Microcin is here

By Denis Legezo

Published: 2020-06-19 · Archived: 2026-04-05 14:52:35 UTC

In February 2020, we observed a Trojan injected into the system process memory on a particular host. The target turned out to be a diplomatic entity. What initially attracted our attention was the enterprise-grade API-like (application programming interface) programming style. Such an approach is not that common in the malware world and is mostly used by top-notch actors.

Due to control server reuse (Choopa VPS service), target profiling techniques and code similarities, we attribute this campaign with high confidence to the SixLittleMonkeys (aka Microcin) threat actor. Having said that, we should note that they haven't previously applied the aforementioned coding style and software architecture. During our analysis we didn't observe any similar open source tools, and we consider this to be the actor's own custom code.

To deliver a new network module with a coding style that we consider enterprise-grade, Microcin used steganography inside photos, including this one of a sock (payload removed here)

SixLittleMonkeys' sphere of interest remains the same – espionage against diplomatic entities. The actor is still also using steganography to deliver configuration data and additional modules, this time from the legitimate public image hosting service cloudinary.com. The images include one related to the notorious GitLab hiring ban on Russian and Chinese citizens. In programming terms, the API-like architecture and asynchronous work with sockets is a step forward for the actor.

Why we consider the current software architecture interesting

By “enterprise-grade API-like programming style” we mean, firstly, asynchronous work with sockets. In terms of Windows user-space entities, it was I/O completion ports. In the OS kernel space, this mechanism is actually a queue for asynchronous procedure calls (APC). We believe there’s a reason for using it in backend applications on the high-loaded server-side. Obviously, however, neither client-side software nor Trojans of this kind need this server-side programming approach. So, it looks to us like the developers have applied some habits from server-side programming.

Secondly, the exported function parameters in the injected library look more like an API: the arguments are two callback functions – encryptor/decryptor and logger. So, if the authors decide to change encryption or logging algorithms, they could do so easily without even touching the network module. Once again, even targeted malicious samples rarely take such architectural issues into consideration.

Another injected library’s exported function parameter is the host name. If the caller doesn’t pass the infected host name as this parameter, the following commands will not be executed. It filters out all messages to other hosts.

Initial infection

Module features	File name	Detection time
Backdoor sideloaded by legit GoogleCrashHandler	version.dll	2019.12.31
Downloader/decryptor inside spoolsv.exe address space	spoolsv.dll	2020.01.16
Bitmap picture with steganography inside	Random .bmp name	2020.01.16
Network module in the same spoolsv.exe address space	Module.dll	2020.01.16

Infection timeline

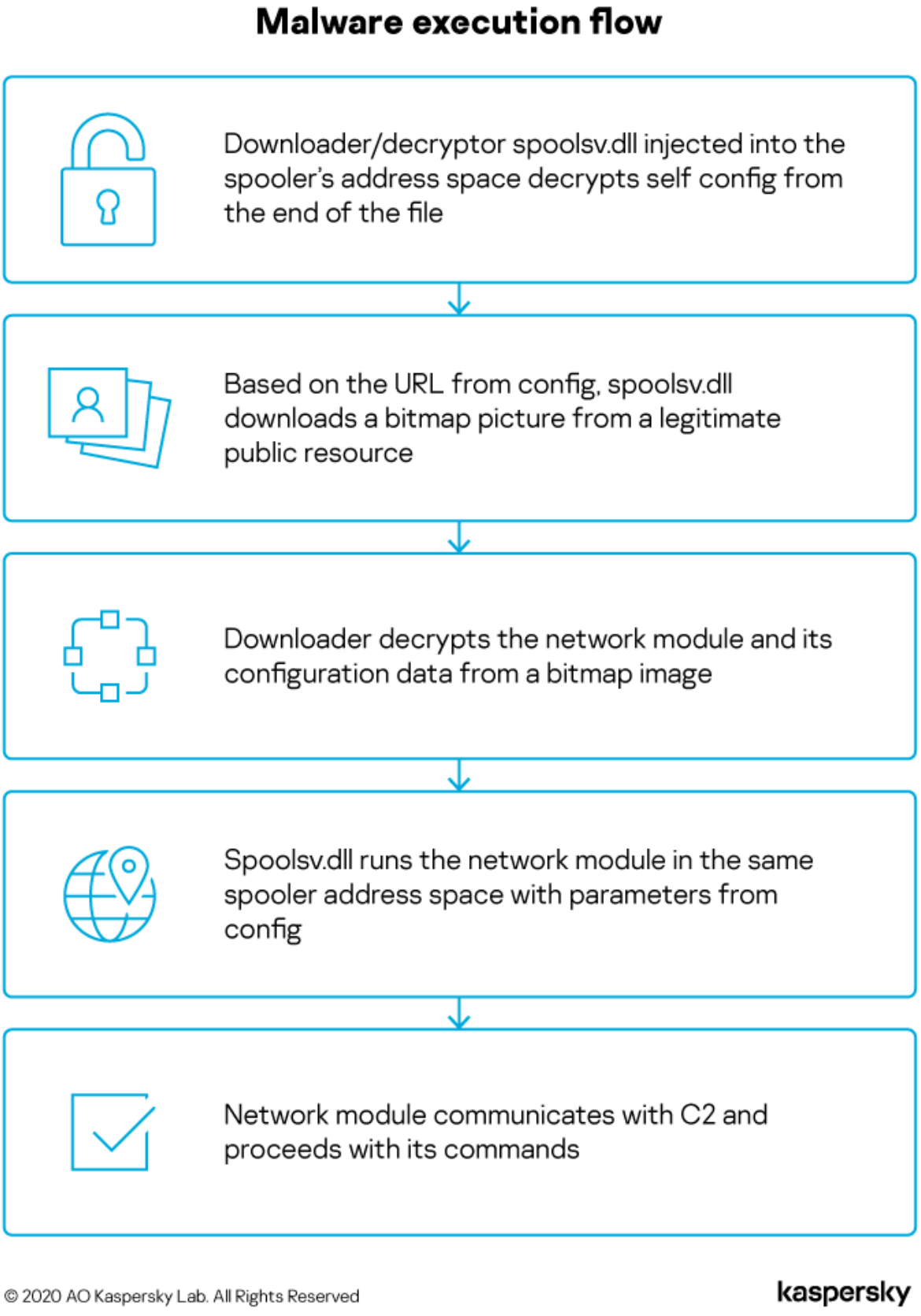
The backdoor is started by GoogleCrashHandler.exe, due to .dll search order hijacking (version.dll). Bitmap files with a steganography downloader and decryptor (spoolsv.dll), injected into the spoolsv.exe API-like network module, are injected into the same system process.

Let’s cover the modules one at a time. Our telemetry shows that another Microcin backdoor was already on the host before this new network module. It’s most probably a reinfection with newer malware.

Backdoor MD5	File name	Compilation timestamp	Size
c9b7acb2f7caf88d14c9a670ebb18c62	version.dll	2020.05.20 02:37:58	407552

This UPX packed .dll was executed with the legitimate GoogleCrashHandler.exe (very common library search order hijacking) just before the New Year. The compilation timestamp is obviously spoofed. In this case we don’t know how the backdoor, along with the legitimate application, was delivered.

We won’t concentrate on this backdoor in this report, because it’s fairly typical for Microcin. We just want to emphasize that the timeline above shows it existed on the host before the analyzed module.



Downloader/decryptor

The campaign in question starts with the 64-bit spoolsv.dll downloader/decryptor module that has to be loaded by spoolsv.exe into its address space.

Downloader/decryptor MD5	Modified time	Size	Build	Target ID
c7e11bec874a088a088b677aaa1175a1	2020.03.04 12:20:13	155291	20200304L02f	@TNozi96
ef9c82c481203ada31867c43825baff4	2019.10.15 11:46:04	145233	20200120L03o	@TNozi96
1169abdf350b138f8243498db8d3451e	2019.01.25 04:58:15	150195	20191119L	123456

So far, we have registered three samples of this module. The file tails contains the following encrypted configuration data.

Parameter	Length (bytes)	Possible values
.bmp URL len	4	82
.bmp URL	.bmp URL len	http://res.cloudinary.com/ded1p1ozv/image/upload/v1579489581/<random_name>.bmp
Sleep time	2	17211 and other non-round random numbers
Module build length	4	15
Module build	Module build length	Date based on the previous table
Target ID length	4	9
Target ID	Target ID length	Readable strings from the previous table
Random ASCII chars	16	Randomly generated on host
Hardcoded canary	4	0x5D3A48B6

We have published the source code of our decryptor for Microcin's configuration and steganography at https://github.com/dlegezo/common.

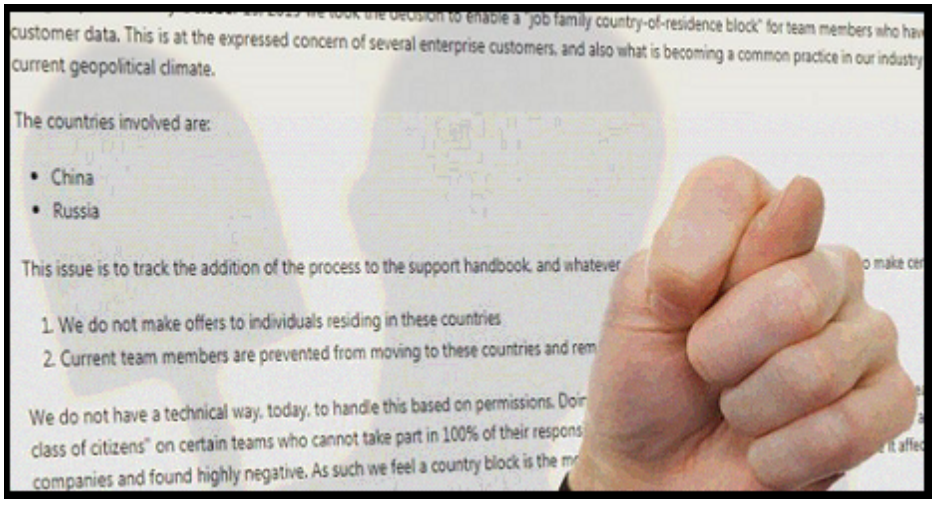
The bitmap URL serves to download the image (like the one with the sock shown above) with the next stage network module. The module build, target ID and random ASCII chars are for the next network module, which includes them in the control server communications.

To get the bitmap, the downloader sends an HTTP GET request to cloudinary.com. The steganography is inside the color palette part of the .bmp file. A typical decryption algorithm includes four stages:

1. 1 Combine neighboring half bytes into one byte
2. 2 Decrypt data length with custom XOR-based algorithm
3. 3 Decrypt six-byte XOR key for main data
4. 4 Decrypt data itself using decrypted length and key

Besides the configuration data and steganography, the same algorithm is used for the C2 traffic. As we mentioned, due to the malware architecture, the latter can easily be changed. Encryption is XOR-based, but the key scheduling is quite specific and tricky. In the corresponding appendix we provide the part of the decryptor containing the algorithm.

Bitmap images and steganography



Besides the sock image, the campaign operators use more social-oriented photos (payload removed here). The background here is the GitLab hiring ban on Russian and Chinese citizens

So far, we have registered four different images. The encrypted content in all cases are PE files with the following network module and C2 domain for the files. This is the only parameter that comes from bitmap; all others are provided by the downloader.

Image content	C2 domain	Network module MD5
Sock in washing machine	apps.uzdarakchi[.]com	445b78b750279c8059b5e966b628950e

Two people in hoodies	forum.mediaok[.]info	06fd6b47b1413e37b0c0baf55f885525
GitLab hiring ban	forum.uzdarakchi[.]com	06fd6b47b1413e37b0c0baf55f885525
Woman with child, militaries	owa.obokay[.]com	06fd6b47b1413e37b0c0baf55f885525

Network in-memory module

The downloader decrypts the configuration data and C2 domain from the bitmap and then everything is ready to start the last stage inside the same spoolsv.exe virtual address space. We consider the architectural approach in this module to be the most interesting part of the chain.

The network module's entry point is the exported function SystemFunction000() with multiple arguments. As a beacon, the Trojan prepares an HTTP POST request with the target's fingerprinting data. And a lot of the parameters become part of the request.

Exported function argument	**Parameter meaning**
Target host name	This has to be the same as the infected machine host name. Only then will the Trojan start and receive commands. Initialized by the downloader
Target ID	We already enumerated these readable ASCII strings from the decrypted downloader's config, e.g., @TNozi96
Build version	Inside these readable ASCII strings the dates are clearly mentioned. The C2 uses them to understand which build it's currently working with
WORD field of fingerprint structure	Initialized with 0x4004 by the downloader. We don't have enough data to describe this field's meaning
C2 IP address and port number	The coordinates of the C2, initialized from the decrypted bitmap image
ASCII string in fingerprint structure	Unique random string generated by the downloader
BYTE to fingerprint structure	Initialized with 0x4004 by the downloader. We don't have enough data to describe this field's meaning
Half of maximum sleep time	Sleep time before the working cycle. Half because the full time is counted <this arg> + <random>%<this arg>. It's effectively a maximum of a maximum sleep time
Logger address	First callback function address. In this case it's a logger function inside the downloader

Encryptor/decryptor address	Second callback function address. In this case it’s an encryptor/decryptor function inside the downloader

The last two arguments illustrate why we call the network module API-like: any encryption and logging routine could be used without even touching the module code. We consider this programming approach as scalable and useful for large systems. Let’s take a look at these two callback arguments.

Callback and its arguments	**Callback features**
Logger takes ASCII string as a log message	Logger function whose parameter is the message text. In this module all the messages are shortenings like “LIOO”, “RDOE”, etc.
Encryptor/decryptor to deal with the traffic between host and C2, takes its length, encryption key, and the flag (0 to encrypt and 1 to decrypt) as argument data	Encryptor/decryptor function first used to encrypt beacon with target’s fingerprint. It then decrypts C2 command structures and encrypts replies to them

The module uses the Windows API function WSAIoctl() – something rarely seen in malware – to get the ConnectEx() address and sends a prepared request. Another Windows API function, GetQueuedCompletionStatus(), is in charge of asynchronous work with I/O. In other words, the malware uses I/O completion ports for Windows user-space entities, which is effectively an APC queue in the OS kernel.

The same data structure is used for both sides of the communication: from host to C2 and back. Let’s describe its main fields here.

Field	**Features**
Command code	One byte in the structure is the command code, which could vary from 0x00 to 0x16 (22). We describe the main network module commands in the table below
Error code	Another byte is used for the error code
Command argument	The main command field that takes all the necessary strings, etc. and also keeps fingerprinting data in the case of the beacon

So far, we have described the infection chain, module architecture, custom encryption and HTTP POST-based C2 communication protocol. Last, but not least, is the command set shown in the table below.

Command code	**Command features**
3	Check if target’s ID meets the parameter
4	List logical drives
5	List files

6	Create directory
7	Remove directory
8	Copy file
9	Move file
10	Delete file
11	Execute PE
12	Execute Windows shell command
14	Terminate program
15	File download
16	Read from downloaded file
17	File upload
18	Write to file
19	Stop
20	Sleep

Infrastructure

Domain	IP	First seen	ASN
apps.uzdarakchi[.]com	95.179.136[.]10	November 11, 2019	20473
forum.uzdarakchi[.]com	172.107.95[.]246	February 7, 2020	40676
forum.mediaok[.]info	23.152.0[.]225	March 19, 2020	8100
owa.obokay[.]com	N/A (now parked)		

To sum up

This time the Microcin campaign has made an interesting step forward, not in terms of a fancy initial infection vector, but as programmers. The API-like network module is much easier to support and update. This improvement is not only about anti-detection or anti-analysis; it’s about software architecture and a step towards a normal non-monolithic framework implementation.

IoC

Downloader

ef9c82c481203ada31867c43825baff4
1169abdf350b138f8243498db8d3451e
c7e11bec874a088a088b677aaa1175a1

Network module

f464b275ba90b3ba9d0a20b8e27879f5
9320180ef6ee8fa718e1ede01f348689
06fd6b47b1413e37b0c0baf55f885525
625a052ddc80efaab99efef70ba8c84f

Domains and IPs

95.179.136.10
apps.uzdarakchi[.]com
forum.uzdarakchi[.]com
forum.mediaok[.]info
owa.obokay[.]com
