Exposing a Russian Campaign Targeting Ukraine Using New Malware Duo: BadPaw and MeowMeow

March 2026

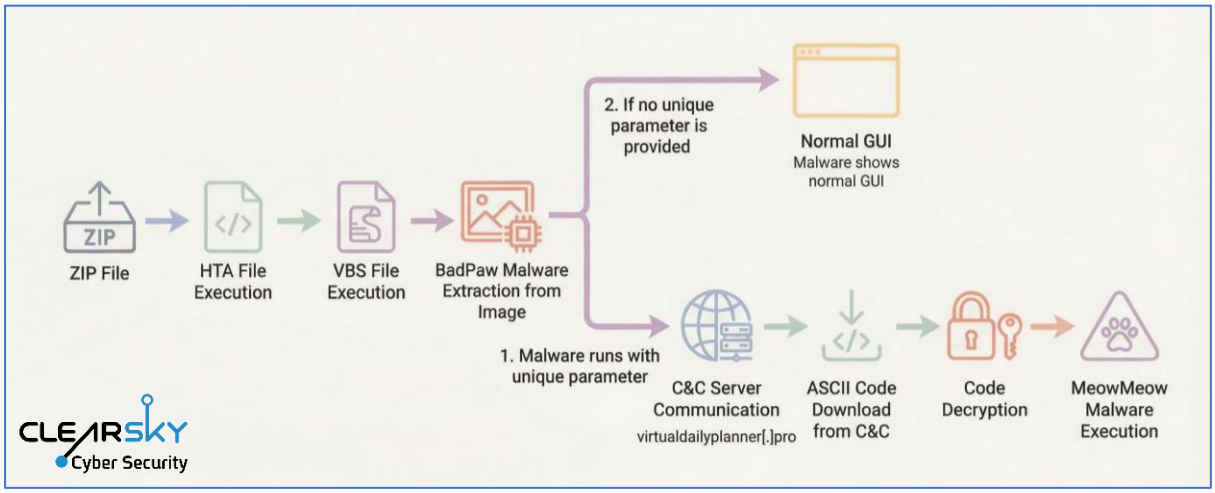
Executive Summary

www.clearskysec.com - info@clearskysec.com - 972-3-6240346

ClearSky Team has identified a targeted Russian cyber campaign against Ukraine utilizing two novel malware strains, BadPaw and MeowMeow.

The attack chain initiates with a phishing email containing a link to a ZIP archive. Once extracted, an initial HTA file displays a lure document written in Ukrainian concerning border crossing appeals to deceive the victim. Simultaneously, the infection triggers the download of BadPaw, a .NET-based loader. Upon establishing command-and-control (C2) communication, the loader deploys MeowMeow, a sophisticated backdoor. To hinder analysis and reverse engineering, both strains are obfuscated using the .NET Reactor packer, signaling a deliberate effort by the threat actors to maintain persistence and evade detection.

To ensure persistence and evade discovery, both malware strains incorporate sophisticated defense mechanisms. The campaign employs strict Parameter Validation; the malicious components remain dormant, running only "dummy" code with a benign GUI, unless executed with specific, predefined parameters. Furthermore, the MeowMeow backdoor features advanced environmental awareness. It actively scans for virtual machines and common analysis tools such as Wireshark, ProcMon, and Fiddler, immediately terminating its execution if a sandbox or researcher environment is detected.



Attack Chain as part of the Russian campaign utilizing new malware

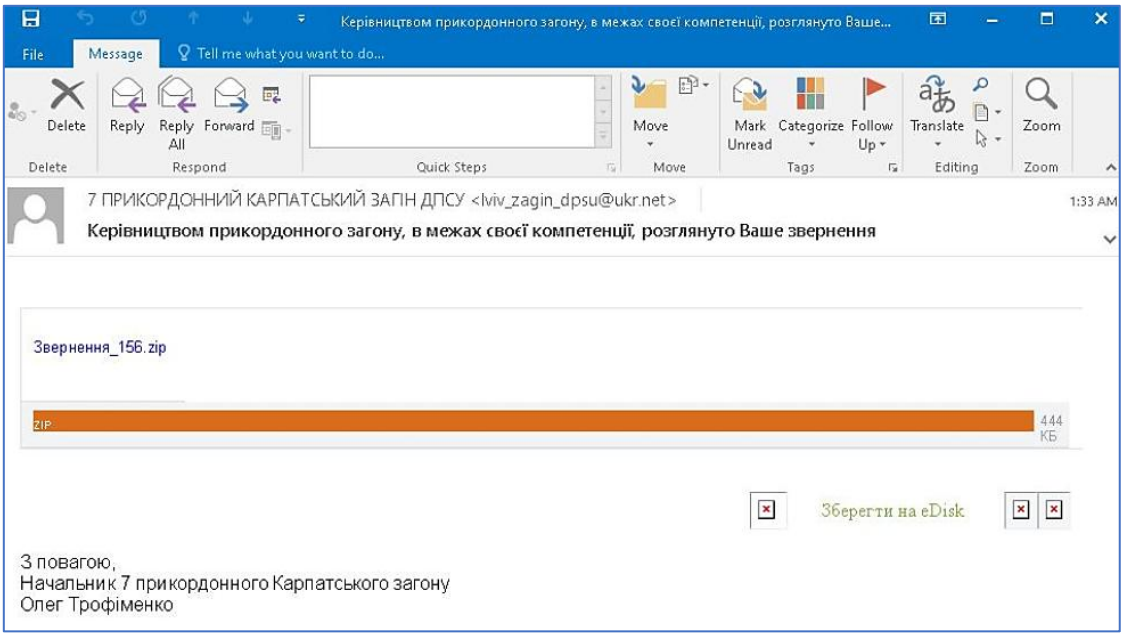
ClearSky attributes this campaign with high confidence to a Russian state-aligned threat actor and with low confidence to the specific group APT28 (Fancy Bear). This assessment is based on a three-pronged analysis:

- Targeting & Victimology: The focus on Ukrainian entities, combined with the geopolitical nature of the lure, aligns with Russian strategic objectives.
- Linguistic Artifacts: The presence of Russian-language strings within the code suggests a development environment native to the region.

- Tactical Overlap: The multi-stage infection chain, the use of .NET-based loaders, and the specific obfuscation techniques mirror established tradecraft observed in previous Russian cyber operations.

Research Details

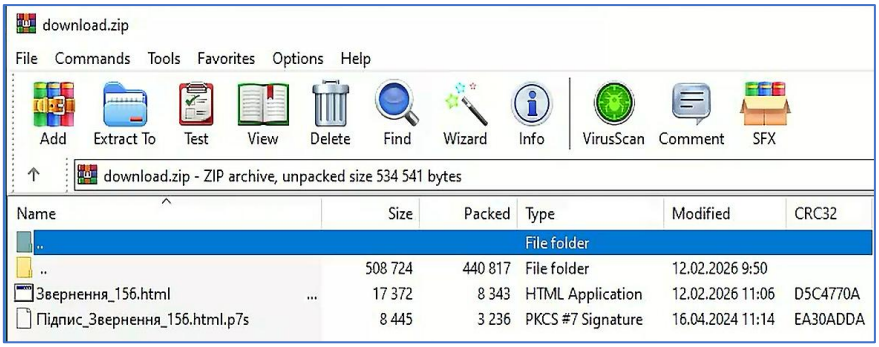
An email file sent from **lviv_zagin_dpsu[@]ukr[.]net** was recently identified following an upload from Ukraine to a public database. While the file is not currently flagged as malicious by defense systems[1], it leverages **ukr[.]net**, a popular Ukrainian email provider. This service has been frequently used by the Russian threat actor APT28 across multiple campaigns to establish legitimacy and secure the trust of targeted victims.



When a user clicks the link purportedly containing the ZIP file, the following two actions are triggered:

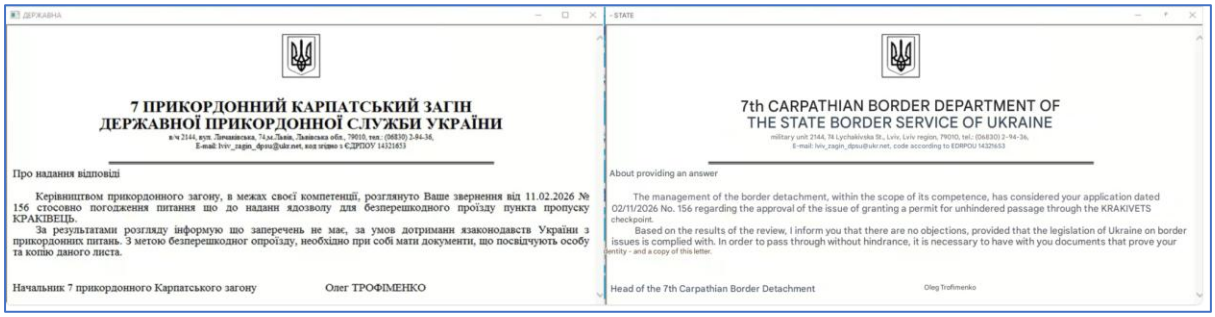
1. **The user is first redirected to the following URL**:
   https[://]infotrackerstatistic[.]live/open?token=2c5ef1a0fed545aaaf803964bb4da2e3
   This URL hosts an exceptionally small image, or tracking pixel. We assess that when this image is loaded, the attacker receives a notification that the link has been clicked. The domain **infotrackerstatistic[.]live** appears to have been specifically registered by the threat actor for this purpose.
2. **Simultaneously, the user is redirected to a secondary link**:
   https[://]cutt[.]ly/0tnHM0nh
   Navigating to this shortened URL initiates the final download of the ZIP archive.

[1] Sha256: 160e40a763dfb518dc6929c2d7838d3f9eafab09eab1e8d0b00c69f6b73d681b



The retrieved ZIP archive contains a file with an .html extension; however, this is a masquerade. Technical analysis reveals the file is actually an HTA (HTML Application).

Upon execution, the HTA file drops and opens a decoy document to distract the user and reduce suspicion. The decoy contains the following text:



Left: Original Ukrainian text | Right: English translation

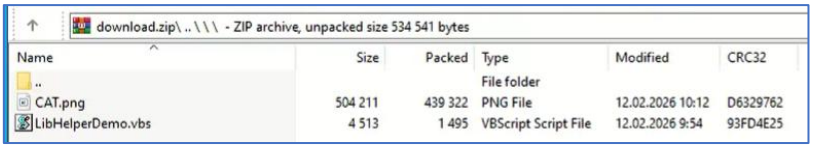
The dropped decoy document serves as a social engineering tactic, presenting a confirmation of receipt for a government appeal regarding a Ukrainian border crossing. This lure is intended to maintain the veneer of legitimacy while the HTA file executes its secondary stages in the background.

To evade detection and identify potential sandbox environments, the HTA file performs an environmental check by inspecting the following Registry key:

HKLM|SOFTWARE|Microsoft|Windows NT|CurrentVersion|InstallDate

By querying this value, the malware calculates the "age" of the operating system. If the system was installed less than ten days prior to execution, the malware terminates. This is a common anti-analysis technique used to avoid execution on freshly provisioned virtual machines or automated analysis sandboxes.

Once the environmental requirements are met, the malware initiates a recursive search for the initial ZIP archive titled "**Звернення**_156" (the original download). Upon locating the archive, it extracts the following secondary components:



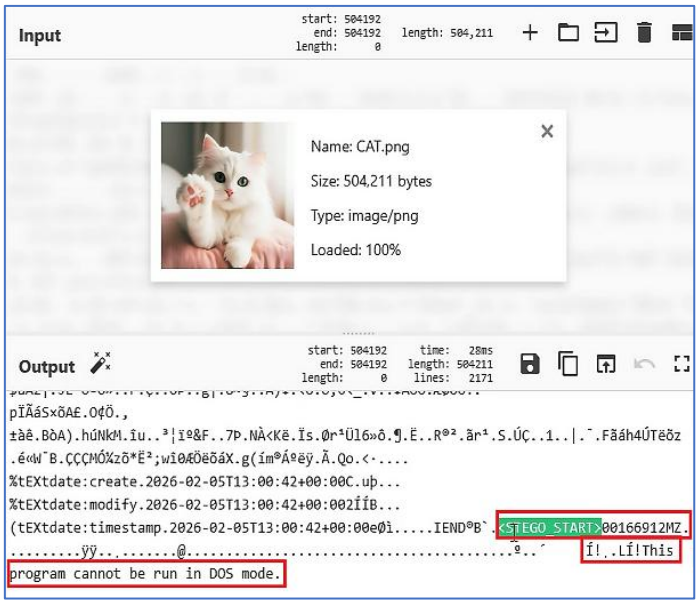
The extracted files are saved under different names to the following path:

86319 ms	936	mshta.exe	C:\Users\admin\AppData\Local\UpdateHelpers\HelperLogo.png	492 Kb	image
86319 ms	936	mshta.exe	C:\Users\admin\AppData\Local\UpdateHelpers\UpdateHelper.vbs	4 Kb	text

To ensure persistence on the infected system, a scheduled task is created to execute the VBS file. Upon execution, LibHelperDemo[.]vbs accesses the accompanying image file to extract hidden data, a technique known as steganography.

The script parses the image to locate a specific marker, <STEGO_START>, which is immediately followed by the embedded file size (represented in hex as 00166912, or 166,912 bytes). Following this marker, the header of a portable executable (PE) file becomes visible. This extracted data is then saved as a standalone file.

The image file and its embedded technical data are shown below:

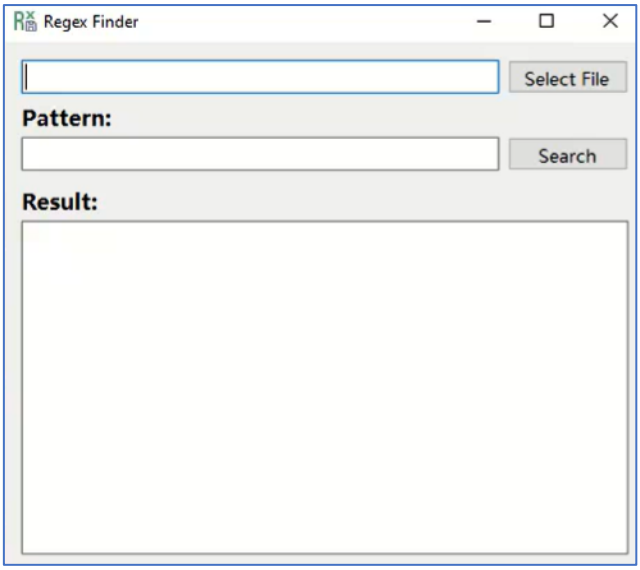


The downloaded file has the following hash:

Sha256: 6cad470e10c09151b5d337a082a088cfe25d697ef295e02759e1e68e8b3bbbcb

At the time of analysis, the file is recognized as malicious by only nine antivirus engines. We have named this malware "BadPaw". Its primary objective is to establish communication with a Command and Control (C2) server to download additional malicious components.

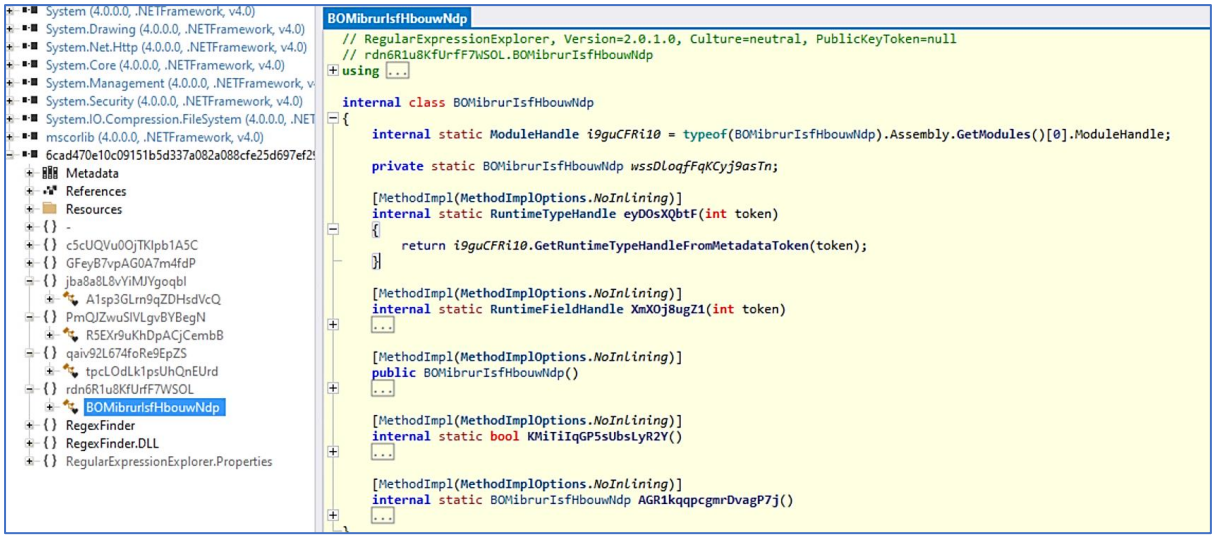
BadPaw incorporates several defensive and evasion capabilities. The first layer of defense is triggered if the file is executed outside of the intended attack chain. In this scenario, the malware executes "fake" code designed to present a legitimate-looking Graphical User Interface (GUI) for a "Regex Finder" tool. This interface allows a user to upload files and search for Regex patterns. By presenting this functional decoy, the malware conceals its embedded malicious logic and prevents the execution of its primary payload.



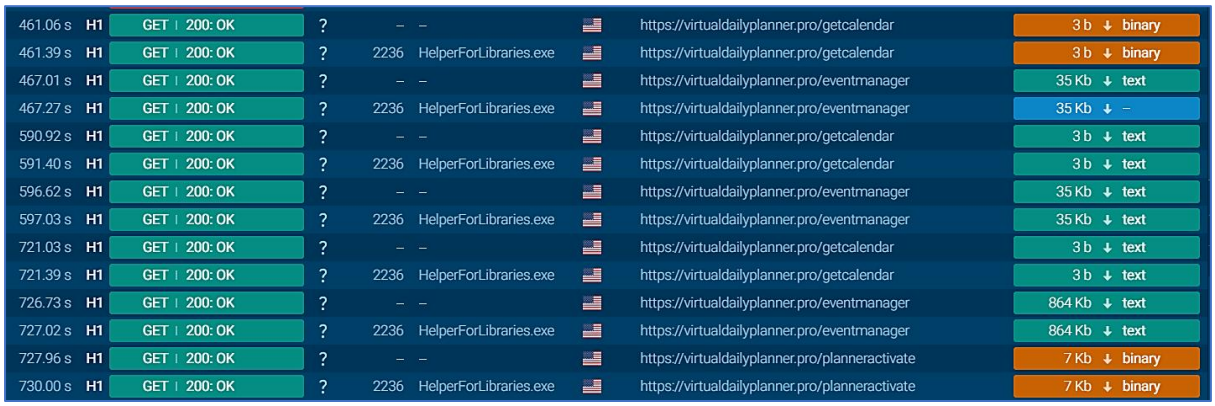
The malicious logic is only activated when the malware is executed using the "-renew" parameter.

```
<Actions Context="Author">
  <Exec>
    <Command>C:\Users\Public\Libraries\HelperForLibraries.exe</Command>
    <Arguments>-renew</Arguments>
    <WorkingDirectory>C:\Users\Public\Libraries</WorkingDirectory>
  </Exec>
 </Actions>
</Task>
```

An additional layer of defense employed by BadPaw is the use of .NET Reactor, a commercial protection and obfuscation tool for .NET assemblies. This packer obfuscates the underlying code to hinder static analysis and reverse engineering. The following image demonstrates the malware's code as it appears under the protection of .NET Reactor:



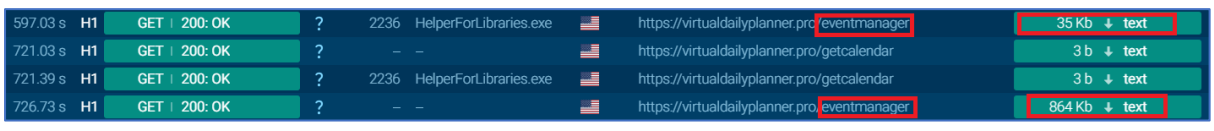
When the executable is run with the required parameter, it establishes communication with the command-and-control (C2) server: virtualdailyplanner[.]pro. This network activity initiates several key actions:



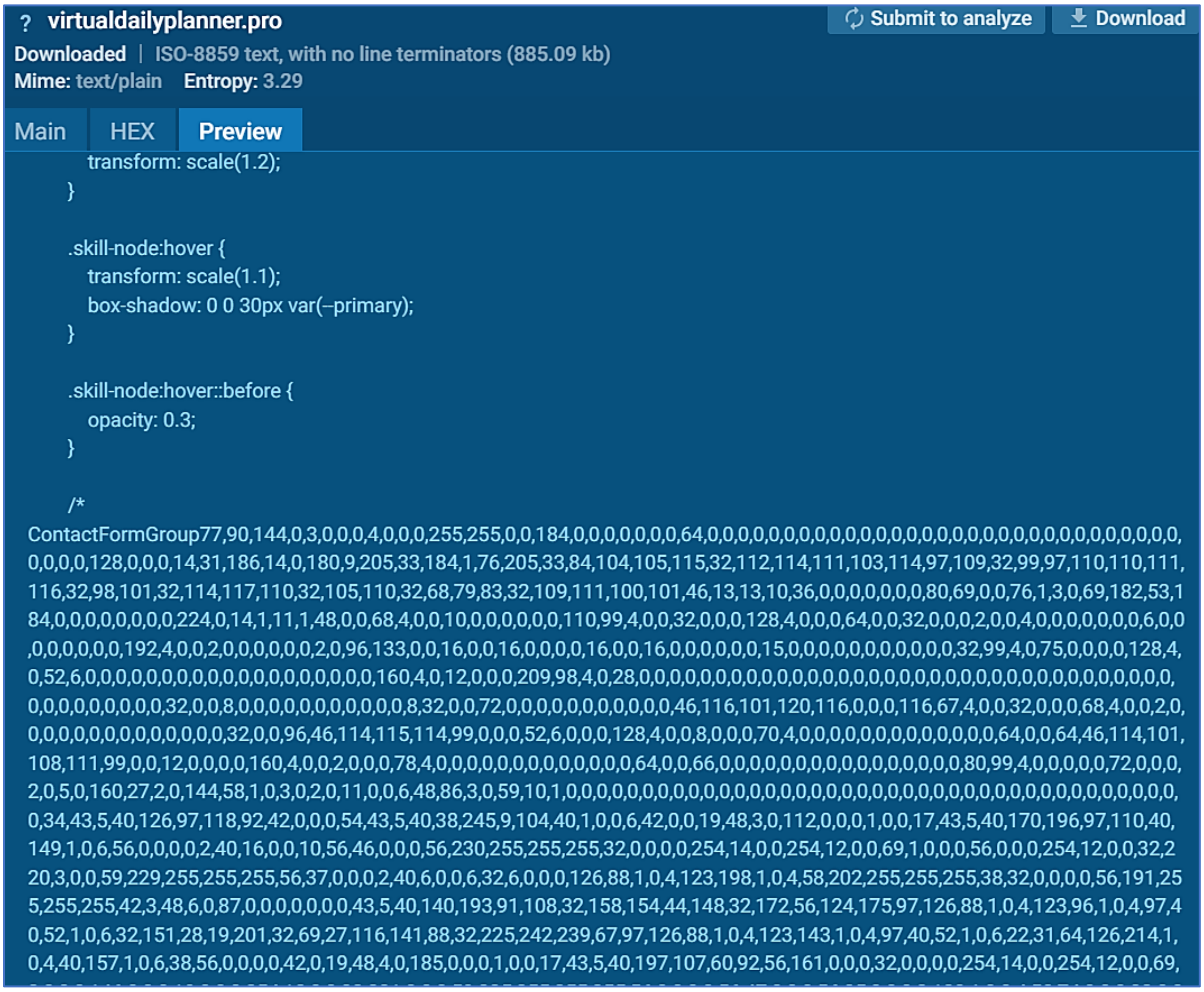
Network requests to the attacker-controlled infrastructure are directed to the following URLs:

- hxxps[://]virtualdailyplanner[.]pro/getcalendar: The server responds with a simple numeric string: "500".
- hxxps[://]virtualdailyplanner[.]pro/eventmanager: The initial response returns a landing page titled: "Telemetry UP!".

Subsequent requests to the /eventmanager endpoint result in the delivery of different data. This behavior indicates a staged or state-aware response mechanism on the C2 server. The image below illustrates the differences between the initial and subsequent data transfers:



The data received in the second stage consists of a standard HTML page containing a block of ASCII code. This payload is encapsulated between the markers /*ContactFormGroup and ContactFormGroup*/.



```
? virtualdailyplanner.pro                                   Submit to analyze   Download
Downloaded | ISO-8859 text, with no line terminators (885.09 kb)
Mime: text/plain   Entropy: 3.29
Main   HEX   Preview
        transform: scale(1.2);
    }

    .skill-node:hover {
        transform: scale(1.1);
        box-shadow: 0 0 30px var(–primary);
    }

    .skill-node:hover::before {
        opacity: 0.3;
    }

    /*
ContactFormGroup77,90,144,0,3,0,0,0,4,0,0,0,255,255,0,0,184,0,0,0,0,0,0,0,64,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,128,0,0,0,14,31,186,14,0,180,9,205,33,184,1,76,205,33,84,104,105,115,32,112,114,111,103,114,97,109,32,99,97,110,110,111,
116,32,98,101,32,114,117,110,32,105,110,32,68,79,83,32,109,111,100,101,46,13,13,10,36,0,0,0,0,0,0,0,80,69,0,0,76,1,3,0,69,182,53,1
84,0,0,0,0,0,0,0,224,0,14,1,11,1,48,0,0,68,4,0,0,10,0,0,0,0,0,0,110,99,4,0,0,32,0,0,0,128,4,0,0,0,64,0,0,32,0,0,0,2,0,0,4,0,0,0,0,0,0,0,6,0,0
,0,0,0,0,0,192,4,0,0,2,0,0,0,0,0,0,2,0,96,133,0,0,16,0,0,16,0,0,0,0,16,0,0,16,0,0,0,0,0,0,15,0,0,0,0,0,0,0,0,0,0,0,32,99,4,0,75,0,0,0,0,128,4,
0,52,6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,160,4,0,12,0,0,0,209,98,4,0,28,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,32,0,0,8,0,0,0,0,0,0,0,0,0,0,0,8,32,0,0,72,0,0,0,0,0,0,0,0,0,0,0,46,116,101,120,116,0,0,0,116,67,4,0,0,32,0,0,0,68,4,0,0,2,0,
0,0,0,0,0,0,0,0,0,0,0,0,32,0,0,96,46,114,115,114,99,0,0,0,52,6,0,0,0,128,4,0,0,8,0,0,0,70,4,0,0,0,0,0,0,0,0,0,0,0,0,0,64,0,0,64,46,114,101,
108,111,99,0,0,12,0,0,0,0,160,4,0,0,2,0,0,0,78,4,0,0,0,0,0,0,0,0,0,0,0,0,0,64,0,0,66,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,80,99,4,0,0,0,0,0,72,0,0,0,
2,0,5,0,160,27,2,0,144,58,1,0,3,0,2,0,11,0,0,6,48,86,3,0,59,10,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,34,43,5,40,126,97,118,92,42,0,0,0,54,43,5,40,38,245,9,104,40,1,0,0,6,42,0,0,19,48,3,0,112,0,0,0,1,0,0,17,43,5,40,170,196,97,110,40,
149,1,0,6,56,0,0,0,0,2,40,16,0,0,10,56,46,0,0,0,56,230,255,255,255,32,0,0,0,0,254,14,0,0,254,12,0,0,69,1,0,0,0,56,0,0,0,254,12,0,0,32,2
20,3,0,0,59,229,255,255,255,56,37,0,0,0,2,40,6,0,0,6,32,6,0,0,0,126,88,1,0,4,123,198,1,0,4,58,202,255,255,255,38,32,0,0,0,0,56,191,25
5,255,255,42,3,48,6,0,87,0,0,0,0,0,0,0,43,5,40,140,193,91,108,32,158,154,44,148,32,172,56,124,175,97,126,88,1,0,4,123,96,1,0,4,97,4
0,52,1,0,6,32,151,28,19,201,32,69,27,116,141,88,32,225,242,239,67,97,126,88,1,0,4,123,143,1,0,4,97,40,52,1,0,6,22,31,64,126,214,1,
0,4,40,157,1,0,6,38,56,0,0,0,0,42,0,19,48,4,0,185,0,0,0,1,0,0,17,43,5,40,197,107,60,92,56,161,0,0,0,32,0,0,0,0,254,14,0,0,254,12,0,0,69,
```

The identified ASCII block is decoded to reconstruct an additional malware component. Following this stage, the malware communicates with the URL:

hxxps[://]virtualdailyplanner[.]pro/planneractivate

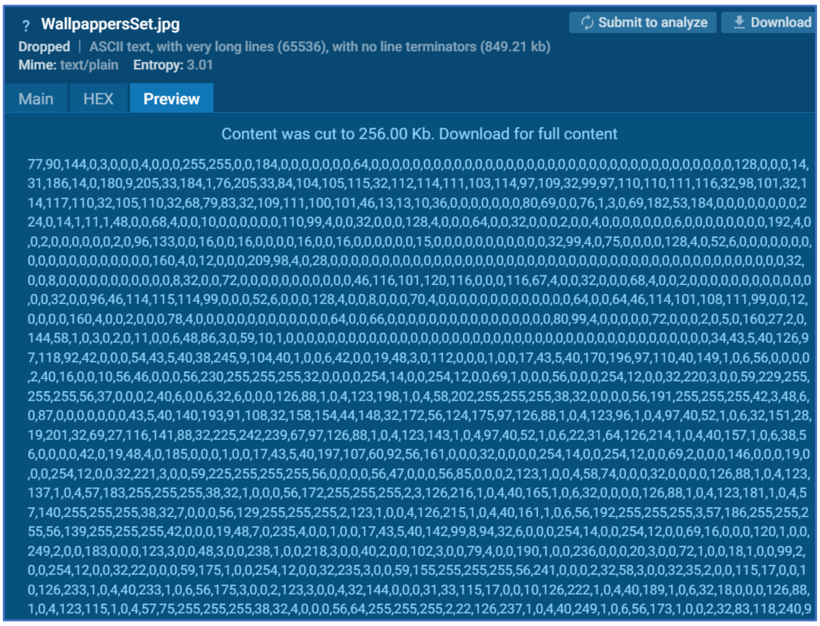
```
 < div class = "row justify-content-center align-items-center" > .. < div class = "col-lg-5 col-12" > .. < button type
= "submit" class = "form-control" > Login < /button>..                                    </div > .... < div class =
"col-lg-5 col-12" > .. < a href = "register.html" class = "btn custom-btn custom-border-btn" > Register <
/a>..                                  </div > .. < /div>....                          </form >
..                            .. < /div>..                         </div > .. < /div>....                <div class="video-
wrap">..                  <video autoplay="" loop="" muted="" class="custom-
video" poster="">..                         <source src="videos/video.mp4" type="video/mp4">....                       Y
our browser does not support the video tag...                        </video > .. < /div>..            </section > .. <
/main>..<!-- AddCssStyle!QVeAiksWIxnlgOhjUqxvaA/pmA31bROyCawP45bXQN7NswH5V32JxZh0UWtMc4pdfM7qCXaU + T+1G7umqIQpzw ==
AddCssStyle!-- > .... < script defer src =
"https://static.cloudflareinsights.com/beacon.min.js/vcd15cbe7772f49c399c6a5babf22c1241717689176015" integrity =
"sha512-ZpsOmlRQV6y907TI0dKBHq9Md29nnaEIPlkf84rnaERnq6zvWvPUqr2ft8M1aS28oN72PdrCzSjY4U6VaAw1EQ==" data - cf - beacon =
'{"version":"2024.11.0","token":"f8f29a13e88b4a0b8fb4e99c3938d87f","r":1,"server_timing":{"name":
{"cfCacheStatus":true,"cfEdge":true,"cfExtPri":true,"cfL4":true,"cfOrigin":true,"cfSpeedBrain":true},"location_startswit
h":null}}' crossorigin = "anonymous" > < /script>.</body > .. <  / html > ..
```

Encrypted information is transferred via the AddCssStyle! parameter. Subsequently, the malware drops the following three files onto the compromised system:

738.78 s	2236	HelperForLibraries.exe	C:\Users\Public\Libraries\config.library-ms	88 b text
751.11 s	2236	HelperForLibraries.exe	C:\Users\admin\Pictures\WallpappersSet.jpg	829 Kb text
799.58 s	2236	HelperForLibraries.exe	C:\Users\admin\AppData\Local\MeowCheck\MeowMeowProgramm.exe	276 Kb executable

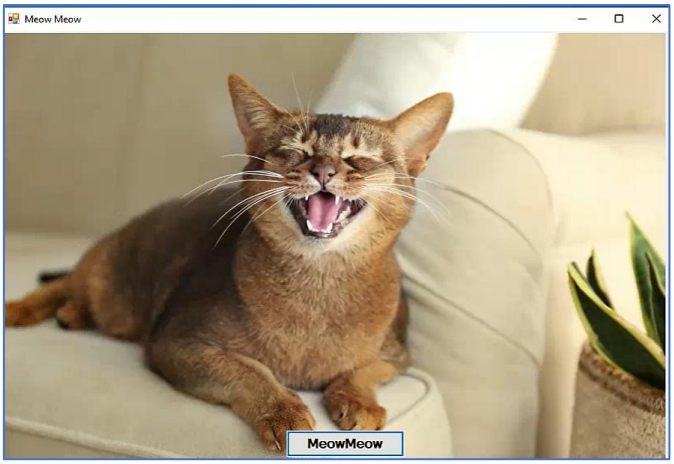
- **Config[.]library-ms**: Contains the configuration data transferred from the server via the /planneractivate endpoint.

- **WallpapperSet[.]jpg**: Contains the raw ASCII code retrieved during the request to the /eventmanager endpoint.



The final component is an additional executable, dropped after the ASCII data is converted back into a standard string. This file, named **MeowMeowProgram[.]exe**, serves as a persistent backdoor.

Consistent with the "BadPaw" tradecraft, if this file is executed independently of the full attack chain, it initiates a dummy code sequence. This decoy execution displays a graphical user interface (GUI) featuring a picture of a cat, aligning with the visual theme of the initial image file from which the primary malware was extracted.



When the "MeowMeow" button within the decoy GUI is clicked, the application simply displays a "Meow Meow Meow" message, performing no further malicious actions. This serves as a secondary functional decoy to mislead manual analysis.

Conversely, the malicious logic is only activated when the malware is executed with a specific parameter, -v, provided by the initial infection chain. The "MeowMeow" backdoor incorporates four distinct layers of protection, mirroring the tradecraft observed in the BadPaw malware:

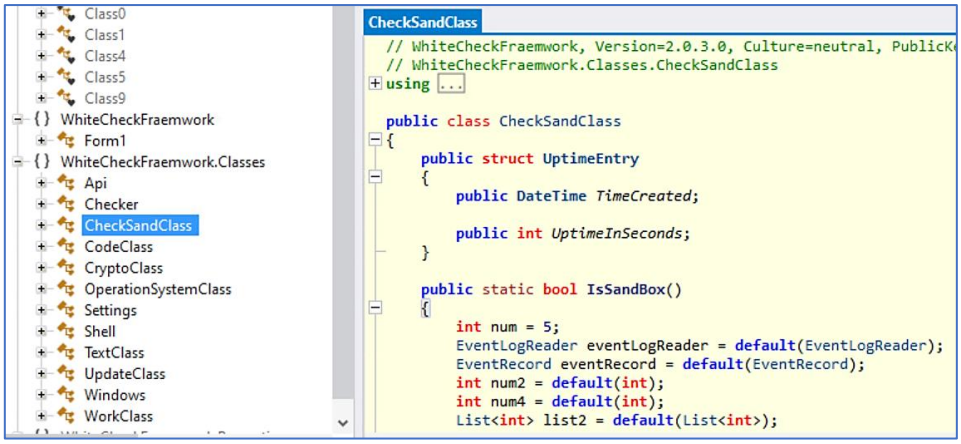
- The requirement of a unique runtime parameter to trigger malicious code.
- The use of .NET Reactor to hinder static analysis.
- Checks to ensure the malware is running on a target system rather than a sandbox.
- Routine monitoring for forensic and monitoring tools running in the background.

The following image, captured after partially stripping the .NET Reactor protection layers, illustrates the specific process checks used to identify analysis tools:

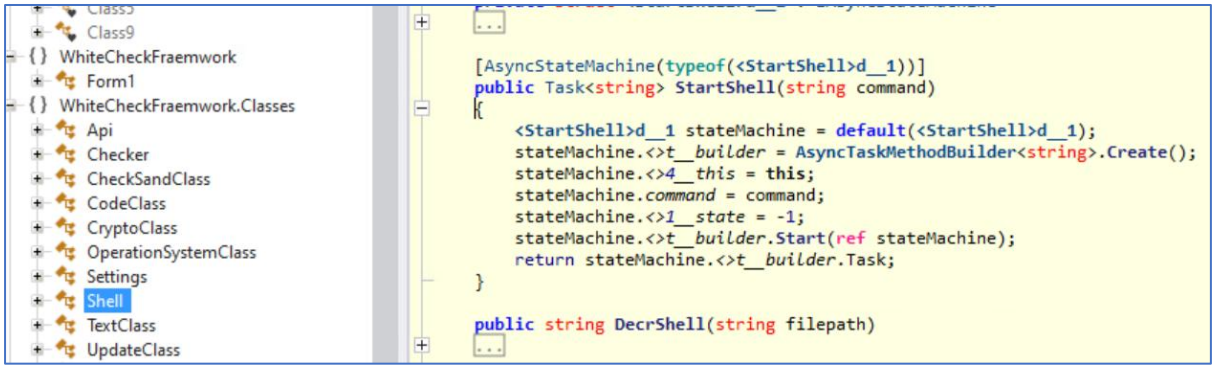
```
private static List<string> smethod_0()
{
    Process[] processes = Process.GetProcesses();
    List<string> list = new List<string>();
    Process[] array = processes;
    foreach (Process process in array)
    {
        list.Add(process.ProcessName.ToLower().Trim());
    }
    return list;
}

private static List<string> smethod_1()
{
    int num = 4;
    while (true)
    {
        List<string> list = new List<string>();
        string[] array = "WiResHaRK TsHARK DumPcAp iDAq oLLydBg proXIFiEr ProCMon NEtmOn GlaSSwire fiDDler cHaRles proCeXp NMap pRtG poSTman".Split(' ');
        while (true)
        {
```

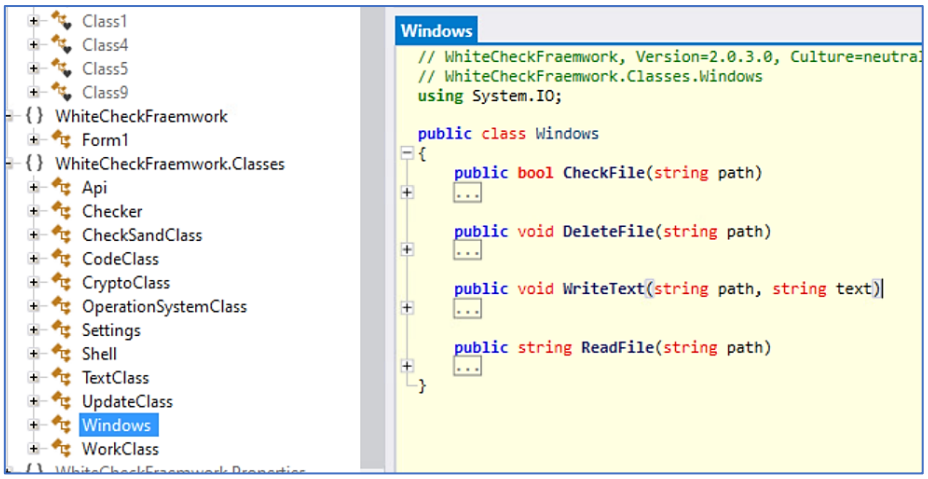
The code reveals checks for well-known forensic and debugging utilities, including **Wireshark**, **Procmon**, **Ollydbg**, and **Fiddler**, among others. Additionally, the malware performs a check to determine if it is executing within a virtualized environment:



The malware features integrated shell capabilities, allowing the threat actor to remotely execute PowerShell commands on the compromised host.



Additionally, the backdoor supports a variety of file system operations. These capabilities include verifying the existence of specific files, as well as the ability to delete, write, and read data from the local storage:



During the analysis of the malware's code, several revealing strings were identified:

```
Match match2 = Regex.Match(string_0, "работоспособного\\sсостояния\\s(\\d+)\\sсек");
if (match2.Success)
{
    return int.Parse(match2.Groups[1].Value);
}
Match match3 = Regex.Match(string_0, "работоспособного\\sсостояния\\s(\\d+)\\sсек");
if (match3.Success)
{
```

Notably, these strings are written in Russian rather than Ukrainian, which supports the assessment of a Russian-based threat actor. When translated, one of the strings reads:
Time to reach working/operational condition: (\d+) seconds
The presence of these Russian-language strings suggests two possibilities: the threat actor committed an operational security (OPSEC) error by failing to localize the code for the Ukrainian target environment, or they inadvertently left Russian development artifacts within the code during the malware's production phase.