АНТИВИРУСНАЯ ЗАЩИТА БИЗНЕС-КЛАССА

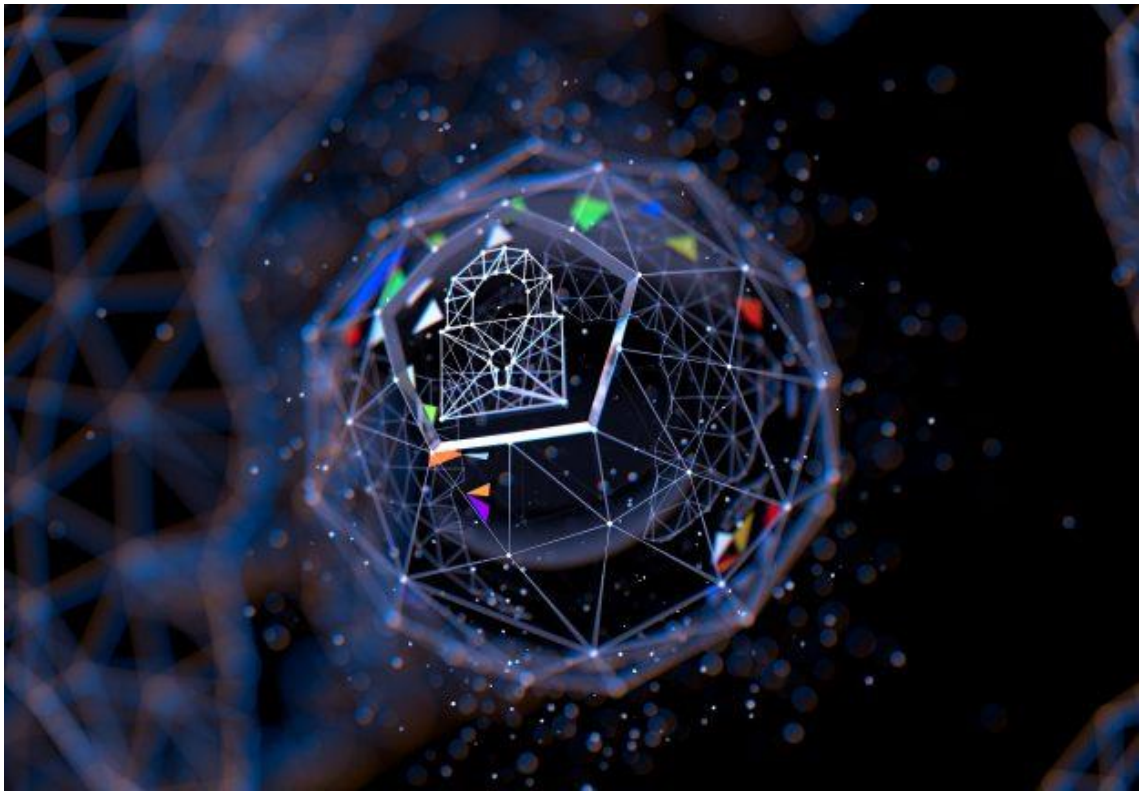
ESET: анализ новых компонентов Zebrocy

23 января 2019 года

Кибергруппа Sednit действует минимум с 2004 и регулярно фигурирует в новостях. Считается, что Sednit (более известные как Fancy Bear) стоят за взломом Национального комитета Демократической партии США перед выборами 2016 года, Всемирного антидопингового агентства (WADA), телевизионной сети TV5Monde и другими атаками. В арсенале группы набор вредоносных инструментов, некоторые из которых мы задокументировали в прошлом отчете.

Недавно мы выпустили отчет о LoJax – UEFI-рутките, который также имеет отношение к Sednit и использовался в атаках на Балканах, в Центральной и Восточной Европе.

В августе 2018 года операторы Sednit развернули два новых компонента Zebrocy, и с этого момента мы наблюдаем всплеск использования этого инструмента. Zebrocy – набор из загрузчиков, дропперов и бэкдоров. Загрузчики и дропперы предназначены для разведки, в то время как бэкдоры обеспечивают персистентность и шпионские возможности. У этих новых компонентов необычный способ эксфильтрации собранных данных через связанные с почтовыми службами протоколы SMTP и POP3.



Жертвы новых инструментов напоминают пострадавших, упомянутых в нашем предыдущем посте про Zebrocy, а также у Лаборатории Касперского. Цели атак находятся в Центральной Азии, Центральной и Восточной Европе, это преимущественно посольства, министерства иностранных дел и дипломаты.

Обзор

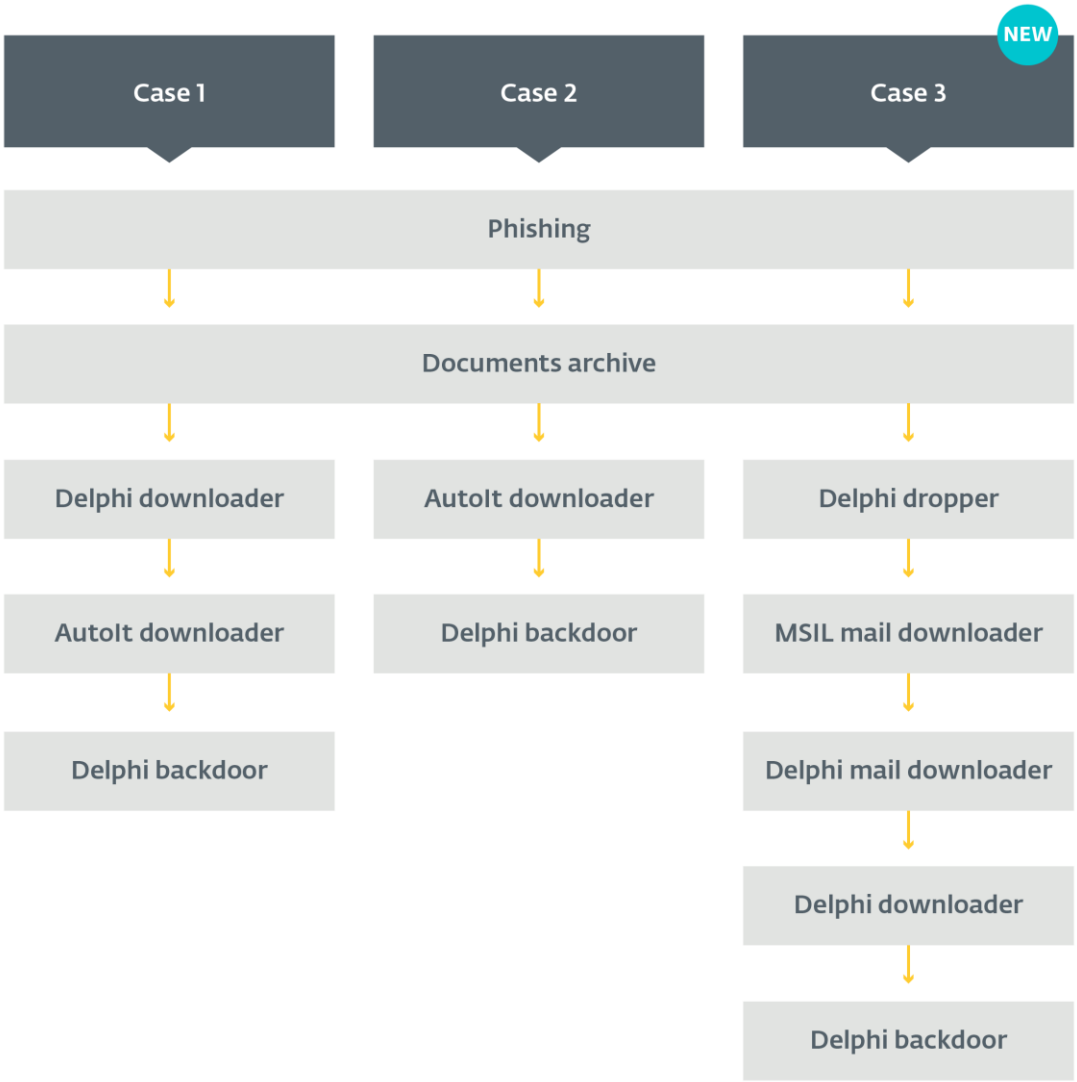


Рисунок 1. Схема старых и новых компонентов Zebrocy

На протяжении двух лет кибергруппа Sednit в качестве вектора заражения Zebrocy использовала фишинговые письма (варианты 1 и 2 в таблице выше). После компрометации атакующие применяли различные загрузчики первого этапа для сбора информации о жертве и, в случае заинтересованности, через несколько часов или дней разворачивали один из бэкдоров второго уровня.

Классическая схема кампании Zebrocy – получение жертвой архива во вложении к письму. Архив содержит два файла, один из которых – безобидный документ, а второй – исполняемый файл. Атакующие пытаются обмануть жертву, называя второй файл типичным для документа или изображения именем и используя «двойное расширение».

В новой кампании (вариант 3 в таблице) применяется более запутанная схема – ее разберем ниже.

Delphi-дроппер

Первый бинарный файл – Delphi-дроппер, что довольно необычно для кампании Zebrocy. В большинстве случаев это, скорее, загрузчик, устанавливаемый в системе жертвы на первом этапе атаки.

С помощью нескольких методов дроппер усложняет реверс-инжиниринг. В исследованных образцах он использует ключевое слово *liver* для обозначения начала и конца ключевых элементов, как показано ниже.

```
$ yara -s tag_yara.yar  SCANPASS_QXWEGRFGCVT_323803488900X_jpeg.exe
find_tag SCANPASS_QXWEGRFGCVT_323803488900X_jpeg.exe
0x4c260:$tag: l\x00i\x00v\x00e\x00r\x00
0x6f000:$tag: liver
0x6f020:$tag: liver
0x13ab0c:$tag: liver
```

Правило YARA выше ищет строку *liver*. Первая строка *liver* используется в коде, но ничего не разделяет, в то время как остальные разделяют дескриптор ключа, изображение (ниже приведен его hexdump) и зашифрованный компонент в дроппере.

```
$ hexdump -Cn 48 -s 0x6f000  SCANPASS_QXWEGRFGCVT_323803488900X_jpeg.exe
0006f000  6c 69 76 65 72 4f 70 65  6e 41 69 72 33 39 30 34  |liverOpenAir3904|
0006f010  35 5f 42 61 79 72 65 6e  5f 4d 75 6e 63 68 65 6e  |5_Bayren_Munchen|
0006f020  6c 69 76 65 72 ff d8 ff  e0 00 10 4a 46 49 46 00  |liver……JFIF.|
```

Сначала данные сохраняются в картинку с именем файла *C:\Users\public\Pictures\scanPassport.jpg*, если такой файл еще не существует.

Интересно, что файл дроппера называется *SCANPASS_QXWEGRFGCVT_323803488900X_jpeg.exe*, что также подсказывает о фишинговой схеме, связанной с паспортами и информацией о путешествиях. Это может означать, что оператор мог знать цель фишингового сообщения. Дроппер открывает изображение и, если файл уже существует, прекращает исполнение. В ином случае он его открывает и получает дескриптор ключа *OpenAir39045_Bayren_Munchen*. Изображение отсутствует, хотя формат и верный – см. рисунок ниже.

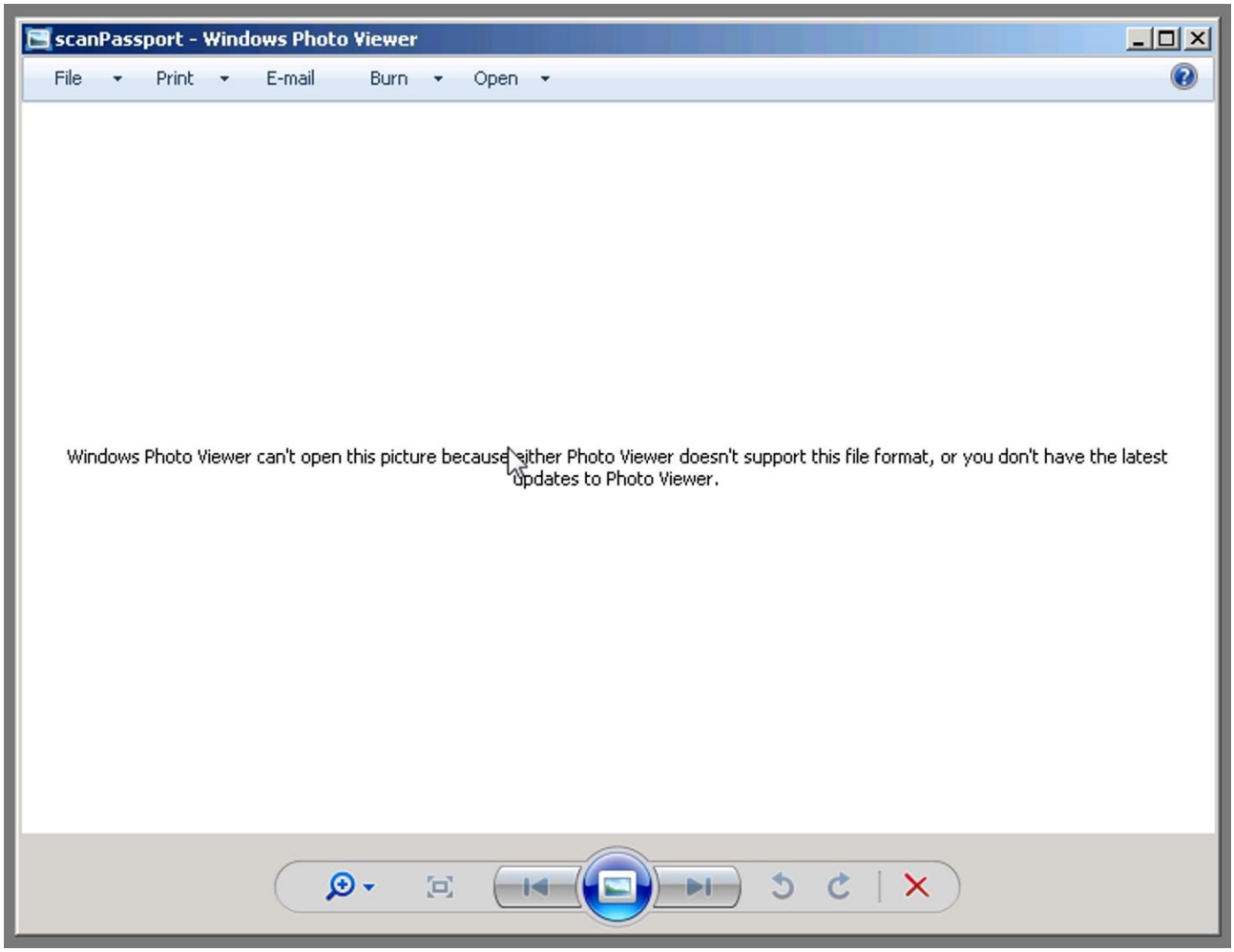


Рисунок 2. ScanPassport.jpg

Строка дескриптора ключа содержит *Bayren_Munchen* – скорее всего, это отсылка к футбольной команде FC Bayern Munich. В любом случае, важно не содержимое дескриптора, а его длина, с помощью которой можно получить ключ XOR для расшифровки компонента.

Чтобы получить ключ XOR, дроппер ищет последнее ключевое слово *liver* и делает от него отступ на длину дескриптора. Длина ключа XOR – 27 (0x1b) байт (идентично длине дескриптора ключа).

С помощью ключа XOR и простого цикла дроппер расшифровывает последнюю часть – зашифрованный компонент, находящийся сразу после последнего тэга до конца файла. Обратите внимание, что MZ заголовок исполняемого компонента начинается сразу после ключевого слова *liver*, а ключ XOR получается из части заголовка PE, обычно являющимся последовательностью 0x00 байтов, восстанавливаемых после расшифровки компонента, как показано на рисунке ниже.

```
$ diff -W200 -y <(xxd -s 0x13ab08 SCANPASS_QXWEGRFGCVT_323803488900X_jpeg.exe) <(xxd -s 0x13ab08 decrypted)
0013ab08: a571 ffd9 6c69 7665 7219 31ee 7559 7269  .q..liver.1.uYri    |    0013ab08: a571 ffd9 6c69 7665 724d 5a90 0003 0000  .q..liverMZ.....
0013ab18: 2826 2b2f 2ebb a67a 62d1 7e75 4456 6e75  (&+/...zb.~uDVnu    |    0013ab18: 0004 0000 00ff ff00 00b8 0000 0000 0000  ................
0013ab28: 7833 7e75 546b 7e75 5a72 6928 222b 2f2e  x3~uTk~uZri("+/.    |    0013ab28: 0040 0000 0000 0000 0000 0000 0000 0000  .@..............
0013ab38: 4459 7a62 697e 7544 566e 7578 737e 7554  DYzbi~uDVnuxs~uT    |    0013ab38: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0013ab48: 6b7e 755a 72e9 2822 2b21 31fe 577a d660  k~uZr.("+!1.Wz.`    |    0013ab48: 0000 0000 0080 0000 000e 1fba 0e00 b409  ................
0013ab58: b354 fc57 22b8 5927 161c 274b 0e07 3515  .T.W".Y'..'K..5.    |    0013ab58: cd21 b801 4ccd 2154 6869 7320 7072 6f67  .!..L.!This prog
0013ab68: 1b49 4f0b 4c4f 2a37 1516 491c 1064 241b  .IO.LO*7..I..d$.    |    0013ab68: 7261 6d20 6361 6e6e 6f74 2062 6520 7275  ram cannot be ru
0013ab78: 1b58 1a10 5510 242d 5537 1d0d 4d0c 2622  .X..U.$-U7..M.&"    |    0013ab78: 6e20 696e 2044 4f53 206d 6f64 652e 0d0d  n in DOS mode...
```

Рисунок 3. Зашифрованный компонент (слева) по сравнению с расшифрованным компонентом (справа)

Компонент сбрасывается в *C:\Users\Public\Documents\AcrobatReader.txt* и преобразует файл в *C:\Users\Public\Documents\AcrobatReader.exe*.

Возможно, это попытка обойти защиту ПК, выдающую предупреждение, когда бинарный файл сбрасывает на диск файл с расширением .exe.

В очередной раз оператор пытается обмануть жертву, и, если она обратит внимание на директорию, то увидит картину как на следующем рисунке:

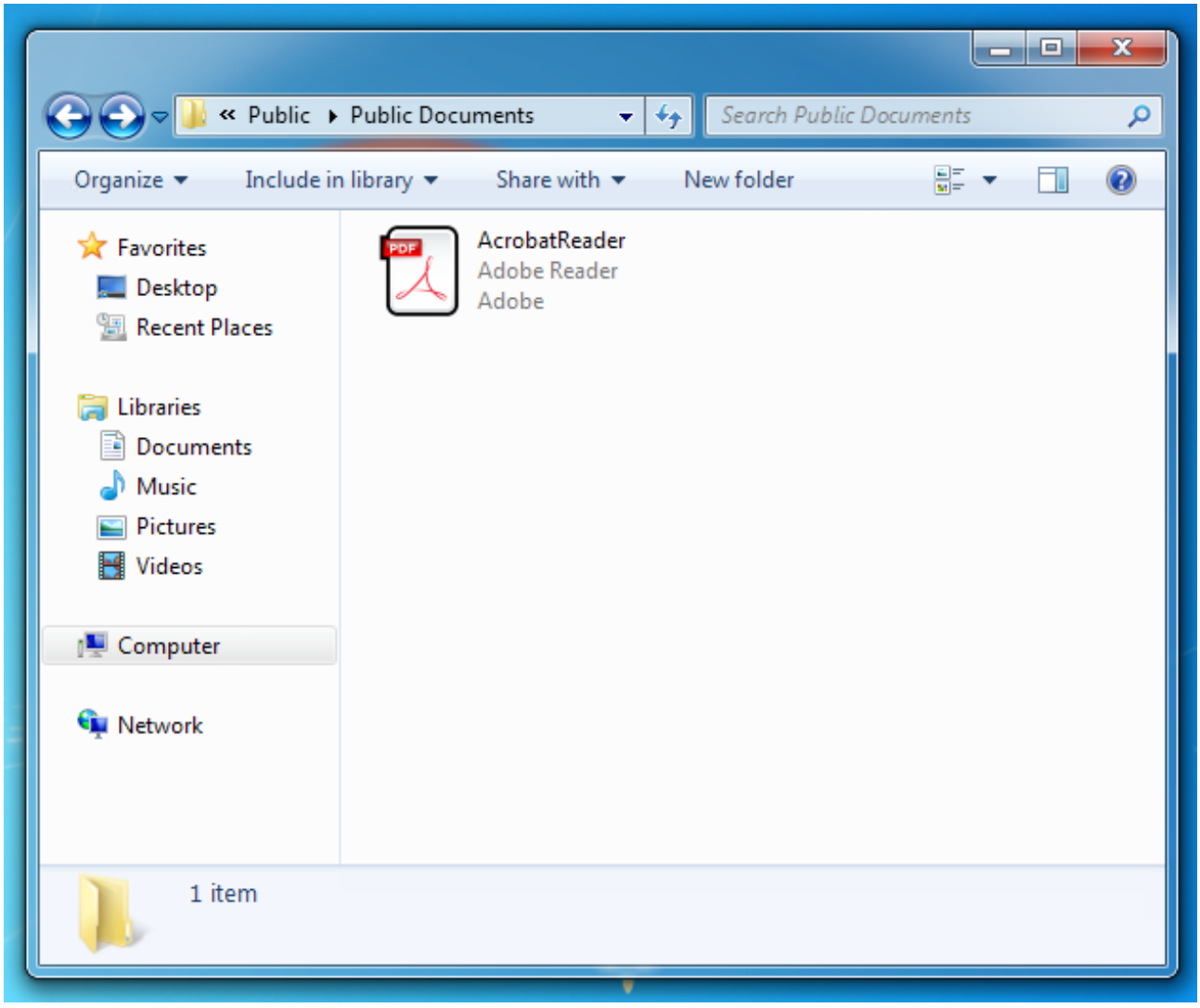


Рисунок 4. Компонент выглядит как файл PDF

По умолчанию Windows скрывает расширение, и этим пользуется злоумышленник, сбрасывающий исполняемый файл в папку «Документы» и маскирующий его под PDF.

Наконец, дроппер выполняет размещенный компонент и завершает работу.

Почтовый загрузчик MSIL

Доставляемый компонент предыдущего дроппера – запакованный с помощью UPX загрузчик MSIL. Для лучшего понимания логика процесса описана ниже, затем приведен исходный код и рассматривается схема управления.

Главный метод вызывает *Run* для запуска приложения, которое затем создает форму *Form1*.

```
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Application.Run((Form) new Form1());
}
```

Form1 назначает множество переменных, включая новый Timer для семи из них.

```
    this.start = new Timer(this.components);
    this.inf = new Timer(this.components);
    this.txt = new Timer(this.components);
    this.subject = new Timer(this.components);
    this.run = new Timer(this.components);
    this.load = new Timer(this.components);
    this.screen = new Timer(this.components);
```

У объекта Timer есть три важных поля:

- Enabled: указывает на включенное состояние таймера
- Interval: время между событиями в миллисекундах
- Tick: коллбэк выполняется по истечении интервала таймера и в случае включенного таймера

Поля обозначены следующим образом:

```
  this.start.Enabled = true;
   this.start.Interval = 120000;
   this.start.Tick += new EventHandler(this.start_Tick);
   this.inf.Interval = 10000;
   this.inf.Tick += new EventHandler(this.inf_Tick);
   this.txt.Interval = 120000;
   this.txt.Tick += new EventHandler(this.txt_Tick);
   this.subject.Interval = 120000;
   this.subject.Tick += new EventHandler(this.subject_Tick);
   this.run.Interval = 60000;
   this.run.Tick += new EventHandler(this.run_Tick);
   this.load.Interval = 120000;
   this.load.Tick += new EventHandler(this.load_Tick);
   this.screen.Interval = 8000;
   this.screen.Tick += new EventHandler(this.screen_Tick);
```

Для каждого объекта выставляется интервал *Interval* от 8 секунд до 2 минут. Коллбэк добавляется в обработчик события. Обратите внимание, что только *start* выставляет значение «истина» для *Enabled*, что означает, что спустя 2 минуты (12 000 миллисекунд = 120 секунд) *start_Tick* будет вызван обработчиком события.

```
    private void start_Tick(object sender, EventArgs e)
    {
        try
        {
            this.start.Enabled = false;
            Lenor lenor = new Lenor();
            this.dir = !Directory.Exists(this.label15.Text.ToString()) ? this.label16.T
ext.ToString() + "\" : this.label15.Text.ToString() + "\";
            this.att = this.dir + "audev.txt";
            this._id = lenor.id(this.dir);
            this.inf.Enabled = true;
        }
```

Далее каждый метод демонстрирует идентичное поведение – меняет значение *Enabled* на *false* в начале. Метод выполняется, а затем меняет значение *Enabled* следующего объекта на *true*, что активирует следующий таймер. Переменная *Enabled* используется оператором для создания нечто вроде машины состояний – если функция не срабатывает, механизм повторяет ее выполнение, пока не получит положительный результат. Время между исполнениями двух функций может использоваться как попытка обойти антивирусную защиту через добавление задержки.

Теперь, после описания структуры каждого метода, перейдем к алгоритму управления. Ниже приведен обзор шагов в виде обмена электронными письмами между ящиками.

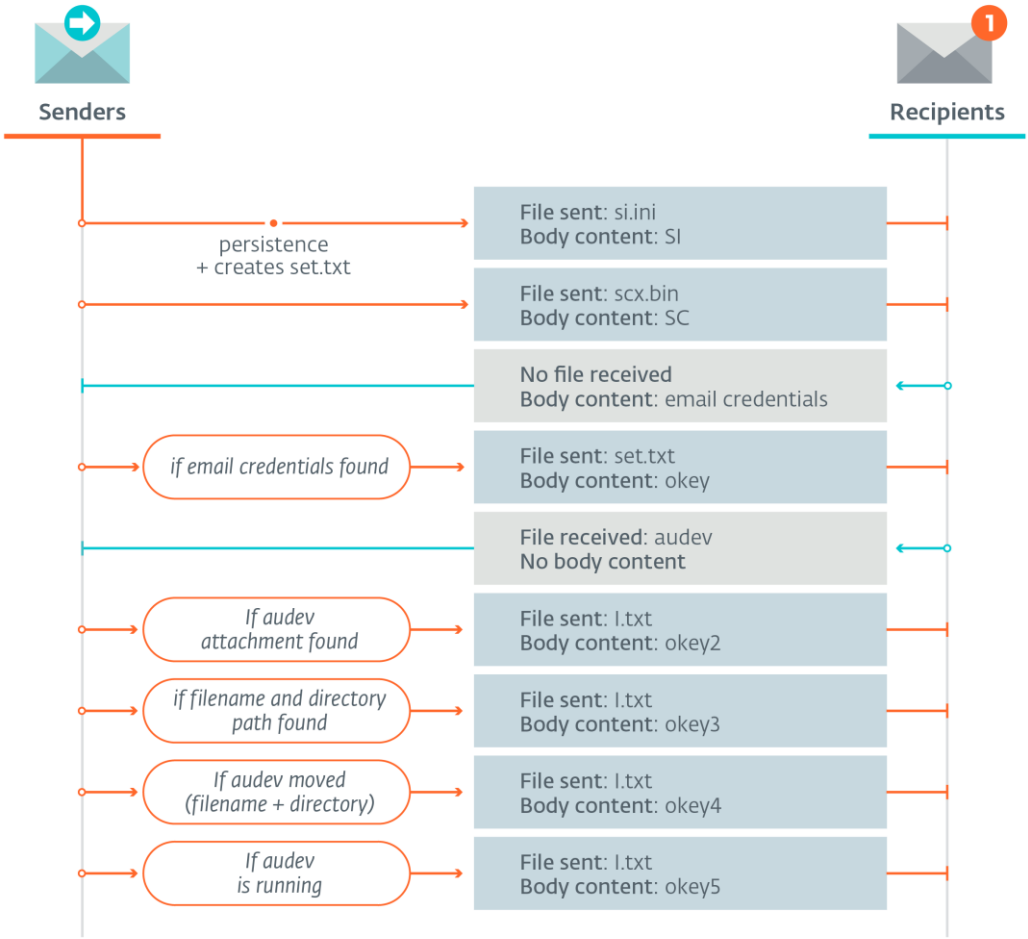


Рисунок 5. Обмен электронными письмами

Малварь проверяет существование определенного пути, используемого для сброса каждого из файлов в процессе исполнения. По возможности используется *C:\Users\Public\Videos*, в ином случае – *C:\Documents and Settings\All Users\Documents* в качестве директории по умолчанию. Обратите внимание, что второй путь специфичен для Windows XP, в то время как первый – для Vista и выше.

16-байтовый *id* генерируется посредством конкатенации серийного номера тома C: и имени пользователя; он хранится в файле *audev.txt*.

Загрузчик собирает следующую информацию:
— текущий путь приложения
— версия операционной системы
— системная директория
— пользовательский домен
— имя машины
— имя пользователя
— текущий часовой пояс
— текущая дата
— список логических дисков и информация о каждом из них (модель, серийный номер, и т.д.)
— листинг директории *C:\Program Files* и *C:\Program Files (x86)*
— список процессов

Эти данные хранятся в файле *C:\Users\Public\Videos\si.ini* и отправляются письмом во вложении через SMTPS с помощью дефолтного порта 465. Тело письма содержит строку *SI* (что, возможно, означает System Information), получатель письма – *sym777.g@post.cz*. Для всего обмена информацией тема писем обозначена как *id*.

Оператор решил завести несколько запасных адресов и отправляет то же письмо двум другим получателям, скорее всего, на случай, если основной адрес не работает. После отправки письма загрузчик удаляет файл *si.ini*.

В процессе первого запуска малвари создается файл *set.txt* с текстом *{System_Parametrs = 10}* внутри и запись в ключе реестра Windows.

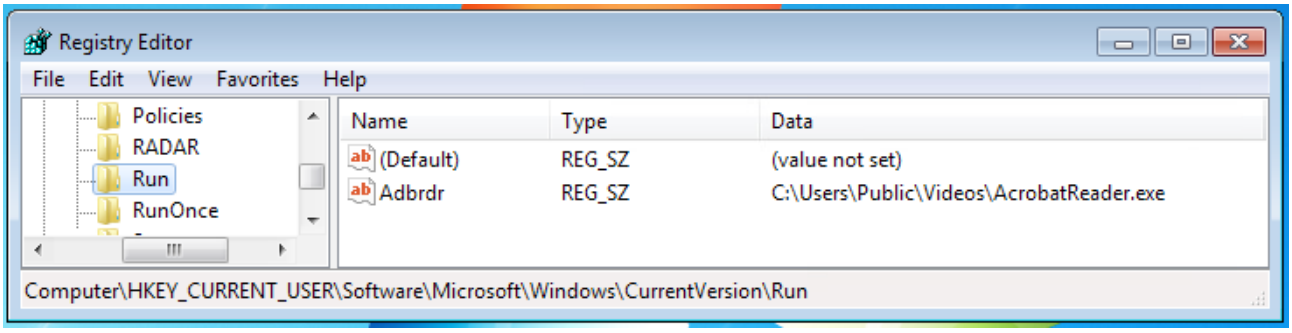


Рисунок 6. Персистентность в реестре

Один скриншот делается под именем *scx.bin* с компьютера жертвы и отправляется вложением по электронной почте с текстом *SC* (что может означать Screenshot) в теле письма.

После отправки малварь связывается с почтовым ящиком *kae.mezhnosh@post.cz* по протоколу POP3 через SSL (порт 995) и ищет сообщения с темой, которая соответствует *id*. Если такое сообщение существует и тело не пусто, малварь расшифровывает его и отправляет сообщение с *okey* в теле на *sym777.g@post.cz*. Содержимое ранее полученного сообщения очищается и парсится следующим образом:

```
string[] strArray = this._adr.Replace("B&", "").Replace("Db", "").Split('%');
        string str1 = strArray[0];
        string str2 = strArray[1];
```

Получаются две строки: первая – это пароль, а вторая – имя пользователя для адреса почты.

Новые учетные данные используются для подключения к полученному почтовому ящику, поиска в нем сообщения с темой, совпадающей с *id* малвари, и приложения со строкой *audev* в имени файла. Если оба условия выполняются, малварь сохраняет приложение и удаляет сообщение с сервера.

Журнал сообщений отправляется на *sym777.g@post.cz*, а сообщения, полученные через POP3, приходят от адресата с недавно полученными пользовательскими данными.

Схема атакующих усложняет расследование. Во-первых, если у вас есть загрузчик с письмами, вы не можете подключиться к почтовому ящику, содержащему следующий этап.

Во-вторых, если вы получаете учетные данные для почты, вы все равно не можете достать следующую полезную нагрузку, потому что она удаляется после получения.

Когда загрузчик успешно записывает вложение на диск, он отправляет в почте сообщение с *okey2* в теле и вложение *l.txt*, содержащее *090*. Этот же файл перезаписывается нулями, и малварь пробует получить другое сообщение. Если это срабатывает, отправляется файл *l.txt* с *okey3* в теле. Содержимое вложения – директория и имя файла. Малварь перемещает файл *audev* по этому адресу. Наконец, малварь отправляет письмо с *okey4* в теле и *l.txt* во вложении. Это запускает исполняемый файл — *audev.exe* и проверяет в списке процессов наличие строки *audev*.

```
Process.Start(this.rn);
 foreach (Process process in Process.GetProcesses())
 {
     if (process.ProcessName.Contains("audev"))
 }
```

Если обнаруживается такое имя, будет отправлено последнее письмо, содержащее в теле *okey5* и *l.txt*во вложении. Наконец, удаляется *l.txt* и *set.txt*, удаляется созданный ключ реестра Windows, и работа программы завершается.

Почтовый загрузчик на Delphi

Основная роль загрузчика – оценить важность скомпрометированной системы и, если она представляется интересной, загрузить и выполнить последний загрузчик Zebrocy.

Бинарный файл написан на Delphi и упакован с помощью UPX. Полное определение объекта *TForm1*можно найти в разделе с его ресурсами, в нем приведены некоторые используемые параметры конфигурации. Следующие разделы описывают инициализацию, возможности и сетевой протокол загрузчика.

Инициализация

Сначала производится расшифровка набора строк, являющихся адресами электронной почты и паролями. Оператор применяет алгоритм шифрования AES ECB. Каждая строка расшифровывается по шестнадцатеричному разряду, где первые четыре байта соответствуют финальному размеру расшифровываемой строки (расшифрованные строки в конце могут содержать некоторые отступы). В объекте *TForm1* содержится два AES ключа: первый используется для шифрования данных, а второй – для их расшифровки.

Адреса почты и пароли используются оператором для отправления команд для малвари, а также для получения информации, собранной с компьютера жертвы. Применяются протоколы связи SMTP и POP3 – оба через SSL. Для использования OpenSSL малварь скидывает и применяет две динамические библиотеки OpenSSL: *libeay32.dll (98c348cab0f835d6cf17c3a31cd5811f86c0388b)* и *ssleay32.dll(6d981d71895581dfb103170486b8614f7f203bdc)*.

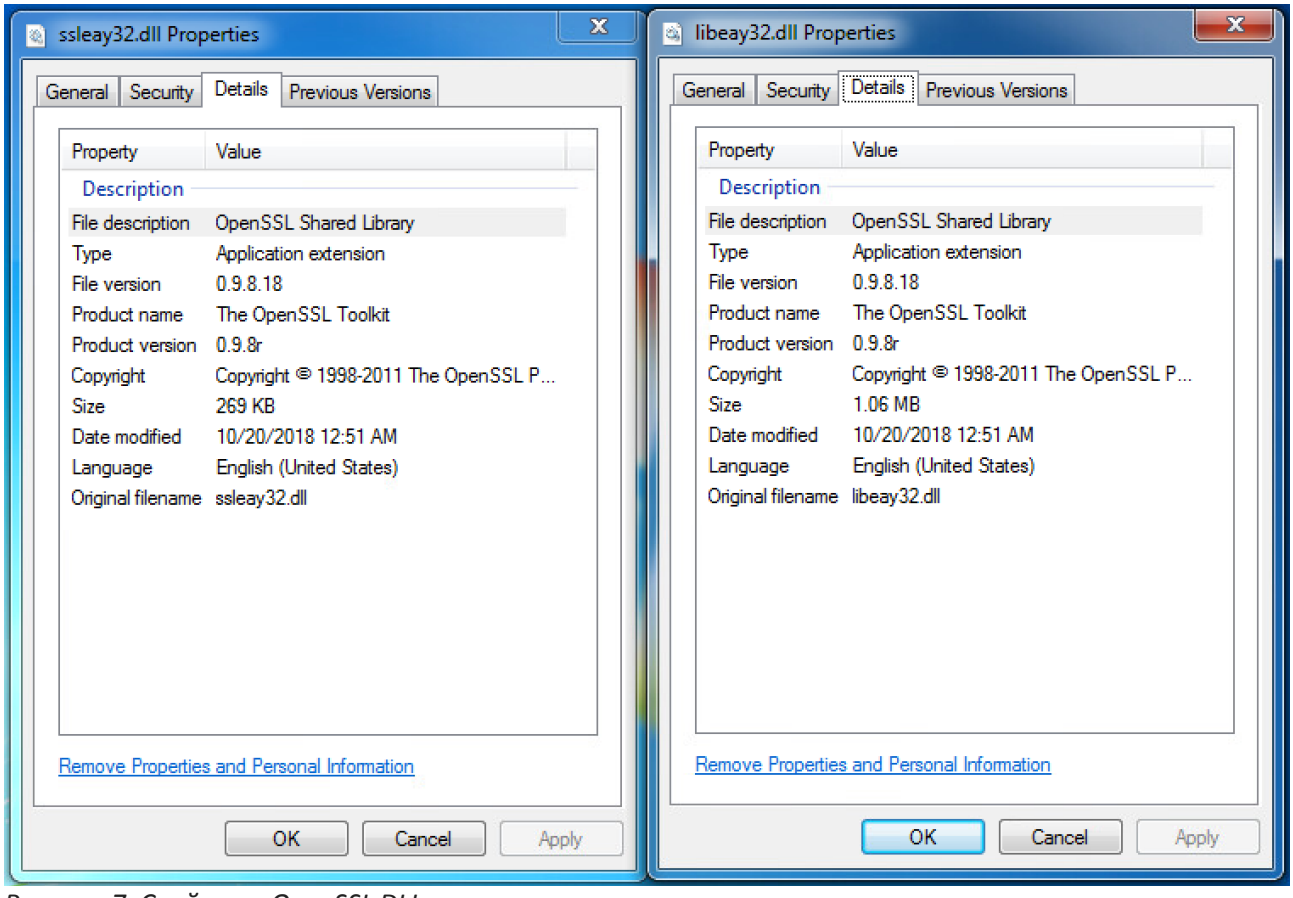


Рисунок 7. Свойства OpenSSL DLL

Обратите внимание, что все файлы сбрасываются в рабочую директорию малвари *C:\Users\Public*.

Персистентность обеспечивается во время первого исполнения малвари посредством известной техники скриптов сценария входа. Создается файл со скриптом *registration.bat* и пишется несколько строк из объекта *TForm1*. Финальный скрипт выглядит следующим образом:

```
reg add HKCU\Environment /v "UserInitMprLogonScript" /t REG_EXPAND_SZ /d "C:\Users\Publ
ic\Videos\audev.exe" /f
del C:\Users\Public\Videos\registr.bat
exit
```

В последнюю (но не по важности) очередь малварь создает *id*, таким же образом, как и в ранее описанных бинарных файлах Zebrocy. Она получает имя пользователя с помощью *GetUserNameW*Windows API и добавляет в начало серийный номер диска C:\.

Возможности

Учитывая, что для процесса сбора информации о жертве есть несколько условий и порядок действий, ниже приведено описание различных его возможностей. Конфигурация сканирования хранится в объекте *TForm1*, где сгруппированы семь различных возможностей для сбора информации с компьютера жертвы.

Начиная с простого сканирования, первая информация, которую может получить малварь, связана с файлами со следующими расширениями: *.docx, .xlsx, .pdf, .pptx, .rar, .zip, .jpg, .bmp, .tiff*. Для каждого из файлов, обнаруженных на диске, малварь получает полный путь и последнюю дату изменения. Эта информация шифруется с помощью ключа AES, о котором мы говорили ранее, и сохраняется в файл *0.txt*. Другое сканирование нацелено на расширения *.dat, .json, .db* и, как и в предыдущем случае, получает полный путь и последнюю дату изменения файла. Затем шифрует их и хранит в файле *57.txt*.

Листинг запущенных процессов – еще одна возможность малвари, позволяющая хранить информацию в файле *08.txt*. Это выглядит следующим образом:

```
======Listing_of_processes=======
[System Process]
System
smss.exe
csrss.exe
wininit.exe
csrss.exe
winlogon.exe
services.exe
lsass.exe
[…]
```

В файле *i.txt* собирается общая информация о компьютере жертвы, а также некоторые данные о малвари (номер версии и путь, по которому она выполняется). См. в примере ниже:

```
v7.00
C:\Users\Public\Videos\audev.txt
========================================
Log_Drivers:
 C: fixed; size= 102297 Mb, free=83927 Mb S/N: [redacted]
 ================================================
 OSV: Windows 7
WinType: 32
WinDir: C:\Windows
Lang: English (United States)
TZ: UTC1:0 Romance Standard Time
HostN: [redacted]-PC
User: [redacted]
==============S_LIST====================
C:\Program Files\Common Files
C:\Program Files\desktop.ini
C:\Program Files\DVD Maker
C:\Program Files\Internet Explorer
C:\Program Files\Microsoft.NET
C:\Program Files\MSBuild
C:\Program Files\Reference Assemblies
C:\Program Files\Uninstall Information
C:\Program Files\Windows Defender
[…]
```

Малварь может делать скриншоты, которые сохраняются в формате *2\[YYYY-mm-dd HH-MM-SS]-Image_001.jpg*, и генерирует другой файл *2\sa.bin*, заполненный списком путей к файлам всех снятых скриншотов. Последняя возможность – перечисление сетевых компонентов и данных о системе, результат записывается в *4.txt*.

Сетевой протокол

Почтовый загрузчик Delphi – сравнительно новое дополнение инструментария Zebrocy, в нем предусмотрен новый способ эксфильтрации данных и получения команд от оператора. Эксфильтрация довольно проста, но производит много шума в сети, так как ранее собранные зашифрованные файлы отправляются через SMTPS, каждая версия файла – трижды.

Sender	Recipient
kevin30@ambcomission.com	rishit333@ambcomission.com
salah444@ambcomission.com	rishit333@ambcomission.com
karakos3232@seznam.cz	antony.miloshevich128@seznam.cz

Тема письма – *id* жертвы, и файл отправляется в виде приложения с ключевым словом, соответствующим содержимому файла. Обратите внимание, что для каждого файла есть отправляемая зашифрованная версия.

files	files encrypted	keywords
-	0.txt	SCAN
57.txt	58.txt	ACC
08.txt	082.txt	PrL
i.txt	i2.txt	SYS_INFO
4.txt	42.txt	GET_NETWORK

Скриншоты и файлы по обоим сканированиям тоже отправляются, но с другими ключевыми словами.

Content	Keywords
screenshots	SC
.docx, .xlsx, .pdf, .pptx, .rar, .zip, .jpg, .bmp, .tiff	FILEs
.dat, .json, .db	D_ACC

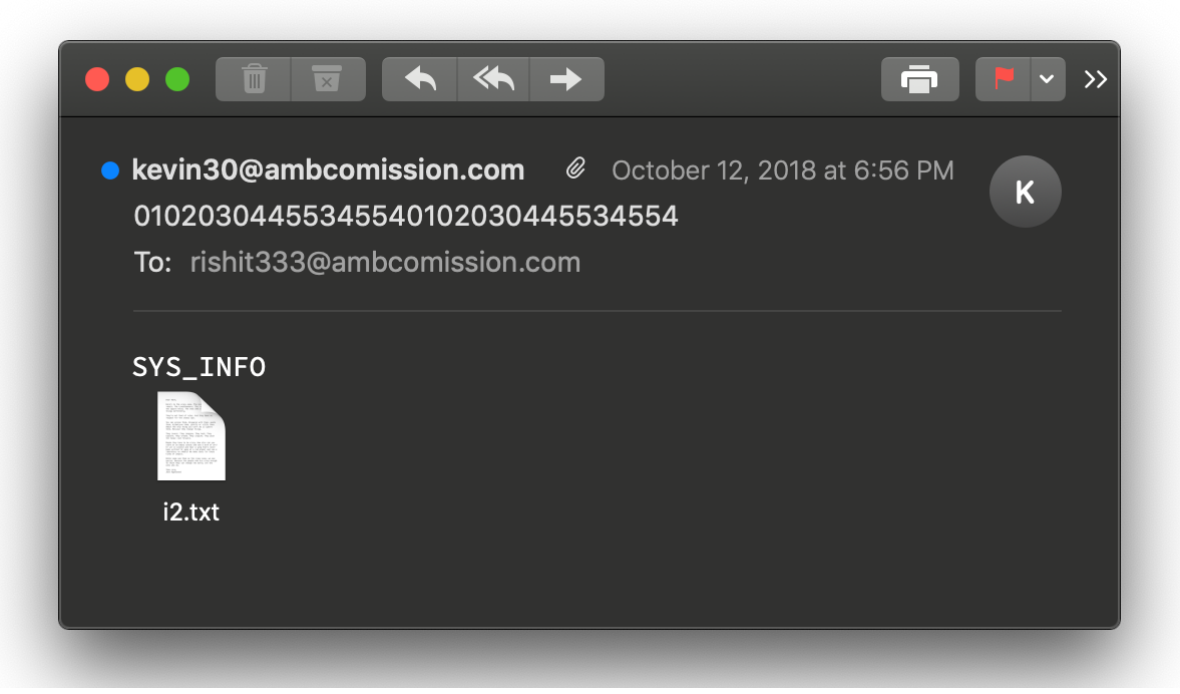


Рисунок 8. Пример письма с передаваемыми данными

В то время как при эксфильтрации данных используется протокол SMTP, бинарный файл связывается с адресом почты *tomasso25@ambcomission.com* через POP3 и парсит письма. Тело письма содержит различные ключевые слова, интерпретируемые малварью в качестве команд.

Keywords	Purpose	Log
scan	scan	
ldfile	scan	
edit34	execute and delete	
pKL90	register	isreg
prlist	process listing	
Start23	execute	isr
net40	enumerating network resources	
dele5	delete file	isd
dele6	delete directory	isd
cd25	create directory	isc
autodel	delete itself	
Co55	copy file	is_cp
Mo00	move file	is_m

После выполнения лог дебаггера и результат команд (если есть) отправляются назад оператору. Например, после команды на сканирование оператор получает файл, содержащий список файлов с совпадающими расширениями вместе с каждым таким файлом.

В то время как у этого загрузчика есть функции бэкдора, он сбрасывает в систему загрузчик на Delphi, уже связанный с данной группой, которые мы описывали в предыдущей статье о Zebrocy.

Заключение

В прошлом мы уже видели пересечения Zebrocy и традиционных для Sednit вредоносных программ. Мы ловили Zebrocy на сбросе в систему XAgent – флагманского бэкдора Sednit, поэтому с высокой долей уверенности приписываем авторство Zebrocy этой кибергруппе.

Тем не менее, анализ бинарных файлов выявил ошибки на уровне языка, а также разработку, указывающую на иной уровень квалификации авторов. Оба загрузчика применяют почтовые протоколы для эксфильтрации данных и идентичные механизмы для сбора одной и той же информации. Однако они создают много шума в сети и системе, создавая множество файлов и пересылая их. В процессе анализа почтового загрузчика на Delphi нам показалось, что некоторые функции пропали, но строки все еще оставались в бинарном файле. Этот набор инструментов используется группой Sednit, но мы полагаем, что он разрабатывается другой командой – менее опытной, в сравнении с создателями традиционных компонентов Sednit.

Компоненты Zebrocy – дополнение в инструментарии Sednit, и недавние события могут объяснить повышенное активное использование бинарных файлов Zebrocy вместо традиционной малвари.

АНТИВИРУСНАЯ ЗАЩИТА БИЗНЕС-КЛАССА

Индикаторы компрометации

Имена файлов, SHA-1 и детектирование продуктами ESET

1. SCANPASS_QXWEGRFGCVT_323803488900X_jpeg.exe — 7768fd2812ceff05db8f969a7bed1de5615bfc5a — **Win32/Sednit.ORQ**
2. C:\Users\public\Pictures\scanPassport.jpg — da70c54a8b9fd236793bb2ab3f8a50e6cd37e2df
3. C:\Users\Public\Documents\AcrobatReader.{exe,txt} — a225d457c3396e647ffc710cd1edd4c74dc57152 — **MSIL/Sednit.D**
4. C:\Users\Public\Videos\audev.txt — a659a765536d2099ecbde988d6763028ff92752e — **Win32/Sednit.CH**
5. %TMP%\Indy0037C632.tmp — 20954fe36388ae8b1174424c8e4996ea2689f747 — **Win32/TrojanDownloader.Sednit.CMR**
6. %TMP%\Indy01863A21.tmp — e0d8829d2e76e9bb02e3b375981181ae02462c43 — **Win32/TrojanDownloader.Sednit.CMQ**

Email

carl.dolzhek17@post.cz
shinina.lezh@post.cz
P0tr4h4s7a@post.cz
carl.dolzhek17@post.cz
sym777.g@post.cz
kae.mezhnosh@post.cz
tomasso25@ambcomission.com
kevin30@ambcomission.com
salah444@ambcomission.com
karakos3232@seznam.cz
rishit333@ambcomission.com
antony.miloshevich128@seznam.cz