由交互式扫描联想到的实时漏洞感知方法

Auth : Cryin'
Date : 2016.05.03
Link : https://github.com/Cryin/Paper

概述

最近刚刚完成了交互式应用程序安全测试(IAST)功能的设计与开发工作，通过对 IAST 相关技术的研究，发现交互式扫描作为 WEB 安全扫描器的一种新型技术，其意义不仅仅是提高漏洞检测率、减少误报率；也让我联想通过交互式扫描技术实现实时漏洞感知的可行方法。

本文将详细介绍交互式扫描技术实现细则，并从 IAST 技术出发，讲述利用交互式扫描技术实现实时漏洞感知并进行实例演示。

IAST 技术原理

交互式应用程序安全测试(IAST—interactive application security testing) 实时动态交互的漏洞检测技术，搜集、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 结合的一种相互关联的运行时安全检测技术。。

IAST 扫描可以发现比传统的 Web 应用程序扫描器更多的安全漏洞，而且产生误报少。此外，交互式扫描可以清楚的告诉你漏洞代码的具体位置，非常便于开发人员快速确定漏洞产生原因并进行修复，国外已经有数款传统扫描器(AWVS、Appscan、seeker)厂商增加了该功能模块或开发了独立的交互式扫描器。高效、准确的漏洞扫描已经得到了国外数家公司的验证和认可。并且国内目前并没有真正意义上支持交互式扫描的安全厂商。

IAST 实现

如上所述，IAST 技术的实现思想是要在服务端部署 Agent 端，也就是说需要根据不同的语言平台分别开发设计 Agent 端，分别为 PHP、JAVA、.NET。由于平台语言不同，骑 Agent 设计、实现细节也不尽相同，但其大致原理基本相同，所以这里以 PHP 程序为例，详细介绍交互式扫描技术对于 sql 注入漏洞的扫描原理及实现。

- Agent 运行方法：

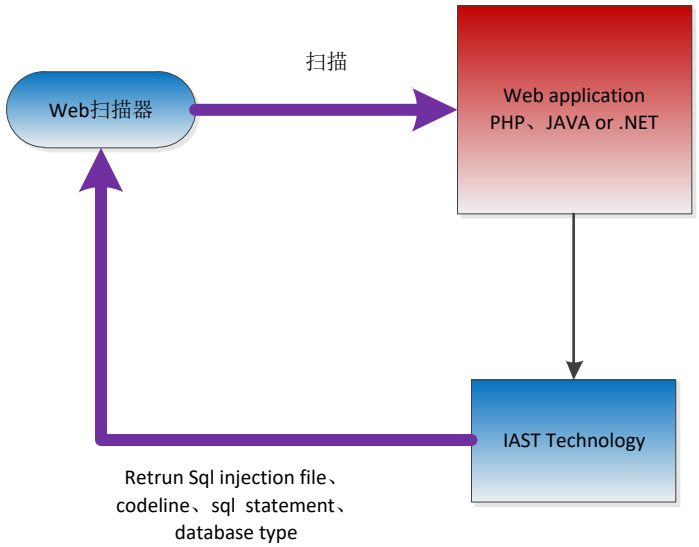
  在 PHP 的配置文件 php.ini 中有个选项 auto_prepend_file，通过这两个选项来设置页眉和脚注，等同于在页面中使用 require 函数。这样就可以保证在每个 PHP 页面中载入我们的 Agent，从而进行与扫描器的交互工作。

- 函数 HOOK 及漏洞检测方法：

Agent 在开始工作后，会对当前页面及 include 的所有页面文件源代码进行解析。将需要进行监控的函数如 sql_query 等关键函数替换为自定义函数完成 hook 操作，并将页面代码保存至 tmpfile 临时文件，通过文件内容及名称的 hash 值确保该临时文件的唯

一性。在 hook 函数时，同时获取当前 sql 查询函数的参数信息，即 sql 语句，Agent 检测 sql 查询语句中是否包含交互式扫描特征，若存在该特征则格式化程序文件、行号、sql 语句到字符串，使用 php 函数 eval 执行临时 php 文件，并将格式化的信息显示到响应页面中返回给扫描器端，由扫描器进一步判断检测是否是 sql 注入漏洞并报出漏洞。

◆ 流程设计:



◆ SQL 注入交互式扫描检测实例:

编写 mysql 查询测试页面，代码如下:

```
function query($query)
{
    $result=mysql_query($query);
    return $result;
}
if(isset($_GET["bookname"]))
{
    $bookname=$_GET["bookname"];
    if(!empty($bookname))
    {
        $query="select * from book where bookname = '$bookname'";
        $result=query($query);
        if($result)
        {
            $result_row=mysql_fetch_row(($result));
            if($result_row)
            {
                echo "十年生死两茫茫<br>";
                echo "不思量，  自难忘<br>";
            }
        }
    }
    else
    {
        echo "IAST TEST!";
    }
}
```

其中参数 bookname 未进行任何过滤及处理，存在 sql 注入漏洞。将 PHP Agent 端按照 Agent 运行方法部署 php 文件并重新启动 Apache，请求端发送 sql 注入检测请求，如下：

http://10.65.10.195/iast/index.php?bookname=xs1WISSTART%27%22elqZV%27WISEND

按照 PHP Agent 的实现，如果检测到 sql 注入漏洞，则将存在 sql 注入的程序文件、行号、sql 语句、数据库类型都返回到响应内容中，如图所示上述请求的响应内容如下:

```
<!DOCTYPE html>
<html>

    <head>
        <meta http-equiv="content-type" content="text/html; charset=utf-8" />
        <title>sql injection - iast test</title>
    </head>
    <body>
        IAST TEST!
    </body>

</html>
<!--WISASPECT:
00000009SQL_Query000000010000003Fselect * from book where bookname = 'xs1WISSTART'"elqZV'WISEND'
0000002ED:/software/PHPnow-1.5.6/htdocs/iast/index.php0000001Aa0000000200000019"mysql_query" was called.
00000000Edatabase=mysql
WISASPECTSPLIT
-->
```

可以看到响应内容末尾信息中，包含了存在 sql 注入的文件为：D:/software/PHPnow-1.5.6/htdocs/iast/index.php；sql 查询漏洞所在代码的行号为十六进制数 1A 即第 26 行；sql 查询语句为:select * from book where bookname = 'xs1WISSTART'"elqZV'WISEND'；数据库类型为 mysql。

至此，我们已经完整的演示了 IAST 技术扫描一个 sql 注入漏洞的完整过程。其它比如代码注入、命令执行等常见漏洞的检测方法也基本类同。

◆ 真实的代码执行漏洞检测实例:

Seacms V6.26 存在远程代码执行漏洞(注该漏洞已提交某第三方漏洞平台)，文件 include/main.class.php 第 3107 行代码中$strIf 变量可控导致带入的代码直接在 eval 函数中执行。

```
3103                if (strpos($strThen,$labelRule3)>=0){
3104                    $elsearray=explode($labelRule3,$strThen);
3105                    $strThen1=$elsearray[0];
3106                    $strElse1=$elsearray[1];
3107                    @eval("if(".$strIf.'){\$ifFlag=true;}else{\$ifFlag=false;}");
```

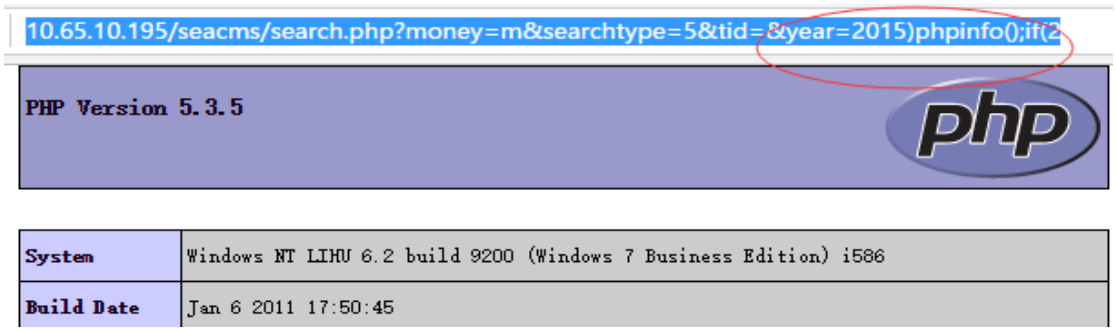
注意代码@eval("if(".$strIf."){\$ifFlag=true;}else{\$ifFlag=false;}");
其中参数 letter、ver、year、jq、area 都能进入该函数，比如 letter=E 传入该函数时会出现如下情况：

if(E==E){\$ifFlag=true;}else{\$ifFlag=false;}");

其它如 year=2015 传入该函数时代码会出现如下情况：

if(2015==2015){\$ifFlag=true;}else{\$ifFlag=false;}");

所以这里只要拼接下就能注入任意代码：比如传入参数$strIf 为 2015)phpinfo();if(2 时，代码会变成这样：
@eval(“if(2015==2015)phpinfo();if(2){$ifFlag=true;}else{$ifFlag=false;}}”)



如果使用传统的动态扫描技术是无法扫描出该漏洞的，因为要正确拼接、闭合双引号内的 php 代码才能执行代码并返回特征到响应内容中。扫描器根据特征才能判断是否存在漏洞。

但是如果使用交互式扫描技术检测的话，只要对 eval 函数进行监控，当参数包含 php 代码执行的特征并且能够进入到最终 eval 函数并执行，那即可确定存在代码执行漏洞。

何为漏洞感知

漏洞感知是近一两年才被一些安全厂商和安全研究人员提出来的一个 Web 漏洞检测、预警的新概念或者新技术，行业内并没有对漏洞感知有明确、详细的定义。在百度搜索关键词仅有三家厂商有该类型产品或相关产品中提到了漏洞感知关键词，如图:

从这些产品及网上的资料，我理解的漏洞感知即实时感知常见安全漏洞并通过短信、邮件、微信等方式进行预警的一个安全系统。

由交互式扫描联想到的漏洞感知实现方法

从交互式扫描技术原理及实现细节可以了解到，不论是 PHP、JAVA、.NET 不同的语言程序，其交互式扫描的总体原理均是在服务器端部署 Agent，扫描器开启交互式扫描功能。Agent 获取对漏洞检测有价值的信息(比如数据库查询执行时的 sql 语句)，并返回给扫描器，由扫描器端对这些信息进行分析、处理从而判断是否存在响应漏洞并报出漏洞。

所以，很容易想到的是如果我丰富 Agent 的功能，将漏洞分析、处理、判断等工作都在 Agent 完成，则不需要扫描器专门去扫描该站点程序，也完全不需要再将信息返回给扫描器客户端处理。在部署了 Agent 的站点如果受到了真实的 web 攻击(如 sql 注入)，Agent 端检测出漏洞后就可以即时报出漏洞并做出预警。这样就实现了一个基本的漏洞感知功能。

扫描器与漏洞感知的结合

考虑到交互式扫描的高检测率、误报少的优势，结合漏洞感知不仅可以很好的应用到白盒测试应用场景帮助开发人员发现并修复漏洞，而且也可以在程序运营正常使用过程中感知漏洞攻击威胁并进行预警。扫描器在请求 header 中添加特征字段，并增加密码校验等机制。Agent 端对 Header 进行对比、校验。如果是正常的 http 请求则不开启交互式扫描仅运行漏洞感知功能。这样也不会影响程序的正常运行和访问。

总结

交互式扫描技术目前国内还没有涉足的安全厂商以及扫描器，但势必接下来几年会陆续有厂商跟进。漏洞感知的概念已经被数家新型公司炒作话题。但相关漏洞感知产品还不成熟，具体实现的技术方法也不尽相同。有些厂家也只是将传统扫描器经过包装，通过实时不间断扫描站点漏洞而实现漏洞的实时感知。但其思路并没有新颖之处，交互式扫描的特点是检测率高、误报少。还需要在服务器端部署 Agent 端。经过对 Agent 进一步丰富开发并结合漏洞感知的功能，可以在网站开发过程中即使发现漏洞并提交工单帮助开发人员修复漏洞，同时也可以在网站程序上线后继续运行，实时感知安全漏洞、攻击威胁。对于研发公司在代码开发阶段、代码安全测试阶段有较大的意义，相信后续市面上也会出现类似的产品。

参考

[1] Interactive Application Security Testing (IAST) | White Paper
[2] AcuSensor Technology: Ups detection rate, reduces false positives
[3] Evolution of Application Security Testing: From Silos to Correlation and Interaction
[4] Synopsys Software Integrity Seeker