PHP弱类型你真的懂了吗？

1. 前言

最近看了几篇关于PHP弱类型的文章，发现虽然弱类型相关的漏洞比较容易，但这类问题是比较容易让人忽略的，所以我对PHP弱类型的漏洞做了一个小结。

1. 概念

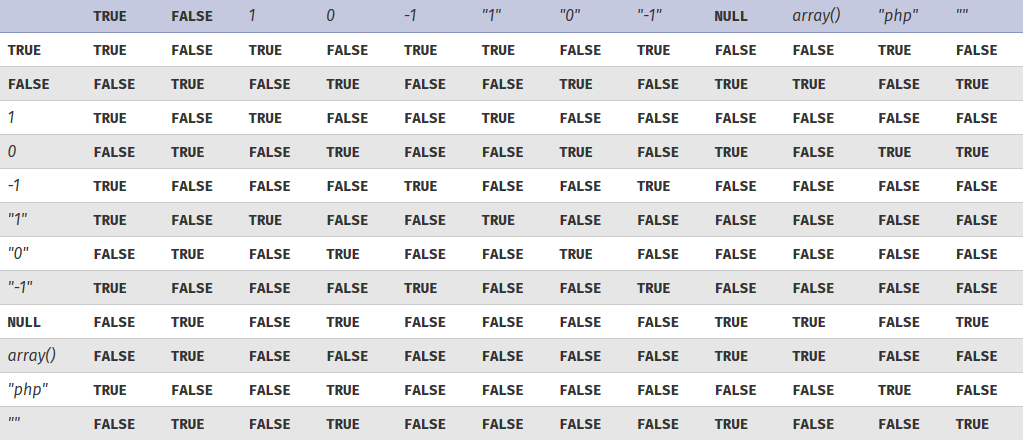
首先先说一下什么是强类型和弱类型？

语言通常被分为强类型和弱类型两种，强类型指的是强制数据类型的语言，换句话说，一个变量一旦被定义了成某个类型，如果不经过强制类型转换，这个变量就一直是这个类型，在变量使用之前必须声明变量的类型和名称，且不经强制转换不允许两种不同类型的变量互相操作。而弱类型可以随意转换变量的类型。

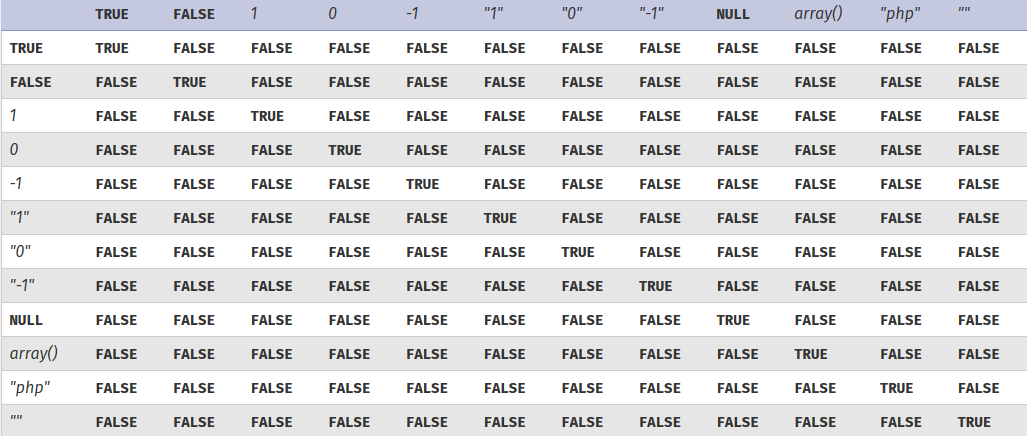
PHP作为最受欢迎的开源脚本语言，越来越多的应用于Web开发领域。同时PHP属于弱类型语言，即定义变量的时候不用声明它是什么类型。弱类型确实给程序员书写代码带来了很大的便利，但是在安全领域，特性既漏洞，这些特性在代码里面经常就是漏洞最容易出现的地方。

PHP官方也给出了类型比较表，表格显示了 PHP [类型](https://www.php.net/manual/zh/language.types.php)和[比较运算符](https://www.php.net/manual/zh/language.operators.comparison.php)在松散和严格比较时的作用。

松散比较==



严格比较===



总结一下PHP弱类型产生漏洞原理

1. PHP在处理哈希字符串时，会利用”!=”或”==”来对哈希值进行比较，它把每一个以”0e”开头的哈希值都解释为0，所以如果两个不同的密码经过哈希以后，其哈希值都是以”0e”开头的，那么PHP将会认为他们相同，都是0。
2. 当不同类型的变量进行比较的时候就会存在变量转换的问题，在转换之后就有可能会存在弱类型问题。例如需要将GET或者是POST的参数转换为int类型，或者是两个变量不匹配的时候，PHP会自动地进行变量转换。但是PHP是一个弱类型的语言，导致在进行类型转换的时候就会产生弱类型相关的漏洞。
3. 函数转换出错时导致的弱类型问题，例如：strcmp函数参数str1不为预期的String类型时(例如数组，)，在PHP 5.3版本之前将返回-1，5.3之后的版本将返回NULL。

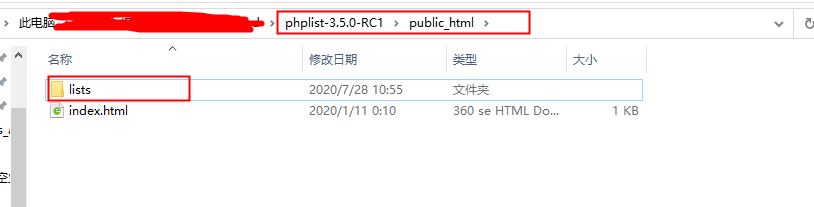
通过以上方式可以产生密码重置、绕过管理员账户的身份验证、注入、cookie伪造等弱类型漏洞。一些CTF试题也会利用其弱类型进行设置一些如MD5碰撞、十六进制转换等问题。

下面通过一些漏洞实例来进行说明。

1. CVE-2020-8547

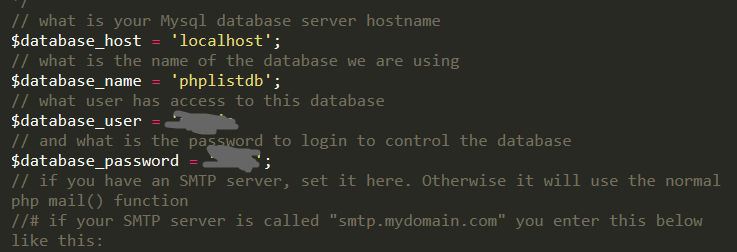
该漏洞是phpList 3.5.0版本存在的安全漏洞，源于程序没有正确处理开头为0e之后全部为数字字符的哈希值。远程攻击者可利用该漏洞绕过管理员账户的身份验证。话不多说还是直接复现一下更为直观。

首先下载phpList 3.5.0-RC1，然后解压找到这个目录



将这个目录拷贝到网站目录下，并重命名。

修改config/config.php配置文件，设置要连接的数据库和账户密码。

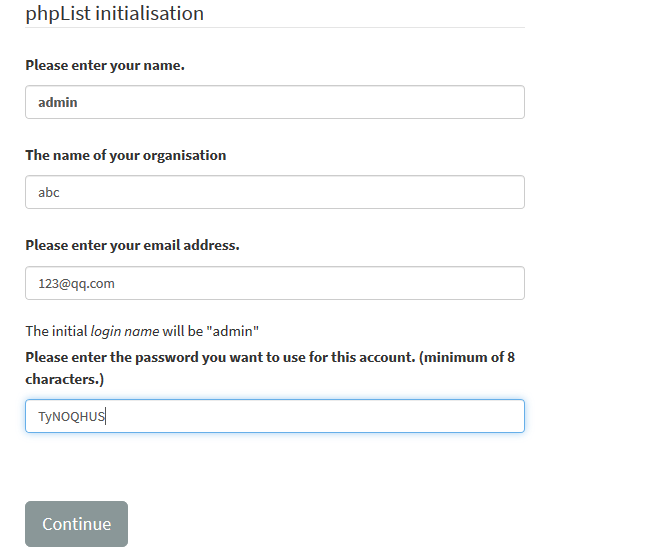


同时建立phplistdb数据库。

访问http://127.0.0.1/phplist/admin/

漏洞利用

初始化安装，设置管理员的账号密码：



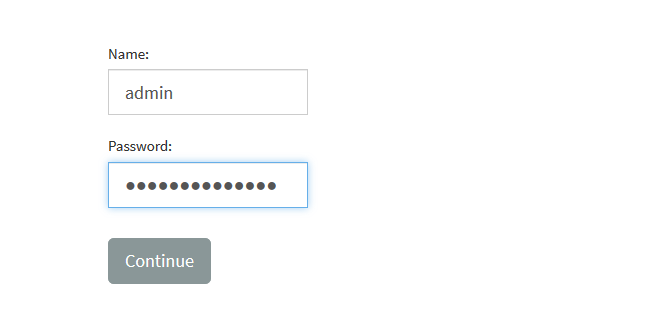
这里的密码要设置成 sha256 后以0e开头的字符串，如 TyNOQHUS

我们再次访问后台，以密码 34250003024812进行登录，其sha256后也是以0e开头

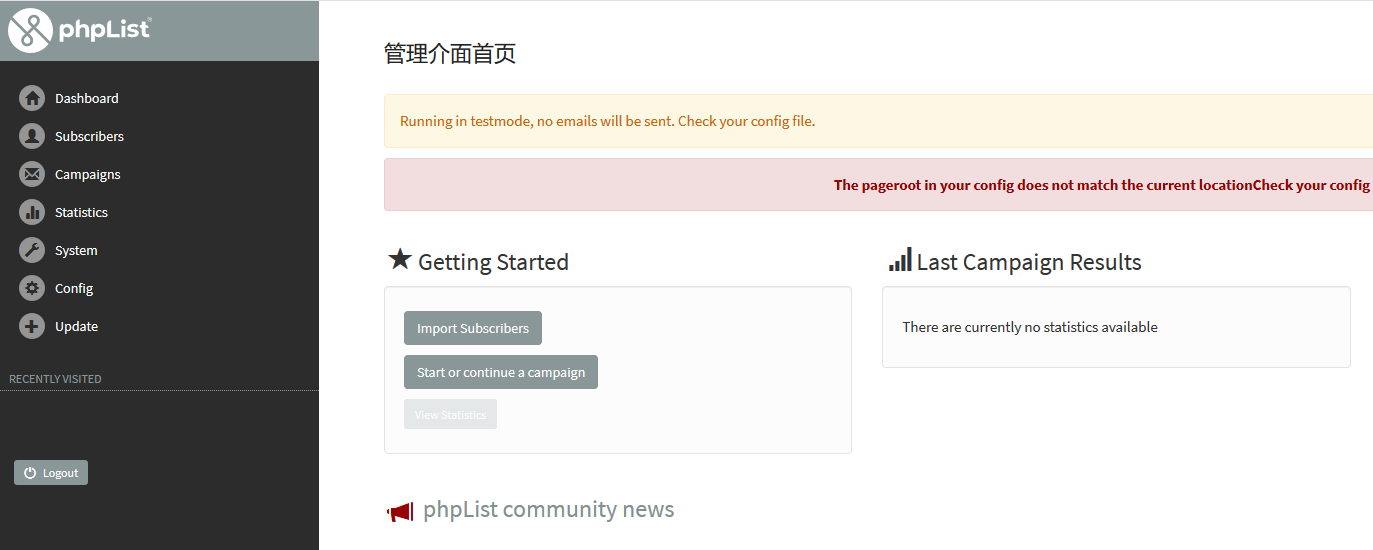
TyNOQHUS hash 0e66298694359207596086558843543959518835691168370379069085300385

34250003024812 hash

0e46289032038065916139621039085883773413820991920706299695051332

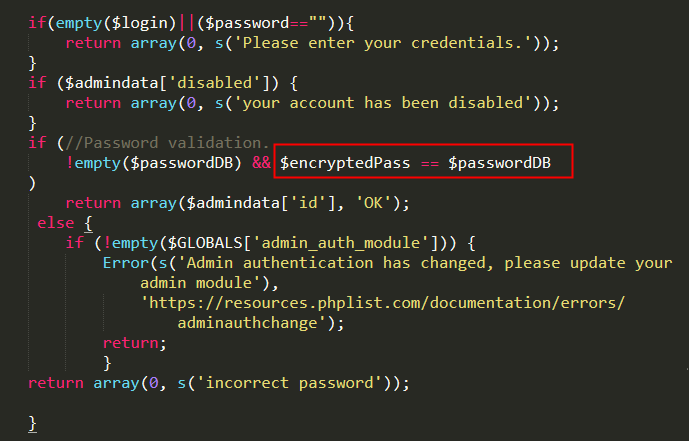


登录成功，漏洞验证成功



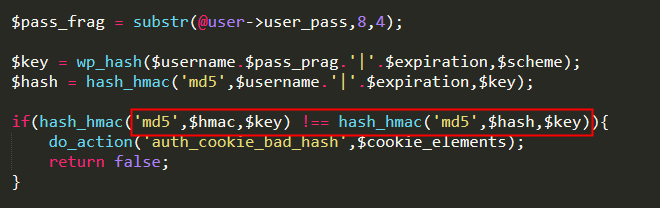
下面还是具体分析一下漏洞处的代码

在验证管理员登录的php文件phpListAdminAuthentication.php中在登录时传入密码，经过if判断后，可以看到$encryptedPass（密码sha256后的值）是使用==来判断和数据库中的值是否一样，即$encryptedPass == $passwordDB。PHP弱类型比较，就会造成0exxxxx == 0eyyyyy（会把每一个以”0e”开头的哈希值都解释为0），具体代码如下图。

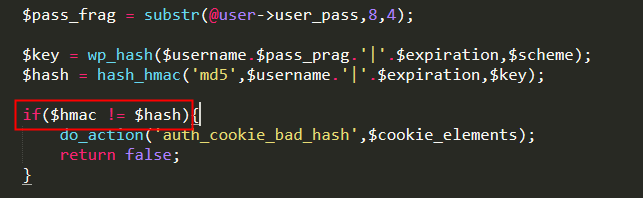


1. CVE-2014-0166（WordPress Cookie伪造）

在wordpress-3.8.2的补丁中有以下代码



而在wordpress-3.8.1的相对应的php代码是这样的。



通过对比我们自然把全部的关注点放到!=与!==上来

再看php manual中给出的例子：

<?php

var\_dump(0 == "a"); // 0 == 0 -> true

var\_dump("1" == "01"); // 1 == 1 -> true

var\_dump("10" == "1e1"); // 10 == 10 -> true

var\_dump(100 == "1e2"); // 100 == 100 -> true

?>

字符串在与数字比较前会自动转换为数字，所以0=="a"了。回到wordpress的验证代码上来。先生成根据用户名($username)、密码($pass\_frag)、cookie有效期 ($expiration)、wp-config.php中的key($key)四个信息计算出对应的$hash, 然后用cookie中取得的$hmac值与之进行比较($hmac != $hash )，从而验证cookie有效性。cookie的格式是这样的：wordpress\_hashofurl=username|expiration|hmac我们能控制的变量有$username和$expiration，其中$username需要固定。于是我们可以通过控制cookie中的$expiration去改变$hash的值，然后将cookie中的$hmac设置为0。只要不断改变$expiration，找到满足$hash=="0"的$hash，就成功伪造了有效的cookie。

1. HDwiki sql注入

在/control/list.php中代码如下



分析代码可知

从GET里获得$doctype，即$doctype = $this->get[2]，接着进入一个switch语句。 我们看到后面直接将$doctype带入SQL语句了的：$count=$this->db->fetch\_total('focus',"type=$doctype")，只要这个switch语句不影响$doctype的值，后面就能注入了。 我们看到case 2和case 3的结果都不会改变$doctype的值，但如果进入default是会将$doctype改为1的。这里就犯了一个“弱类型”的错误，当一个字符串和一个数字比较的时候，是会先将字符串强制类型转换后再与数字比较。 所以我传入doctype是2xxxx的时候，实际上是会进入case 2而不是default。出了switch语句后，doctype的值还是2xxxx，而不是2. 所以之后的$count=$this->db->fetch\_total('focus',"type=$doctype");，$doctype带入SQL语句造成注入。

1. CTF中的利用

《MD5碰撞》

<?php

if (isset($\_GET['Username']) && isset($\_GET['password'])) {

$logined = true;

$Username = $\_GET['Username'];

$password = $\_GET['password'];

if (!ctype\_alpha($Username)) {$logined = false;}

if (!is\_numeric($password) ) {$logined = false;}

if (md5($Username) != md5($password)) {$logined = false;}

if ($logined){

echo "successful";

}else{

echo "login failed!";

}

}

?>

这个题目的意思是输入一个数字和一个字符串，并且让他们的MD5值相同，才可以得到successful, 0e在比较的时候会将其视作为科学计数法，所以无论0e后面是什么，0的多少次方还是0。所以我们只需要输入一个数字和字符串进行MD5加密之后都为0e的即可得出答案。md5('240610708') == md5('QNKCDZO')成功绕过。

《起名字真难》

<?php

function noother\_says\_correct($number)

{

$one = ord('1');

$nine = ord('9');

for ($i = 0; $i < strlen($number); $i++)

{

$digit = ord($number{$i});

if ( ($digit >= $one) && ($digit <= $nine) )

{

return false;

}

}

return $number == '54975581388';

}

$flag='\*\*\*\*\*\*\*';

if(noother\_says\_correct($\_GET['key']))

echo $flag;

else

echo 'access denied';

?>

题目大致的意思就是输入一串key,key呢不可以是数字的形式，但是却要求与数字54975581388相等才可以拿到flag，看完题目就知道要求字符串和数字进行比较，所以$number == '54975581388'看到这就想到了弱类型，54975581388与之匹配的十六进制的字符串是0xccccccccc。全不是数字，自然就绕过了，得到flag。

1. 结束

看到这里相信大家都对PHP弱类型比较了解了，当然还有很多其他的漏洞实例，这里就不一一列举了。下面列举一些以0e开头的字符串的md5的hash值

s214587387a

0e848240448830537924465865611904

s1502113478a

0e861580163291561247404381396064

s1091221200a

0e940624217856561557816327384675

s1665632922a

0e731198061491163073197128363787

s1885207154a

0e509367213418206700842008763514

s1836677006a

0e481036490867661113260034900752

s1665632922a

0e731198061491163073197128363787

s878926199a

0e545993274517709034328855841020

QLTHNDT

0e405967825401955372549139051580

QNKCDZO

0e830400451993494058024219903391

EEIZDOI

0e782601363539291779881938479162

TUFEPMC

0e839407194569345277863905212547

UTIPEZQ

0e382098788231234954670291303879

UYXFLOI

0e552539585246568817348686838809

IHKFRNS

0e256160682445802696926137988570

240610708

0e462097431906509019562988736854

314282422

0e990995504821699494520356953734

571579406

0e972379832854295224118025748221

903251147

0e174510503823932942361353209384

1110242161

0e435874558488625891324861198103

1320830526

0e912095958985483346995414060832

1586264293

0e622743671155995737639662718498