Legitimate Sites used as Cobalt Strike C2s against Indian Government

telsy.com/legitimate-sites-used-as-cobalt-strike-c2s-against-indian-government/

March 4, 2022

Cyber Threat Intelligence
04 Mar

Introduction

Telsy Threat Intelligence team observed an attack against members of the Indian government or local institutions, which uses social engineering themes as an investigation for a cyber attack or the classic COVID-19 theme.

The campaign, probably carried out via a spear phishing e-mail, starts with the opening of a legitimate PDF attachment containing a malicious URL from which to download an ISO file. The ISO file contains LNK files and a malicious DLL that executes a Cobalt Strike beacon in memory.

Using a legitimate portal as C2 and encrypted HTTPS communication makes the campaign very silent.

Cobalt Strike is a commercial penetration testing tool, which gives security testers access to a large variety of attack capabilities.

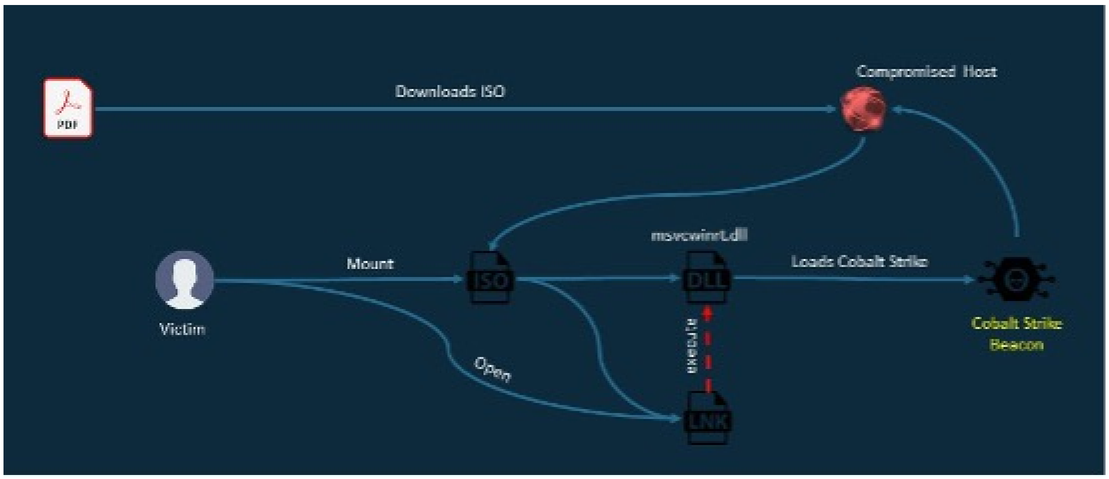
This powerful network attack platform combines social engineering, unauthorized access tools, network pattern obfuscation, and a sophisticated mechanism for deploying malicious executable code on compromised systems.

Therefore Cobalt Strike although a legitimate tool used by ethical hackers is also widely used by threat actors to launch real attacks against organizations.

Most threat actors either use stolen/cracked versions of Cobalt Strike, or simply patch out the watermark value to disrupt attribution attempts.

Cobalt Strike’s watermark 1359593325 and the analyzed infection chain might lead one to think of the threat actor Nobelium aka APT29 due the similarities, both in components and how the target is infected as previously described by security companies Volexity and Microsoft.

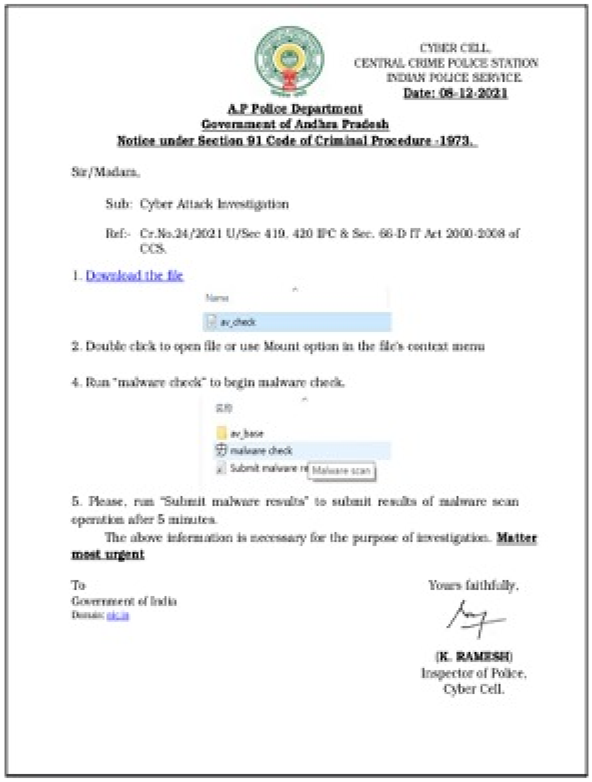
Unfortunately, there is no clear evidence to attribute these campaigns to this threat actor.



Analysis

First PDF Analysis

The 1st PDF found, with hash 0b1cc9a276712b1d6f379b43504bd1f1d8a49cfd, has been uploaded to VirusTotal on: 2021-12-08.



CYBER CELL
CENTRAL CRIME POLICE STATION
INDIAN POLICE SERVICE
Date: 08-12-2021

A.P Police Department
Government of Andhra Pradesh
Notice under Section 91 Code of Criminal Procedure -1973.

Sir/Madam,

Sub: Cyber Attack Investigation

Ref:- Cr.No.24/2021 U/Sec 419, 420 IPC & Sec. 66-D IT Act 2000-2008 of CCS.

1. Download the file

2. Double click to open file or use Mount option in the file's context menu

4. Run "malware check" to begin malware check.

5. Please, run "Submit malware results" to submit results of malware scan operation after 5 minutes.

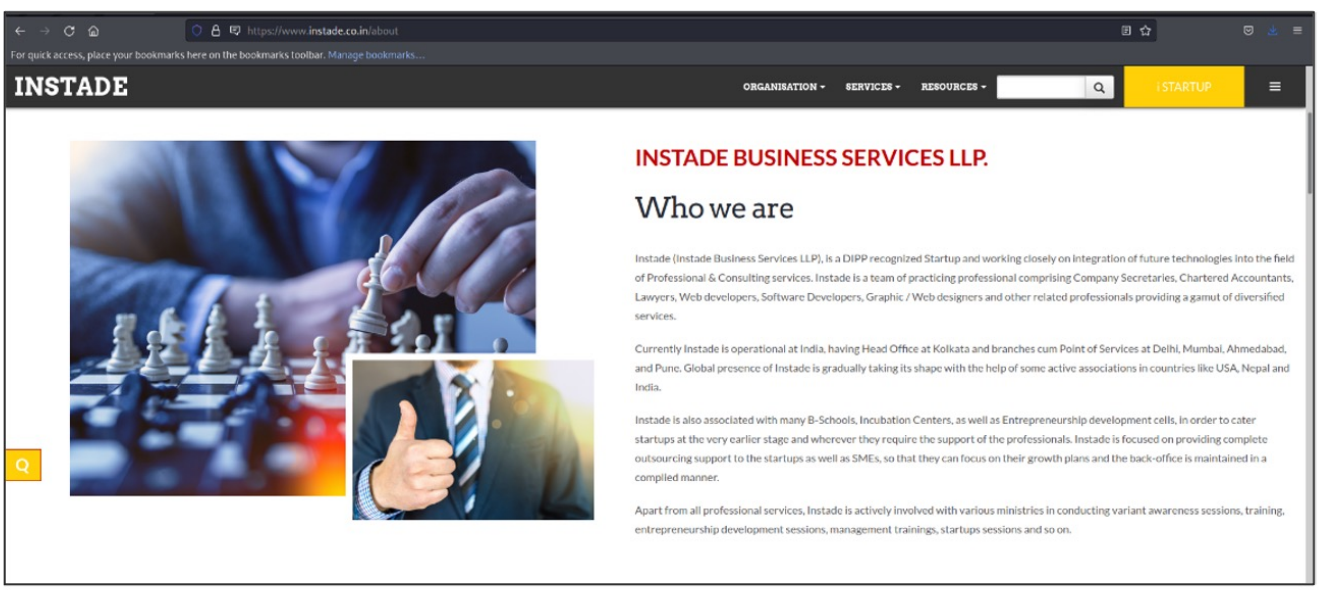
The above information is necessary for the purpose of investigation. **Matter most urgent**

To
Government of India
Domain: nic.in

Yours faithfully,

(K. RAMESH)
Inspector of Police,
Cyber Cell.

The PDF is intended to trick the user by downloading an ISO from "hxxps://www.instade.co.in/assets/frontend/av_check.iso" which is still active at the time of writing. The domain "instade.co.in" appears to be legitimate, it uses a certificate issued by Sectigo and according to information in the Whois registry was registered in 2015.



The downloaded ISO, with hash d5edd698c944accea764ff74978ca3d86067afab, contains the following files:

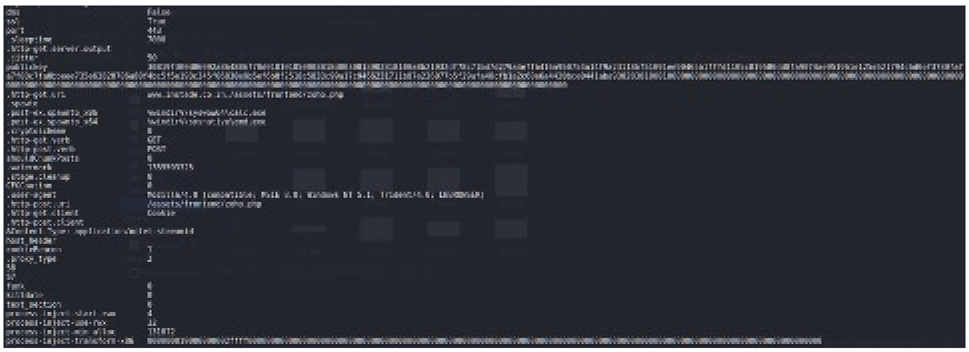
• 2dcbe02294e633f49806c2d5d0d1f1207a0b1959 – 'malware check.lnk'

• 9152e25c2574cccba6c7bfed2e598f9ce2afdcd0 – 'Submit malware report.doc.lnk'

• 44ee7f74ca1553af0e5484213dea676c66371e53 – 'av_base/msvcwinrt.dll'

Opening one of the LNK files causes the DLL to be executed and consequently, the Cobalt Strike beacon infects the system. The DLL is executed via rundll32.exe by specifying the exported 'InitShut()' function to be executed.

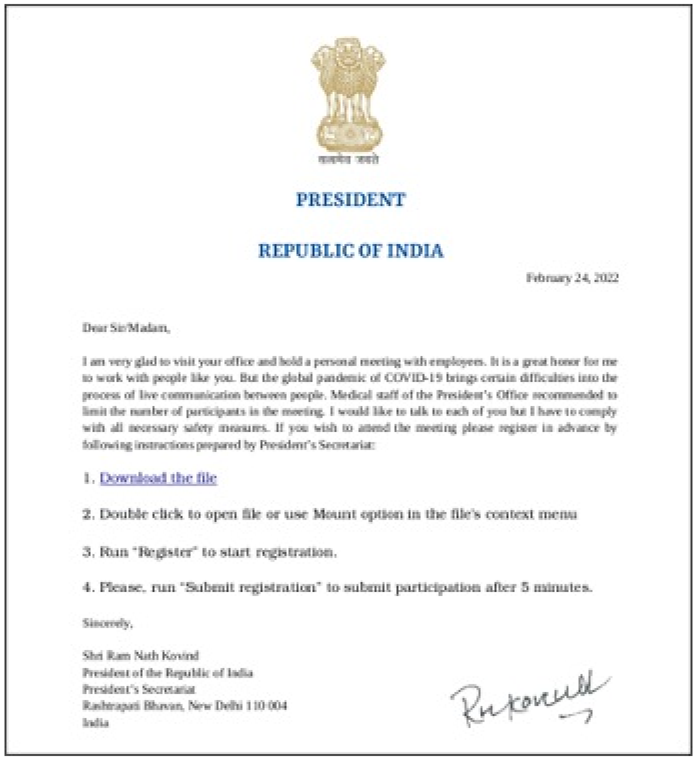
The DLL, as said it's just a Cobalt Strike loader, the Cobalt Strike beacon configuration is the following.



The Cobalt Strike beacon uses the same compromised domain as C2, as seen above for the ISO download.

Second PDF Analysis

The 2nd PDF analysed, with hash e648483ce584211520a20a155ebcd3f70166fa93 and named 'President-Kovind-special-visit-2022.02.24.pdf', is more recent and was uploaded to VirusTotal on 2022-02-24. This PDF uses COVID-19 prevention as a decoy before the meeting with the Indian president.



सत्यमेव जयते

PRESIDENT

REPUBLIC OF INDIA

February 24, 2022

Dear Sir/Madam,

I am very glad to visit your office and hold a personal meeting with employees. It is a great honor for me to work with people like you. But the global pandemic of COVID-19 brings certain difficulties into the process of live communication between people. Medical staff of the President's Office recommended to limit the number of participants in the meeting. I would like to talk to each of you but I have to comply with all necessary safety measures. If you wish to attend the meeting please register in advance by following instructions prepared by President's Secretariat:

1. Download the file

2. Double click to open file or use Mount option in the file's context menu

3. Run "Register" to start registration.

4. Please, run "Submit registration" to submit participation after 5 minutes.

Sincerely,

Shri Ram Nath Kovind
President of the Republic of India
President's Secretariat
Rashtrapati Bhavan, New Delhi 110 004
India

The targets of this campaign are most likely the participants of the event advertised on the Indian government portal as members of one or more of these organizations:

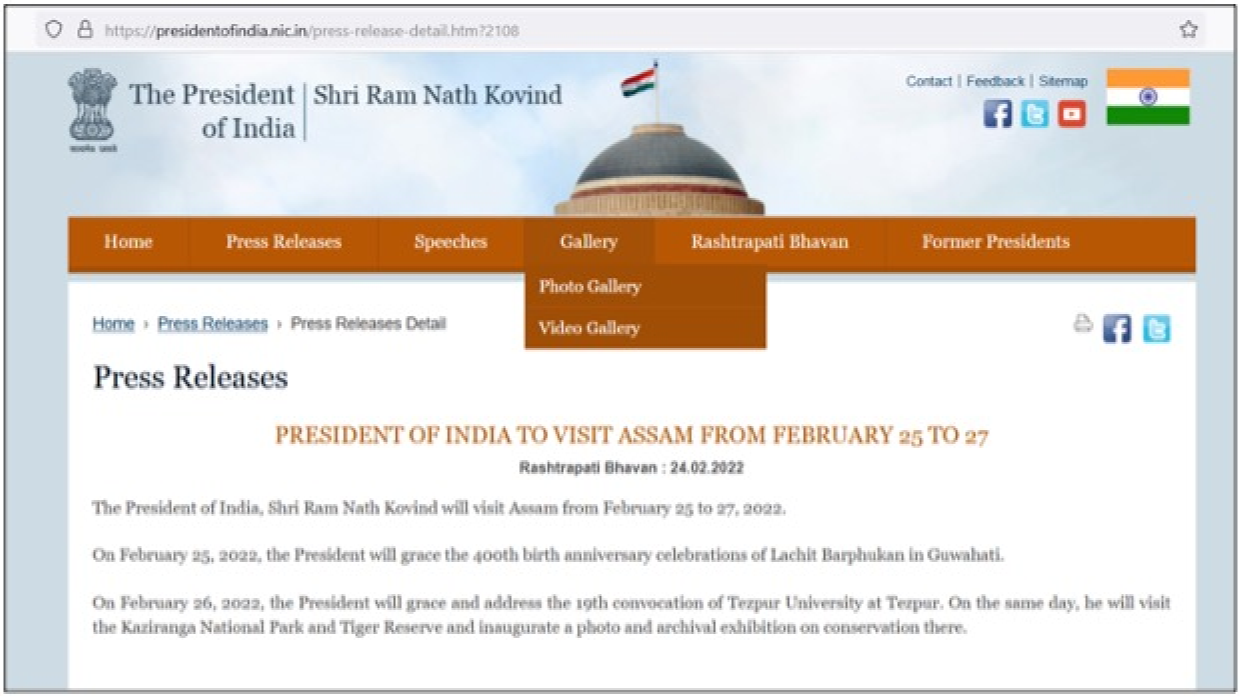
– Assam State

– Guwahati Municipal

– Tezpur University

– Kaziranga National Park

– Tiger Reserve



The PDF, as the previous tricks the user in downloading an ISO from the following URL hxxps://tiny.one/covid22. Also in this case the link is still up on the time being.

The downloaded ISO, with hash e2ff656f52dccc9fb70e90dc94c4fce8ab14e8ed, contains the following files:

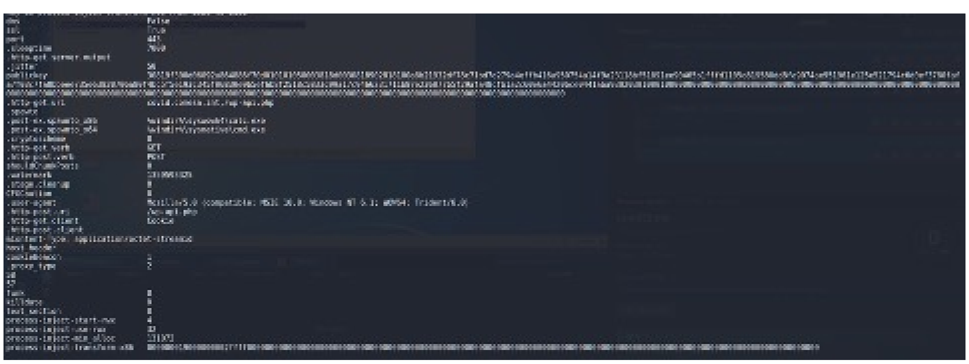
• b2a095b6e1dad70df03763a385ff04a1036065be – 'Register.lnk'

• bd165723292f62e4be7ae60d12c25461900519fb – 'Submit registration.lnk'

• f80ee71efcea4736b41d6ffed777ff1bb5621043 – 'data/msvcwinrt.dll'

Once the LNK file is opened the malicious DLL is run through rundll32.exe specifying the exported function named 'AwaitProperty()".

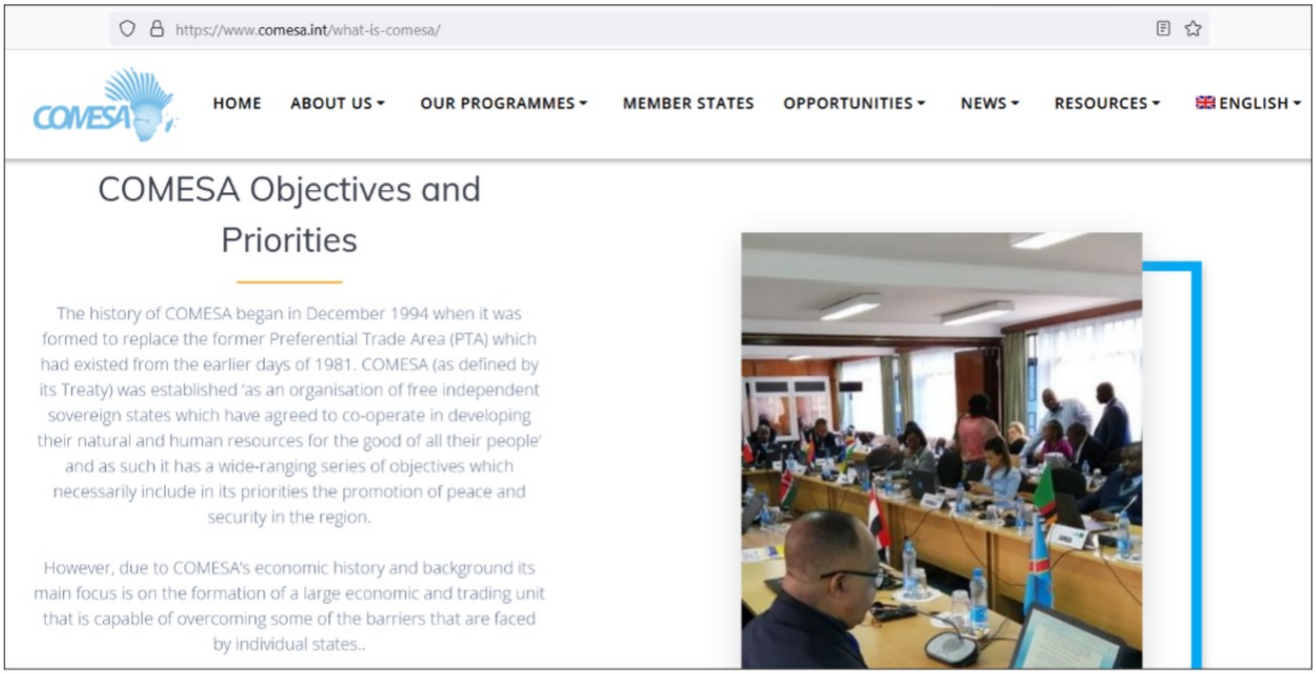
Also in this case the DLL is a Cobalt Strike loader and the beacon has the following configuration.

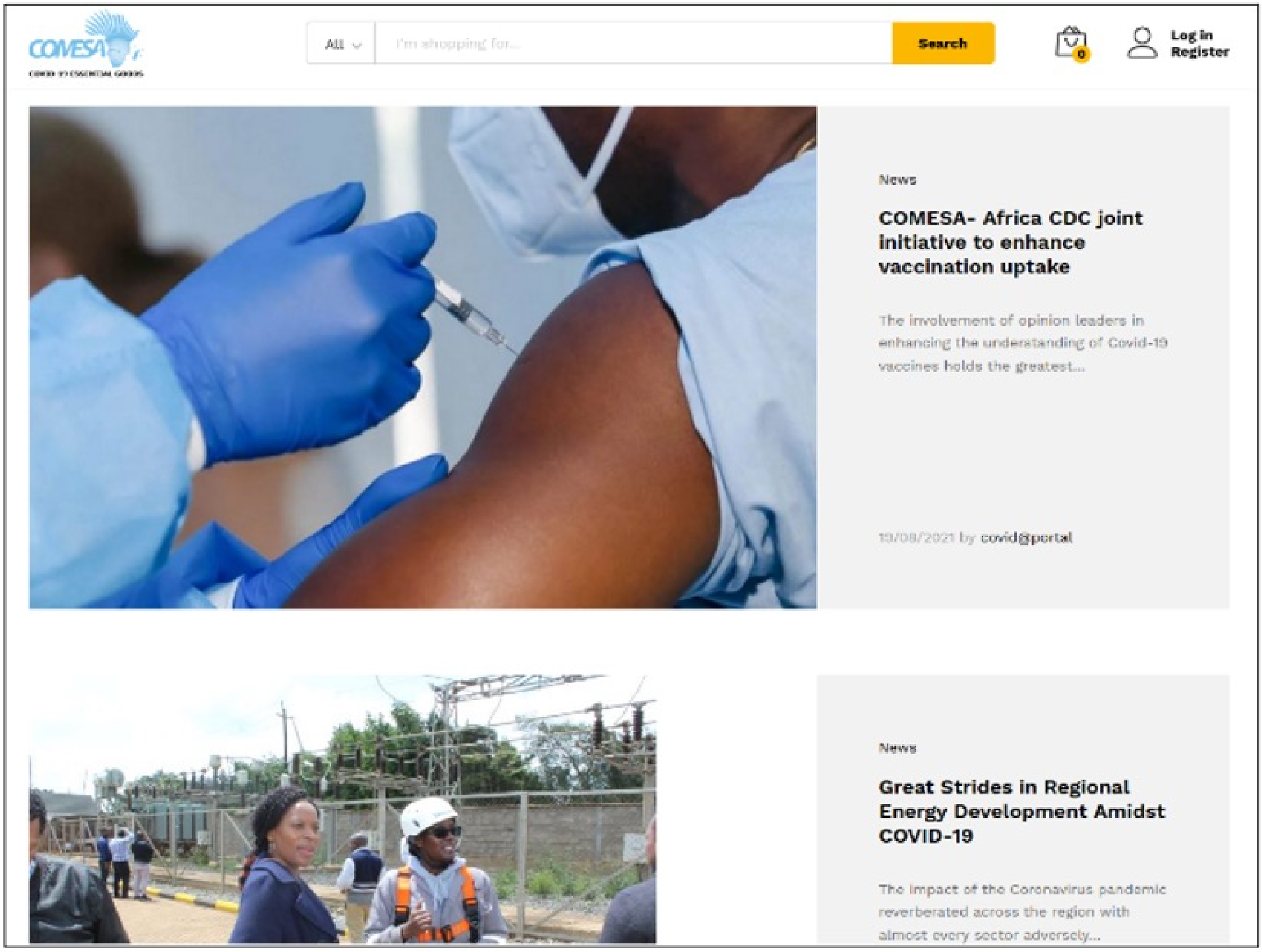


The public key and the watermark is the same as the previous beacon but the C2 is the domain 'covid.comesa.int'.

The same domain hosts the ISO, with hash e2ff656f52dccc9fb70e90dc94c4fce8ab14e8ed, in the following path: 'hxxps://covid.comesa.int/wp-content/uploads/covid.iso'.

The domain appears to be compromised, as “comesa.int” is the official website of the Common Market of Eastern and Southern Africa.



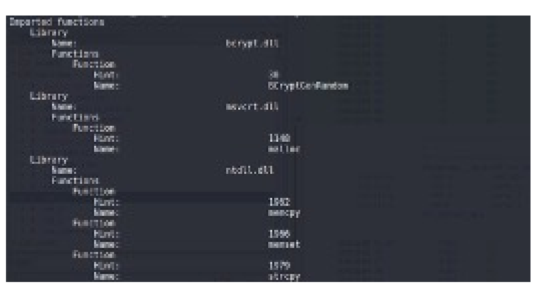


Cobalt Strike dropper analysis

Both the infection chains ends in a Cobalt Strike loader and the DLLs are pretty the same so the analysis has been conducted on the following hash: f80ee71efcea4736b41d6ffed777ff1bb5621043.

As said, the purpose of this DLL is to load and execute a Cobalt Strike beacon, indeed the sample appears very simple, even though the author has inserted some stub call between the significant code.

The sample imports a minimum set of functions, so it needs to load at runtime libraries and APIs.

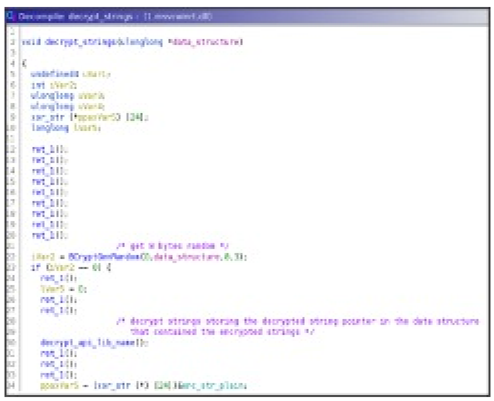


Libraries and APIs names are stored encrypted via XOR operation in the data section using a basic data structure. Every encrypted string has its own xor key stored in the same data structure.

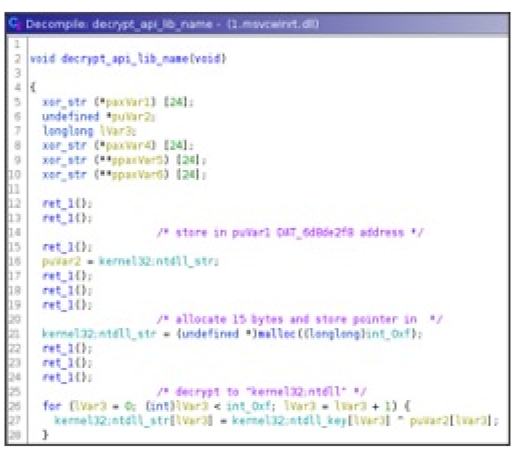
The structure is a basic array of struct, every item is 16 bytes long and the array with the encrypted string contains 24 items like the array with the xor key.

```
struct xor_strings
{
    void * ptr_buf;
    long len;
};
```

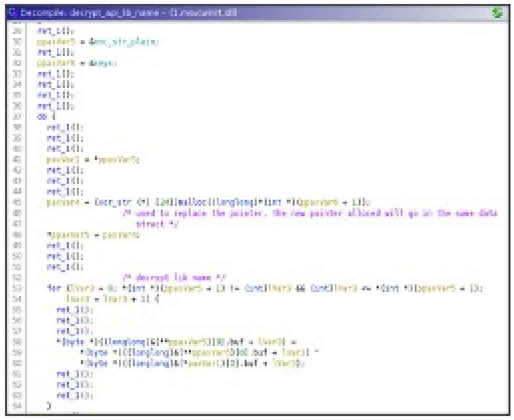
Before, entering in the specific function used to decrypt the strings, it takes a random string 8 bytes long.



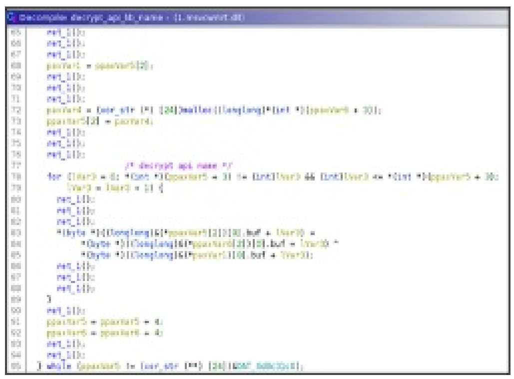
Then it decrypts the string 'kernel32;ntdll' using again the xor operator and a dedicated key.



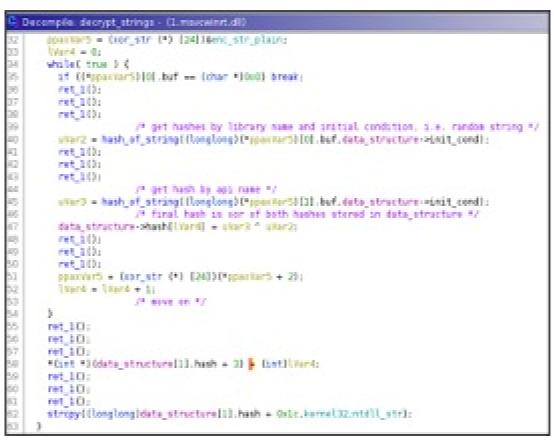
Finally, it decrypts the library name and the API name, notice how the threat actor use allocation to store the decrypted string instead of using existing space doing in-place decryption.



After that libraries and APIs are decrypted the strings are hashed with a custom algorithm and stored in the structure named 'data_structure'. Every hash will take 8 bytes.



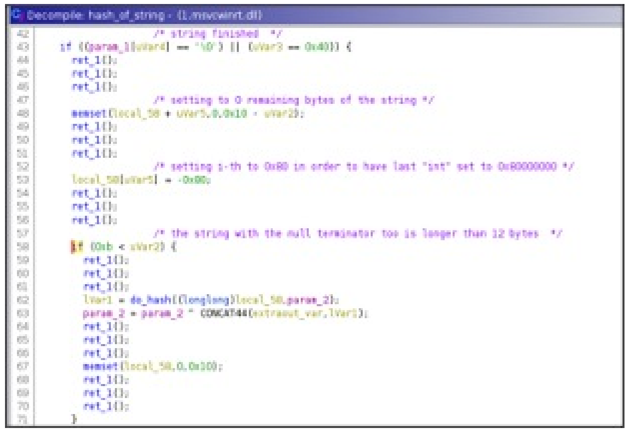
The data structure will contain all the hashes and the initial condition obtained randomically.



```
data_structure
{
    long init_condition_hash;
    long api_lib_hash[12];
}
```

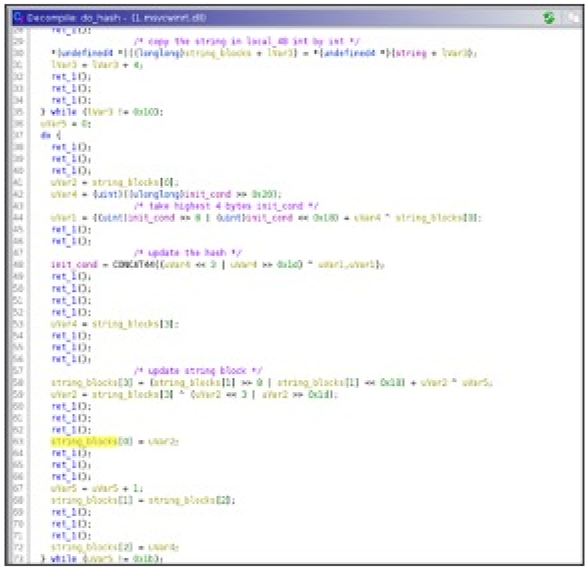
The string is hashed 16 bytes per time, the string, of course, can be of arbitrary length.

When the string is smaller than 16 bytes, it is aligned to 16 bytes adding 0x80 bytes and then setting the remainder to 0.



On the other hand, if the string is larger than 16 bytes the hash is calculated in chunks of 16 bytes and the remainder will follow the logic shown before. Of course the calculated hash is incremental, i.e. the hash of the n-th chunk is xored to the hash of the (n-1) chunk and so on.

The hash is computed, starting from a generated random initial condition that according to the string value is updated multiple times in a loop and finally returned.

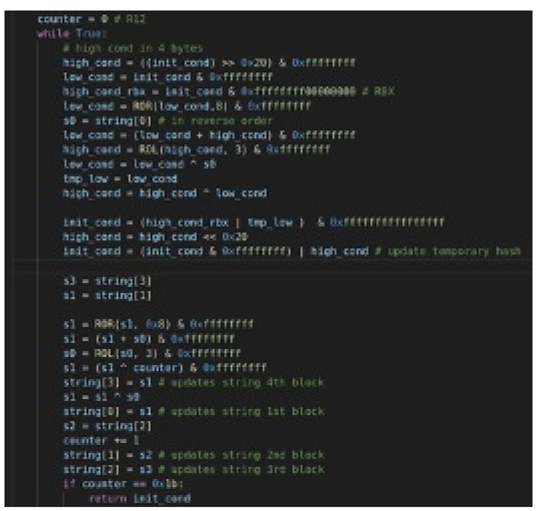


Basically, it treats the string/chunk, since it is 16 bytes long, in blocks 4 bytes long doing some shuffle and binary operation between the blocks self and the initial condition that is updated from time to time.

In particular, the final hash is due to 0x1b iteration of the hashing algorithm.

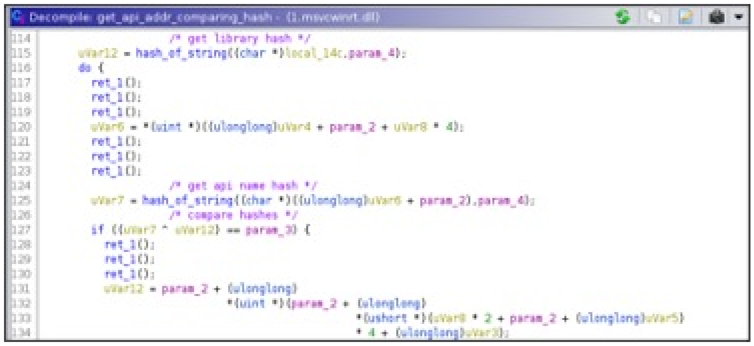
In the first for loop (line 30) the string API is copied as 4 block bytes long in an integer vector.

Then, in the second while loop the initial condition are updated according (line 48) to the string blocks and they are updated too in the same while, the code seems contorted, below a basic re-implementation.

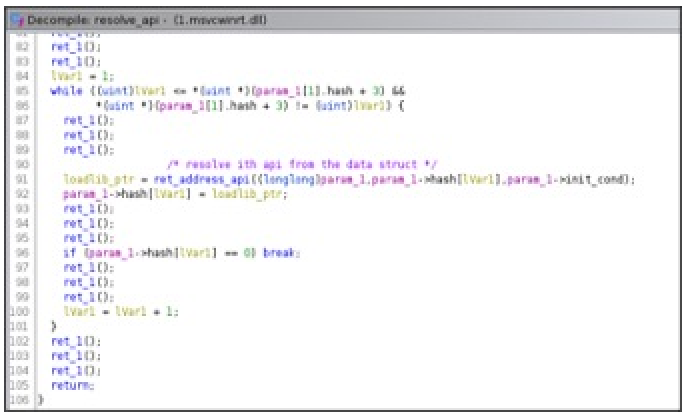


As said, the hash of every API to load are stored in the data structure then the API addresses are searched doing a basic walk into the PEB and checking the hash.

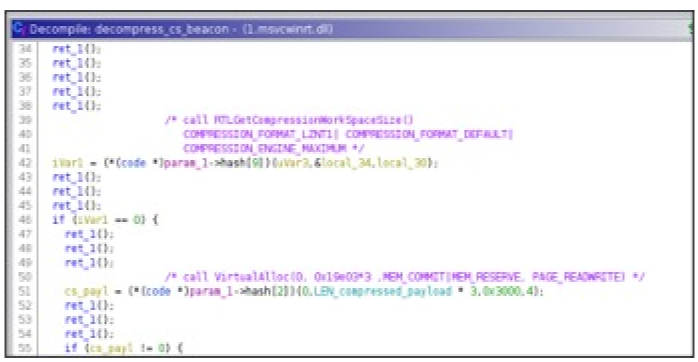
Every module and API found is hashed and compared with the hash of the API string obtained initially.



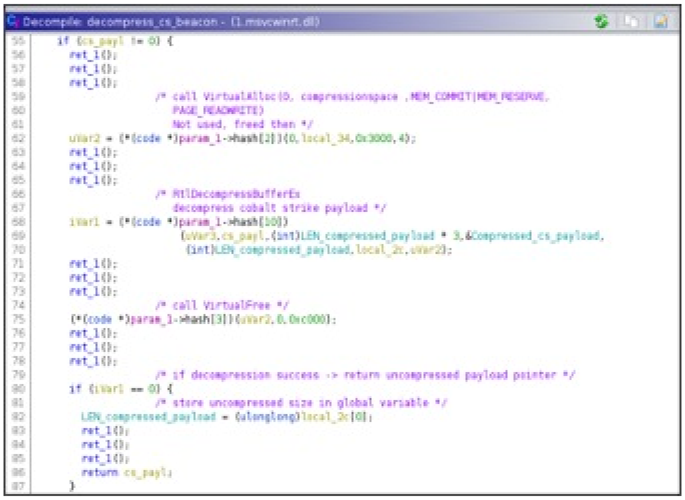
Finally, it loads all the APIs.



The payload is embedded in the binary using the compression algorithm: LNZT1.



Indeed, after allocing the required RW memory using VirtualAlloc() the payload is decompressed and the pointer is returned.



Not knowing the actual size of the decompressed payload, the memory allocated to contain it is allocated using the size of the compressed payload * 3 as its size.

Then the author wanted, perhaps for greater security, to insert a further step, i.e. allocate a new memory area, equal exactly to the decompressed payload size, copy into it and execute it.

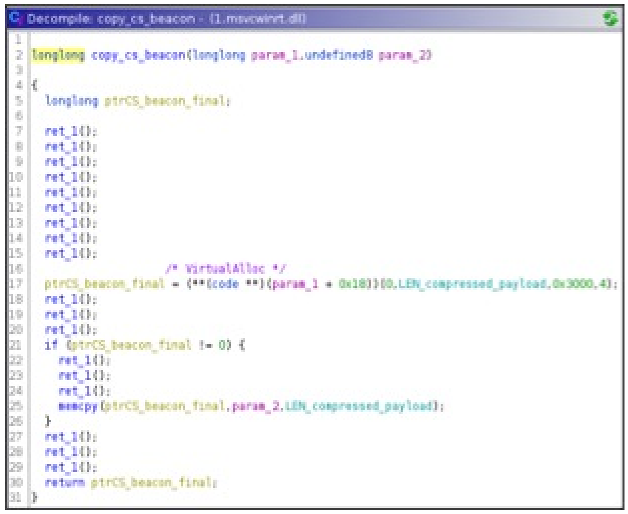
This way to write the code is not very logical nor correct.

Indeed assuming that the decompressed payload will take less of the initial space allocated there will be no problem in running directly it.

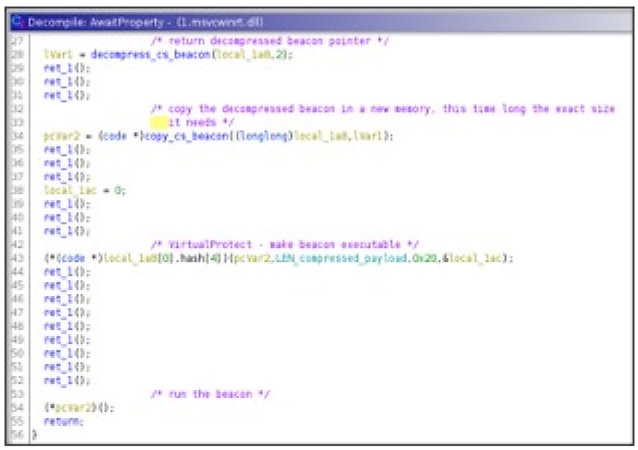
On the other hand, assuming what scares the author, i.e. decompressed payload longer than the allocated memory, the RtlDecompressBufferEX() will return an error, STATUS_BAD_COMPRESSION_BUFFER and will lead to a NULL pointer access of the code, very bad and basic error.

Another weird point is the use of the hash to resolve APIs. Usually, the hash is used to obfuscate strings and make harder analysis. Here the approach is hybrid, indeed doing a trace of the sample all the required API are uncovered due to the initial decryption step.

This behavior shows that the sample likely has been written by a not so skilled programmer or it is product of confused cut and paste of multiple code's pieces.



Anyway, in the end the new memory is made executable and run.

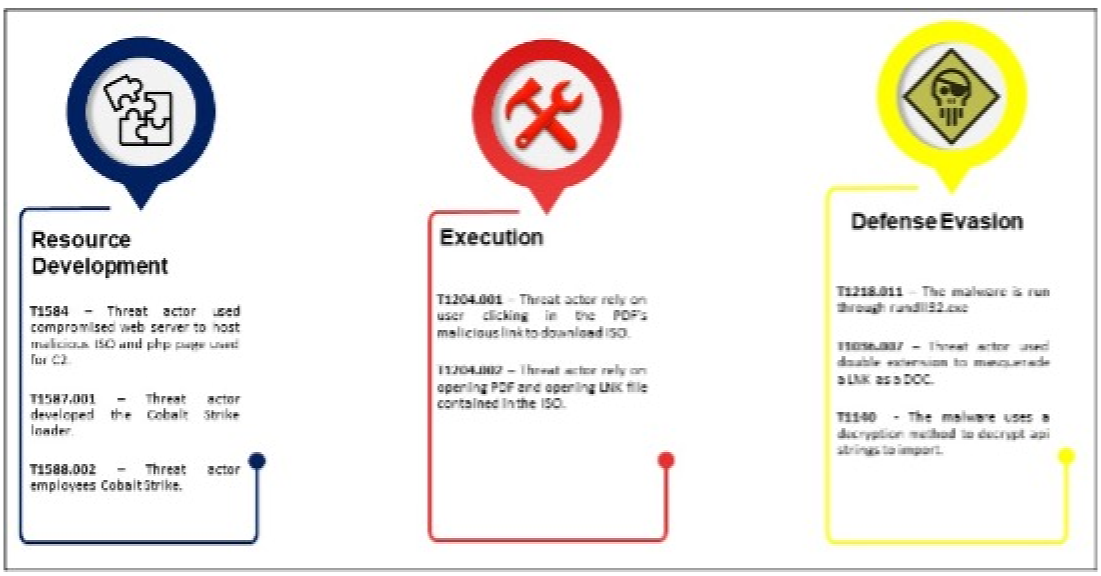


Indicators of Compromise

TYPE	HASH	NAME
PDF	*0b1cc9a276712b1d6f379b43504bd1f1d8a49cfd*	*Letter No.24-2021 of PS Dt. 08-12-2021.pdf*
ISO	*d5edd698c944accea764ff74978ca3d86067afab*	*av_check.iso*
LNK	*2dcbe02294e633f49806c2d5d0d1f1207a0b1959*	*malware check.lnk*
LNK	*9152e25c2574cccba6c7bfed2e598f9ce2afdcd0*	*Submit malware report.doc.lnk*
DLL	*44ee7f74ca1553af0e5484213dea676c66371e53*	*msvcwinrt.dll (Cobalt Strike Loader)*
PDF	*e648483ce584211520a20a155ebcd3f70166fa93*	*President-Kovind-special-visit-2022.02.24.pdf*
ISO	*e2ff656f52dccc9fb70e90dc94c4fce8ab14e8ed*	*covid.iso*
LNK	*b2a095b6e1dad70df03763a385ff04a1036065be*	*Register.lnk*
LNK	*bd165723292f62e4be7ae60d12c25461900519fb*	*Submit registration.lnk*
DLL	*f80ee71efcea4736b41d6ffed777ff1bb5621043*	*msvcwinrt.dll (Cobalt Strike Loader)*

DOMAIN - IP - URL
https://covid.comesa.int/wp-content/uploads/covid.iso (Domain Legit)
https://covid.comesa.int/wp-api.php (Domain Legit)
https://www.instade.co.in/assets/frontend/av_check.iso (Domain Legit)
https://www.instade.co.in/assets/frontend/zoho.php (Domain Legit)
https://tiny.one/covid22
tiny.one

ATT&CK Matrix



Fill out the form below to download the full report

Check other cyber reports on [our blog](#).

This report was produced by Telsy’s “Cyber Threat Intelligence” team with the help of its CTI platform, which allows to analyze and stay updated on adversaries and threats that could impact customers’ business.