Understanding SalatStealer: Features and Impact

blog.dexpose.io/understanding-salatstealer-features-and-impact/

M4lcode March 15, 2025

Introduction

Salat Stealer is a stealthy malware developed in the Go programming language, designed to infiltrate systems and extract sensitive data. Once it infects a device, it gathers extensive system information, such as hard drive details, screen resolution, running processes, and active windows. One of its most alarming features is its ability to live-stream the victim's desktop and capture audio and video using the device's microphone and camera, creating serious privacy concerns. Furthermore, Salat Stealer is capable of exfiltrating files from the compromised machine. Its presence can result in significant risks, including identity theft, financial loss, and severe breaches of privacy.

Capabilities and Functionality

Data Theft and Credential Harvesting
• Gathers stored credentials from web browsers (e.g., Chrome, Firefox, Edge).
• Extracts login credentials from local email clients.
• Accesses cryptocurrency wallet files to steal private keys or funds.
• Searches for unsecured credentials stored in plaintext files.
• Exfiltrates files from the compromised system, potentially leading to severe privacy breaches, financial loss, and identity theft.

Live Desktop Monitoring
• Possesses live-streaming capabilities, allowing attackers to monitor the victim's desktop activity in real time.
• Can record audio and video through the device's microphone and camera, posing serious privacy risks.

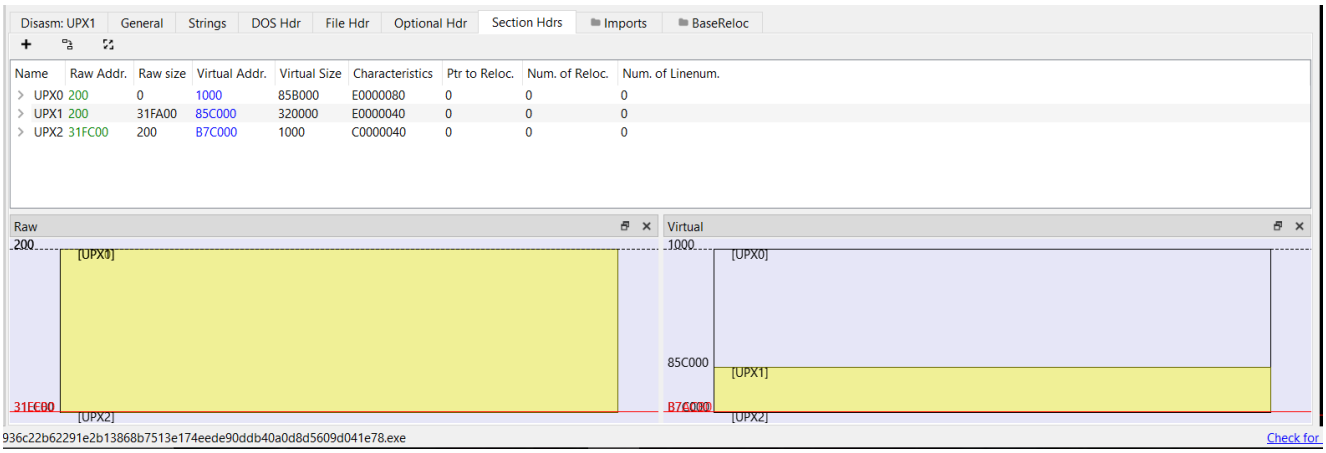
Persistence and Evasion Techniques
• Writes files to critical system directories (Windows, System32, Drivers, Program Files)
• Modifies Windows Registry (Run key) to ensure automatic execution at startup.
• Uses User Account Control (UAC) bypass techniques to escalate privileges.
• Executes dropped payloads to extend its attack chain.
• Employs UPX packing to obfuscate its code and evade signature-based detection.

Attack Chain Overview

1. **Initial Infection:**
    - Distributed through phishing emails, malicious attachments, and drive-by downloads.
    - Can also be delivered via cracked software or trojanized applications.
2. **Execution and Persistence:**
    - The malware executes upon user interaction, such as opening a malicious file.
    - It abuses the Windows Registry (Run key) to achieve persistence.
    - Attempts to bypass User Account Control (UAC) for elevated privileges.
3. **System Reconnaissance & Data Collection:**
    - Enumerates running processes and active windows.
    - Scans for stored credentials in web browsers, email clients, and cryptocurrency wallets.
    - Checks system language settings to potentially avoid infecting specific regions.
4. **Data Exfiltration:**
    - Extracted credentials and system details are sent to an attacker-controlled command-and-control (C2) server.
    - The malware may attempt to remove traces of its activity to avoid detection.

UPX Packing

All known Salat Stealer samples discovered so far have been packed using the UPX packer.



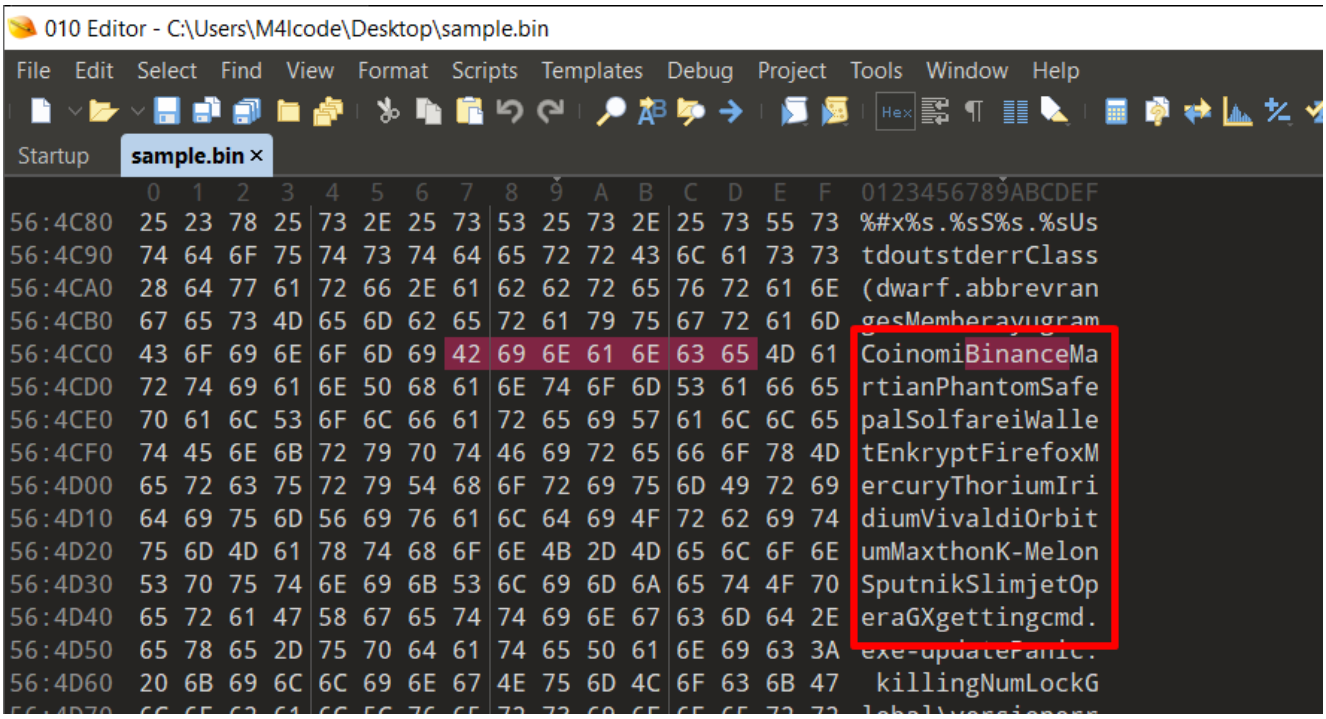
To unpack it just easily use upx -d file_name

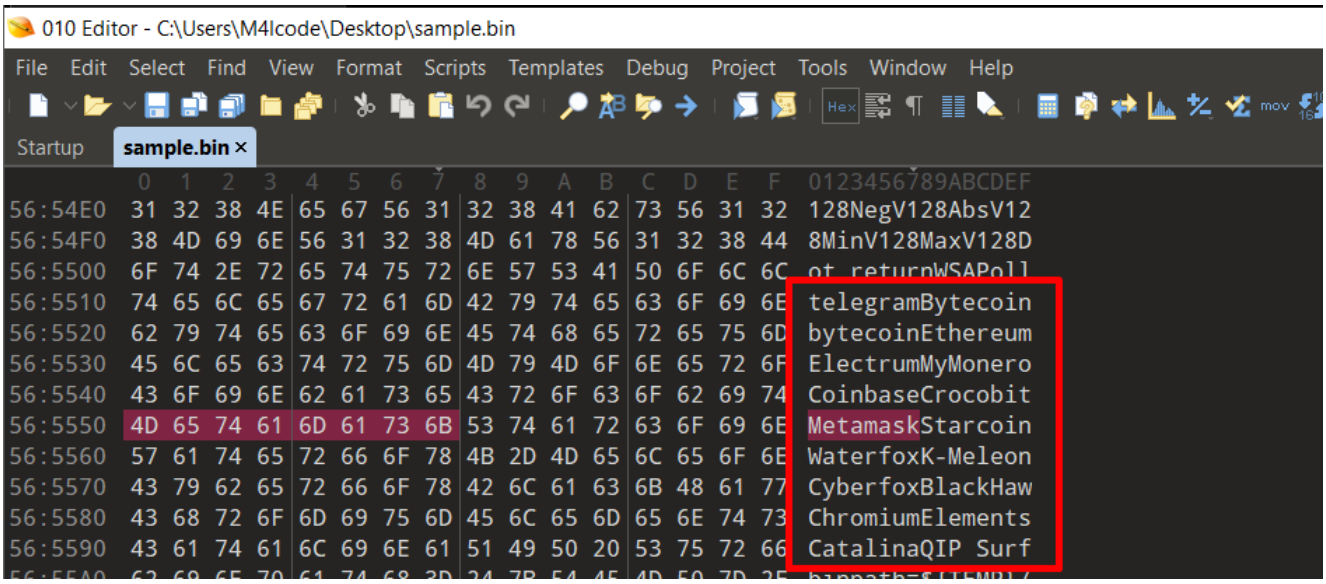
```
C:\Users\M4lcode>upx -d C:\Users\M4lcode\Desktop\SalatStealer.exe
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2025
UPX 5.0.0       Markus Oberhumer, Laszlo Molnar & John Reiser   Feb 20th 2025

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
  11596800 <-   3276288   28.25%    win32/pe     SalatStealer.exe

Unpacked 1 file.
```

During static analysis of the **Salat Stealer** binary, multiple hardcoded strings were found, indicating targeted applications, including cryptocurrency wallets and web browsers.

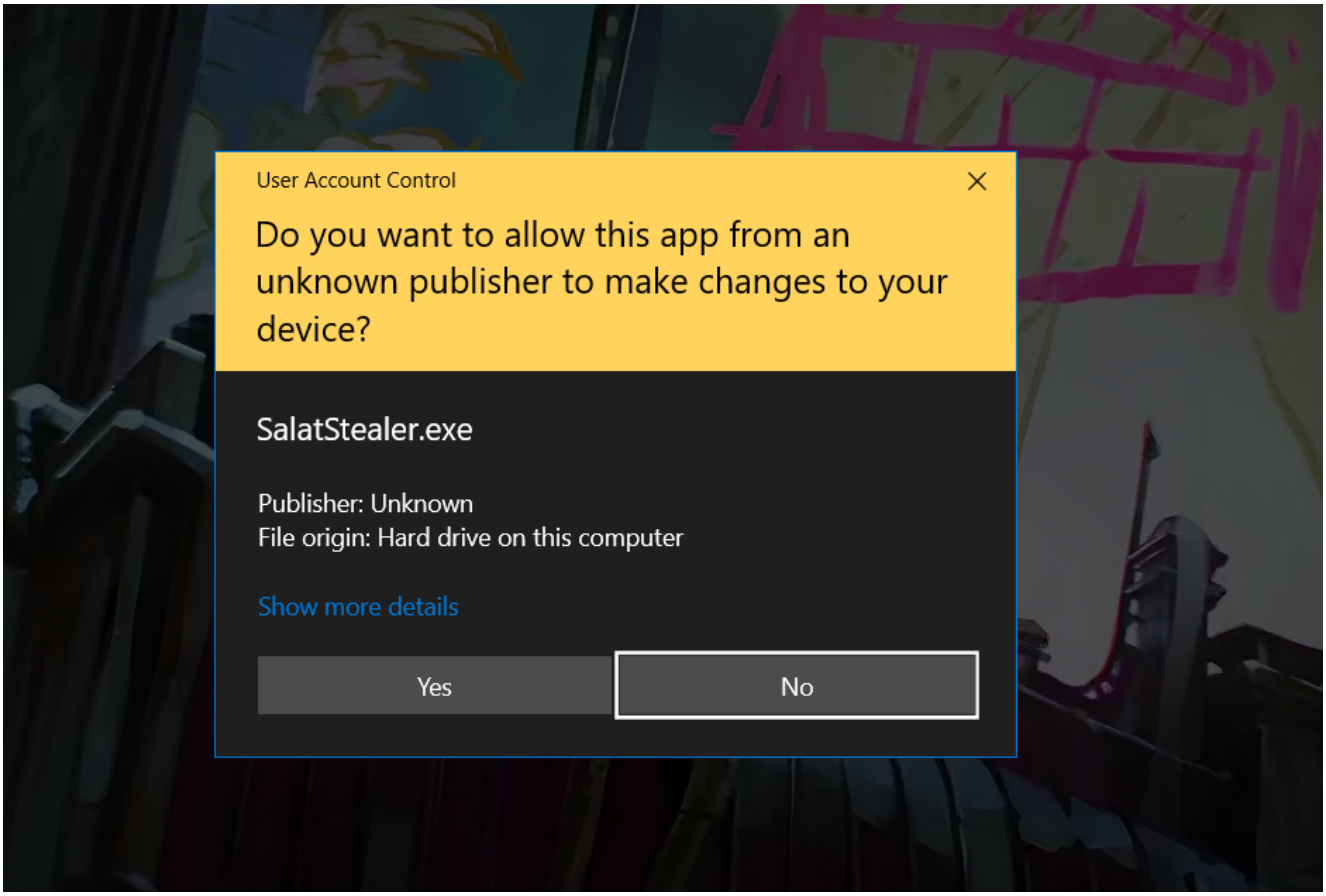




User Account Control (UAC) Bypass

SalatStealer employs **User Account Control (UAC) bypass techniques** to gain elevated privileges without alerting the user. By setting **EnableLUA** key in \REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\EnableLUA to “0” (Disables UAC).

By setting **EnableLUA** to 0, the malware disables UAC enforcement, allowing it to execute administrative commands and maintain persistence while bypassing security restrictions. However, this change **does not take effect immediately**—a system reboot is required for UAC to be fully disabled. Until the system restarts, the UAC prompt will still appear when executing the malware.



Persistence Mechanism of SalatStealer

SalatStealer establishes persistence by copying itself in random directories, in my case it copied it self in:

- C:\Program Files (x86)\Windows Defender\
- C:\Program Files (x86)\Windows NT\

It then creates **Run keys** in the Windows Registry to ensure execution at startup:

\REGISTRY\USER\<USER_SID>\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\smss = “C:\\Program Files (x86)\\Windows Defender\\smss.exe”

\REGISTRY\USER\<USER_SID>\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\dllhost = “C:\\Program Files (x86)\\Windows NT\\dllhost.exe”

Targeted Data by SalatStealer

SalatStealer retrieves hardware identifier (HWID) for the system.

It calls **golang_org_x_sys_windows_registry_OpenKey()** with: HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography as argument

```
77   v29 = v18;
78   v19 = encoding_hex_Decode(v34, v12, v16, v34, v12, v16);
79   if ( v19 > v29 )
80     runtime_panicSliceAcap();
81   v30 = v19;
82   v13 = golang_org_x_sys_windows_registry_OpenKey(-2147483646, &aAbiNewnameTagT[10750], 31, 1);// SOFTWARE\Microsoft\Cryptography
83   v32 = v17;
84   v31 = v13;
85   v14 = main_dec(v34, v30, v29, 1);
86   v5 = v14;
87   dword_EC785C = v17;
.rdata:0097A4D4                 db  en out of rangeGODEBUG: unknown cpu feature  subtle.XORBytes: dst
.rdata:0097A4D5                 db ' too shortreflect: Out of non-func type MapIter.Key called before'
.rdata:0097A516                 db ' Nexttrailing garbage after addressmissing validateFirstLine func'
.rdata:0097A557                 db 'mime: duplicate parameter nameApplication error %#x (%s): %ssqlit'
.rdata:0097A598                 db 'e3: abort due to ROLLBACKsqlite3: another row availablesqlite3_in'
.rdata:0097A5D9                 db 'voke_busy_handler_gotransform: short source bufferfailed to read '
.rdata:0097A61A                 db 'misc opcode: %v%s requires data count sectioninvalid immediate va'
.rdata:0097A65B                 db 'lue for %sminimum size mismatch: %d > %dmaximum size mismatch: %d'
.rdata:0097A69C                 db ' < %doperationKindAtomicRMW8Cmpxchgerror decoding export kind: %w'
.rdata:0097A6DD                 db 'unit length has reserved valueLocal\Sputnik\Sputnik\User DataSOFT'
.rdata:0097A71E                 db 'WARE\Microsoft\Cryptographyhttps://1.1.1.1/dns-query?name=failed '
.rdata:0097A75F                 db 'to enable privileges: %vfailed to set proxy blanket: %vbad certif'
.rdata:0097A7A0                 db 'icate status responsetls: unsupported public key: %TTLS_RSA_WITH_'
.rdata:0097A7E1                 db 'AES_128_CBC_SHA256TLS_RSA_WITH_AES_128_GCM_SHA256TLS_RSA_WITH_AES'
.rdata:0097A822                 db '_256_GCM_SHA384TLS: sequence number wraparoundCLIENT_HANDSHAKE_TR'
.rdata:0097A863                 db 'AFFIC_SECRETSERVER_HANDSHAKE_TRAFFIC_SECRETtls: failed to sign ha'
.rdata:0097A8A4                 db 'ndshake: json: invalid number literal %qin literal true (expectin'
.rdata:0097A8E5                 db 'g ',27h,'r',27h,')in literal true (expecting ',27h,'u',27h,')in l'
.rdata:0097A90F                 db 'iteral true (expecting ',27h,'e',27h,')in literal null (expecting'
```

This is where **MachineGuid** is stored, which is commonly used for HWID tracking.

Then it calls **golang_org_x_sys_windows_registry_Key_GetStringValue** to retrieve **MachineGuid** value

```
109  }
110  v33[0] = main_GetHWID_deferwrap1;
111  v33[1] = v6;
112  v35 = v33;
113  StringValue = golang_org_x_sys_windows_registry_Key_GetStringValue(v6, "MachineGuidUser Name: ", 11);
114  if ( v19 )
115  {
116    v15 = golang_org_x_sys_windows_registry_OpenKey(-2147483646, &aAbiNewnameTagT[10750], 31, 257);
```

Capturing Active Window

`main_getActiveWin` is responsible for capturing the title of the currently active window to track user activity. It first checks if the system is idle using `main_IsIdle()`, likely to avoid detection or log only meaningful interactions. If not idle, it retrieves the active window handle via `main_getForegroundWindow()` and extracts the window title using `main_GetWindowText()`.

```
IDA View-A   Pseudocode-A   Hex View-1   Structures   Enums
15      runtime_morestack_noctxt();
16    LOBYTE(v4) = main_IsIdle(-647710720, 69);
17    if ( v4 )
18    {
19      v1 = runtime_concatstring2(0, 0, 0, &aErmssse3avx2bm[1486], 6);
20      v0 = v5;
21    }
22    else
23    {
24      v0 = 0;
25      v1 = 0;
26    }
27    v7 = v1;
28    v6 = v0;
29    main_getForegroundWindow();
30    if ( v2 )
31    {
32      WindowText = main_GetWindowText(v2);
33      runtime_concatstring2(0, v7, v6, WindowText, v4);
34    }
35    main_errorHandler(v2);
36  }
0047D12E main.getActiveWin:36 (87DD2E)
```

Targeted Web Browsers

Salat Stealer aims to extract stored credentials, cookies, and session data from the following browsers:

```
Chrome
Firefox
Mercury
Thorium
Iridium
Vivaldi
Orbitum
Maxthon
K-Meleon
Sputnik
Slimjet
Opera GX
SeaMonkey
IceDragon (Comodo)
Pale Moon
DCBrowser
Waterfox
BlackHawk
Cyberfox
Chedot
Kometa
Fenrir
Coowon
Liebao
Dragon (Comodo Dragon)
CocCoc
Yandex Browser
Chrome SxS
360Browser
UR Browser
```

It extracts the following data:

- Cookies
- Saved Login Credentials
- Authentication Tokens
- Extensions

```
IDA View-A    Pseudocode-B    Pseudocode-A    Hex View-1
39  else
40  {
41    v27[1] = a2;
42    v27[0] = a1;
43    v27[3] = 15;
44    v27[2] = "Network\\Cookies";
45    v13 = path_filepath_join(v27, 2, 2);
46    LOBYTE(v22) = a9;
47    main_getChromeCookies(a3, a4, a5, a6, a7, a8, v13, v17, 0, v22);
48    v26[1] = a2;
49    v26[0] = a1;
50    v26[3] = 10;
  else
  {
    v25[1] = a2;
    v25[0] = a1;
    v25[3] = 10;
    v25[2] = "Login Data";
    v15 = path_filepath_join(v25, 2, 2);
    main_getChromeLogins(a3, a4, a5, a6, a7, a8, v15, v18);
    v24[1] = a2;
    v24[0] = a1;
    v24[3] = 8;
    v24[2] = "Web DatapostOpentaskkill/config/SoftwareAccountspostopenCurveID(finishedexporterGoString01234
    v16 = path_filepath_join(v24, 2, 2);
    LOBYTE(v23) = a9;
    main_getChromeToken(a3, a4, a5, a6, a7, a8, v16, v19, 0, v23);
    main_errorHandler(v11);
  }
```

Targeted Cryptocurrency Wallets

Salat Stealer specifically targets multiple crypto wallets to steal private keys including:

```
Metamask
TonKeeper
SuiWallet
Coinomi
Binance Wallet
Martian Wallet
Phantom Wallet
SafePal Wallet
Solfare Wallet
Enkrypt Wallet
Exodus Wallet
Guarda Wallet
Bitapp Wallet
Coin98 Wallet
Fewcha Wallet
Finnie Wallet
Iconex Wallet
Kaikas Wallet
Oxygen Wallet
Pontem Wallet
Saturn Wallet
Sollet Wallet
Wombat Wallet
Starcoin Wallet
Electrum Wallet
MyMonero Wallet
Crocobit Wallet
PaliWallet
ExodusWeb3
Armory
XinPay
XMR.PT
Atomic Wallet
Jaxx Wallet
```

Targeted Messaging Applications

```
Telegram Desktop
Kotatogram
```

Screen and Live Desktop Monitoring

Salat Stealer can continuously capture screenshots and even stream the victim's desktop live to the attacker's command-and-control (C2) server, providing real-time visibility into their activities.

Clipboard Data Theft

Salat actively monitors clipboard activity, allowing it to intercept copied text, including passwords, cryptocurrency addresses, and other sensitive data.

Keylogging – Keystroke Interception

Salat Stealer records everything typed on the victim's system, capturing credentials, messages, and any other input in real-time.

Audio and Video Espionage

Salat Stealer functions as a full-fledged spyware tool, capable of:

Microphone Recording: Capturing and transmitting audio from the victim's microphone.
Webcam Access: Recording video from the system's webcam.
Live Streaming: Broadcasting real-time audio and video feeds to the attacker's remote server.

Exfiltration

Salat Stealer employs a stealthy exfiltration process to transmit stolen data to its C2 server, using AES encryption to obfuscate sensitive information before transmission. Files are bundled into ZIP archives

Conclusion

SalatStealer is a stealthy and persistent malware designed to steal sensitive data while evading detection. By harvesting credentials, exfiltrating files, and enabling real-time surveillance, it poses severe risks to victims, including financial loss, identity theft, and privacy breaches. Its use of UAC bypass and registry modifications, making removal difficult. Additionally, its encryption mechanisms and UPX packing further complicate analysis and detection. Given its broad targeting of web browsers, cryptocurrency wallets, and messaging applications, SalatStealer remains a significant threat, emphasizing the need for continuous monitoring and advanced security measures to counter its impact.

IOCs

```
IP: 104[.]21[.]84[.]111
Hash:
e2797eec2f82e9f93bed5c70adacecab791441199814ae333c45c7bf1c70ab6b
cfdf1d2768ed773c3f5b2c2a03d7892551ea79b181068c23a765f1e09a8c90b1
52a1750dc75795faa2bfdd3405ee027ee7a4fe78928027a5439372312479ca33
57d78bce936c22b62291e2b13868b7513e174eede90ddb40a0d8d5609d041e78
e2c1f8f1db1d2c47bbe60e2d4daf5422865639bcafca1933c9f807e353d98e5b
Registry:
\REGISTRY\USER\USER_SID\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\smss = "C:\\Prograı
\REGISTRY\USER\USER_SID\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\dllhost = "C:\\Pro
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\EnableLUA =
```

Yara Rule

```
rule Detect_SalatStealer_MEMORY
{
    meta:
        author = "Dexpose Team"
        date = "2025-03-12"
        description = "Detects SalatStealer in Memory"
        version = "1.1"
        sharing = "TLP:CLEAR"
        hash1 = "e2797eec2f82e9f93bed5c70adacecab791441199814ae333c45c7bf1c70ab6b"
        hash2 = "cfdf1d2768ed773c3f5b2c2a03d7892551ea79b181068c23a765f1e09a8c90b1"
        hash3 = "52a1750dc75795faa2bfdd3405ee027ee7a4fe78928027a5439372312479ca33"
        hash4 = "57d78bce936c22b62291e2b13868b7513e174eede90ddb40a0d8d5609d041e78"
        hash5 = "e2c1f8f1db1d2c47bbe60e2d4daf5422865639bcafca1933c9f807e353d98e5b"
    strings:
        $hex1 = {
            89 41 4C 89 0C 24 E8 ?? ?? ?? ?? 8B 44 24 04 8B 4C 24 08 85 C0 74 ??
            C7 44 24 20 00 00 00 00 C7 44 24 24 00 00 00 00 C7 44 24 28 00 00 00 00
            C7 44 24 2C 00 00 00 00 8D 15 ?? ?? ?? ?? 89 54 24 20 8D 15 ?? ?? ?? ??
            89 54 24 24 74 ?? 8B 40 04
        }
                // mov     [ecx+4Ch], eax
                // mov     [esp+3Ch+var_3C], ecx ; _ptr_exec_Cmd
                // call    os_exec__ptr_Cmd_Start
                // mov     eax, [esp+3Ch+var_38]
                // mov     ecx, dword ptr [esp+3Ch+var_34]
                // test    eax, eax
                // jz      short loc_XXXXXX
                // mov     [esp+3Ch+var_1C], 0
                // mov     [esp+3Ch+var_18], 0
                // mov     [esp+3Ch+var_14], 0
                // mov     [esp+3Ch+var_10], 0
                // lea     edx, RTYPE_string
                // mov     [esp+3Ch+var_1C], edx
                // lea     edx, "An error occurred"
                // mov     [esp+3Ch+var_18], edx
                // jz      short loc_XXXXXX
                // mov     eax, [eax+4]

                $hex2 = {
                E8 ?? 7? 00 00 8B 44 24 28 8B 4C 24 24 8B 54 24 30 89 94 24 ?? 03 00 00
                3C 89 9C 24 ?4 02 00 00 8B 6C 24 ?? 89 AC 24 AC 0? 00 00 8B 74 24 ??}
                // call    main_getChrome
                // mov     eax, [esp+718h+var_6F0]
                // mov     ecx, [esp+718h+var_6F4]
                // mov     edx, [esp+718h+var_6E8]
                // mov     [esp+718h+var_3EC], edx
                // mov     ebx, [esp+718h+var_6DC]
                // mov     [esp+718h+var_464], ebx
                // mov     ebp, [esp+718h+var_6E4]
                // mov     [esp+718h+var_56C], ebp
                // Partial wildcard for unknown instruction
                // mov     esi, [esp+718h+var_6D8]
```

```
    condition:
                uint16(0) == 0x5A4D and
        filesize &gt; 10MB and filesize &lt; 12MB and
        all of them
}
```

MITRE ATT&CK Techniques

Tactic	Technique ID	Technique Name
Boot or Logon Autostart Execution	T1547	Registry Run Keys / Startup Folder
Privilege Escalation	T1547.001	Abuse Elevation Control Mechanism
Defense Evasion	T1548	Bypass User Account Control
Defense Evasion	T1548.002	Boot or Logon Autostart Execution
Impair Defenses	T1562	Disable or Modify Tools
Impair Defenses	T1562.001	Modify Registry
Credential Access	T1555	Credentials from Password Stores
Credential Access	T1555.003	Credentials from Web Browsers
Unsecured Credentials	T1552	Credentials In Files
Unsecured Credentials	T1552.001	Credentials In Files
Discovery	T1614	System Location Discovery
Discovery	T1614.001	System Language Discovery
Collection	T1005	Data from Local System