

2021第一届网刃杯网络安全大赛-藏在s7里的秘密

原创

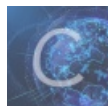
夜白君 于 2021-09-13 10:40:27 发布 1269 收藏 1

分类专栏: [2021第一届网刃杯网络安全大赛](#) 文章标签: [网络安全](#) [python](#) [uncf](#) [网刃杯](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: https://blog.csdn.net/qq_43264813/article/details/120262297

版权



[2021第一届网刃杯网络安全大赛](#) 专栏收录该内容

4 篇文章 2 订阅

订阅专栏

2021第一届网刃杯网络安全大赛-藏在s7里的秘密

难度系数: 3.0

题目描述: 某工厂的安全设备捕获了攻击者向PLC中写入恶意数据的数据包, 你能分析出并找到其中隐藏的数据吗?

解题思路:

下载查看流量包发现无法打开, 尝试对流量包进行修复

```
← → ↻ https://f00l.de/hacking/pcapfix.php
导入书签... 火狐官方网站 新手上路 常用网址 京东商城 JD 京东商城 2021年工业信息安全... 新标签页

Home f00l's blog pcapfix

pcapfix - online pcap / pcapng repair service

pcapfix 1.1.7-DEVEL (c) 2012-2021 Robert Krause

[*] Reading from file: /tmp/phpEgjuD6
[*] Writing to file: fixed_phpEgjuD6
[*] File size: 111581 bytes.
[*] Unknown file type. Assuming PCAP format.
[*] Analyzing Global Header...
[-] Magic number: 0xffffffff
[-] Major version number: 65535
[-] Minor version number: 65535
[+] GTM to local correction: 0
[+] Accuracy of timestamps: 0
[+] Max packet length: 65535
[+] Data link type: 1
[-] The global pcap header seems to be corrupt! ==> CORRECTED
[*] Analyzing packets...
[+] Packet #1 at position 24 (0 | 0 | 215 | 215).
[+] Packet #2 at position 255 (0 | 0 | 110 | 110).
[+] Packet #3 at position 381 (0 | 0 | 215 | 215).
[+] Packet #4 at position 612 (0 | 0 | 42 | 42).
[+] Packet #5 at position 670 (0 | 0 | 110 | 110).
[+] Packet #6 at position 796 (0 | 0 | 42 | 42).
[+] Packet #7 at position 854 (0 | 0 | 82 | 82).
[+] Packet #8 at position 952 (0 | 0 | 82 | 82).
[+] Packet #9 at position 1050 (0 | 0 | 66 | 66).
[+] Packet #10 at position 1132 (0 | 0 | 66 | 66).
[+] Packet #11 at position 1214 (0 | 0 | 54 | 54).
```

CSDN @夜白君

分析流量发现存在一个png图

The image shows a Wireshark interface with a packet list and a packet details pane. The packet list shows several packets, with packet 484 selected. The details pane shows the structure of the selected packet, including Ethernet II, Internet Protocol Version 4, and a raw data section. The raw data section shows a hex dump and a corresponding ASCII representation, where a PNG image is identified.

No.	Time	Source	Destination	Protocol	Length	Info
470	1970-01-01 08:00:00.000000	192.168.139.131	192.168.139.2	NBNS	110	Refresh NB <01><02>__MSBROWSE_<02><01>
471	1970-01-01 08:00:00.000000	VMware_c0:00:08	Broadcast	ARP	42	Who has 192.168.139.2? Tell 192.168.139.1
472	1970-01-01 08:00:00.000000	192.168.139.131	192.168.139.2	NBNS	110	Refresh NB <01><02>__MSBROWSE_<02><01>
473	1970-01-01 08:00:00.000000	VMware_c0:00:08	Broadcast	ARP	42	Who has 192.168.139.2? Tell 192.168.139.1
474	1970-01-01 08:00:00.000000	VMware_c0:00:08	Broadcast	ARP	42	Who has 192.168.139.2? Tell 192.168.139.1
475	1970-01-01 08:00:00.000000	192.168.139.131	192.168.139.2	NBNS	110	Refresh NB WIN-41B2ENRCCOP<20>
476	1970-01-01 08:00:00.000000	VMware_9b:a9:78	VMware_f6:2d:0f	ARP	42	Who has 192.168.139.2? Tell 192.168.139.131
477	1970-01-01 08:00:00.000000	VMware_f6:2d:0f	VMware_9b:a9:78	ARP	42	192.168.139.2 is at 00:50:56:f6:2d:0f
478	1970-01-01 08:00:00.000000	192.168.139.1	192.168.139.2	NBNS	110	Refresh NB LAPTOP-L6RRNEL4<20>
479	1970-01-01 08:00:00.000000	192.168.139.131	192.168.139.2	NBNS	110	Refresh NB WIN-41B2ENRCCOP<20>
480	1970-01-01 08:00:00.000000	192.168.139.131	192.168.139.2	DNS	85	Standard query 0xd421 A teredo.ipv6.microsoft.com
481	1970-01-01 08:00:00.000000	192.168.139.131	192.168.139.2	NBNS	110	Refresh NB WIN-41B2ENRCCOP<20>
482	1970-01-01 08:00:00.000000	192.168.139.1	192.168.139.131	S7COMM	91	ROSCTR:[Userdata] Function:[Request] -> [Block functions] -> [C
483	1970-01-01 08:00:00.000000	192.168.139.131	192.168.139.1	S7COMM	165	ROSCTR:[Userdata] Function:[Response] -> [Block functions] -> [C
484	1970-01-01 08:00:00.000000	192.168.139.1	192.168.139.131	S7COMM	541	ROSCTR:[Job] Function:[Write Var]

> Frame 484: 541 bytes on wire (4328 bits), 541 bytes captured (4328 bits)
> Ethernet II, Src: VMware_c0:00:08 (00:50:56:c0:00:08), Dst: VMware_9b:a9:78 (00:0c:29:9b:a9:78)
> Internet Protocol Version 4, Src: 192.168.139.1, Dst: 192.168.139.131

```
0000 00 0c 29 9b a9 78 00 50 56 c0 00 08 08 00 45 00  ..).x.P V....E.  
0010 02 0f a9 0c 40 00 80 06 b8 06 c0 a8 8b 01 c0 a8  ....@.....  
0020 8b 83 f2 fb 00 66 1e 44 b4 36 e7 56 b9 a2 50 18  ....f.D .6.V..P.  
0030 10 0a b7 30 00 00 03 00 01 e7 02 f0 80 32 01 00  ...-0....-2..  
0040 00 81 00 00 0e 01 c8 05 01 12 0a 10 02 01 c4 00  .........PNG..  
0050 01 84 00 00 00 00 04 0e 20 89 50 4e 47 0d 0a 1a  ....IHD R.....sR  
0060 0a 00 00 00 0d 49 48 44 52 00 00 00 7f 00 00 00  ....GB.....gAMA.  
0070 16 08 06 00 00 00 09 9b c9 1b 00 00 00 01 73 52  ....a...pHYs.  
0080 47 42 00 ae ce 1c e9 00 00 00 04 67 41 4d 41 00  
0090 00 b1 8f 0b fc 61 05 00 00 00 09 70 48 59 73 00
```

将图片导出并查看发现图片长度缺失，使用010打开发现提升CRC校验有问题，疑是需要爆破图片真实高度，使用网络上的脚本进行爆破

```

import zlib
import struct
import binascii

file = 'aaaa.png'
fr = open(file, 'rb').read()
data = bytearray(fr[12:29])

#crc32key = eval(str(fr[29:33]).replace('\x', '').replace("b'", '0x').replace("'", ''))
crc32key = struct.unpack('>I', fr[29:33])[0]&0xffffffff
# print(hex(fr[29:33]))
#data = bytearray(b'\x49\x48\x44\x52\x00\x00\x01\xf4\x00\x00\x01\xf1\x08\x06\x00\x00\x00')
n = 4096
for w in range(n):
    width = bytearray(struct.pack('>i', w))
    for h in range(n):
        height = bytearray(struct.pack('>i', h))
        for x in range(4):
            data[x+4] = width[x]
            data[x+8] = height[x]
            #print(data)
        crc32result = zlib.crc32(data)
        if crc32result == crc32key:
            print(crc32key)
            print(width,height)
            print(data)
            newpic = bytearray(fr)
            for x in range(4):
                newpic[x+16] = width[x]
                newpic[x+20] = height[x]
            fw = open(file.split('.')[0]+'_cracked+'.png, 'wb')
            fw.write(newpic)
            fw.close

```

将爆破出的高度修改查看拿到flag

