2017年最佳算法提名勒索软件（sega）分析

菠菜 / 2017-03-09 15:34:00 / 浏览数 4584 技术文章 技术文章 顶(0) 踩(0)

一．前言

在电视剧中，经常看到这样的场景：男主与女主相亲相爱结婚后生下一可爱的孩子，谁知男主遇见小三后移情别恋并最终与小三一起，时隔多时，发现孩子得了血液类疾病，

在2017年1月，新型的加密勒索软件以Sega之姿重出江湖，这款新型勒索软件的加密方式正如上面描述的电影场景一般。加密过程涉及三方角色：入侵者(相当于女主)，系统

这款勒索软件的原理，就是通过强大的curve25519密钥协商加密算法，对每个文件生成不同的加密密钥后，使用chacha20对文件进行加密。勒索软件之所以采用curve255

勒索软件最大的亮点一方面在于巧妙设计的算法结构，涉及的三个角色的加密变换，此外软件代码中涉及到的加密方面的算法有murmurhash，sha1,base64,chcha20, curve25519等，使用这些算法并不能显示出软件作者的牛B之处，作者的牛B之处在于把这些种加密算法的使用都恰到好处，可见软件作者对加密算法有深入理解。另一方面

被此勒索软件加密后，显示效果图如下：

二．技术分析：

程序代码做了各种混淆，包括跳转混淆，内存加载，SMC等反调试技术，我们只对跳转混淆的处理方法做下详细说明，对于SMC等其它的阻碍调试的方法等不在本文中展开

跳转混淆处理

代码的跳转做了跳转混淆，跳转混淆标志为push XXX;CALL XX

使用ida-python脚本去除掉跳转混淆的代码。算法的思想为：

遍历程序代码段，找到混淆函数后，通过交叉引用找到所有调用混淆函数的地方，在调用混淆函数的地方向上搜索找到push语句，根据push的操作数计算出最终要跳转到的

通过分析，代码中有多处混淆函数，每个混淆函数代码都是固定的。

混淆函数对应的十六进制为：

```
51 52 8B 54 24 0C 8B 4C  24 08 81 C1 FF 00 00 00 29 D1 41 41 89 4C 24 08  5A 59 C2 04 00
```

搜索上面的十六进制，定位到混淆函数。使用IDA的交叉引用功能找到调用函数的位置。

最终，通过下面的代码去除跳转混淆代码。

```
def process_jmp(push_ea, to_ea):

   i = idautils.DecodeInstruction(push_ea)
   if i.get_canon_mnem() == 'push':
       jmp_start_ea = NextHead(push_ea)
       code_distance = to_ea - jmp_start_ea - 5
       print '%x :jmp to: %x delta:%x' % (jmp_start_ea, to_ea, code_distance)
       x86_nop_region(push_ea, jmp_start_ea)

       code_distance = code_distance & 0xffffffff
       code_distance = '%08x' % code_distance
       distance_len = len(code_distance)
       ret_str =[]
       temp_buf = code_distance[::]
       for i in range(distance_len/2):
           bytes = temp_buf[-2::]
           ret_str.append(bytes)
           temp_buf=temp_buf[:-2:]


       fill_value = 'E9%s' % ''.join(ret_str)
       buf = binascii.unhexlify(fill_value)
       idaapi.patch_many_bytes(jmp_start_ea, buf)


if __name__=="__main__":
   confusion_fun_list = findConfusionBinary()
   for confusion_fun in confusion_fun_list:
```

```
    for x in XrefsTo(confusion_fun,flags = 0):
        addr_push = idc.PrevHead(x.frm)
        # push mem format
        if GetMnem(addr_push) == 'push' and  GetOpType(addr_push, 0)==5:
            push_arg=GetOpnd(addr_push, 0)
            ret_arg=NextHead(x.frm)
            push_arg=int(push_arg.strip('h'),16)
            to_ea = myfun(push_arg, ret_arg)

            process_jmp(addr_push, to_ea)
```

去除跳转混淆后代码前后比较：

加载函数过程

获取函数名称时没有采用shellcode中常采用的ROR/L算法，而是采用了murmurhash，代码中的murmurhash及解密出的函数地址如下：

CC地址生成过程：

CC地址以加密的方式保存在文件的尾部。

解密过程为：读取文件尾部的密钥与密文，使用CHACHA20算法进行解密

如下图所示文件尾部的密文与加密key

打开原始文件，定位到文件尾部读：

通过salsa20算法，解密出CC地址：

最终解密出来的地址部分为：“rzunt3u2.com
er29sl.com”，此解密出来的内容做为最终通信地址的后缀部分，通信地址的前面部分内容硬编码在文件中，最后得到的通信地址为：mbfce24rgn65bx3g.rzunt3u2.com和

程序中硬编码的通信地址的部分内容如下图

```
from Crypto.Cipher import ChaCha20
secrect = '\x2E\x4B\x59\xD7\xA3\x9B\xFC\xBC\x3E\xA1\xCD\x70\x60\xBF\xB9\xBB\x15\xF7\xEB\x9F\xBA\xE3\x3E\xE0\x4B\x34\x4E\xF1\x6
cipter_code = '\x1D\xA3\xF3\x7D\xB2\x3D\x85\x66\x9D\x67\x2B\xDA\x6A\x9A\x51\x58\x64\x4A\x4F\x03\xC1\x1B\x7D\x11\xE8'
nonce = '\x00\x00\x00\x00\x00\x00\x00\x00'
cipher = ChaCha20.new(nonce=nonce ,key=secrect)
print cipher.decrypt(cipter_code)
```

字符生成算法

程序中的字符生成算法可以保证在同一台机器中相同str_id可以生成相同的字符内容，而不同机器的相同str_id生成的字符内容不同。

代码中用到的随机字符生成算法为

GenStrByID(unsigned int str_len, char str_id)

参数一表示要生成的字符长度，参数二表示生成的字符id.

程序从根据str_id和从注册表中读取到的machine_id进行sha1运算，对sha1的结果进行base64编码，最后根据字符长度从base64结果中选择指定长度的字符，如果指定长

文件遍历过程

文件遍历过程并不是使用的常规的FindFirstFile/FindNextFile类函数进行遍历。而是通过ZwQueryDirectoryFile函数历文件。

对文件后缀名进行判断：

文件夹白名单：

与CC地址通信的内容也经过格式化为packmessage格式：

三．加密与解密：

3.1 加密过程

对文件的加密使用流加密算法CHACHA20,CHACHA20的密钥来自于curve25519密钥协商体系。

加密过程涉及到三个角色，分别为入侵者，系统，文件。

加密过程涉及到两次密钥的协商过程，分别为入侵者与系统协商共享密钥，系统与每个文件协商共享密钥。其中的协商密钥算法为curve25519。

其中硬编码的i_public如下所示

密钥生成过程如下图。

其中f_public保存在加密后文件的尾部，每个被加密文件的f_public不同。

S_public1与s_public2密钥保存在临时文件中。共0x40字节。

入侵者与系统通过协商出共享密钥（图中的共享密钥1），随后系统与文件协商密钥的过程中，使用这个共享密钥做为系统方的私钥（即s_secret2就是共享密钥1）。

入侵者与系统之间协商密钥的过程：

系统生成随机的私钥,与入侵者硬编码的公钥，计算出系统与入侵者之间的共享密钥，同时把系统方的公钥(s_public1)保存在临时文件的前0x20字节处。此阶段协商出来的共

随后在系统与文件这两个角色协商密钥过程中，用共享密钥1作为系统方的私钥（s_secret2），系统方的s_public2由s_secret2生成后保存在临时文件中的后0x20字节处。此

文件方使用2中协商出来的通信密钥(共享密钥2)做为chacha20密钥，加密文件内容。并将文件方的公钥(f_public)保存在加密文件的尾部。

文件加密密钥的生成过程：

使用chacha20加密文件的代码：

重命名文件，生成.sega文件后缀

3.2 解密过程分析：

由上面的分析，要想解密文件，就必须知道chacha20的密钥，这个密钥是文件方与系统方协商密钥的共享密钥（即共享密钥2）。从上面可以看到只有得到s_secret2才能解

3.3 动态IP生成算法：

程序运行后，会动态生成0x2000个IP地址，然后向这些IP地址发送感染用户的用户信息。关于IP地址的生成在参考1中也有了详细的说明，本文就不再赘述。

四． 其它操作：

4.1 对用户地理区域的判断：

通过GetKeyboardLayoutList函数得到键盘布局信息，用来判断用户的所在的地址区域。这些区域信息在参考1中都有提及。

4.2 自删除

4.3 删除用户的还原镜像

4.4 添加启动项：

五．总结：

对于这款勒索软件尚没有可以解密文件的方法。建议做好预防措施。使用使用安全软件或设置以避免已知的勒索软件运行；做好重要数据的备份工作。此外，这款勒索软件会

六．参考：

https://www.govcert.admin.ch/blog/27/sage-2.0-comes-with-ip-generation-algorithm-ipga

https://www.cert.pl/en/news/single/sage-2-0-analysis/

https://isc.sans.edu/forums/diary/Sage+20+Ransomware/21959/

https://www.bleepingcomputer.com/forums/t/634978/sage-file-sample-extension-sage/

上一篇：CSRF漏洞挖掘 下一篇：关于update的一个小注入

1. 0 条回复

- 动动手指，沙发就是你的了！

先知社区

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板