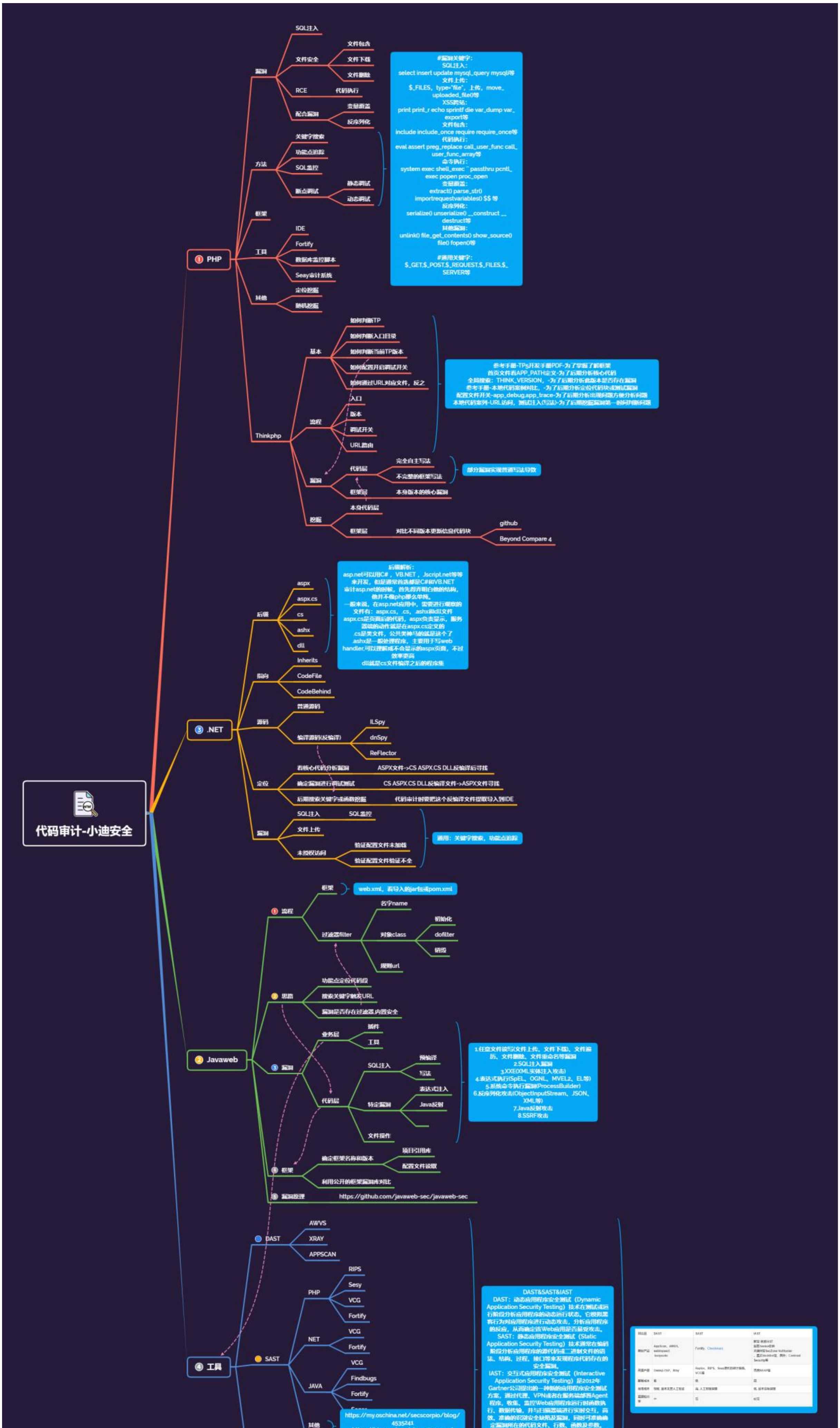
代码审计-SAST&IAST 项目&PHP&Java&NET&Python&Js&Go 等测评


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
#通用关键字:
$_GET,$_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-TP5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, -为了后期分析此版本是否存在漏洞
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
部分漏洞实现普通写法导致
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的时候, 首先得弄明白他的结构, 他并不像php那么单纯。
一般来说, 在asp.net应用中, 需要进行观察的文件有: aspx.cs, .cs, .ashx和dll文件
aspx.cs是页面后的代码, 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, 公共类神马的就是这个了
.ashx是一般处理程序, 主要用于写web handler,可以理解成不会显示的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
代码审计时要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
2 Javaweb
1 流程
框架
web.xml, 看导入的jar包或pom.xml
过滤器filter
名字name
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(XML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
4 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
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, 分析应用程序的反应, 从而确定该Web应用是否易受攻击。
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、接口等来发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应用程序运行时函数执行、数据传输, 并与扫描器端进行实时交互, 高效、准确的识别安全缺陷及漏洞, 同时可准确确定漏洞所在的代码文件、行数、函数及参数。
DAST
SAST
IAST

#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

#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

#章节点：

1、语言审计-PHP&.Net&Java&Python

2、漏洞审计-注入&上传&RCE&未授权等

3、框架审计-ThinkPHP&Spring&Flask 等

4、工具审计-RIPS&VCG&Fortify&Bandit 等

5、技术审计-动静态调试&DAST&SAST&IAST 等

#简要点：

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

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-雳鉴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

演示案例：

➢ 代码审计利器-SAST-单语言

➢ 代码审计利器-SAST-多语言

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，分析应用程序的反应，从而确定该 Web 应用是否易受攻击。

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、接口等来发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 应用程序运行时函数执行、数据传输，并与扫描器端进行实时交互，高效、准确的识别安全缺陷及漏洞，同时可准确确定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时安全检测技术。

目前还有些商业版平台未介绍如下：

静态：CheckMarx 奇安信代码卫士等

IAST：悬镜灵脉 IAST 默安雳鉴 IAST 等

PHP -Seay RIPS CheckMarx Fortify VCG Kunlun-M

NET -VCG Fortify CheckMarx

Java-Fortify CheckMarx

Python-Bandit CheckMarx

JS-Kunlun-M NodeJsScan CheckMarx

Go-Gosec CheckMarx

Bandit

参考：https://bandit.readthedocs.io/

安装：pip install bandit

linux:

安装后会在当前 Python 目录下 bin

使用：bandit -r 需要审计的源码目录

windows:

安装后会在当前 Python 目录下 script

使用：bandit -r 需要审计的源码目录

D:\Python3\Scripts>bandit.exe -r F:\python_webapp\www\

Kunlun-M

1、安装依赖库：pip install -r requirements.txt

2、配置文件启用：cp Kunlun_M/settings.py.bak Kunlun_M/settings.py

3、初始化数据库：python kunlun.py init initialize

4、加载规则数据库：python kunlun.py config load

Web 使用：D:\Python38\python.exe kunlun.py web -p 9999

Cli 使用：D:\Python38\python.exe kunlun.py scan -t

涉及资源：

补充：涉及录像课件资源软件包资料等下载地址