PyViper - A previously unknown APT campaign leveraging PyInstaller

Report Id: 20231109
Version: 1.0 (24.Nov.2023)

Executive Summary

In August 2023 we discovered previously unknown malicious activity targeting companies in Russia and China active in the IT and hospitality industries. We analyzed this activity and discovered three malicious samples related to this attack. The original source code of the samples could be restored and we found that the main purpose of this threat is espionage. The operation cannot be tied to a known actor at this point.

One interesting feature of this threat is that the original malicious samples were written in Python and compiled to executable samples using PyInstaller. This is an atypical procedure for cyber attacks in generall, since the resulting executable become rather large size, up to 7.7Mb for this campaign.

We discovered various modules of this APT carrying capabilities of a keylogger, a screenshot functionality and the ability to search the infected system for specified files for future exfiltration. The infection vector is yet unknown to us.

This report in a nutshell:

- We discovered a previously unknown APT campaign called PyViper that usedsPyInstaller to create its modules from Python source code;
- Based on our telemetry, there are three victims of this campaign, but one hardcoded artefact in a malicious module suggests that it was developed to specifically target one of the victims based in Russia;
- This campaign stores the payload on a publically available Github repository, which is still active at the time of writing.

Techniques, Tactics and Procedures specific to this campaign:

Infrastructure

Github used as storage for malware and C2 addresses

VPS hosted at commercial hosting provider based in Luxembourg

Infection vector

Unknown

Implants

Previously unknown implant written in Python

Victimology

Three known victims, based in Russia and China, IT and hospitality verticals

MITRE´s ATT&CK® mapping (full details in Appendix III):

Tactic	Techniques
Execution	T1059.006, T1569.002
Persistence	T1543.003
Defense Evasion	T1140, T1036.004, T1036.005, T1027.002,
Credential Access	T1056.001
Discovery	T1083, T1057, T1082, T1039, T1074.001
Collection	T1560.002, T1119, T1115, T1056.001, T1113
Command and Control	T1071.001, T1132.001, T1573.001, T1102.001
Exfiltration	T1041

For more information please contact: intelreports@kaspersky.com

Table of Contents

Technical Details

Background

We discovered malicious activity in three cases as per our telemetry that we analyzed, restored the original source code, and came to the conclusion with medium level of confidence that this malicious activity has signs of APT and that the main purpose of this attack is, probably, espionage. All three samples are binaries created by PyInstaller. The second sample was found on three victims located in China and Russia, with one being from the hospitality industry.

We analyzed all three samples to find out their functionality and to understand their roles in this attack.

Searcher module

MD5	`4aebc2c53aaf777ec83c5aa14a43318f`
SHA1	`de942eaf447fa277136caccf741303bd9f958310`
SHA256	`f04c1bc048617641eadc42d9fe68c07f509ec5ee25350291b47e3440a8fb6b26`
Link time	`Sun Jan 05 15:15:27 2020 UTC+3`
File type	`PE (x64)`
Compiler	`Visual C/C++, Microsoft Visual Studio (2015)` `(PyInstaller generated)`
File size	`3,983,589`
File name	`GoogleUpdate.exe`

We started our analysis with the searcher module. This sample was built using PyInstaller, a tool that merges a Python application and all its dependencies into a single package. Such packages can be unpacked and decompiled and as a result we can obtain original Python files as well as added binaries. Among the extracted Python files the main Python application can be found, which again we unpacked and decompiled. As a result, the payload was found in the file called `search.py`.

The file name already gives away what the main purpose of this sample is, which is finding files carrying specified names. Once found, this script copies their paths to the file `res.txt`, located in the hard coded folder "`C:\Users\Public\Downloads\`". The file names to be searched are specified and hard coded within the sample's code. One such file name suggests that it was developed to specifically target one of the victims.

Based on its functionality, we called this module 'Searcher'.

Spy module

MD5	aa0e6fc2495b89ff0800465703f1d548
SHA1	b7f298bd37a0e6d70d077c2fd556aefbfd316a0f
SHA256	c1e570d29911da858f7f22a8ebefc0aaf03cde4428bb7d51e459e9c9c496be14
Link time	`Tue Sep 04 17:43:33 2018 UTC+3`
File type	`PE (x32)`
Compiler	`Visual C/C++, Microsoft Visual Studio (2015)` `(PyInstaller generated)`
File size	`7,783,528`
File name	`IEProxy.exe, NvStereo.exe`

The Spy module was also found to be created using PyInstaller and presents itself as a python module, the main payload can be found in the file called `mon.py`. This module carries the following espionage capabilities: it can create screenshots, steals the clipboard content and acts as a keylogger.

This module exhibits the following routines:

- **shot_screen** - creates screenshots, compresses them and saves them with files names in the format "`%TMP%\ChromeJob\YYYY_MM_DD_HH_MM_SS_MMM`", where YYYY year, MM month, DD day, HH hours, MM minutes, SS seconds, MMM microseconds (only first three digits).
- **get_current_process** - collects information about the current process (PID, module name, window title, adds timestamp in a format shown above), compresses it and stores it in file with names in the format "`%TMP%\ChromeJob\tmpMMDD`" (where MM - current month and DD - current day)
- **KeyStroke** - the main routine that is called each time when any key is pressed, it uses two previous routines, collects information about pressed keys, grabs clipboard content each time when user is trying to paste that data, compresses and stores collected data in files with names like "`%TMP%\ChromeJob\tmpMMDD`" (except screenshots, format of their filenames is described above).

Like the previous Searcher module (MD5 `4aebc2c53aaf777ec83c5aa14a43318f`), this module also does not have any functionality for data exfiltration.

Downloader

MD5	b874476b29ad120b8b5498f2a74ec14a
SHA1	8a07a1e74cce79e818d41565638088643d75ff54
SHA256	ca4ecea5cae3ff7ab2f2b807f6a9d5029e3ab0782e6a01718de4a4e26b5c4f2b
Link time	`Tue Sep 04 17:42:13 2018 UTC+3`
File type	`PE (x32)`
Compiler	`Visual C/C++, Microsoft Visual Studio (2015)` `(PyInstaller generated)`
File size	4 793 748 bytes
File name	`winevents.exe`

After unpacking and decrypting this module, the original Python application can be found in a file named `remote_runs.py`, which has two routines:

- **`decodedata`** - retrieves encoded and encrypted implant from a Github repository;
- **`getdata`** - routine to decrypt and decode implant previously retrieved by `decodedata` routine.

This module installs the executable as a service named `EventLogs` on a victim system and executes it by calling the `servicemanager start` method. Once launched, it downloads the payload from the Github resource at "`https://github[.]com/TheclaMcentire/test/blob/main/demo`" and runs it in memory. The payload is placed between two groups of brackets (6 brackets in each), encoded with base64, XOR'ed with 1 byte and packed with zlib[1]. This repository is still active at the time of writing:

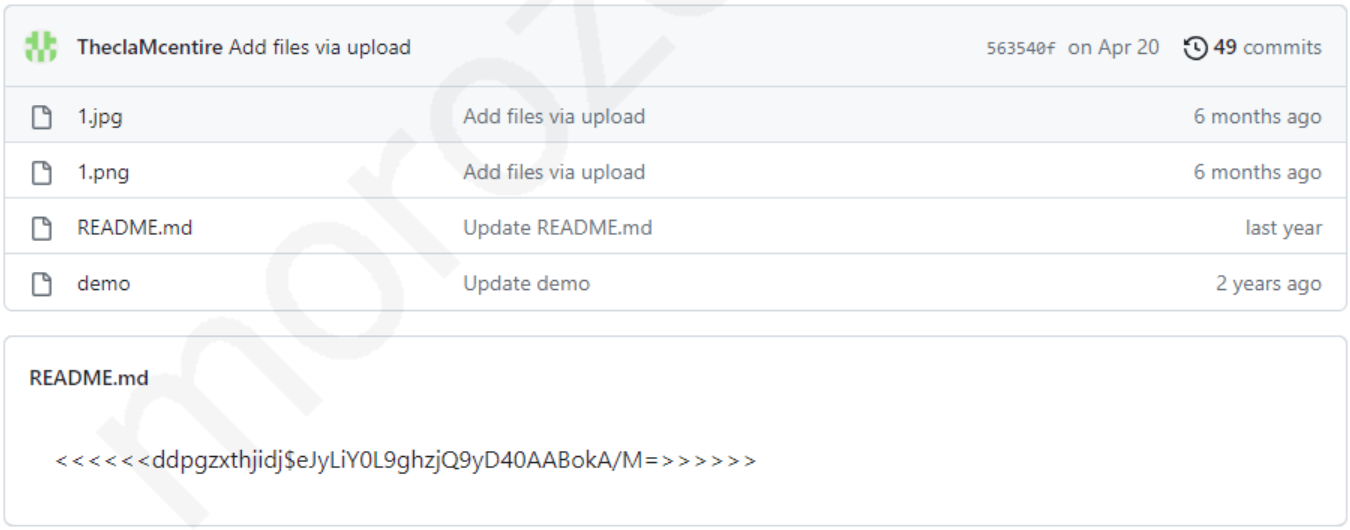

Fig. 1 The Github repository containing the malicious payload

[1] https://www.zlib.net

Main malware implant

MD5	4712c8b27f90d99a28b3535f0dc5c0bf
SHA1	97f38c8d41942996b442000f22cab28486e75ab3
SHA256	7a585bd5a8122a831b095730b81e92a411aca97de98ff09efa38370798d66497
File type	Plain text
File size	7147 bytes
File name	demo

This is the main payload, it is base64-encoded and XORed with one byte (0x3F). After decryption, the file presents itself as a python script:

MD5	a86cab2ad9919597b1805aba011587d4
SHA1	a75242536f11f84c4c0a25c7ebf82b71e89f4bed
SHA256	f6208fec88155c8dc621f5cf230fe75e42163cbc7e10387e76871dd9d80ab773
File type	Python script
File size	24271 bytes

This payload has the following configuration parameters:

N	Parameter name and its value	Description
1	iplist = [r'https://github[.]com/TheclaMcentire/test/blob/main/README.md']	URL where the list with addresses of C2 is located.
2	LOGONPAGE="/logon.aspx"	Path on the C2 used to upload data with fingerprint info and to receive commands for the implant.
3	USERPAGE="/errorFE.aspx"	Not used.
4	COMPAGE="/logoff.aspx"	Not used.
5	RETPAGE="/signout.aspx"	Path on the C2 used to upload results of command executions to CnC.
6	RESPAGE="/frowny.aspx"	Not used.
7	POSTFORM='username'	String used as a key in the 'key:value' structure to be sent to C2.

The C2 is able to control the implant via commands and receive results of their execution, but also to download and run both binaries and python script and to receive fingerprinting information for reconnaissance. The communication protocol behind these functions is rather basic and based on trivial POST requests, which are transmitted and received compressed via zlib and encoded with base64.

The implant sends each request with the following hardcoded headers:

```
headers = {'Accept': 'text/html, application/xhtml+xml, image/jxr, */*',
           'Accept - Encoding':'gzip, deflate',
           'Accept-Language':'en-US,en;q=0.5',
           'Connection':'Keep-Alive',
           'Host':'google.com',
           'User-Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/92.0.2743.116 Safari/537.36 Edge/15.15063'
           }
```

Additionally, it adds one more header - 'password', it contains the string representation of the python routine's output of `time.time()`. This routine returns the time in seconds since the epoch (January 1, 1970, 00:00:00 (UTC)) as a floating point number.

Based on the parameters listed above, we can conclude that the available C2 addresses are stored in the file located at "`https://github[.]com/TheclaMcentire/test/blob/main/README.md`". The following string represents the content of that file at the time of research:

```
<<<<<<ddpgzxthjidj$eJyLiY0L9ghzjQ9yD40AABokA/M=>>>>>>
```

The ciphertext is placed between two groups of characters: '<<<<<<' and '>>>>>>'. The decryption procedure for this ciphertext text is trivial and can be found in the payload source code:

```
def d64zlib(data):
    return zlib.decompress(base64.b64decode(data))

def c2decode(key, data):
    try:
        global d64zlib,log
        data = d64zlib(data)
        ret = ''
        for i in range(0,len(data)):
            ret += chr(ord(key[i]) ^ ord(data[i]))
        return ret
    except Exception as e:
        log('c2decode:'+str(e))
```

The ciphertext mentioned above consists of two parts with the dollar character '$' used as a delimiter between them. The first part is the key for the 'c2decode' routine (this key is used to decrypt the C2 addresses), while the second part is the encrypted C2 address. The implant supports the following commands:

Command	Description
UPLINE='1'	Unused.
CMDMES="2"	Execute command with cmd.exe via pipes (remote shell).
DOWNLOADANDEXEC="3"	Download and run.
UPLOADFILE='7'	Download a file.
DLANDPY="4"	Run python script.
CHANGETIME='5'	Change time between C2 requests.
DOWNFILE='0'	Upload a file.
GETTASK = '8'	Get task list from a victim machine.
KILLTASK = '9'	Run taskkill on a victim machine (via PID).
PMINIT = '10'	Run socks5 server or establish a connection to specified IP address or port number.
PMAP = '11'	Close all connections or send data, depending on parameters.

Additional projects

We inspected the GitHub repository in which the implant is located and discovered other projects that are related to the implant described above with a medium level of confidence. As an example, they share the same IP address encoding like the implant's C2 IP address with all resulting IP addresses within the same ASN.

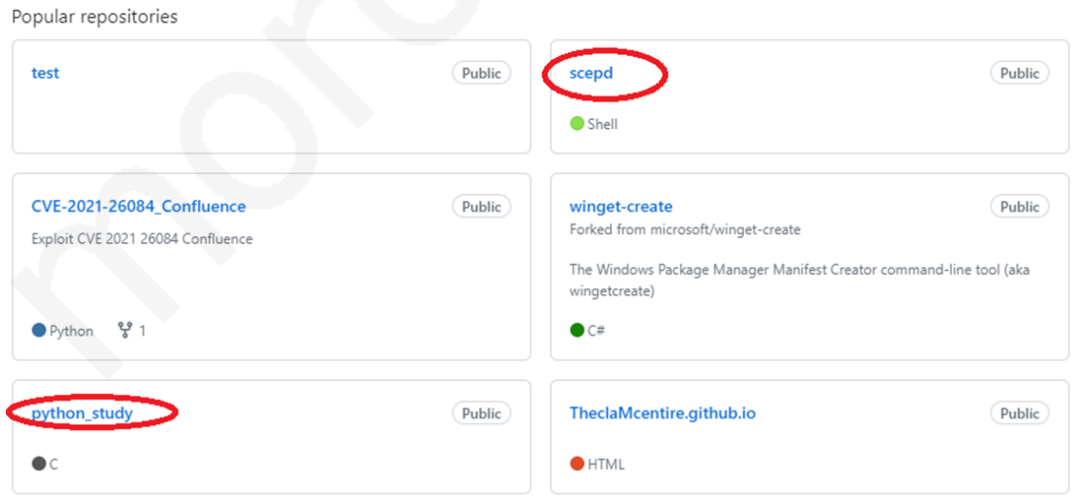

Fig. 2 Other projects at that repository (circled interesting ones)

We think that two of these other projects could be interesting due to their similarities in style and that they could be related to the campaign described in this report, although with a medium level of confidence.

scepd

The first such project is called scepd. Here is the content of that project at the time of analysis:

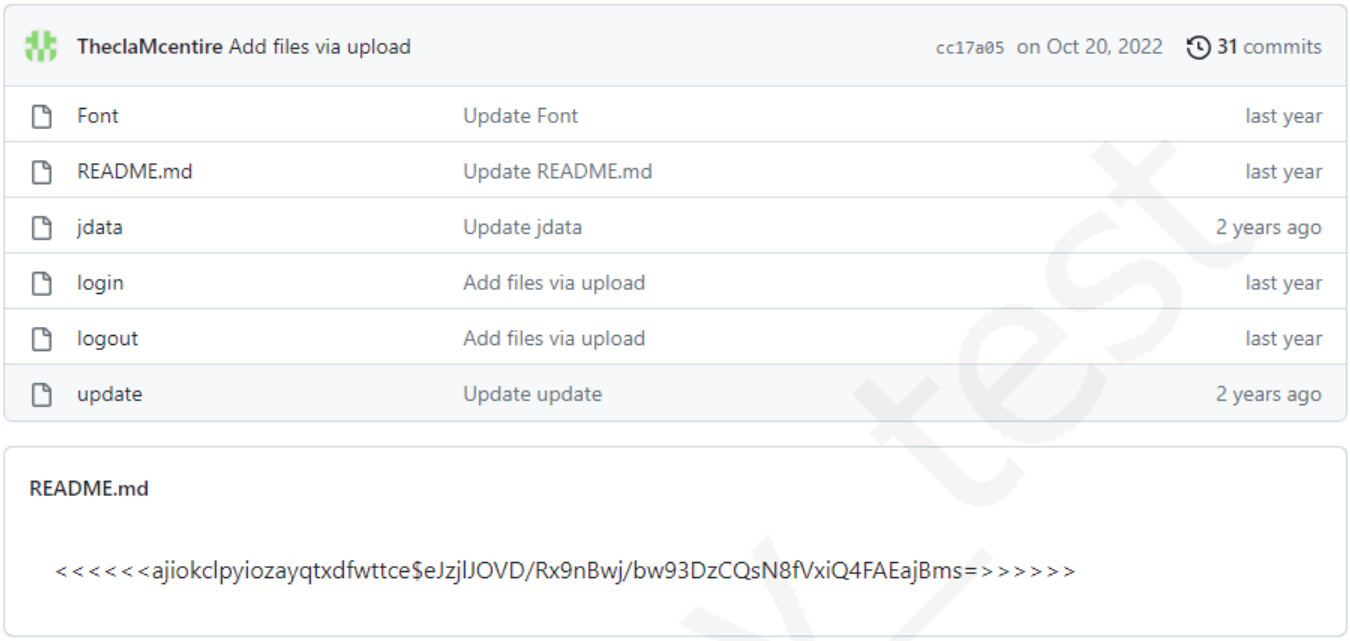

Fig. 3 The scepd project content

We applied the same decryption routine that was used to decrypt the implant's C2 address, described in the "Malware implant" section, and were able to decrypt the `README.md` file.

File name	File size	File md5	URL
`README.md`	82 bytes	`6d356514b438c1069ad014e89adadf75`	`hxxps://github[.]com/TheclaMcentire/scepd/blob/main/README.md`

After decrypting and decoding the string, the IP address is: `hxxp://89[.]42.178.12:8080`. The same IP address was used as the implant C2, but the port number is additionally specified in this case. Two other files of this project also contain encrypted IP addresses:

File name	`Font`
File md5	`6d356514b438c1069ad014e89adadf75`
URL	`https://github[.]com/TheclaMcentire/scepd/blob/main/Font`
Decrypted content	**`hxxp://89[.]42.178.12:8080`**
Analysis date and time	`18th Oct 2023, 15:52 UTC`

File name	Login
File md5	f538dd3be1aa73470e83202bdc29eb87
URL	https://github[.]com/TheclaMcentire/scepd/blob/main/Login
Decrypted content	**hxxp://89[.]42.178.13:8080**
Analysis date and time	18th Oct 2023, 15:59 UTC

We also found 3 other files related to this project, which resemble the encrypted implant. It might indicate with medium level of confidence that they were encrypted with the same algorithm like the implant and therefore could be related to the PyViper campaign. These files are named `jdata`, `logout` and `update`. These samples look like they are encrypted, but here we do not have the encryption keys.

python_study

The next project from the same Github repository is called python_study:

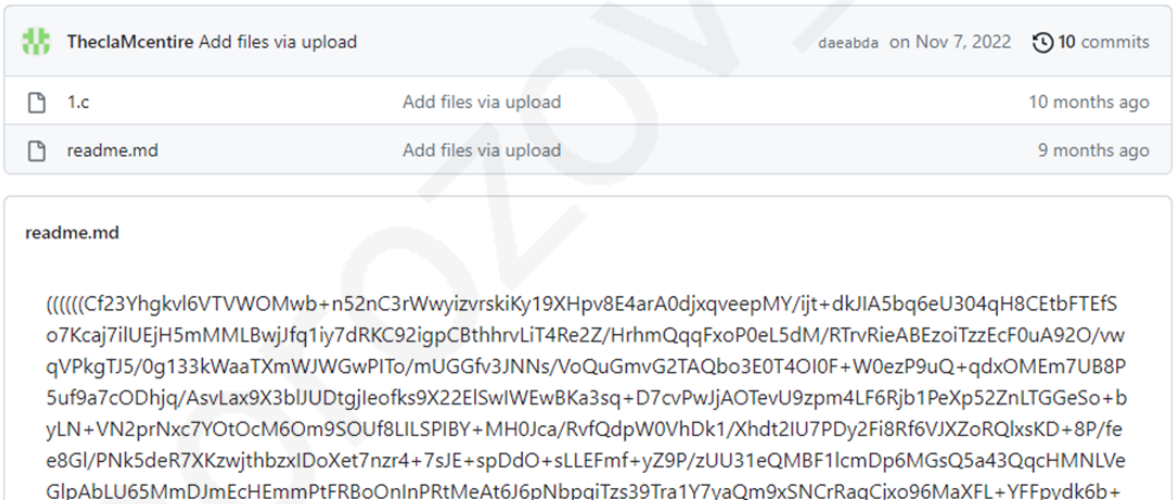

Fig. 4 The python_study project content

File name	File size	File md5	URL
README.md	3760 bytes	0b4ff18339dd6e69e929bf6caa429bbd	hxxps://github[.]com/TheclaMcentire/python_study/blob/main/readme.md

The file `readme.md` looks like the encrypted implant, so it may be related to the PyViper campaign. We were not able to decrypt the content because of the missing key.

One other file from this repository is named `1.c`:

File	File size	File md5	URL
1.c	87 bytes	ab696fa547cbd629e3bd0600cd24731f	hxxps://github[.]com/TheclaMcentire/python_study/blob/main/1.c

The content of the file `1.c` shows the following string:

```
--------vdqwydcseqhgwcogjdhykemc...eJyTE2Bld/b28Y6JjwmNDIqIdwmNdg4ODQ0GAEjIBrU=________
```

We tried to apply the decryption scheme that is used by the implant, however with another delimiter - '...' and the removal of 8 characters from the beginning and end of the line, and were able to decrypt it. The usage of the same encryption scheme means that this project is related to the implant with medium level of confidence. The decrypted string shows another web address, again with an IP address incremented by 1: hxxp://89[.]42.178.13:8080.

Infrastructure

This APT campaign stores its main payload in a GitHub repository. Also, it uses one IP address as its C2 server. We also discovered a second IP address only differing in the last digit, encoded with the same encoding scheme. This leads us to assume with medium level of confidence that this IP address (89[.]42.178.13) is also related to the PyViper campaign.

Domain	IP	First seen	ASN
-	89[.]42.178.12	-	50757
-	89[.]42.178.13	-	50757

Victims

At the moment only three victims are known to us: two victims are from Russia (hospitality and IT industry) and one located in China. With a medium level of confidence, we suppose that the real victim is only one of these entities, because one of the samples contain a hardcoded filename that suggests that it was developed for the specific target.

Attribution

We do not attribute this APT to any known APT actor at this point.

Conclusions

The usage of publicly available resources to store payloads and C2 addresses is not a new approach, but traffic involving such resources looks benign and an attacker has more chances to keep malicious payloads available over longer periods of time compared to dedicated C2 infrastructure to spread malware.

The malware found in this APT contains some debug data, such as commented local IP addresses, and even inaccuracies in its command handling. These aspects give us reason to believe that the developer of this APT had mediocre experience in developing malicious software at best. However, it must be recognized that, often, even technically simple APTs are able to achieve the goal, which in this case is believed to be cyberespionage.

Appendix I – Indicators of Compromise

Note: The indicators in this section are valid at the time of publication. Any future changes will be directly updated in the corresponding .ioc file.

Downloader
b874476b29ad120b8b5498f2a74ec14a winevents.exe

Unpacked and decompiled downloader
e53b7fd69fb42dfd2d62e890e299ba0c remote_runs.py

Searcher
4aebc2c53aaf777ec83c5aa14a43318f GoogleUpdate.exe

Unpacked and decompiled searcher
db26b97e4336c353fe009696cf3bced2 search.py

Spy module
aa0e6fc2495b89ff0800465703f1d548 IEProxy.exe, NvStereo.exe

Unpacked and decompiled spy module
88c2c28670e86164e6dadfacaef11f42 mon.py

Second Stage (encoded and encrypted)
4712c8b27f90d99a28b3535f0dc5c0bf demo

Second Stage (decoded and decrypted)
a86cab2ad9919597b1805aba011587d4 -

File paths
c:\windows\temp\log.txt
c:\windows\temp\ietmp.tmp
c:\Users\Public\Downloads\res.txt
c:\Windows\WinEvents.exe
C:\Windows\GoogleUpdate.exe
c:\Program Files\internet explorer\ieproxy.exe
c:\Program Files\nvidia corporation\3d vision\nvStereo.exe
c:\nvidia corporation\3d vision\nvStereo.exe

Domains and IPs
89[.]42.178.12 PyViper C2
89[.]42.178.13 PyViper C2

Github repository, legitimate resource:
hxxps://github[.]com/TheclaMcentire/

Appendix II – MITRE ATT&CK Mapping

This table contains all the TTPs identified in the analysis of the activity described in this report.

Execution	**T1059.006**	**Command and Scripting Interpreter: Python** Uses Python to execute all its modules, as well as for downloading and executing the main implant body.
	T1569.002	**System Services: Service Execution** Installs the EXE that holds the downloader module as a service called EventLogs to achieve persistence.
Persistence	**T1543.003**	**Create or Modify System Process: Windows Service** Uses Windows services to run the downloader module at computer startup.
Defense Evasion	**T1140**	**Deobfuscate/Decode Files or Information** Uses base64 to encode exfiltrated data and to decode the main implant, implant CnC addresses and commands received from CnCs.
	T1036.004	**Masquerading: Masquerade Task or Service** Creates the service to achieve persistence of the downloader module named EventLogs, this service name looks like legitimate one.
	T1036.005	**Masquerading: Match Legitimate Name or Location** Adversaries named the components of this APT as GoogleUpdate.exe, IEProxy.exe, NVStereo.exe and winevents.exe. These names look like legitimate ones.
	T1027.002	**Obfuscated Files or Information: Software Packing** Uses zlib to pack exfiltrated data and to unpack the main implant, implant CnC addresses and commands received from CnCs.
Credential Access	**T1056.001**	**Input Capture: Keylogging** The Spy module can log keystrokes from the victim's machine.

Discovery	**T1083**	**File and Directory Discovery** The Enumerator module can enumerate the directory listing to find files with specific hardcoded names.
	T1057	**Process Discovery** The implant has the capability to obtain a listing of running processes by tasklist executing.
	T1082	**System Information Discovery** The implant can collect information about victim's machine (Local time, hostname, host ip, information about OS version, etc) and send it to the CnC.
	T1039	**Data from Network Shared Drive** The Enumerator module can enumerate the directory listing (included on network shared drives, if mapped) to find files with specific hardcoded names.
	T1074.001	**Data Staged: Local Data Staging** The Spy module can collect sensitive information (keylogger logs, screenshots, grabbed clipboard data) and stores them locally prior to exfiltration.
Collection	**T1560.002**	**Archive Collected Data: Archive via Library** The implant and the spy module can compress data with ZLIB prior to sending it back to the C2 server.
	T1119	**Automated Collection** The implant can automatically collect data about the compromised system . The spy module can automatically collect keystrokes, and screen images before exfiltration.
	T1115	**Clipboard Data** The Spy module can steal data from the victim's clipboard.
	T1056.001	**Input Capture: Keylogging** The Spy module can perform keylogging.
	T1113	**Screen Capture** The Spy module can capture screenshots on compromised hosts.

Command and Control	**T1071.001**	**Application Layer Protocol: Web Protocols** The implant uses HTTPS for communication with command and control servers.
	T1132.001	**Data Encoding: Standard Encoding** The implant has used base64 to encode command and control traffic.
	T1573.001	**Encrypted Channel: Symmetric Cryptography** Uses XOR-based encoding to decode the CnCs addresses obtained from the github repository.
	T1102.001	**Web Service: Dead Drop Resolver** This APT uses GitHub repository to store main implant body as well as CnCs addresses.
Exfiltration	**T1041**	**Exfiltration Over C2 Channel** The implant exfiltrates gathered data over C2 channels.