WinorDLL64: A backdoor from the vast Lazarus arsenal?

welivesecurity.com/2023/02/23/winordll64-backdoor-vast-lazarus-arsenal/

February 23, 2023

The targeted region, and overlap in behavior and code, suggest the tool is used by the infamous North Korea-aligned APT group

Vladislav Hrčka
23 Feb 2023 - 11:30AM

The targeted region, and overlap in behavior and code, suggest the tool is used by the infamous North Korea-aligned APT group

ESET researchers have discovered one of the payloads of the Wslink downloader that we uncovered back in 2021. We named this payload WinorDLL64 based on its filename WinorDLL64.dll. Wslink, which had the filename WinorLoaderDLL64.dll, is a loader for Windows binaries that, unlike other such loaders, runs as a server and executes received modules in memory. As the wording suggests, a loader serves as a tool to load a payload, or the actual malware, onto the already compromised system. The initial Wslink compromise vector has not been identified.

The initially unknown Wslink payload was uploaded to VirusTotal from South Korea shortly after the publication of our blogpost, and hit one of our YARA rules based on Wslink's unique name WinorDLL64. Regarding Wslink, ESET telemetry has seen only a few detections – in Central Europe, North America, and the Middle East.

The WinorDLL64 payload serves as a backdoor that most notably acquires extensive system information, provides means for file manipulation, such as exfiltrating, overwriting, and removing files, and executes additional commands. Interestingly, it communicates over a connection that was already established by the Wslink loader.

In 2021, we did not find any data that would suggest Wslink is a tool from a known threat actor. However, after an extensive analysis of the payload, we have attributed WinorDLL64 to the Lazarus APT group with low confidence based on the targeted region and an overlap in

both behavior and code with known Lazarus samples.

Active since at least 2009, this infamous North-Korea aligned group is responsible for high-profile incidents such as both the Sony Pictures Entertainment hack and tens-of-millions-of-dollar cyberheists in 2016, the WannaCryptor (aka WannaCry) outbreak in 2017, and a long history of disruptive attacks against South Korean public and critical infrastructure since at least 2011. US-CERT and the FBI call this group HIDDEN COBRA.

Based on our extensive knowledge of the activities and operations of this group, we believe that Lazarus consists of a large team that is systematically organized, well prepared, and is made up of several subgroups that utilize a large toolset. Last year, we discovered a Lazarus tool that took advantage of the CVE-2021-21551 vulnerability to target an employee of an aerospace company in the Netherlands, and a political journalist in Belgium. It was the first recorded abuse of the vulnerability; in combination, the tool and the vulnerability led to the blinding of the monitoring of all security solutions on compromised machines. We also provided an extensive description of the structure of the virtual machine used in samples of Wslink.

This blogpost explains the attribution of WinorDLL64 to Lazarus and provides an analysis of the payload.

Links to Lazarus

We have discovered overlaps in both behavior and code with Lazarus samples from Operation GhostSecret and the Bankshot implant described by McAfee. The description of the implants in both GhostSecret and Bankshot articles contains overlaps in the functionality with WinorDLL64 and we found some code overlap in the samples. In this blogpost we will only use the FE887FCAB66D7D7F79F05E0266C0649F0114BA7C sample from GhostSecret for comparison against WinorDLL64 (1BA443FDE984CEE85EBD4D4FA7EB1263A6F1257F), unless specified otherwise.

The following details summarize the supporting facts for our low confidence attribution to Lazarus:

1. Victimology

> Fellow researchers from AhnLab confirmed South Korean victims of Wslink in their telemetry, which is a relevant indicator considering the traditional Lazarus targets and that we have observed only a few hits.

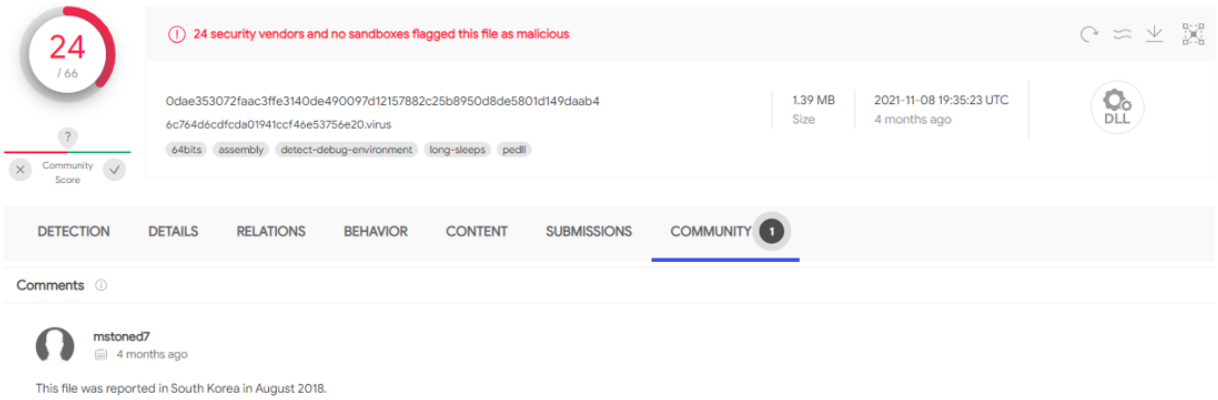

Figure 1. Reported South Korean victim, where mstoned7 is the researcher from Ahnlab

2. Malware

- The latest GhostSecret sample reported by McAfee (FE887FCAB66D7D7F79F05E0266C0649F0114BA7C) is from February 2018; we spotted the first sample of Wslink in late 2018 and fellow researchers reported hits in August 2018, which they disclosed after our publication. Hence, these samples were spotted a relatively short period of time apart.
- The PE rich headers indicate that the same development environment and projects of similar size were used in several other known Lazarus samples (e.g., 70DE783E5D48C6FBB576BC494BAF0634BC304FD6; 8EC9219303953396E1CB7105CDB18ED6C568E962). We found this overlap using the following rules that cover only these Wslink and Lazarus samples, which is an indicator with a low weight. We tested them on VirusTotal's retrohunt and our internal file corpus.

```
rich_signature.length == 80 and
pe.rich_signature.toolid(175, 30319) == 7 and
pe.rich_signature.toolid(155, 30319) == 1 and
pe.rich_signature.toolid(158, 30319) == 10 and
pe.rich_signature.toolid(170, 30319) >= 90 and
pe.rich_signature.toolid(170, 30319) <= 108
```

This rule can be translated to the following notation that is more readable and used by VirusTotal, where one can see the product version and build ID (VS2010 build 30319), number and type of source/object files used ([LTCG C++] where LTCG stands for Link Time Code Generation, [ASM], [C]), and number of exports ([EXP]) in the rule:

```
[LTCG C++] VS2010 build 30319 count=7
[EXP] VS2010 build 30319 count=1
[ASM] VS2010 build 30319 count=10
[ C ] VS2010 build 30319 count in [ 90 .. 108 ]
```

- The GhostSecret article described “a unique data-gathering and implant-installation component that listens on port 443 for inbound control server connections” that additionally ran as a service. This is an accurate description of Wslink downloader behavior, apart from the port number, which can vary based on the configuration. To sum it up, even though the implementation is different, both serve the same purpose.
- The loader is virtualized by Oreans’ Code Virtualizer, which is a commercial protector that is used frequently by Lazarus.
- The loader uses the MemoryModule library to load modules directly from memory. The library is not commonly used by malware, but it is quite popular among North Korea-aligned groups such as Lazarus and Kimsuky.
- Overlap in the code between WinorDLL64 and GhostSecret that we found during our analysis. The results and the significance in attribution are listed in Table 1.

Table 1. Similarities between WinorDLL64 and GhostSecret and their significance in attributing both to the same threat actor

Other similarities between WinorDLL64 and GhostSecret	**Impact**
Code overlap in code responsible to get processor architecture	Low
Code overlap in current directory manipulation	Low
Code overlap in getting the process list	Low
Code overlap in file sending	Low
Behavior overlap in listing processes	Low
Behavior overlap in current directory manipulation	Low
Behavior overlap in file and directory listing	Low
Behavior overlap in listing volumes	Low
Behavior overlap in reading/writing files	Low
Behavior overlap in creating processes	Low
Considerable behavior overlap in secure removal of files	Low
Considerable behavior overlap in termination of processes	Low
Considerable behavior overlap in collecting system information	Low

Code overlap in the file sending functionality is highlighted in Figure 2 and Figure 3.

```
FileW = kernel32_CreateFileW(file_name, 0x80000000, 3, 0, 3, FileAttributesW, 0);
if ( FileW == -1 )
  return sendEncryptedDword(0xB6BE);
FileSize = kernel32_GetFileSize(FileW, &file_size);
...
if ( sendEncryptedDword(0xB6BD) && sendEncrypted(&file_size, 8u) && recvDecryptedDword(&offset) )
{
  ...
  kernel32_SetFilePointer(FileW, offset, &file_ptr, FILE_BEGIN);
  if ( (_DWORD)remaining_bytes )
  {
LABEL_12:
    for ( bytes_to_read = 0x3800; ; bytes_to_read = HIDWORD(remaining_bytes) )
    {
      kernel32_ReadFile(FileW, buff + 2, bytes_to_read, &curr_bytes_read, 0);
      buff_size = curr_bytes_read;
      LODWORD(remaining_bytes) = (__PAIR64__(remaining_bytes, HIDWORD(remaining_bytes)) - curr_bytes_read) >> 32;
      HIDWORD(remaining_bytes) -= curr_bytes_read;
      if ( !remaining_bytes )
        buff_size = curr_bytes_read + 0x8000;
      *(_WORD *)buff = buff_size;
      if ( !sendEncrypted((const void *)buff, bytes_to_read + 2) )
        break;
      if ( (_DWORD)remaining_bytes )
        goto LABEL_12;
      ...
```

Figure 2. GhostSecret sending a file

```
FileW = CreateFileW(config + 4, GENERIC_READ, 3u, 0i64, 3u, FILE_ATTRIBUTE_NORMAL, 0i64);
if ( FileW == -1i64 )
  return (comm_stru->symmetric_encrypt_send_dword)(comm_stru->tls_context, 18i64);
...
(comm_stru->symmetric_encrypt_send_dword)(comm_stru->tls_context, 17i64);
v11 = GetFileSizeEx(FileW, &file_size_then_ptr);
file_size = file_size_then_ptr;
(comm_stru->symmetric_encrypt_send)(tls_context, &file_size, 8i64);
SetFilePointerEx(FileW, *config, &file_size_then_ptr, FILE_BEGIN);
offset = *config;
...
if ( *config < file_size )
{
  read_bytes = 0i64;
  while ( 1 )
  {
    bytes_to_read = file_size - offset - read_bytes;
    if ( bytes_to_read > 0x7800 )
      LODWORD(bytes_to_read) = 0x7800;
    ...
    succ = ReadFile(FileW, buff, bytes_to_read, &curr_read_bytes, 0i64);
    if ( !succ )
      break;
    if ( !(comm_stru->symmetric_encrypt_send)(tls_context, buff, curr_read_bytes) )
    ...
    file_size_then_ptr = (curr_read_bytes + file_size_then_ptr);
    read_bytes = file_size_then_ptr;
    ...
```

Figure 3. Wslink sending a file

Technical analysis

WinorDLL64 serves as a backdoor that most notably acquires extensive system information, provides means for file manipulation, and executes additional commands. Interestingly, it communicates over a TCP connection that was already established by its loader and uses some of the loader's functions.

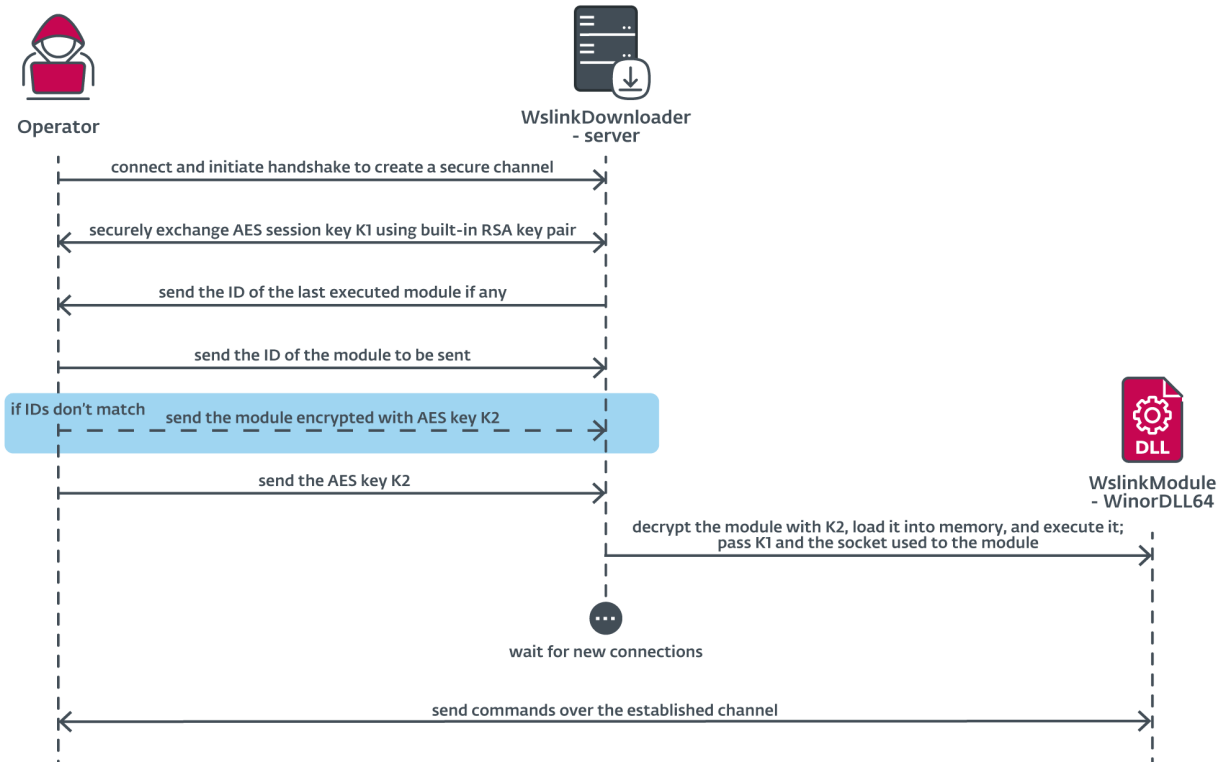

Figure 4. Visualization of Wslink's communication

The backdoor is a DLL with a single unnamed export that accepts one parameter – a structure for communication that was already described in our previous blogpost. The structure contains a TLS-context – socket, key, IV – and callbacks for sending and receiving messages encrypted with 256-bit AES-CBC that enable WinorDLL64 to exchange data securely with the operator over an already established connection.

The following facts lead us to believe with high confidence that the library is indeed part of Wslink:

- The unique structure is used everywhere in the expected way, e.g., the TLS-context and other meaningful parameters are supplied in the anticipated order to the correct callbacks.
- The name of the DLL is WinorDLL64.dll and Wslink's name was WinorLoaderDLL64.dll.

WinorDLL64 accepts several commands. Figure 5 displays the loop that receives and handles commands. Each command is bound to a unique ID and accepts a configuration that contains additional parameters.

```
while ( (comm_stru->symmetric_receive_decrypt_dword)(tls_context, &received_id)
     && (comm_stru->symmetric_receive_decrypt_dynamic_length)(comm_stru->tls_context, &config, &config_len) )
{
  result = 0;
  if ( received_id == 0xF )
  {
    (comm_stru->symmetric_encrypt_send_dword)(comm_stru->tls_context, 17i64);
    ret_val = 1;
  }
  else if ( received_id == 0x10 )
  {
    result = (comm_stru->symmetric_encrypt_send_dword)(comm_stru->tls_context, 17i64);
  }
  else
  {
    id = command_list.id;
    cmd_list_entry = &command_list;
    while ( id )
    {
      if ( id == received_id )
      {
        callback = cmd_list_entry->callback;
        if ( callback )
          result = callback(comm_stru, config, config_len);
        break;
      }
      ++cmd_list_entry;
      id = cmd_list_entry->id;
    }
  }
```

Figure 5. The main part of the backdoor's command-receiving loop

The command list, with our labels, is in Figure 6.

```
command_list    command <1, offset get_extensive_system_info>
                                    ; DATA XREF: WinorDll64_1:loc_4C592E↑r
                                    ; WinorDll64_1+5C↑o
                command <2, offset create_and_pipe_process_output>
                command <3, offset pipe_powershell_command>
                command <5, offset set_get_current_directory>
                command <6, offset get_volume_info_or_file_list>
                command <7, offset append_to_file>
                command <8, offset read_file>
                command <9, offset download_directory>
                command <0Ah, offset secure_remove_list_of_files>
                command <0Bh, offset create_process_opt_as_user>
                command <0Ch, offset get_process_list_or_kill_processes>
                command <0Dh, offset disconnect_session_or_get_session_list>
                command <0Eh, offset get_connection_time>
```

Figure 6. The command list

Table 2 contains a summary of the WinorDLL64 commands, where modified, and old categories refer to the relationship to the previously documented GhostSecret functionality. We highlight only significant changes in the modified category.

Table 2. Overview of backdoor commands

Category	Command ID	Functionality	Description

Category	Command ID	Functionality	Description
New	0x03	Execute a PowerShell command	WinorDLL64 instructs the PowerShell interpreter to run unrestricted and to read commands from standard input. Afterwards, the backdoor passes the specified command to the interpreter and sends the output to the operator.
0x09	Compress and download a directory	WinorDLL64 recursively iterates over a specified directory. The content of each file and directory is compressed separately and written to a temporary file that is afterwards sent to the operator and then removed securely.	
0x0D	Disconnect a session	Disconnects a specified logged-on user from the user's Remote Desktop Services session. The command can also perform different functionality based on the parameter.	
0x0D	List sessions	Acquires various details about all sessions on the victim's device and sends them to the operator. The command can also perform different functionality based on the parameter.	
0x0E	Measure connection time	Uses the Windows API GetTickCount to measure the time required to connect to a specified host.	
Modified	0x01	Get system info	Acquires comprehensive details about the victim's system and sends them to the operator.
0x0A	Remove files securely	Overwrites specified files with a block of random data, renames each file to a random name, and finally securely removes them one by one.	

Category	Command ID	Functionality	Description
0x0C	Kill processes	Terminates all processes whose names match a supplied pattern and/or with a specific PID.	
Old	0x02/0x0B	Create a process	Creates a process either as the current or specified user and optionally sends its output to the operator.
0x05	Set/Get current directory	Attempts to set and subsequently acquire the path of the current working directory.	
0x06	List volumes	Iterates over drives from C: to Z: and acquires the drive type and volume name. The command can also perform different functionality based on the parameter.	
0x06	List files in a directory	Iterates over files in specified directory and acquires information such as names, attributes, etc. The command can also perform different functionality based on the parameter.	
0x07	Write to a file	Downloads and appends the stated amount of data to specified file.	
0x08	Read from a file	The specified file is read and sent to the operator.	
0x0C	List processes	Acquires details about all running processes on the victim's device and additionally sends ID of the current process.	

Conclusion

Wslink's payload is dedicated to providing means for file manipulation, execution of further code, and obtaining extensive information about the underlying system that possibly can be leveraged later for lateral movement, due to specific interest in network sessions. The Wslink

loader listens on a port specified in the configuration and can serve additional connecting clients, and even load various payloads.

WinorDLL64 contains an overlap in the development environment, behavior, and code with several Lazarus samples, which indicates that it might be a tool from the vast arsenal of this North-Korea aligned APT group.

ESET Research offers private APT intelligence reports and data feeds. For any inquiries about this service, visit the ESET Threat Intelligence page.

IoCs

SHA-1	ESET detection name	Description
1BA443FDE984CEE85EBD4D4FA7EB1263A6F1257F	Win64/Wslink.A	Memory dump of discovered Wslink payload WinorDll64.

MITRE ATT&CK techniques

This table was built using version 12 of the ATT&CK framework. We do not mention techniques from the loader again, only the payload.

Tactic	ID	Name	Description
Resource Development	T1587.001	Develop Capabilities: Malware	WinorDLL64 is a custom tool.
Execution	T1059.001	Command and Scripting Interpreter: PowerShell	WinorDLL64 can execute arbitrary PowerShell commands.
T1106	Native API	WinorDLL64 can execute further processes using the CreateProcessW and CreateProcessAsUserW APIs.	
Defense Evasion	T1134.002	Access Token Manipulation: Create Process with Token	WinorDLL64 can call APIs WTSQueryUserToken and CreateProcessAsUserW to create a process under an impersonated user.

Tactic	ID	Name	Description
	T1070.004	Indicator Removal: File Deletion	WinorDLL64 can securely remove arbitrary files.
Discovery	T1087.001	Account Discovery: Local Account	WinorDLL64 can enumerate sessions and list associated user, and client names, among other details.
	T1087.002	Account Discovery: Domain Account	WinorDLL64 can enumerate sessions and list associated domain names –among other details.
	T1083	File and Directory Discovery	WinorDLL64 can obtain file and directory listings.
	T1135	Network Share Discovery	WinorDLL64 can discover shared network drives.
	T1057	Process Discovery	WinorDLL64 can collect information about running processes.
	T1012	Query Registry	WinorDLL64 can query the Windows registry to gather system information.
	T1082	System Information Discovery	WinorDLL64 can obtain information such as computer name, OS and latest service pack version, processor architecture, processor name, and amount of space on fixed drives.
	T1614	System Location Discovery	WinorDLL64 can obtain the victim's default country name using the GetLocaleInfoW API.
	T1614.001	System Location Discovery: System Language Discovery	WinorDLL64 can obtain the victim's default language using the GetLocaleInfoW API.

Tactic	ID	Name	Description
	T1016	System Network Configuration Discovery	WinorDLL64 can enumerate network adapter information.
	T1049	System Network Connections Discovery	WinorDLL64 can collect a list of listening ports.
	T1033	System Owner/User Discovery	WinorDLL64 can enumerate sessions and list associated user, domain, and client names – among other details.
Collection	T1560.002	Archive Collected Data: Archive via Library	WinorDLL64 can compress and exfiltrate directories using the quicklz library.
	T1005	Data from Local System	WinorDLL64 can collect data on the victim's device.
Impact	T1531	Account Access Removal	WinorDLL64 can disconnect a logged-on user from specified sessions.

23 Feb 2023 - 11:30AM

Sign up to receive an email update whenever a new article is published in our Ukraine Crisis – Digital Security Resource Center

Newsletter

Discussion