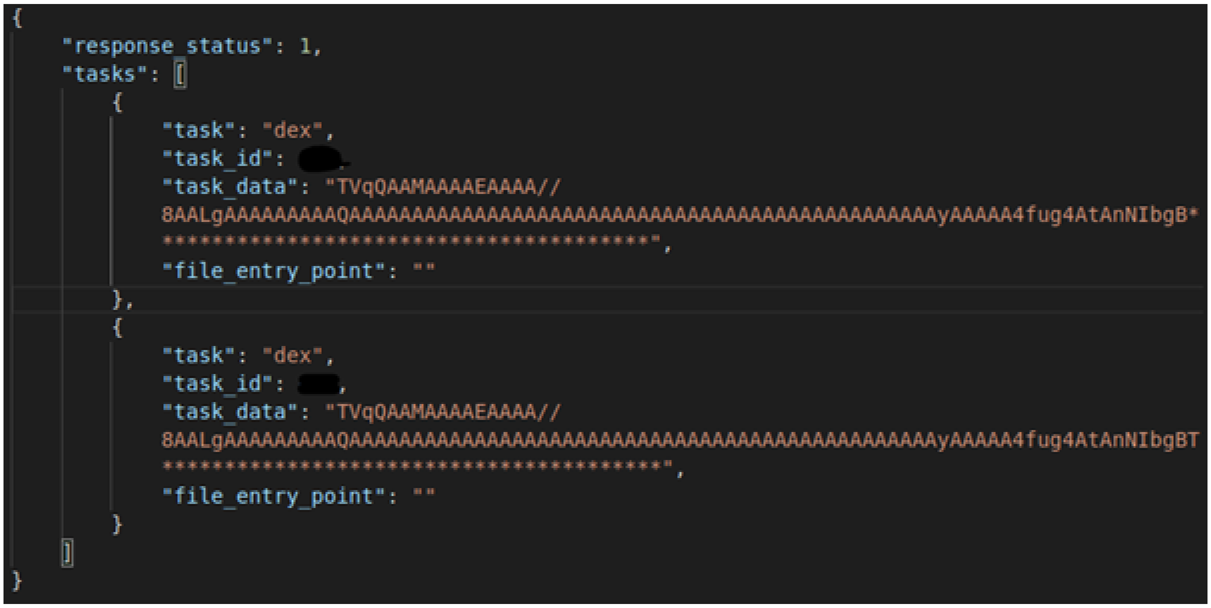
Adventures in the land of BumbleBee – a new malicious loader

research.nccgroup.com/2022/04/29/adventures-in-the-land-of-bumblebee-a-new-malicious-loader/

April 29, 2022



```
{
    "response_status": 1,
    "tasks": [
        {
            "task": "dex",
            "task_id": [redacted],
            "task_data": "TVqQAAMAAAAEAAAA//
            8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAyAAAAA4fug4AtAnNIbgB*
            ***************************************",
            "file_entry_point": ""
        },
        {
            "task": "dex",
            "task_id": [redacted],
            "task_data": "TVqQAAMAAAAEAAAA//
            8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAyAAAAA4fug4AtAnNIbgBT
            ****************************************",
            "file_entry_point": ""
        }
    ]
}
```

Authored by: Mike Stokkel, Nikolaos Totosis and Nikolaos Pantazopoulos

tl;dr

BUMBLEBEE is a new malicious loader that is being used by several threat actors and has been observed to download different malicious samples. The key points are:

- BUMBLEBEE is statically linked with the open-source libraries OpenSSL 1.1.0f, Boost (version 1.68). In addition, it is compiled using Visual Studio 2015.
- BUMBLEBEE uses a set of anti-analysis techniques. These are taken directly from the open-source project [1].
- BUMBLEBEE has Rabbort.DLL embedded, using it for process injection.
- BUMBLEBEE has been observed to download and execute different malicious payloads such as Cobalt Strike beacons.

Introduction

In March 2022, Google's Threat Analysis Group [2] published about a malware strain linked to Conti's Initial Access Broker, known as BUMBLEBEE. BUMBLEBEE uses a comparable way of distribution that is overlapping with the typical BazarISO campaigns.

In the last months BUMBLEBEE, would use three different distribution methods:

- Distribution via ISO files, which are created either with StarBurn ISO or PowerISO software, and are bundled along with a LNK file and the initial payload.
- Distribution via OneDrive links.
- Email thread hijacking with password protected ZIPs

BUMBLEBEE is currently under heavy development and has seen some small changes in the last few weeks. For example, earlier samples of BUMBLEBEE used the user-agent ‘bumblebee’ and no encryption was applied to the network data. However, this functionality has changed, and recent samples use a hardcoded key as user-agent which is also acting as the RC4 key used for the entire network communication process.

Technical Analysis

Most of the identified samples are protected with what appears to be a private crypter and has only been used for BUMBLEBEE binaries so far. This crypter uses an export function with name **SetPath** and has not implemented any obfuscation method yet (e.g. strings encryption).

The BUMBLEBEE payload starts off by performing a series of anti-analysis checks, which are taken directly from the open source Khasar project[1]. After these checks passed, BUMBLEBEE proceeds with the command-and-control communication to receive tasks to execute.

Network Communication

BUMBLEBEE’s implemented network communication procedure is quite simple and straightforward. First, the loader picks an (command-and-control) IP address and sends a HTTPS GET request, which includes the following information in a JSON format (encrypted with RC4):

Key	Description
client_id	A MD5 hash of a UUID value taken by executing the WMI command *‘SELECT * FROM Win32_ComputerSystemProduct’*.
group_name	A hardcoded value, which represents the group that the bot (compromised host) will be added.
sys_version	Windows OS version
client_version	Default value that’s set to 1
user_name	Username of the current user
domain_name	Domain name taken by executing the WMI command *‘SELECT * FROM Win32_ComputerSystem’*.

task_state	Set to 0 by default. Used only when the network commands **‘ins’** or **‘sdl’** are executed.
task_id	Set to 0 by default. Used only when the network commands **‘ins’** or **‘sdl’** are executed.

Once the server receives the request, it replies with the following data in a JSON format:

Key	**Description**
response_status	Boolean value, which shows if the server correctly parsed the loader’s request. Set to 1 if successful.
tasks	Array containing all the tasks
task	Task name
task_id	ID of the received task, which is set by the operator(s).
task_data	Data for the loader to execute in Base64 encoded format
file_entry_point	Potentially represents an offset value. We have not observed this being used either in the binary’s code or during network communication (set to an empty string).

Tasks

Based on the returned tasks from the command-and-control servers, BUMBLEBEE will execute one of the tasks described below. For two of the tasks, **shi** and **dij**, BUMBLEBEE uses a list of predefined process images paths:

- C:\Program Files\Windows Photo Viewer\ImagingDevices.exe
- C:\Program Files\Windows Mail\wab.exe
- C:\Program Files\Windows Mail\wabmig.exe

Task name	**Description**
shi	Injects task’s data into a new process. The processes images paths are embedded in the binary and a random selection is made
dij	Injects task’s data into a new process. The injection method defers from the method used in task ‘dij’. The processes images paths are embedded in the binary [1] and a random selection is made.
dex	Writes task’s data into a file named **‘wab.exe’** under the Windows in AppData folder.

sdl	Deletes loader’s binary from disk.
ins	Adds persistence to the compromised host.

For the persistence mechanism, BUMBLEBEE creates a new directory in the Windows AppData folder with the directory’s name being derived by the **client_id** MD5 value. Next, BUMBLEBEE copies itself to its new directory and creates a new VBS file with the following content:

Set objShell = CreateObject(“Wscript.Shell”)

objShell.Run “rundll32.exe my_application_path, IternalJob”

Lastly, it creates a scheduled task that has the following metadata (this can differ from sample to sample):

1. Task name – Randomly generated. Up to 7 characters.
2. Author – Asus
3. Description – Video monitor
4. Hidden from the UI: True
5. Path: %WINDIR%\\System32\\wscript.exe VBS_Filepath

Similarly with the directory’ name, the new loader’s binary and VBS filenames are derived from the ‘client_id’ MD5 value too.

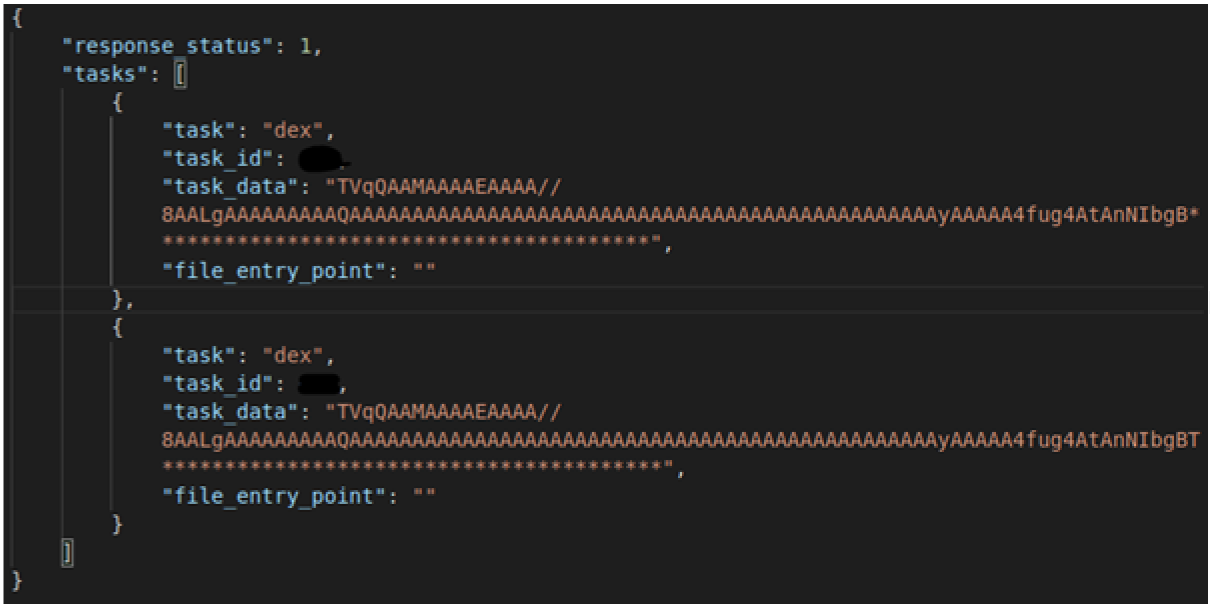
Additional Observations

This sub-section contains notes that were collected during the analysis phase and worth to be mentioned too.

- The first iterations of BUMBLEBEE were observed in September 2021 and were using “**/get_load**” as URI. Later, the samples started using “**/gate**“. On 19th of April, they switched to “**/gates**“, replacing the previous URI.
- The **“/get_load”** endpoint is still active on the recent infrastructure – this is probably either for backwards compatibility or ignored by the operator(s). Besides this, most of the earlier samples using URI endpoint are uploaded from non-European countries.
- Considering that BUMBLEBEE is actively being developed on, the operator(s) did not implement a command to update the loader’s binary, resulting the loss of existing infections.
- It was found via server errors (during network requests and from external parties) that the backend is written in Golang.
- As mentioned above, every BUMBLEBEE binary has an embedded group tag. Currently, we have observed the following group tags:

VPS1GROUP	ALLdll
VPS2GROUP	1804RA
VS2G	1904r
VPS1	2004r
SP1	1904l
RA1104	25html
LEG0704	2504r
AL1204	2704r
RAI1204	

- As additional payloads, NCC Group’s RIFT has observed mostly Cobalt Strike and Meterpeter being sent as tasks. However, third parties have confirmed the drop of Sliver and Bokbot payloads.
- While analyzing NCC Group’s RIFT had a case where the command-and-control server sent the same Meterpeter PE file in two different tasks in the same request to be executed. This is probably an attempt to ensure execution of the downloaded payload (Figure 1). There were also cases where the server initially replied with a Cobalt Strike beacon and then followed up with more than two additional payloads, both being Meterpeter.



```
{
    "response_status": 1,
    "tasks": [
        {
            "task": "dex",
            "task_id": [redacted],
            "task_data": "TVqQAAMAAAAEAAAA//
            8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAyAAAAA4fug4AtAnNIbgB*
            ****************************************",
            "file_entry_point": ""
        },
        {
            "task": "dex",
            "task_id": [redacted],
            "task_data": "TVqQAAMAAAAEAAAA//
            8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAyAAAAA4fug4AtAnNIbgBT
            *****************************************",
            "file_entry_point": ""
        }
    ]
}
```

Figure 1 – Duplicated received tasks

In one case, the downloaded Cobalt Strike beacon was executed in a sandbox environment and revealed the following commands were executed by the operator(s):

```
net group “domain admins” /domain
ipconfig /all
netstat -anop tcp
execution of Mimikatz
```

Indicators of Compromise

Type	Description	Value
IPv4	Meterpreter command-and-control server, linked to Group ID **2004r** & **25html**	23.108.57[.]13
IPv4	Meterpreter command-and-control server, linked to Group ID **2004r** & **2504r**	130.0.236[.]214
IPv4	Cobalt Strike server, linked to Group ID **1904r**	93.95.229[.]160
IPv4	Cobalt Strike server, linked to Group ID **2004r**	141.98.80[.]175
IPv4	Cobalt Strike server, linked to Group ID **2504r** & **2704r**	185.106.123[.]74

IPv4	BUMBLEBEE command-and-control servers	103.175.16[.]45 103.175.16[.]46 104.168.236[.]99 108.62.118[.]236 108.62.118[.]56 108.62.118[.]61 108.62.118[.]62 108.62.12[.]12 116.202.251[.]3 138.201.190[.]52 142.234.157[.]93 142.91.3[.]109 142.91.3[.]11 149.255.35[.]167 154.56.0[.]214 154.56.0[.]216 168.119.62[.]39 172.241.27[.]146 172.241.29[.]169 185.156.172[.]62 192.236.198[.]63 193.29.104[.]176 199.195.254[.]17 199.80.55[.]44 209.141.59[.]96 209.151.144[.]223 213.227.154[.]158 213.232.235[.]105 23.106.160[.]120 23.106.160[.]39 23.227.198[.]217 23.254.202[.]59 23.81.246[.]187 23.82.140[.]133 23.82.141[.]184 23.82.19[.]208 23.83.133[.]1 23.83.133[.]182 23.83.133[.]216 23.83.134[.]110 23.83.134[.]136 28.11.143[.]222 37.72.174[.]9 45.11.19[.]224 45.140.146[.]244 45.140.146[.]30 45.147.229[.]177 45.147.229[.]23 45.147.231[.]107 49.12.241[.]35 71.1.188[.]122 79.110.52[.]191 85.239.53[.]25 89.222.221[.]14 89.44.9[.]135 89.44.9[.]235 91.213.8[.]23 91.90.121[.]73

References

[1] – https://github.com/LordNoteworthy/al-khaser

[2] – https://blog.google/threat-analysis-group/exposing-initial-access-broker-ties-conti/