以修改删除数据库，还可能获取执行命令的权限，进而完全控制服务器。

* **命令执行漏洞**

漏洞：系统中使用了一些调用操作系统函数的命令，在调用过程中，如果命令的来源不可信，系统可能执行恶意命令。

影响：攻击者有可能把要执行的命令替换成恶意命令，如删除系统文件。

* **日志伪造漏洞**

漏洞：将未经验证的用户输入写入日志。

影响：攻击者可以利用该漏洞伪造日志条目或将恶意内容注入日志。

* **参数篡改**

漏洞：一些重要参数可能会被篡改。

影响：攻击者能够通过篡改重要参数或方法对系统进行攻击。

* **密码明文存储**

漏洞：配置文件中存储明文密码。

影响：在配置文件中存储明文密码可能会危及系统安全，攻击者可以轻易获取到系统密码。

* **配置文件缺陷**

漏洞：配置文件内容存在缺陷，例如未设置统一的错误响应页面。

影响：攻击者能够利用配置文件的缺陷对系统进行攻击。

* **路径操作错误**

漏洞：用户输入没有有效的安全控制手段就直接对文件进行操作。

影响：攻击者可以控制路径参数，访问或修改其他受保护的文件。

* **资源管理**

漏洞：使用完资源后没有关闭，或者可能关闭不成功。

影响：攻击者有可能通过耗尽资源池的方式发起拒绝服务攻击，导致服务器性能降低，甚至宕机。

* **不安全的Ajax调用**

漏洞：系统存在不安全的Ajax调用。

影响：攻击者能够利用该漏洞绕过验证程序或直接编写脚本调用Ajax方法实现越权操作。

* **系统信息泄露**

漏洞：异常捕获泄露系统信息。

影响：攻击者可以从泄露的信息中找到有用信息，发起有针对性的攻击。

* **调试程序残留**

漏洞：代码包含调试程序，如：主函数。

影响：调试程序会在应用程序中建立一些意想不到的入口点被攻击者利用。

## 1.8 输出成果及客户收益

输出成果：

《信息系统代码审计报告》

本项服务给XXXXXXXXXXX带来收益如下：

1. 从提高系统的安全性及稳定性出发应该源代码进行质量控制，以保证源代码的质量；
2. 确保系统稳定，高效的运行，控制不利因素提升质量。