

RGB-Team: хактивисты, стоящие за CМoon, и происхождение стилера

SL securelist.ru/rgb-team/115109

Kaspersky

April 6, 2026



Введение

RGB-Team — хактивистская группа, позиционирующая себя как проукраинская и нацеленная на организации в России. Группа была активна вплоть до июля 2024 года и в настоящий момент бездействует. Однако, если не считать публичных высказываний самих злоумышленников, атаки RGB-Team до сих пор были мало описаны.

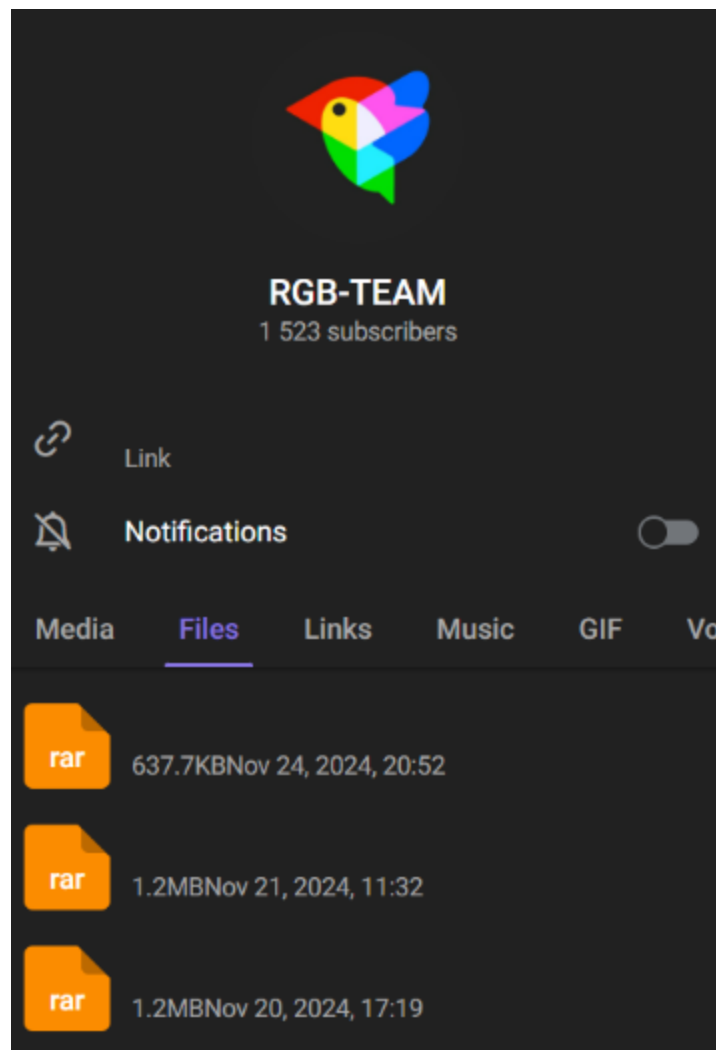
В отличие от многих хактивистских групп, использующих шифровальщики для нанесения ущерба жертвам, RGB-Team атакует с целью извлечь конфиденциальные данные. Например, в последней известной кампании злоумышленники атаковали газовую компанию, украли данные и опубликовали их в своем Telegram-канале. Общий список жертв включал муниципальные образования в некоторых регионах, химическую промышленность и государственно-гражданские службы.

Для проведения атак злоумышленники использовали самописный [червь CМoon](#), который до сих пор никто публично не приписывал какой-либо группе. В этой статье мы проанализируем новые версии червя, докажем его принадлежность RGB-Team, а также разберем его происхождение и связь с еще одной группировкой — вымогателями Eternity.

Краткий обзор последней известной кампании RGB-Team

Цепочка атаки мало чем отличается от стандартного для хактивистов сценария. Изначально злоумышленники скомпрометировали сервис WordPress газодобывающей компании, заразили ее системы червем CMoon, а также получили доступ к панели управления и опубликовали при помощи техники Drive-By Compromise вредоносные образцы по URL-пути **wp-content/uploads/exe/**. В дальнейшем атакующие отправляли жертвам письма с фишинговыми ссылками, которые вели на размещенное на этом ресурсе вредоносное ПО CMoon, замаскированное под офисные файлы при помощи двойного расширения. Большинство файлов представляли собой самораспаковывающиеся архивы с документом-приманкой и собственно вредоносной нагрузкой. Однако были также исполняемые файлы, которые скачивались без архива, например образец **1_dogovor-o-tehnicheskomo-obsluzhivanii-i-remonte-vnutridomovogo-gazovogo.docx.exe**. CMoon скомпилирован на .NET версии 4.7.2 и обфусцирован при помощи собственного алгоритма со вставками лишнего кода и ложными ветвлениями, которые увеличили его размер. При запуске он считывает прописанную в коде конфигурацию, запускает набор модулей и ждет команд от оператора.

Мы [описывали эту кампанию CMoon](#) летом 2024 года и на тот момент не нашли связи с какими-либо известными группами злоумышленников. Однако позже RGB-Team взяли на себя ответственность за проведенные атаки и опубликовали украденные данные в своем Telegram-канале.



Данные жертв в Telegram-канале RGB-Team

Новый образец CМoon

В процессе мониторинга угроз мы обнаружили новый образец CМoon в виде бинарного файла с двойным расширением **zayavka-dlya-yur-lits.docx.exe**. Этот образец датируется октябрём 2024 года и относится к ранее неизвестной кампании. Как и более старые образцы, он скомпилирован на .NET версии 4.7.2. Распространялся он тоже через фишинговые письма, однако в качестве вложения.

Запустившись, этот вариант проверяет, не заражено ли устройство уже, при помощи мьютекса **sexmachine3000**: если такой мьютекс создан, значит, CМoon есть в системе. Также в нем присутствует AntiVM-модуль, который детектирует наличие виртуальной среды, песочниц и отладчиков. Этот модуль использует WinAPI-вызовы для сокрытия активности от средств защиты. Для детектирования виртуальной среды он делает запрос к классу WMI Win32_ComputerSystem и ищет в ответе строки **VIRTUAL** и **vmware**. Также он проверяет модели подключенных дисков через ключевые слова **DADY HARDDISK** и **QEMU HARDDISK**.

```
// Token: 0x06000169 RID: 361 RVA: 0x00009C80 File Offset: 0x00007E80
public static bool DetectVirtualEnviroment()
{
    using (ManagementObjectSearcher managementObjectSearcher = new
        ManagementObjectSearcher("Select Manufacturer, Model from Win32_ComputerSystem"))
    {
        using (ManagementObjectCollection managementObjectCollection =
            managementObjectSearcher.Get())
        {
            foreach (ManagementBaseObject managementBaseObject in
                managementObjectCollection)
            {
                string text = managementBaseObject["Manufacturer"].ToString().ToLower();
                string text2 = managementBaseObject["Model"].ToString();
                if ((text == "microsoft corporation" && text2.ToUpperInvariant
                    ().Contains("VIRTUAL")) || text.Contains("vmware"))
                {
                    return true;
                }
            }
        }
    }
    return false;
}
```

Фрагмент AntiVM-модуля

Помимо описанных выше CМoon использует менее заметные способы проверки среды запуска — загружает набор библиотек, которые часто используются в песочницах: cuckoomon, SxIn и другие, представленные на рисунке ниже.

```
public static bool DetectSandBox()
{
    string[] array = new string[]
    {
        "SbieDll",
        "cmdvrt32",
        "cmdvrt64",
        "SxIn",
        "cuckoomon"
    };
    for (int i = 0; i < array.Length; i++)
    {
        if (AntiVM.GetModuleHandle(array[i] + ".dll").ToInt32() != 0)
        {
            return true;
        }
    }
    using (ManagementObjectSearcher managementObjectSearcher = new ManagementObjectSearcher
        ("SELECT Model FROM Win32_DiskDrive"))
    {
        foreach (ManagementBaseObject managementBaseObject in managementObjectSearcher.Get())
        {
            string text = managementBaseObject["Model"].ToString();
            if (text.Contains("DADY HARDDISK") || text.Contains("QEMU HARDDISK"))
            {
                return true;
            }
        }
    }
}
```

AntiSandbox-модуль

В дополнение к этому образец запрашивает свойство своего процесса `ProcessDebugPort` для проверки наличия отладчика. Если вернулось значение 0, он продолжает работу, а если иное, то заканчивает. После успешного прохождения всех проверок CМoon закрепляется через [Alternate Data Streams \(ADS\)](#). Он создает директорию с названием установленного антивируса в качестве имени и размещает в ней одноименный маскировочный файл с расширением `.sqlite`. Если антивируса на устройстве нет, то именем директории и файла становится строка `system`. После этого образец копирует себя в альтернативный поток данных маскировочного файла, а затем создает ярлык, запускающий этот поток, и помещает его в папку автозапуска. Фрагмент кода, отвечающий за закрепление, представлен на рисунке ниже.

```
// Token: 0x00000111 RID: 273 RVA: 0x00000908 File Offset: 0x00000808
private static string SmartInstall()
{
    string[] array = Identification.QueryInstalledAntivirus();
    DirectoryInfo directoryInfo = new DirectoryInfo(Path.Combine(SmartStartup.LocalAppDataDirectory.FullName, array[1]));
    FileInfo fileInfo = new FileInfo(Path.Combine(directoryInfo.FullName, Path.GetFileNameWithoutExtension(SmartStartup.CurrentExecutable.Name) + ".sqlite"));
    FileInfo fileInfo2 = new FileInfo(Path.Combine(SmartStartup.ShellStartupDirectory.FullName, array[1] + ".lnk"));
    bool flag = ADS.IsPathOnDFS(directoryInfo.FullName);
    if (!directoryInfo.Exists)
    {
        directoryInfo.Create();
    }
}
```

Закрепление через ADS

В попытке избежать анализа антивирусом образец подменяет дату и время создания директории и маскировочного файла на заданные в коде: 2013.05.22, 10:32:16. Также в нем используется довольно интересный механизм обфускации. Изначально зловред запускается в необфусцированном виде: переменные и названия функций не переименованы, а значения не зашифрованы. В процессе работы он проверяет свою контрольную сумму по алгоритму MD5 в поисковом движке Google по запросу URL

<https://www.google.com/search?q=> и, если обнаруживает хотя бы одну страницу в поисковой выдаче, то начинает процесс обфускации. Вероятно, злоумышленники считают, что пока файл не описан в публичном поле, детектирование ему не угрожает.

Для обфускации образец копирует данные из ADS в новый поток процесса, шифрует значения и переименовывает функции и переменные. После этого сохраняет новый образец во временной папке и создает новый ADS в том же маскировочном файле.

```
cmd.exe "/c chcp 65001 && type "{temp_file}" > "{current_file}:{ADS_name}" &&
del /F /S /Q /A "{temp_file}" && timeout /t 3 && "{current_file}:{ADS_name}"
```

Скрипт для перезаписи образца

Образец пытается избежать поведенческого анализа, запуская в качестве дочернего процесса пустышку с одним из имен утилит, жестко прописанных в коде. Сам исполняемый файл дочернего процесса компилируется на лету и после запуска создает поток с функцией `Thread.Sleep(5000)`.

```

// Token: 0x02000040 RID: 64
internal sealed class Decoy
{
    // Token: 0x0600016D RID: 365 RVA: 0x00009F14 File Offset: 0x00008114
    public static void Deploy()
    {
        string tempPath = Path.GetTempPath();
        foreach (string str in Decoy.DecoyNames)
        {
            string text = Path.Combine(tempPath, str + ".exe");
            if (!File.Exists(text))
            {
                try
                {
                    if (Compiler.Generate("\r\n                class Program\r\n                {\r\n                {\r\n                System.Threading.Thread.Sleep(5000);\r\n                {\r\n                Process.Start(text);\r\n                }\r\n                }\r\n            ")
                    {
                        Process.Start(text);
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine(ex.Message);
                }
            }
        }
    }
}

// Token: 0x040000EB RID: 235
private const string DecoySource = "\r\n                class Program\r\n                {\r\n                {\r\n                System.Threading.Thread.Sleep(5000);\r\n                {\r\n                }\r\n            }\r\n";

// Token: 0x040000EC RID: 236
private static readonly string[] DecoyNames = new string[]
{
    "ida",
    "ida64",
    "ollydbg",
    "vboxservice",
    "ProcessHacker",
    "wireshark"
};
}

```

Создание дочернего процесса

После этого СМoon внедряет руткит в процессы с открытыми окнами, создавая для них удаленные потоки. Руткит также содержит функцию **Thread.Sleep**. Ее множественное использование позволяет выйти за пределы допустимого времени исполнения файла в песочнице, а также помешать корреляции происходящих событий в SIEM-системах.

```

// Token: 0x0600011F RID: 287 RVA: 0x00002841 File Offset: 0x00000A41
private static void ScanThread()
{
    while (RootKitModule.RootKitEnabled)
    {
        RootKitModule.scanActiveWindow();
        Thread.Sleep(1000);
    }
}

// Token: 0x06000120 RID: 288 RVA: 0x0000285B File Offset: 0x00000A5B
public static void Start()
{
    if (!RootKitModule.RootKitEnabled)
    {
        RootKitModule.RootKitEnabled = true;
        RootKitModule.RootKitThread = new Thread(new ThreadStart(RootKitModule.ScanThread));
        RootKitModule.RootKitThread.Start();
    }
}

```

Инъекция руткита

Все представленные функции управляются жестко прописанным в коде троянца конфигурационным файлом, который создается при сборке образца. Управление CMoon осуществляется через C2-сервер, а данные эксфильтруются через Telegram.

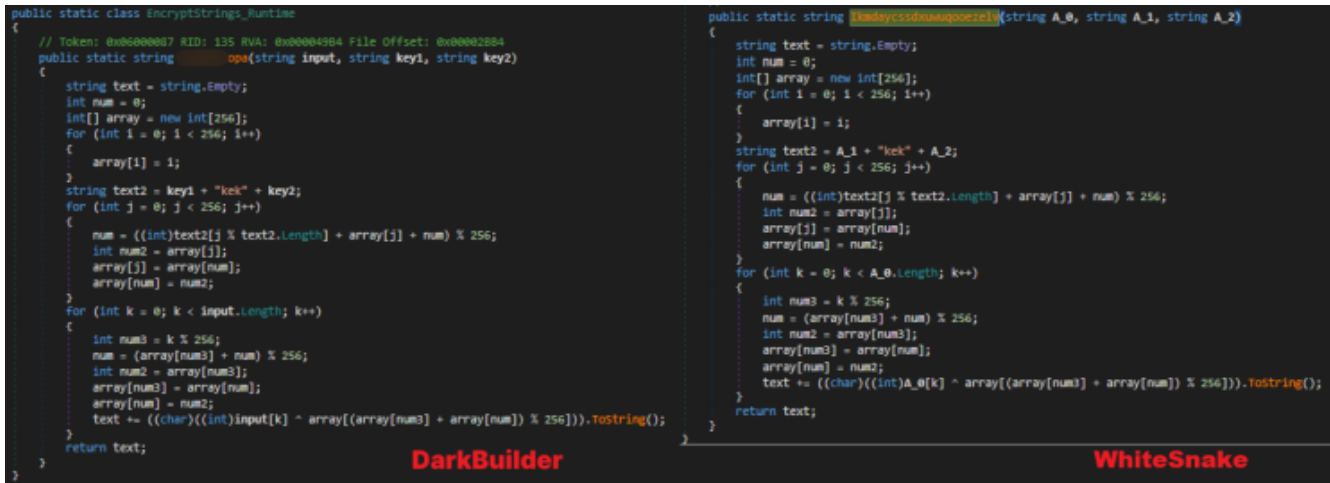
Коммуникации с C2

Как и предыдущие версии, новые образцы CMoon общаются с C2-сервером по протоколу TCP. При этом исходящие пакеты по-прежнему начинаются с автографа зловреда (строки CMOON\$), что позволяет использовать одни и те же правила для детектирования всех известных образцов червя.

Связь CMoon и DarkBuilder

К сожалению, нам не удалось найти билдер CMoon, но на одном из хостов, зараженных этим вредоносным ПО, также находился архив **ct.rar**, содержащий билдер под названием DarkBuilder, в котором используются те же функции, что и в образце CMoon.

Кроме того, на том же хосте мы нашли образец [стилиера WhiteSnake](#), в коде которого есть ссылки на C2-сервер, связанный с CMoon. Анализ этого образца показал, что он был собран с помощью ранее найденного DarkBuilder. В частности, в образце и билдере используются одинаковые функции обфускации. Механизм обфускации в DarkBuilder использует «соль», которая отличается от версии к версии. В образцах WhiteSnake и DarkBuilder, найденных на хосте, соль была одинаковой.



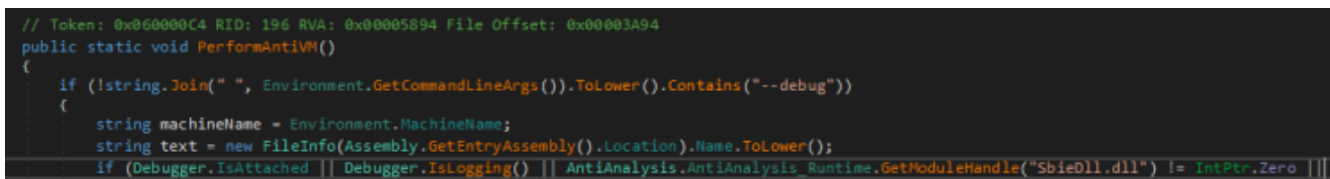
Сравнение DarkBuilder и WhiteSnake

Стоит отметить, что к некоторым образцам CМоон злоумышленники применили аналогичный, но более продвинутый алгоритм обфускации.

Обзор ранней версии DarkBuilder

В процессе исследования угроз мы столкнулись с множеством примеров использования DarkBuilder в реальной среде. При этом хосты, на которых он был обнаружен, никак не связаны друг с другом, что может указывать на компрометацию исходного кода билдера. Подкрепляет эту гипотезу и тот факт, что образец билдера доступен на одном из мультисканеров. Однако нам удалось получить одну из первых версий, сравнить ее с новыми и выяснить, кто за ней стоит.

В ранней версии билдера часть функциональности реализована иначе, чем в более поздних. Например, в ней проверяется наличие прикрепленных отладчиков с помощью функций WinAPI, тогда как в новых этих функций нет. Однако основные функции и кодовая структура одинаковы во всех версиях.



WinAPI-вызов в одной из ранних версий DarkBuilder

Помимо WinAPI-функций, в обнаруженной ранней версии DarkBuilder присутствуют метаданные, ссылающиеся на профиль на GitHub с именем LightM4n (на момент публикации уже удален).

```

using System;
using System.Diagnostics;
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;
using System.Runtime.Versioning;

[assembly: AssemblyVersion("1.5.0.1")]
[assembly: CompilationRelaxations(8)]
[assembly: RuntimeCompatibility(WrapNonExceptionThrows = true)]
[assembly: Debuggable(DebuggableAttribute.DebuggingModes.IgnoreSymbolStoreSequencePoints)]
[assembly: AssemblyTitle("DarkBuilder")]
[assembly: AssemblyDescription("Library for writing builder applications for malware")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("https://github.com/LightM4n")]
[assembly: AssemblyProduct("DarkBuilder")]
[assembly: AssemblyCopyright("Copyright © LightMan 2021")]
[assembly: AssemblyTrademark("")]
[assembly: ComVisible(false)]
[assembly: Guid("f999b3ef-49b0-4088-bcd3-7310abc3fb31")]
[assembly: AssemblyFileVersion("1.5.0.1")]
[assembly: TargetFramework(".NETFramework,Version=v4.7.2", FrameworkDisplayName = ".NET Framework 4.7.2")]

```

Фрагмент кода DarkBuilder с указанием автора

Этот профиль связан с вымогательской группой Eternity, которая также [использовала DarkBuilder](#). У нее есть Telegram-канал, в профиле которого указан контакт разработчика.

Группа Eternity разработала и распространяла по модели MaaS (Malware-as-a-Service) набор вредоносного программного обеспечения: стилер, клиппер, майнер, червь и шифровальщик. К этому ПО прилагался набор инструкций, в том числе — по установке и настройке управляющего сервера. В этой инструкции также упоминается файл DarkBuilder.dll. Ранее мы уже связывали как минимум один из собранных с помощью DarkBuilder образцов с группой Eternity.

C2-сервер стилера WhiteSnake

Изучив обнаруженные образцы DarkBuilder, мы смогли выйти на исходники управляющего сервера WhiteSnake, включающие сам C2-сервер и веб-панель для сборки образцов. Аналогичный набор входил и в ранние версии WhiteSnake, находящиеся в открытом доступе, однако в отличие от них обнаруженный исходник содержит множество функций, схожих с теми, что используются в образцах CMoon. Проект имеет несколько способов развертывания, один из которых — .bat-скрипт.

```

1  :: Add exclusion to Windows Defender to prevent stub deletion while building
2  echo [+] Adding WD exclusion ...
3  powershell -inputformat none -outputformat none -NonInteractive -Command Add-MpPreference -ExclusionPath "%Temp%"
4  powershell -inputformat none -outputformat none -NonInteractive -Command Add-MpPreference -ExclusionPath "%InstallPath%"
5
6  :: Enable Office macro editor (Required if you wanna build Word/Excel files)
7  echo [+] Enable ms office macro editor
8  reg ADD "HKCU\Software\Microsoft\Office\16.0\Word\Security" /v "VBAWarnings" /t REG_DWORD /d 1 /f
9  reg ADD "HKCU\Software\Microsoft\Office\16.0\Word\Security" /v "AccessVBOM" /t REG_DWORD /d 1 /f
10 reg ADD "HKCU\Software\Microsoft\Office\16.0\Excel\Security" /v "VBAWarnings" /t REG_DWORD /d 1 /f
11 reg ADD "HKCU\Software\Microsoft\Office\16.0\Excel\Security" /v "AccessVBOM" /t REG_DWORD /d 1 /f
12
13 :: Copy all files
14 echo [+] Copying files ...
15 mkdir "%InstallPath%"
16 xcopy data\main "%InstallPath%" /s /h /e /k /f /c /i /y
17
18 :: Python reqs
19 echo [+] Installing python requirements
20 python -m pip install setuptools twine
21
22 :: Install WhiteSnake certificate
23 echo [+] Installing root certificate
24 "%InstallPath%\certificate.cer
25 echo [+] Performing initialization
26 "%InstallPath%\dumper.exe --init
27 echo.
28 echo [+] Installation finished
29 timeout /t 3

```

Развертывание C2-сервиса

Скрипт добавляет исключения для Windows Defender, включает редактор макросов, загружает необходимые для выполнения проекта зависимости и добавляет корневой сертификат. Злоумышленники поставляют вместе с билдером самоподписанный корневой сертификат, который позволяет не регистрировать у регуляторов SSL-сертификаты сервера, чтобы затруднить обнаружение C2-серверов. Подготовив устройство, скрипт устанавливает C2-сервер и веб-панель, которая содержит функциональность конструктора образцов, что позволяет партнерам MaaS с низкими техническими навыками создавать собственные варианты стилера.

Custom C2 endpoint (Optional; Has higher priority when exfiltrating data; in case of failure we will try upload it to Telegram)

Telegram BOT token ([Automatic setup](#))

Telegram account or channel chat ID

Build TAG

Loader (Direct links separated by comma; leave empty to disable)

Log encryption method [Log can be decrypted only by your panel]

☒ RSA+RC4 encryption
☐ RC4 encryption

Grabber commands

Функциональность конструктора образцов в веб-панели

При генерации образца можно выбрать удобный для злоумышленника формат C2-коммуникаций. В панели предложено две опции: по протоколу HTTP, обращаясь к адресу сервера вида «http://IP-адрес:порт», либо через Telegram-бота. Помимо этого, партнер MaaS может добавить в конструктор собственный загрузчик, предназначенный для доставки полезной нагрузки в целевую систему. Также можно загрузить файл конфигурации в формате XML, содержащий команды для сбора файлов определенных типов, или ссылку на такую конфигурацию. Наконец, злоумышленник может выбрать, будет его образец закрепляться на устройстве (создатели стилера назвали эту опцию «резидентным режимом») или нет («нерезидентный режим»).

.NET framework version

4.7 (Windows 8, 10, 11)

File size pumper [Disabled]

Execution method [Steal data and install beacon for remote access later.]

☐ Non-resident

☒ Resident

Resident mode connection method

Serveo (You will receive a special code in the telegram bot, which must be pasted into the terminal window before access)

☒ Obfuscate ☐ AntiVM ☐ Auto-Keylogger ☐ Clipper ☒ Random resources



Функциональность конструктора в веб-панели: выбор режима выполнения, типа соединения и т. д.

Если выбрать опцию с закреплением на устройстве, в конструкторе нужно уточнить тип соединения с узлом C2: через сеть Tor или через сервис обратного проксирования Serveo. Сам C2-сервер реализован в виде приложения, написанного на Flutter. Перед началом общения он аутентифицирует образец по паре асимметричных ключей. Сервер использует несколько конечных точек для разных целей.

Конечные точки	Тип запроса	Функции
/auth	POST	Использование открытого и закрытого ключей RSA для проверки подлинности образца
/mnemonic-verify/<machine_id>/<owner_id>	GET	Проверка наличия образца на узле, чтобы избежать повторного заражения
/f/<template>	GET, POST	Управление шаблонами для сборки образцов: GET-запрос позволяет получить шаблон, а POST-запрос — создать новый
/sendData	POST	Получение информации из модуля кражи данных и изменение флага в поле данных 'WSR\$' или 'LWSR\$'
/sendSpoofing	POST	Получение данных из модуля клиппера и изменение флага в поле данных 'WSR\$' или 'LWSR\$'
/api/download/<id>	GET	Скачивание отчета
/api/remove/<id>	POST	Удаление отчета

Выводы

Хотя пик активности RGB-Team пришелся на 2024 год, до сих пор деятельность группы практически не была описана. В этой статье мы выявили связь злоумышленников с кампанией стилера CMoon в июле 2024 года, проанализировали ранее не описанную новую версию этого ПО, а также вышли на инструменты, на основе которых он, вероятнее всего, создан.

Как можно видеть из анализа, вместо собственных разработок злоумышленники полагаются на попавшие в открытый доступ инструменты, которые ранее распространяла по модели Malware-as-a-Service киберпреступная группа Eternity.

При этом стоит отметить, что хактивисты RGB-Team модифицируют и переписывают эти инструменты под свои задачи. Новые обнаруженные образцы демонстрируют возможности и уровень знаний злоумышленников в области разработки вредоносного ПО: они используют продвинутые методы обфускации, включая вставку «мусорного» кода и метаморфические преобразования, которые изменяют структуру и логику загрузчика, чтобы обходить сигнатурные методы обнаружения.

Для обхода динамических средств защиты злоумышленники применяют функции задержки (sleep), а для сокрытия сетевой инфраструктуры используют CDN-сети, VPS-серверы и сервисы обратного проксирования.

Группа RGB-Team атаковала российские организации, относящиеся к химической промышленности и государственному сектору. На данный момент мы не наблюдаем никакой новой активности этих злоумышленников, однако нет гарантии, что они не возобновят атаки, как [иногда поступают](#) хактивисты.

Детектирование решениями «Лаборатории Касперского»

Одним из способов обнаружения атак RGB-Team является мониторинг сетевого трафика с целью выявления характерных аномалий. Вредоносная активность в нем может быть обнаружена с помощью .

Ниже показан интерфейс KATA NDR с оповещением об обнаружении C2-коммуникаций CMoon по протоколу TCP. В данном случае сработало правило *Worm.CMoon.TCP.C&C*, проверяющее наличие уникального значения в начале пакета, явно идентифицирующего это семейство.

Kaspersky
Anti Targeted
Attack Platform

Dashboard
Alerts
Network traffic events
Custom rules
Storage
Assets
Reports
Logs
Settings
Sensor servers

All alerts ▸ Alert #39 ☆

Show related ▾

Assign to @Me ▾

Close alert

State New

Importance High

Data source SPAN Sensor 10.70.246.70 (2026-03-10 15:01:05)

Time created 2026-03-10 15:01:05

Time updated 2026-03-10 15:01:07

Scan results

IDS rule

Payload

Length = 87

000 43 40 4F 4F 4E 24 07 41 42 FD 37 FD FD FD FD 00 CMOON\$-Abe7mnm.

010 00 00 00 00 00 00 23 44 FD FD 42 FD 22 28 FD 00eeBx"ee

020 2E FD 07 2C FD FD 02 50 38 66 FD 3F 19 67 FD FD .w-.wR]8fa?sgae

030 5A FD FD 65 32 4E FD 35 31 66 63 65 63 38 33 34 Zwaee2Na51fcec834

040 64 34 56 35 39 35 39 64 34 34 56 33 56 61 63 63 d465039d44656ecc

050 62 35 32 31 31 63 31 b5211c1

Rule details

Header alert:tcp any any -> any any

flow to_server,established

content "CMOON\$"

nocase true

depth 8

threshold type both,track by_arc,count 1,seconds 60

sid 72004777

Network event

Локальную активность CMoon можно детектировать с помощью .

explorer.exe 23

build.exe 2

VHO:Worm.MSIL.CMoon.gen

cmd.exe 3

cmd.exe 1

VHO:Worm.MSIL.CMoon.gen

build.dat 41

66.254.114.41-1

chcp.com 1

timeout.exe 1

Windows Defender link

Isolate te-ws10-vm-alc-h.lab2.local

Create prevention rule

Create task ▾

File created

IOA tags file_creating_with_suspicious_extension_in_startup_folder

File "C:\Users\user_ca\AppData\Roaming\Microsoft\Windows\Start Menu\Progra...

MD5 f068b89a8a8d9eee7af627be8dcfed56

SHA256 99b4634b64795f541531420e721838c38dba5c4ff2834c6becb613dc5fb0341c

File type LNK file

Size 3 KB

Attributes A

Zone ID 0

Event time 2026-03-20 16:50:24.665

Time created 2013-05-22 10:32:16.000

Time modified 2013-05-22 10:32:16.000

Event initiator

File "C:\distr\build.exe"

Launch parameters "C:\distr\build.exe"

Find events

MD5 132404f2b1c1f5a4d76bd38d1482bdfa

SHA256 a4be526be5359ad2981f439457fe652895731ad56c10c113c22a7836a9591e5d

System info

Host name te-ws10-vm-alc-h.lab2.local

Host IP 10.10.10.210

User name LAB2\user_ca

OS version Microsoft Windows Professional 10.0.19045

Один из ярких артефактов этой активности — закрепление через папку Startup. Злоумышленник может разместить вредоносный файл или ярлык в директориях автозагрузки, чтобы обеспечить автоматический запуск своего кода при каждом входе

14/16

пользователя в систему. Решение KEDR Expert детектирует эту активность с помощью правила file_creating_with_suspicious_extension_in_startup_folder.

Индикаторы компрометации

Хэши файлов

CMOON

[132404F2B1C1F5A4D76BD38D1402BDFAD1C927C3668C16DB3C8D716F9B935790B201A94E80F8CC1AD7CC2448AE5B9259D77A015F022D8A0CBF00204AC3AE496D46AC6CDA371B83CE910C404A08059B25CA5B677820C0CD1B65697DF3A684359361D8073566892D3E010E3F745B0237DFDDFA9440FCCCBA572FB800C8D14400D1F9AF6743D3BD7E084810C70F8D9B10F6CCFED1DC8D319A0EB55082D91243A33264942A8EC9B2796C8F9522D6E7F2C2488DE3B8AC5FA4CC80FBA51AA1DCA9F953](#)

DarkBuilder

[150618981088B0B6BEBA03B459B3494090261F729F296A8A65488286F9C52DF1](#)

WhiteSnake Server

[142EE417E051FF1205FB11E4CC94DEA4](#) — release.zip

Имена файлов CMOON

zayavka-dlya-yur-lits.docx.exe
протокол.exe
пример заполнения протокола.exe
build (2).rar

Имена файлов DarkBuilder

darkbuilder.dll

C2-серверы

[93.185.167\[.\]95](#)
[45.59.104\[.\]152](#)

Группа RGB-Team: кто стоит за стилером CMoon и откуда он взялся

Ваш e-mail не будет опубликован. Обязательные поля помечены *

Имя *

E-mail *

[Cancel](#)

Δ