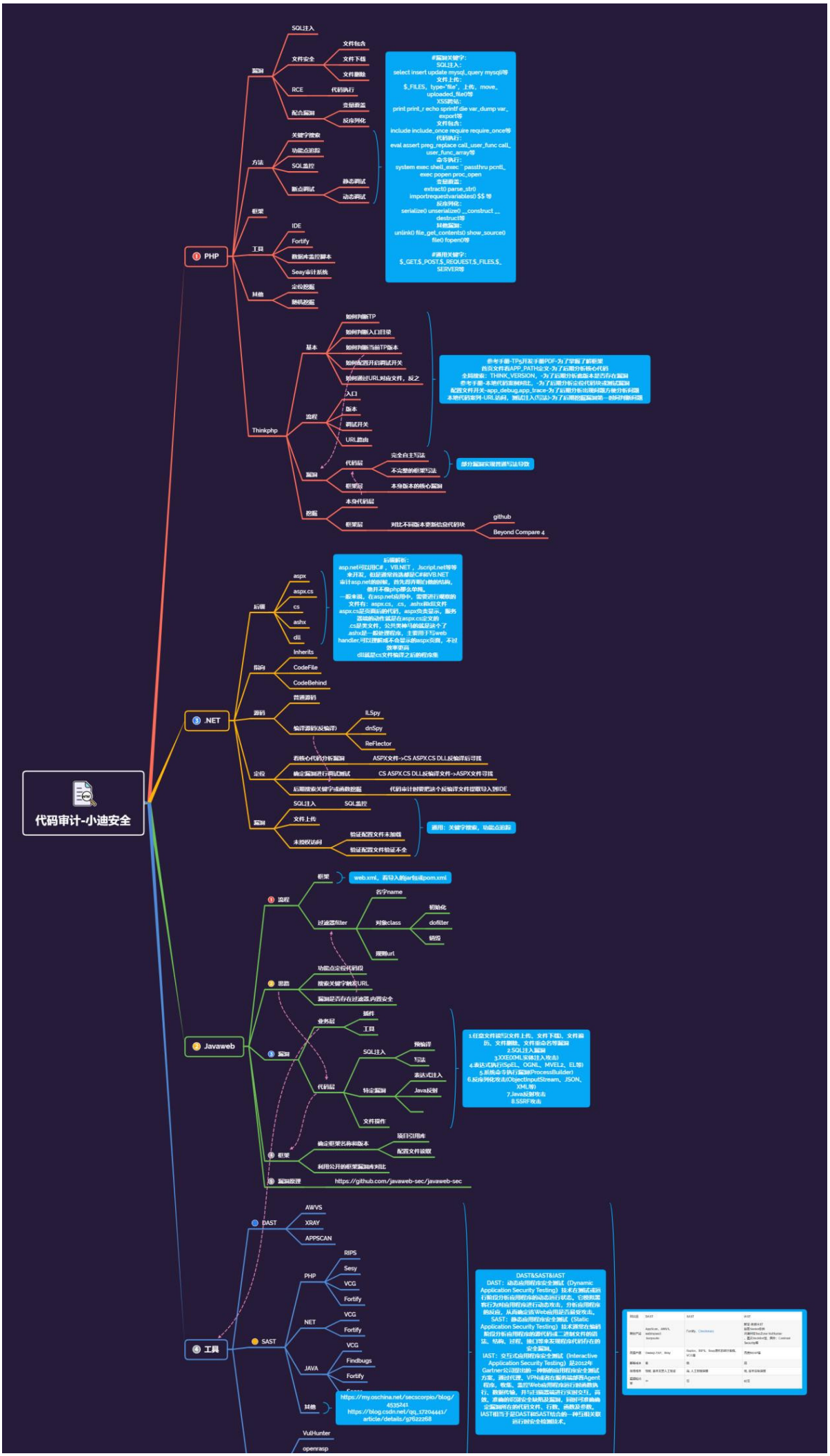
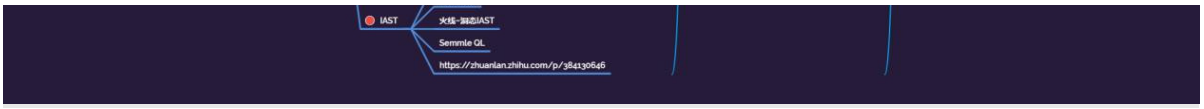
Day96　代码审计-SAST&IAST项目&火线洞态&Agent部署&插件安装&产品测评


代码审计-小迪安全
1 PHP
漏洞
SQL注入
文件安全
文件包含
文件下载
文件删除
RCE
代码执行
配合漏洞
变量覆盖
反序列化
方法
关键字搜索
功能点追踪
SQL监控
断点调试
静态调试
动态调试
#漏洞关键字:
SQL注入:
select insert update mysql_query mysqli等
文件上传:
$_FILES、type="file"、上传、move_uploaded_file等
XSS跨站:
print print_r echo sprintf die var_dump var_export等
文件包含:
include include_once require require_once等
代码执行:
eval assert preg_replace call_user_func call_user_func_array等
命令执行:
system exec shell_exec ` passthru pcntl_exec popen proc_open
变量覆盖:
extract() parse_str()
importrequestvariables() $$ 等
反序列化:
serialize() unserialize() __construct __destruct等
其他漏洞:
unlink() file_get_contents() show_source() file() fopen()等
#调用关键字:
$_GET,$_POST,$_REQUEST,$_FILES,$_SERVER等
框架
工具
IDE
Fortify
数据库监控脚本
Seay审计系统
其他
定位挖掘
随机挖掘
Thinkphp
基本
如何判断TP
如何判断入口目录
如何判断当前TP版本
如何配置开启调试开关
如何通过URL对应文件，反之
参考手册-TPs开发手册PDF-为了掌握了解框架
首页文件看APP_PATH定义-为了后期分析核心代码
全局搜索：THINK_VERSION，-为了后期对比查版本是否存在漏洞
参考手册-本地代码案例对比，-为了后期分析定位代码块或测试漏洞
配置文件开关-app_debug,app_trace-为了后期分析出现问题方便分析问题
本地代码案例-URL访问，测试注入(写法)-为了后期挖掘漏洞第一时间判断问题
流程
入口
版本
调试开关
URL路由
漏洞
代码层
完全自主写法
不完整的框架写法
部分漏洞实现需要写法导致
框架层
本身版本的核心漏洞
挖掘
本身代码层
框架层
对比不同版本更新信息代码块
github
Beyond Compare 4
3 .NET
后缀
aspx
aspx.cs
cs
ashx
dll
后缀解析:
asp.net可以用C#，VB.NET，Jscript.net等等来开发，但是通常首选都是C#和VB.NET
审计asp.net的前提，首先得弄明白他的结构，
他开不开php那么垃圾，
一般来说，在asp.net应用中，需要进行编译的文件有：aspx.cs，cs，ashx和dll文件
aspx.cs是页面后台代码，aspx负责显示，服务器端的动作 就是在aspx.cs定义的
.cs是类文件，公共类库与编程语言个了
ashx是一般处理程序，主要用于写web handler.可以理解成不含显示的aspx页面，不过效率更高
dll就是cs文件编译之后的程序集
指向
Inherits
CodeFile
CodeBehind
源码
普通源码
编译源码(反编译)
ILSpy
dnSpy
ReFlector
定位
看核心代码分析漏洞
ASPX文件->CS ASPX.CS DLL反编译后寻找
确定漏洞进行调试测试
CS ASPX.CS DLL反编译文件->ASPX文件寻找
后期搜索关键字或函数挖掘
代码审计时需把这个反编译文件提取导入到IDE
漏洞
SQL注入
SQL监控
文件上传
未授权访问
验证配置文件未加载
验证配置文件验证不全
通用：关键字搜索，功能点追踪
4 Javaweb
框架
web.xml，看导入的jar包或pom.xml
1 流程
名字name
过滤器filter
对象class
初始化
dofilter
销毁
规则url
2 思路
功能点定位代码段
搜索关键字触发URL
漏洞是否存在过滤器内置安全
3 漏洞
业务层
插件
工具
代码层
SQL注入
预编译
写法
特定漏洞
表达式注入
Java反射
文件操作
1.任意文件读写(文件上传、文件下载)、文件遍历、文件删除、文件重命名等漏洞
2.SQL注入漏洞
3.XXE(OML实体注入攻击)
4.表达式执行(SpEL、OGNL、MVEL2、EL等)
5.系统命令执行漏洞(ProcessBuilder)
6.反序列化攻击(ObjectInputStream、JSON、XML等)
7.Java反射攻击
8.SSRF攻击
4 框架
确定框架名称和版本
项目引用库
配置文件读取
利用公开的框架漏洞库对比
5 漏洞原理
https://github.com/javaweb-sec/javaweb-sec
5 工具
DAST
AWVS
XRAY
APPSCAN
SAST
PHP
RIPS
Sesy
VCG
Fortify
NET
VCG
Fortify
JAVA
VCG
Findbugs
Fortify
其他
https://my.oschina.net/secscorpio/blog/4535241
https://blog.csdn.net/qq_17204441/article/details/97822268
VulHunter
openrasp
DAST&SAST&IAST
DAST：动态应用程序安全测试（Dynamic Application Security Testing）技术在测试或运行阶段分析应用程序的动态运行状态。它模拟黑客行为对应用程序进行动态攻击，分析应用程序响应后，从而确定该Web应用是否容易受攻击。
SAST：静态应用程序安全测试（Static Application Security Testing）技术通常在编码阶段分析应用程序的源代码或二进制文件的语法、结构、过程、接口等实现程序代码存在的安全漏洞。
IAST：交互式应用程序安全测试（Interactive Application Security Testing）是2012年Gartner公司提出的一种新的应用程序安全测试方案，通过代理、VPN或者在服务端部署Agent程序，收集、监控Web应用程序运行时内部执行、数据传输，并与扫描器或进行实时交互，高效、准确的识别安全缺陷及漏洞，同时可准确地定位漏洞所在的代码文件、行数、函数及参数。IAST相当于是DAST和SAST结合的一种互相关联运行的安全检测技术。



1.知识点

- 1、代码审计-开源版&商业版
- 2、代码审计-单语言&多语言
- 3、代码审计-DAST&SAST&IAST

对比项	DAST	SAST	IAST
测试对象	Web应用程序	Web应用程序 APP的漏洞	Web应用程序 APP的漏洞
部署成本	低	低	高
使用成本	较低，基本无需人工验证	高，人工排除误报	低，基本没有误报
漏洞检出率	中	高	较高
脏数据	非常多	较少	几乎没有
研发流程集成	测试/线上运营阶段	研发阶段	测试阶段
误报率	低	高	极低（几乎为0）
测试覆盖度	低	高	高
检查速度	随测试用例数量稳定增加	随代码量呈指数增长	实时检测
逻辑漏洞检测	支持部分	不支持	支持部分
影响漏洞检出率因素	与测试payload覆盖度相关 企业可优化和扩展	与检测策略相关 企业可在定制策略	与检测策略相关 企业可定制测量
第三方组件漏洞检测	支持	不支持	支持
支持语言	不区分语言	区分语言	区分语言
支持框架	不区分框架	区分框架	区分框架
侵入性	较高，脏数据	低	低
风险程度	较高，扫挂/脏数据	低	低
漏洞详情	中，请求	较高，数据流+代码行数	高，请求+数据流+代码行数
CI/CD集成	不支持	支持	支持
持续安全测试	不支持	支持	支持
工具集成	无	开发环境集成 构建工具、问题跟踪工具	构建工具、自动化
其他	无法定位漏洞的具体代码行数和产生漏洞的原因		不支持C，C ++和Golang等语言

对比项	DAST	SAST	IAST
商业产品	AppScan、AWVS、webinpsect burpsuite	Fortify、Checkmarx	默安-雷鉴IAST 新思Seeker软件 开源网安SecZone VulHunter、墨云VackBot等，国外：Contrast Security等
开源产品	Owasp ZAP、Xray	Raptor、RIPS、Seay源代码审计系统、VCG等	百度RASP等
部署成本	低	低	高
使用成本	较低，基本无需人工验证	高，人工排除误报	低，基本没有误报
漏洞检出率	中	高	较高

2.Java审计知识点

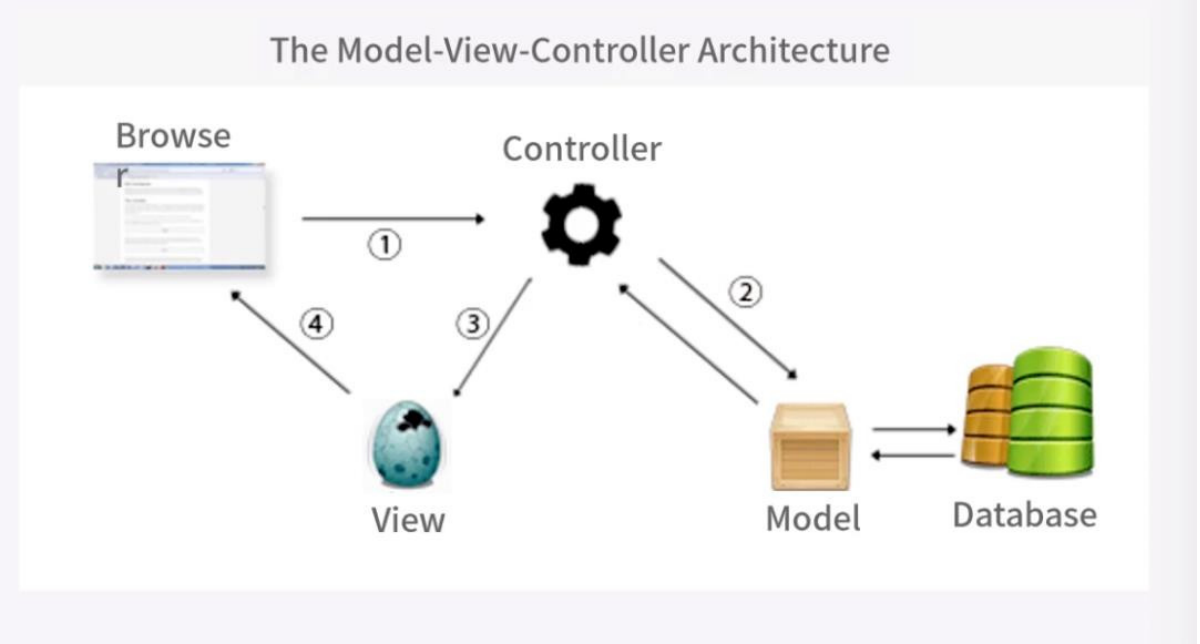
- https://xz.aliyun.com/t/7945 java代码审计常规思路和方法.pdf
- SQL注入，XSS跨站，RCE执行，反序列化，身份验证，SPEL，SSTI，三方组件安全等

3.详细点

- 1、代码审计必备知识点： 环境搭建使用，工具插件安装使用，掌握各种漏洞原理及利用,代码开发类知识点。
- 2、代码审计开始前准备： 审计目标的程序名，版本，当前环境(系统,中间件,脚本语言等信息),各种插件等。
- 3、代码审计挖掘漏洞根本： 可控变量及特定函数，不存在过滤或过滤不严谨存在绕过导致的安全漏洞。
- 4、代码审计教学计划： 审计项目漏洞原理->审计思路->完整源码->应用框架->验证并利用漏洞。
- 5、代码审计教学内容： PHP,Java,.NET,Python 网站应用，引入框架类开发源码，相关审计工具及插件使 用。

4.补充点

-MVC 模型：



当访问动态网页时，以 MVC 框架为例，浏览器提交查询到控制器（①），如是动态请求， 控制器将对应 sql 查询送到对应模型（②），由模型和数据库交互得到查询结果返回给控 制器（③），最后返回给浏览器（④）

-动态调试配置：phpStudy + PhpStorm + XDebug **https://blog.csdn.net/nzjdsds/article/details/100114242**

- 1、先确定 PHP 版本有 Xdebug
- 2、php.ini 配置写入并开启 Xdebug
- 3、PhpStorm 设置端口及 IDEY 并测试
- 4、PhpStorm 开启监听并运行断点访问

PHP5配置:**https://blog.csdn.net/weixin_40418199/article/details/79088365**
PHP7配置:**https://www.jb51.net/article/195840.htm**

-文件代码比对：Beyond Compare 4

-Javaweb身份验证访问控制：
开发做访问控制身份验证有几种技术方案实现：

- 1、传统代码-登录性判断文件代码看
- 2、Shiro框架引用-看配置看引用看外部库
- 3、Filter过滤器-看配置看过滤器目录分析代码
- 4、JWT技术-看看引用看外部库搜关键函数代码

审计此类漏洞：
搞清楚代码的验证方式

5.演示案例

5.1 代码审计利器-IAST-火线洞态测评报告

```
文档：https://doc.dongtai.io/docs/introduction
安装：https://github.com/HXSecurity/DongTai
控制台：https://iast.huoxian.cn/project/projectManage
测试台：https://labs.iast.huoxian.cn/#/images/index
主要测评：应用漏洞&组件安全
在线靶场测评：openrasp&spring
本地应用测评：IDEA_Plugin插件
Agent部署测评：DongTai OpenApi
1、IDEA_Plugin插件
https://github.com/HXSecurity/DongTai-Plugin-IDEA/releases
https://doc.dongtai.io/docs/getting-started/agent/plugin/java-agent-idea
2、Agent部署
```

```
13  https://doc.dongtai.io/docs/category/agent-%E5%AE%89%E8%A3%85%E6%8C%87%E5%8D%97
```

5.2 代码审计利器-IAST-SemmleQL测评报告

```
1  https://lgtm.com/ 代码平台接入测评
2  测评：https://lgtm.com/
```

5.3 补充说明

```
1  目前免费可试用的四个IAST平台：
2  1、openrasp-iastopenrasp-iast 是一款灰盒扫描工具，目前开源的IAST扫描器，通过安装Agent和扫描器，能够结合应用内部hook点信息，针对获取到的url请求参数进行fuzz，从而检测到安全漏洞。
3  支持的编程语言：Java、PHP。
4  2、VulHunter检测原理是通过在应用程序的字节码中动态插桩检测“探针”，来获取应用程序运行时的各种上下文信息。在应用程序运行时，实时分析程序的安全弱点。与基于SAST和DAST技术的产品相比，VulHunter的最大不同点是，通过字节码插桩应用程序获得更多准确的运行时信息。
5  支持的编程语言：java、node.js。
6  3、火线~洞态IAST洞态IAST提供SAAS平台，个人用户通过填写问卷注册登录，下载Agent进行应用程序部署，正常访问应用，就可以触发漏洞检测。漏洞结果提供比较详细的HTTP数据包和污点流图，可用于快速验证和复现漏洞。
7  支持的编程语言：Java、PHP，Python，Go。
8  4、Semmle QLSemmle公司声称以一种独特的方法寻找代码中的漏洞。技术核心是把代码当成数据，将分析问题变成对数据库的请求。SemmleQL是一个声明式的面向对象的查询语言。
9  支持的编程语言：Java，JavaScript，Python，TypeScript，C#，Go，C/C ++。
```