Исследование направленной атаки

Недавно мы проанализировали четыре вредоносных инструмента, которые использовались в направленных атаках на пользователей Тайваня и Вьетнама. С использованием системы телеметрии ESET мы зафиксировали, что это вредоносное ПО доставлялось пользователям через кампании по распространению фишинговых сообщений. Один из файлов вредоносной программы был доставлен в систему пользователя через почтовый веб-интерфейс вьетнамского государственного учреждения. Злоумышленники использовали специальные фишинговые почтовые сообщения, которые содержали убедительный текст, а также специальные фальшивые документы для заманивания пользователя и усиления эффекта атаки.

По результатам этого исследования нам удалось установить следующие факты:

- Один и тот же компонент вредоносного кода несколько раз использовался в этих кампаниях.
- Атаки требовали ручного вмешательства злоумышленника (оператор).
- В атаках обнаружены следы известной группы APT1 (aka Comment crew).
- Для первоначального вектора атаки, т. е. для установки вредоносного кода, использовались фишинговые сообщения, которые якобы содержали важные документы.
- Плохая подготовка злоумышленников: опечатки в конфигурации, наивная реализация криптографических алгоритмов, недостаточная практика в написании кода.
- Вредоносные программы с четко заданным поведением: одна из угроз не сохраняет свое присутствие после перезагрузки, а другая не выполняет никаких вредоносных функций до перезагрузки.

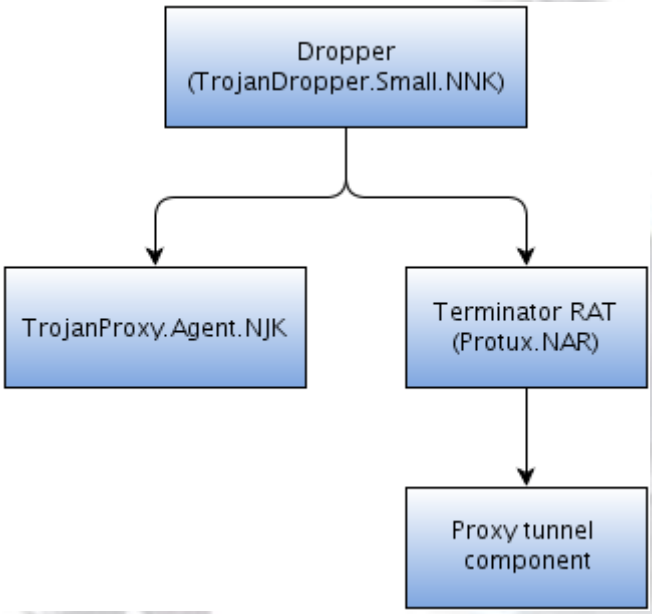

Рис. Анализируемые угрозы

На рисунке выше видно, что первоначальный дроппер осуществляет установку в систему двух других угроз, которые называются **Agent.NJK** и **Terminator RAT**.

Далее, зная характеристики этого дроппера, нам не составило труда найти в нашей коллекции другие фрагменты связанного с ним вредоносного кода. Ниже в таблице дана их характеристика. Видно, что злоумышленники специально выбрали для них маскировку файлов Office.

Угроза	Имя файла	Имя файла [перевод]
Win32/TrojanProxy.Agent.NJK	Bao cao ket qua.doc**[137 spaces]**.exe	Результаты отчета (вьетнамский)
Terminator RAT (Win32/Protux.NAR)	檢驗報告.exe	Отчет о проверке (китайский)

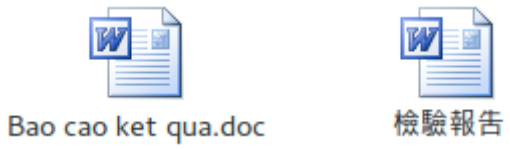

Рис. Замаскированные под документы MS Word исполняемые файлы угрозы.

В процессе запуска эти дропперы будут расшифровывать свои параметры конфигурации с использованием простого XOR с однобайтовым ключом. Псевдокод этого алгоритма на python представлен ниже. Данные конфигурации хранятся в конце последней секции исполняемого файла и занимают 32 байта. Внутри этих данных хранится контрольная сумма, некоторые смещения до важных частей файла, а также их длины; все это организовано в структуру формат которой представлен ниже. Поле контрольной суммы из структуры используется для сравнения со значением, которое жестко зашито в код дроппера. Таким образом вредоносный код проверяет целостность расшифрованных данных.

```
def xor_decrypt(ciphertext, key):
    for i in range(len(ciphertext)):
        c = ciphertext[i]
        if c:
            if c != 0xff:
                c ^= key
            if (c and c != 0xff):
                ciphertext[i] = c
    return ciphertext
```

```
struct hidden_segment_data
{
  int checksum;
  char delimiter;
  char unused[3];
  int pe_file_offset;
  int pe_file_size;
  char unused[4];
  int doc_file_offset;
  int doc_file_size;
  char xorkey;
  char unused[2];
  char last;
};
```

Рис. Псевдокод алгоритма шифрования и формат служебной структуры.

Первое, что делает дроппер – извлекает из себя два файла, один из них представляет исполняемый файл, а другой является фальшивым документом Word, который будет показан пользователю. Оба файла помещаются во временную папку пользователя, при этом каждый из них расшифровывается с использованием простого XOR, который был приведен выше. После извлечения файлов дроппер запускает исполняемый файл, перед этим копируя свое тело в другой каталог, а затем исполняет его оттуда с

дополнительными параметрами командной строки. Параметры представляют собой полный путь к файлу оригинального дроппера и путь к документу Word.

Описание файла	Имя файла
Полезная нагрузка	%TEMP%\~hCb58.tmp
Документ Word	%TEMP%\~hC29f.doc
Копия дроппера	%TEMP%\~hCb37.tmp

Запустившийся из новой директории дроппер дожидается завершения процесса своего родителя, после чего удаляет его файл, а также перемещает файл фишингового документа в каталог с временными файлами. На последней стадии он вызывает функцию *ShellExecuteW* для открытия этого офисного документа. Документ открывается соответствующей программой, которая указана в реестре для открытия .doc-файлов. Подобная функция выполняется с целью имитации вызова операции, которую ожидает пользователь, запустивший на исполнение файл дроппера замаскированного под документ.

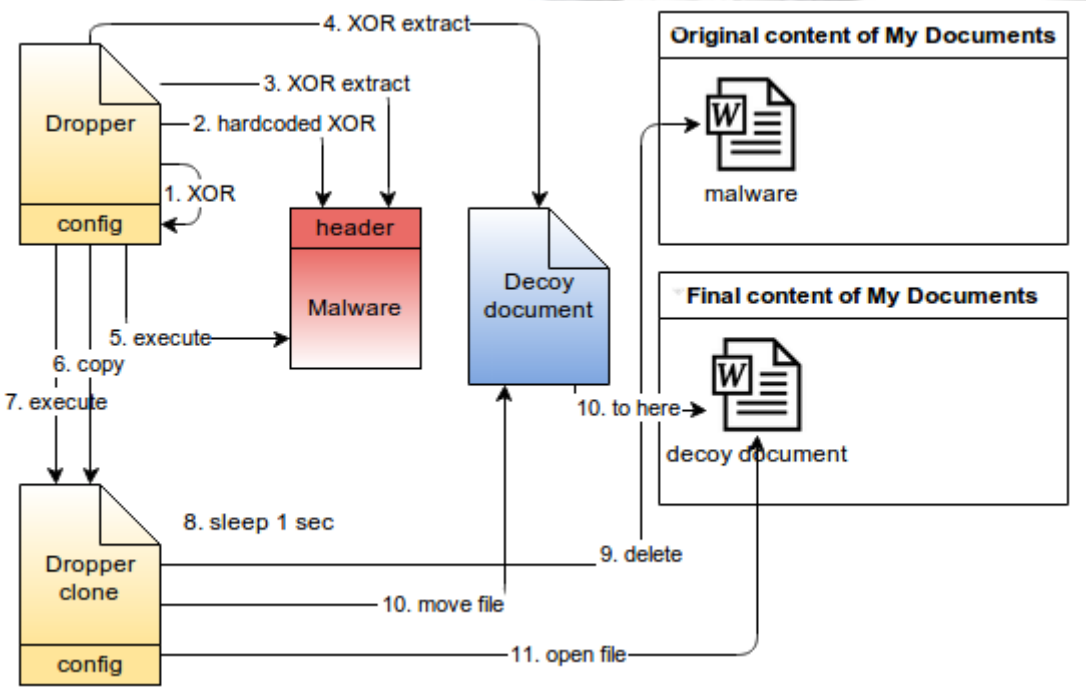

Рис. Выполняемые дроппером операции.

Такой, с первого взгляда, простой подход для обмана пользователей является довольно эффективным. Все процессы выполняемые дроппером происходят незаметно для пользователя, после чего для него открывается запрашиваемый документ. В случае, если вредоносная программа выполняет свои первоначальные действия с минимальной задержкой, то пользователь может вообще ничего не заметить. Подобная тактика компрометации пользователей использовалась группой APT1, о которой сообщали Mandiant. Заметьте, в таком случае злоумышленникам не нужно прибегать к услугам эксплойтов для установки вредоносного кода.

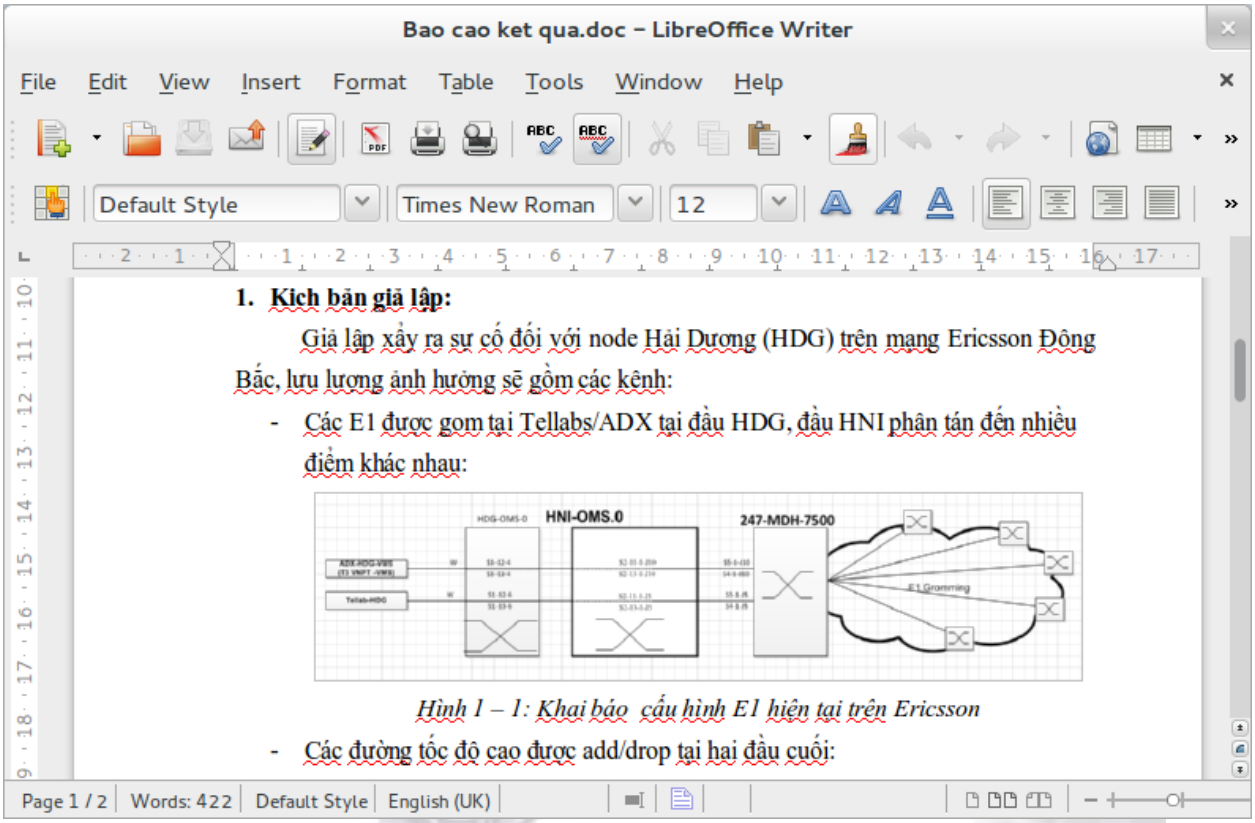

Рис. Фишинговый документ на вьетнамском языке.

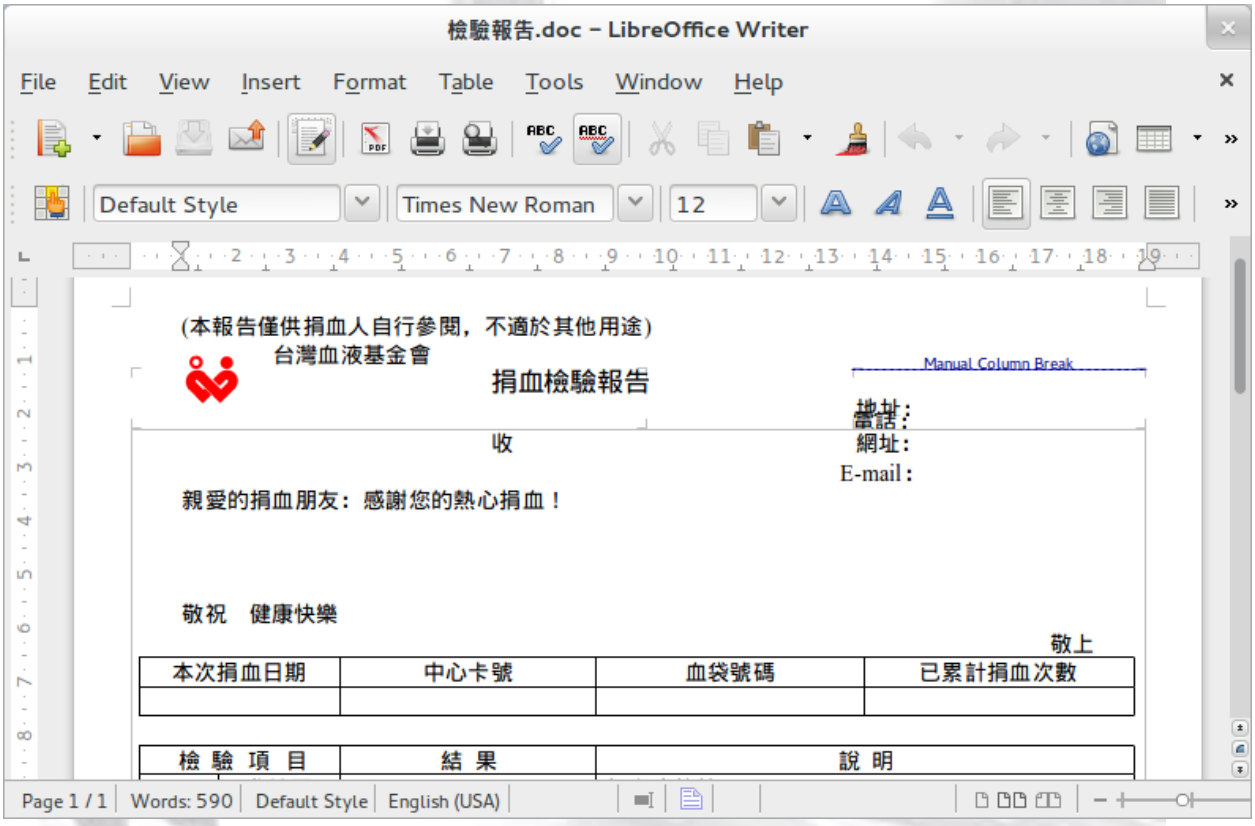

Рис. Схожий документ для пользователей, проживающих на Тайване.

Дроппер использует два различных метода для сокрытия вызовов API-функций: содержит собственную реализацию *GetProcAddress* с открытыми строками имен

функций в коде, а также использует стандартную реализацию функции с зашифрованными именами. При анализе кода видно, что он не написан на языке высокого уровня: некоторые объекты используются с максимальными правами доступа, copy-paste различных частей кода в его теле.

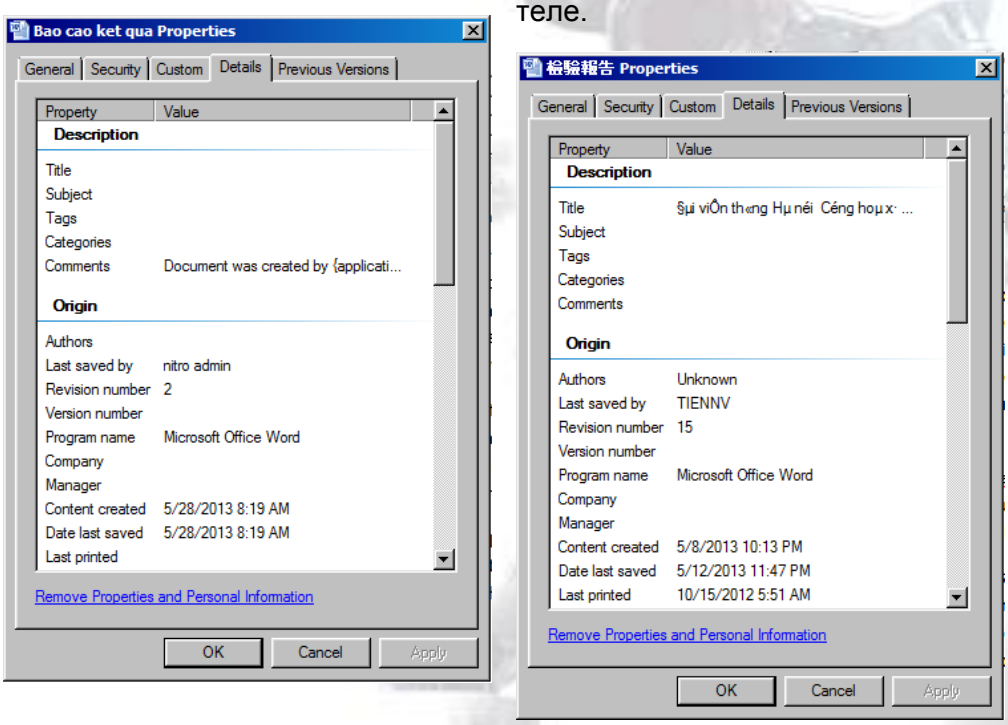

Рис. Метаданные вьетнамского и тайваньского документов.

Win32/TrojanProxy.Agent.NJK

Первый исполняемый файл, который мы проанализировали, обнаруживается нашими продуктами как **Win32/TrojanProxy.Agent.NJK**. Эта вредоносная программа написана на Visual C++ и способна взаимодействовать с удаленными C&C серверами через HTTP протокол. В анализируемой модификации было обнаружено использование трех адресов управляющих серверов, которые используют один и тот же доменный адрес vietnam.vnptnet.info, но с подключением к разным портам (80, 443 и 5050).

Работа с C&C организована через цикл с 15-минутной задержкой, в процессе которого вредоносный код пытается контактировать с сервером используя три вышеуказанных порта. Интересно, что угроза не предпринимает никаких действий для обеспечения своей выживаемости после перезагрузки, т. е. на скомпрометированном компьютере она будет активна только до завершения работы или перезагрузки. Сам вредоносный файл не содержит какой-либо обфускации используемых им строк.

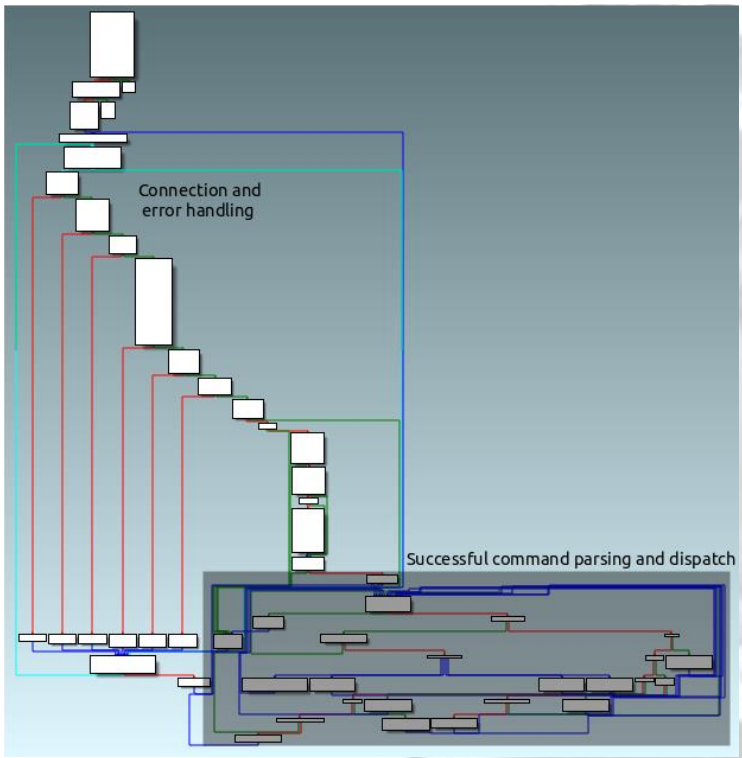

Рис. Основной цикл работы вредоносного кода.

При взаимодействии с управляющим сервером вредоносный код отправляет некоторую информацию о системе через GET запрос HTTP протокола, используя для этого специальную строку для поля User-Agent. Пользовательские данные составляют 105 байт и будут отправлены в компоненте строки запроса GET метода как hex-последовательность. Эти данные содержат следующую информацию: идентификатор вредоносной кампании; внутренний IP-адрес хоста; имя компьютера; Windows Version ID; название учетной записи, под которой исполняется процесс вредоносного кода. Странным является и тот факт, что к передаваемым данным не применяется шифрование, они передаются в открытом виде.

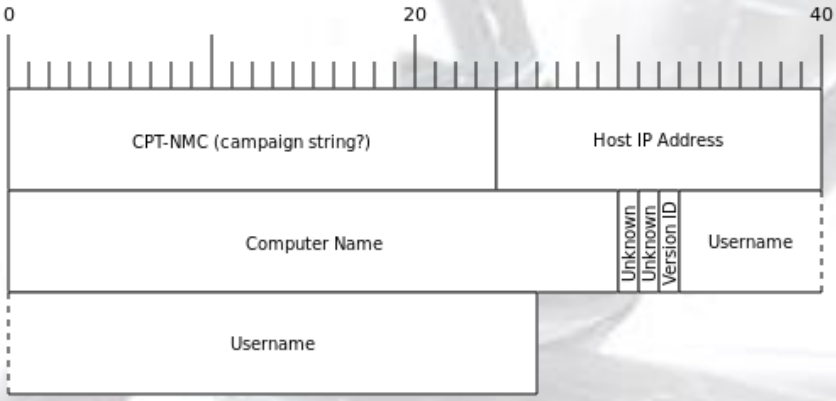

Рис. Формат данных, отправляемых вредоносным кодом на удаленный сервер.

Передаваемые данные вместе с запросом выглядят так.

```
GET
/4350542D4E4D43000000000000000000000000000000000003139322E3136382E3136362E3134
3500555345522D393839424331333535350000000000000000000000000000000000001077573657
2000000000000000000000000000000000000000000000000000000000000000 HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0; .NET CLR
1.1.9527)
Host: vietnam.vnptnet.info
```

Рис. Пример GET-запроса с передаваемыми данными.

На этот запрос сервер ответит обычным заголовком, за исключением добавленного к нему поля Accept со значением «x-wav/y-img». Вредоносная программа не примет ответ от сервера, если в заголовке ответа не будет этого поля. Заметьте, что поле Accept обычно используется в заголовках запросов от клиента в обычных HTTP запросах, но в этом случае все наоборот, сервер должен ответить с заголовком, который содержит данное поле. Мы заметили, что посылаемые C&C команды всегда имеют размер 796 байт, при этом первое целочисленное значение в данных команды является ее идентификатором (command ID). Следующие команды поддерживаются вредоносным кодом.

Command ID	Описание
1000	Передаваемая командная строка исполняется ботом, а данные потоков результатов stdout и stderr отсылаются обратно управляющему серверу.
2000	Бот отвечает «\r\n\r\nRecieve KeepAlive Commond\r\n\r\n».
3004	Загрузить файл на компьютер.
3005	Передать файл на C&C. Поддерживается передача части файла по смещению, заданному в аргументе к команде.
3006	Сменить текущую директорию на указанную в команде.
3007	Установить задержку для исполнения командной строки. Передается как аргумент к использованию для *WaitForSingleObject*.
3008	Отправить управляющему серверу информацию об установленных в системе логических дисках, включая размер, свободное место, букву диска и метку.
3009	Отправить на сервер информацию о файлах в заданном каталоге. Эта информация включает в себя: имя файла, дата последней модификации, размер.
3010	Удалить заданный файл.
3011	Запустить процесс с командной строкой, которая задана как параметр к команде.

В коде Agent.NJK присутствует забавная строка «I want to go to THE GREAT WALL, inner Mongolia very much» и учетные данные для доступа к прокси-сервису somnuek.bu / 044253516. В то же время по анализу вредоносного кода видно, что эти учетные данные нигде не используются. Поиск в Google предоставил нам некоторую информацию, которая ведет к человеку с таким именем в социальной сети. У нас нет данных о том какое отношение он имеет к данной кампании.

Следует отметить, что строка CPT-NMC, которая идентифицирует саму кампанию (идентификатор кампании), отправляется на удаленный сервер в дальнейшем еще раз. CPT расшифровывается как Central Post and Telecommunications Department, это учреждение является отделением вьетнамского правительства. Также можно увидеть, что домен верхнего уровня для URL C&C сервера (vnptnet.info) очень похож на вьетнамский адрес vnpt.nv, который принадлежит тому же учреждению CPT. Вероятно

такое название было выбрано для маскировки домена в журналах систем типа Intrusion Detection System (IDS), которые фиксируют трафик, проходящий через компьютер или компьютеры в сети. Фишинговые документы, о которых мы писали в начале и которые используются для заманивания пользователей, содержат текст, в котором идет речь о системах телекоммуникации. Сами документы содержат в себе различные графики и диаграммы, пытаясь максимально возможно усыпить внимание потенциальной жертвы и убедить в том, что документ настоящий. Похоже, что эта атака была направлена на вьетнамское учреждение CPT и вьетнамские официальные лица сообщали об атаке на их инфраструктуру в этом году.

Мы наблюдали каким образом оператор взаимодействует с зараженной системой. Ниже приводятся несколько доказательств (лог) такого взаимодействия.

```
1. client <-
    command id/name: 3008/Get Drives Infos

   client ->

    label: C:
    type:  3 (DEVICE_FIXED)
    free:  7828
    total: 10228

    label: D:(8
    type:  5 (DEVICE_CDROM)
    free:  0
    total: 589

2. client <-
    command id/name: 1000/ExecuteCommandLine
    executed: netsta -ano

   client ->

3. client <-
    command id/name: 1000/ExecuteCommandLine
    executed: netstat -ano

   client ->
    ...


другие команды:

4.  set
5.  dir C:\DOCUME~1\user\recent /od
6.  dir C:\DOCUME~1\user\desktop
7.  dir c:\
```

Рис. Команды, посылаемые оператором боту Agent.NJK.

Эти выполняемые операции похожи на разведывательные, т. е. относятся к сбору данных о скомпрометированной системе: netsta(t) используется для просмотра текущих сетевых подключений, далее происходит получение информации о логических дисках в системе, просмотр переменных окружения, получение информации о расположении некоторых файлов. Опечатка в слове netsta, вместо netstat, во второй команде, свидетельствует о том, что эти команды боту подаются не автоматизированной системой, а за ними стоит физический оператор. Мы наблюдали такой подход и прежде в случае с направленной атакой Syndicasec.

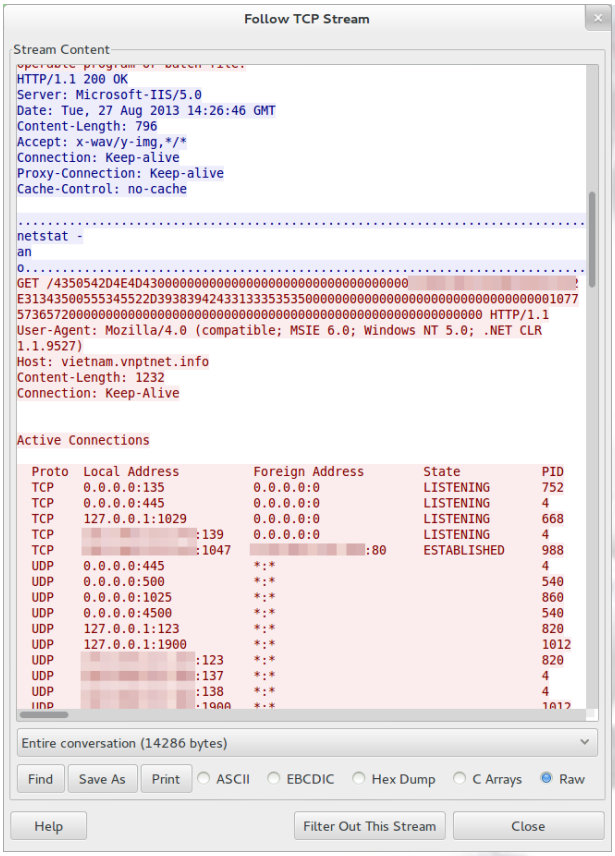

Рис. Бот передает данные о текущих сетевых подключениях в открытом виде.

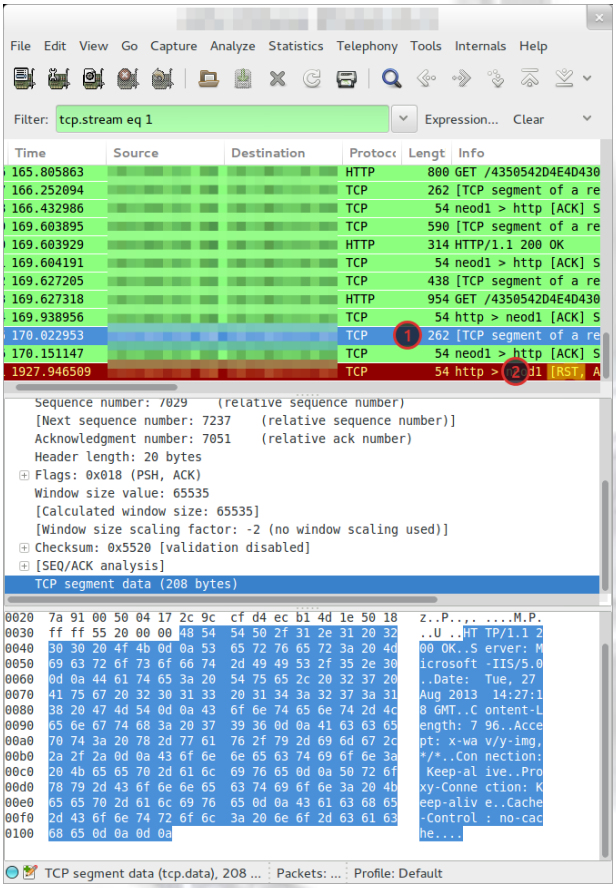

Рис. Закрытие сессии работы C&C с ботом.

На последнем скриншоте под пунктом 2 видно, что сервер разрывает соединение отправляя команду RST (reset). После отправки команды reset поведение сервера изменяется и как только его получит клиент, он в дальнейшем уже не сможет подключиться ни к одному из трех портов.

No.	Time	Source	Destination	Protoc	Lengt	Info
471	1927.946509			TCP	54	80 > 1047 [RST, ACK] Se
472	1927.948839			TCP	62	1051 > 80 [SYN] Seq=0 W
473	1928.254824			TCP	54	80 > 1051 [RST, ACK] Se
474	1928.682556			TCP	62	1051 > 80 [SYN] Seq=0 W
475	1928.990145			TCP	54	80 > 1051 [RST, ACK] Se
476	1929.448046			TCP	62	1051 > 80 [SYN] Seq=0 W
477	1929.755978			TCP	54	80 > 1051 [RST, ACK] Se
478	1929.759015			DNS	80	Standard query 0xfa6f
479	1930.087203			DNS	96	Standard query response
480	1930.103688			TCP	62	1052 > 443 [SYN] Seq=0
481	1930.411510			TCP	54	443 > 1052 [RST, ACK] S
482	1930.869968			TCP	62	1052 > 443 [SYN] Seq=0
483	1931.176899			TCP	54	443 > 1052 [RST, ACK] S
484	1931.635589			TCP	62	1052 > 443 [SYN] Seq=0
485	1931.945291			TCP	54	443 > 1052 [RST, ACK] S
486	1931.946655			TCP	62	1053 > 5050 [SYN] Seq=0
487	1932.253739			TCP	54	5050 > 1053 [RST, ACK]
488	1932.729282			TCP	62	1053 > 5050 [SYN] Seq=0
489	1933.038712			TCP	54	5050 > 1053 [RST, ACK]
490	1933.385577			TCP	62	1053 > 5050 [SYN] Seq=0
491	1933.695482			TCP	54	5050 > 1053 [RST, ACK]
492	1933.696478			TCP	62	1054 > 80 [SYN] Seq=0 W
493	1934.005039			TCP	54	80 > 1054 [RST, ACK] Se

Рис. Сервер отказывает в подключении клиенту, если перед этим им была отправлена команда сброса TCP-соединения.

Отсутствие во вредоносном коде механизмов выживаемости после перезагрузки укрепляет нашу гипотезу о направленности такой кибератаки, поскольку нападающие заинтересованы в том, чтобы оставить как можно меньше следов в скомпрометированной системе. Обычной практикой при использовании такого вредоносного кода было бы выяснить потенциальных жертв в организации, отправить фишинговое письмо, подождать ответа от бота и в конце исследовать скомпрометированные компьютеры через него. В нашем же случае бот поддерживает команды на загрузку других исполняемых файлов в систему (3004) и их последующую установку (3011), поэтому может быть использован для выполнения несколько иных действий.

Terminator RAT (aka FAKEM RAT)

Обнаруживается нашими антивирусными продуктами как **Win32/Protux.NAR**. На момент анализа части этого вредоносного ПО, которое реализует шифрование и отвечает за работу с C&C, мы обнаружили, что эта угроза анализировалась ранее malware.lu и Trend Micro с названиями Terminator RAT и FAKEM RAT. В то же время анализируемые нами образцы отличались от тех, с которыми имели дело эти компании. Еще нужно отметить, что компания FireEye также выпустила свой анализ одной из модификаций этого вредоносного ПО.

По сравнению с **Agent.NJK**, эта угроза является более сложной. Во-первых, данные конфигурации и строки зашифрованы с использованием несколько модифицированной версии алгоритма XTEA. XTEA использует 128-битный ключ и работает с блоками длиной 64 бита.

```
; signed int __cdecl xtea_ecb(void *key, char *ciphertext, signed int len)
xtea_ecb proc near

key= dword ptr  4
ciphertext= dword ptr  8
len= dword ptr  0Ch

mov     eax, [esp+len]
push    esi
mov     esi, [esp+4+ciphertext]
sar     eax, 3
test    eax, eax
jle     short loc_402A96
```

```
push    edi
mov     edi, eax
```

```
loc_402A83:
push    esi
push    [esp+0Ch+key]
call    xtea
pop     ecx
add     esi, 8
dec     edi
pop     ecx
jnz     short loc_402A83
```

```
pop     edi
```

```
loc_402A96:
pop     esi
retn
xtea_ecb endp
```

Рис. XTEA как оно реализовано в исследуемой модификации Terminator RAT.

Реализация XTEA довольно наивна, поскольку использует худший, блочный режим шифрования, как показано на скриншоте ниже. 64-битные блоки, состоящие из нулей, всегда шифруются в один и тот же зашифрованный текст.

```
00404198  23 31 37 82 C2 C3 E1 D3  23 31 37 82 C2 C3 E1 D3  #17.....#17.....
004041A8  23 31 37 82 C2 C3 E1 D3  23 31 37 82 C2 C3 E1 D3  #17.....#17.....
```

Рис. Пример зашифрованного текста, который начинается по адресу 0x404198 и содержит очевидный шаблон.

```
00404198  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
004041A8  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
```

Рис. Расшифрованный текст по тому же адресу.

```
00404148  9A C9 BC 4E 82 9D DB 60  3F CA 65 D4 F5 B7 B1 F0  ...N...`?.e.....
00404158  9E D9 66 08 C0 60 B0 F1  23 31 37 82 C2 C3 E1 D3  ..f..`..#17.....
00404168  23 31 37 82 C2 C3 E1 D3  50 A5 9A 58 81 04 E0 66  #17.....P..X...f
00404178  05 50 32 A8 63 CE C9 56  ED BC 10 4D ED 23 4D 26  .P2.c..V...M.#M&
00404188  88 3B 4A FA 7A 52 B0 79  56 CA AC 72 10 98 10 5B  .;J.zR.yV..r...[
00404198  9A 37 04 90 73 C9 CE 07  94 19 B0 B8 13 91 F7 CE  .7..s...........
004041A8  ED 37 36 05 CB 07 13 EB  23 31 37 82 C2 C3 E1 D3  .76.....#17.....
004041B8  23 31 37 82 C2 C3 E1 D3  23 31 37 82 C2 C3 E1 D3  #17.....#17.....
004041C8  23 31 37 82 C2 C3 E1 D3  23 31 37 82 C2 C3 E1 D3  #17.....#17.....
004041D8  FA BD F5 93 DD AA 8F 92  54 1D 1C E5 99 3D F4 E9  ........T....=..
004041E8  23 31 37 82 C2 C3 E1 D3  23 31 37 82 C2 C3 E1 D3  #17.....#17.....
004041F8  23 31 37 82 C2 C3 E1 D3  23 31 37 82 C2 C3 E1 D3  #17.....#17.....
00404208  23 31 37 82 C2 C3 E1 D3  00 00 00 00 00 00 00 00  #17.............
00404218  28 23 00 00 82 23 00 00  9A C9 BC 4E 82 9D DB 00  (#...#.....N....
00404228  12 08 39 0D B9 70 45 55  F4 EB 7C D8 69 E3 1F F5  ..9..pEU..|.i...
00404238  2B 98 38 1D 2C 83 AB E6  CA 89 F1 77 07 1B 9D CE  +.8.,......w....
00404248  00 3D 44 0B 6F FB 47 CD  33 2A E8 C4 37 A1 F8 AD  .=D.o.G.3*..7...
00404258  3E EF 34 81 94 35 B8 93  B7 B4 3F 99 A6 69 38 A5  >.4..5....?..i8.
00404268  11 43 02 19 2A D1 00 BD  AA BA 04 68 1B 00 D7 54  .C..*......h...T
00404278  3B 7A C9 04 CF 4C 55 02  C2 60 66 7B 93 BC C4 79  ;z...LU..`f{...y
00404288  89 89 25 67 12 85 A3 D8  59 29 9B D7 42 46 EC 22  ..%g....Y)..BF."
00404298  D9 52 58 8B 20 7E 43 12  74 52 38 EC 26 8C 23 00  .RX. ~C.tR8.&.#.
004042A8  B5 1B DC 7C 90 37 65 13  DF 0B FB E4 24 E3 98 31  ...|.7e.....$..1
004042B8  76 99 C5 AA CE 6D C0 89  B0 66 14 8E 1A 6A 31 8C  v....m...f...j1.
004042C8  41 33 F5 2B 29 7B ED D1  23 31 37 82 C2 C3 E1 D3  A3.+){..#17.....
004042D8  23 31 37 82 C2 C3 E1 D3  23 31 37 82 C2 C3 E1 D3  #17.....#17.....
004042E8  23 31 37 82 C2 C3 E1 D3  00 00 00 00 00 00 00 00  #17.............
004042F8  00 00 00 00 00 00 00 00  00 00 00 00 32 00 00 00  ............2...
00404308  69 00 00 00 74 00 00 00  7E 37 00 00 5C 00 00 00  i...t...~7..\...
00404318  41 70 70 00 55 73 65 72  73 5C 00 00 67 72 61 6D  App.Users\..gram
00404328  73 5C 53 74 61 00 00 00  66 74 5C 57 69 6E 64 6F  s\Sta...ft\Windo
00404338  77 73 5C 53 74 61 00 00  72 74 75 70 00 00 00 00  ws\Sta..rtup....
00404348  5C 41 70 70 44 61 74 61  5C 52 6F 00 72 74 20 4D  \AppData\Ro.rt M
00404358  65 6E 75 5C 50 72 6F 00  61 6D 69 6E 67 5C 4D 69  enu\Pro.aming\Mi
00404368  63 72 6F 73 6F 00 00 00  2E 2E 00 00 2E 00 00 00  croso...........
00404378  6E 69 00 00 2E 69 00 00  5C 2A 2E 2A 00 00 00 00  ni...i..\*.*....
```

Рис. Часть конфигурации вредоносного кода в зашифрованном виде.

(1) ключ XTEA, (2) два номера портов (9000, 9090) и некоторые другие данные, (3) обфусцированные промежуточными нулями различные строки.

```
9A C9 BC 4E 82 9D DB 60  3F CA 65 D4 F5 B7 B1 F0  (1) N...`?.e.....
18 17 40 1F 32 30 31 39  00 00 00 00 00 00 00 00  .@.2019.........
00 00 00 00 00 00 00 00  73 76 63 68 6F 73 74 20  ........svchost
2E 65 78 65 00 00 00 00  00 00 00 00 77 69 6E (2) .exe........wins
6C 6F 67 6F 6E 2E 65 78  65 00 00 00 00 00 00 00  logon.exe.......
63 61 74 6C 6F 76 65 72  73 2E 32 35 75 2E 63 6F  catlovers.25u.co
6D 20 00 00 00 00 00 00  00 00 00 00 00 00 00 00  m ..............
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ..... (3) ......
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
64 72 79 62 6F 78 73 2E  34 64 71 2E 63 6F 6D 00  dryboxs.4dq.com.
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
28 23 00 00 82 23 00 00  9A C9 BC 4E 82 9D DB 00  (#...#.....N....
41 63 63 65 6C 6F 72 61  74 6F 72 00 53 4F 46 (4) Accelerator.SOFT
57 41 52 45 5C 4D 69 63  72 6F 73 6F 66 74 5C    WARE\Microsoft\W
69 6E 64 6F 77 73 5C 43  75 72 72 65 6E 74 56 65  indows\CurrentVe
72 73 69 6F 6E 5C 45 78  70 6C 6F 72 65 72 5C 53  rsion\Explorer\S
68 65 6C 6C 20 46 6F 6C  64 65 72 73 00 00 00 00  hell Folders....
00 00 53 4F 46 54 57 41  52 45 5C 4D 69 63 72 6F  ..SOFTWARE\Micro
73 6F 66 74 5C 57 69 6E  64 6F 77 73 5C 43 75 72  soft\Windows\Cur
72 65 6E 74 56 65 72 73  69 6F 6E 5C 45 78 70 6C  rentVersion\Expl
6F 72 65 72 5C 55 73 65  72 20 53 68 65 6C 6C 20  orer\User Shell
46 6F 6C 64 65 72 73 00  00 00 00 00 00 53 74 61  Folders......Sta
72 74 75 70 00 00 00 00  00 00 00 00 00 00 00 00  rtup.......... (5)
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00  00 00 00 00 32 00 00 00  ............2...
69 00 00 00 74 00 00 00  7E 37 00 00 5C 00 00 00  i...t...~7..\...
41 70 70 00 55 73 65 72  73 5C 00 00 67 72 61 6D  App.Users\..gram
73 5C 53 74 61 00 00 00  66 74 5C 57 69 6E 64 6F  s\Sta...ft\Windo
77 73 5C 53 74 61 00 00  72 74 75 70 00 00 00 00  ws\Sta..rtup....
5C 41 70 70 44 61 74 61  5C 52 6F 00 72 74 20 4D  \AppData\Ro.rt M
65 6E 75 5C 50 72 6F 00  61 6D 69 6E 67 5C 4D 69  enu\Pro.aming\Mi
63 72 6F 73 6F 00 00 00  2E 2E 00 00 2E 00 00 00  croso..........
6E 69 00 00 2E 69 00 00  5C 2A 2E 2A 00 00 00 00  ni...i..\*.*....
65 00 00 00 63 6C 74 2E  65 78 00 00 5C 77 75 61  e...clt.ex..\wua
75 00 00 00 61 6D 65 65  00 00 00 00 61 74 61 00  u...amee....ata.
41 70 70 44 00 00 00 00  01 00 00 00 00 00 00 00  AppD............
```

Рис. Расшифрованные данные конфигурации.

(1) директория, в которую была установлена вредоносная программа (располагается в %APPDATA%), (2) имена файлов, которые даны установленным компонентам, (3) домен C&C, (4) название директории в секции ресурсов, где хранится полезная нагрузка, (5) раздел реестра, который используется для запуска после перезагрузки (обеспечивает выживаемость).

После старта **Protux.NAR** динамически заполняет переменные адресов функций в теле своего файла (адреса функций, которые не декларировались в таблице импорта). Для этого используется своя реализация функции *GetProcAddress*, как и в случае с **TrojanProxy.Agent.NJK**. Строки, которые используются для получения адресов функций, не зашифрованы и хранятся в теле вредоносного кода в открытом виде.

Далее Win32/Protux.NAR изменяет в системном реестре значение переменной, которая задает путь к Startup Folder (директория «Автозагрузка», используется для организации автозапуска) на новое значение «%APP_DATA%\2019», копирует существующие файлы по старому пути в новое расположение и обеспечивает перемещение своего тела в то же новое расположение («%APP_DATA%\2019) под именем «svchost .exe» через *MoveFileEx* с флагом MOVEFILE_DELAY_UNTIL_REBOOT. Кроме этого, он достает из своего тела еще один PE-файл и копирует его в каталог под именем «winslogon.ini», задавая его отложенное

переименование через MOVEFILE_DELAY_UNTIL_REBOOT в файл «winslogon.exe» (используется как компонент для организации proxy tunnel).

Рис. Поток выполнения кода, который демонстрирует вышеописанные операции.

Как можно увидеть вредоносный код в значительной степени опирается на флаг MOVEFILE_DELAY_UNTIL_REBOOT функции *MoveFileEx*. Этот способ используется как механизм ОС для перемещения исполняемых файлов, которые запущены на момент вызова функции. Также этот способ помогает предотвратить активацию эвристики антивирусного продукта или прочих технологий обнаружения вредоносного кода.

Для того, чтобы избежать своего обнаружения со стороны security-продуктов при последующих запусках он будет пытаться копировать свой файл в каталог со временными файлами (GetTempPath() + "~7ti2") и записывать несколько случайных байт в конец файла. После этих операций файл осуществляет свою перезапись с использованием функции *MoveFileEx* и флагов MOVEFILE_DELAY_UNTIL_REBOOT, MOVEFILE_REPLACE_EXISTING.

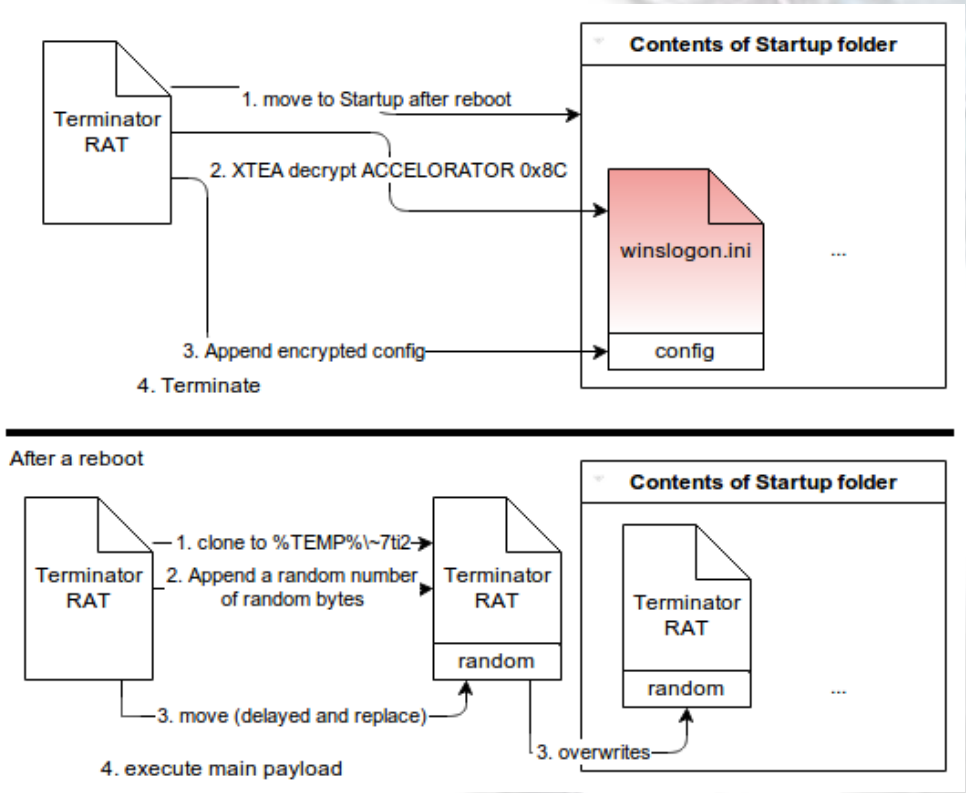

Рис. Более наглядное представление операций вредоносного кода, описанных выше.

После перезагрузки, когда Windows запускает каждый исполняемый файл из каталога Автозагрузки, будут исполнены два файла вредоносной программы «svchost .exe» (основной компонент) и «winslogon.exe» (компонент proxy tunnel). Основной компонент выполняет расшифровку конфигурации и строк, а также запускает вспомогательный поток, который исполняет разные функции в зависимости от того, из какой директории оригинальный файл был запущен.

Далее вредоносный код выделяет блок памяти, копирует элемент секции ресурсов с идентификатором 0x8A в директории (ресурсов) под именем ACCELORATOR в выделенный блок памяти и расшифровывает его через байтовый XOR с ключом 0x32. Отметим, что название указанной директории ресурсов именно ACCELORATOR, а не ACCELERATOR. Эти расшифрованные данные представляют собой исполняемый код, который может исполняться вне зависимости от своего первоначального расположения (position-independent code). Перед непосредственной передачей управления на этот блок памяти, вредоносный код получает текущий IP-адрес компьютера, шифрует через XOR его и копирует вместе с номером порта 8000 по заданному смещению в коде. Операции шифрования с использованием XOR выполняются с использованием того же статического ключа 0x32.

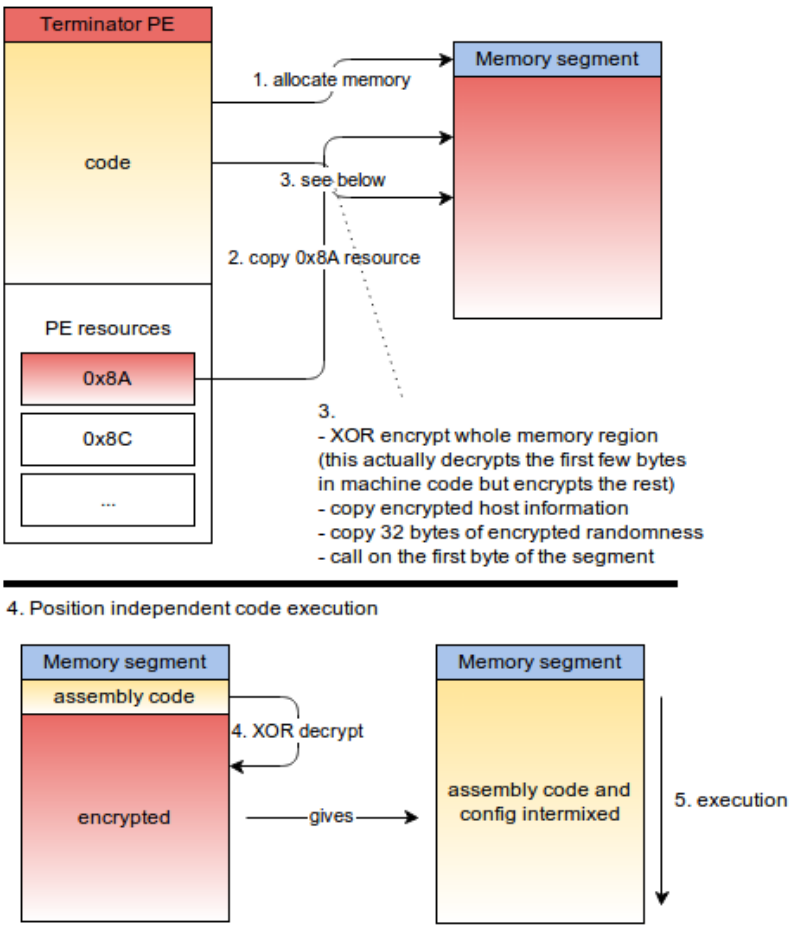

Рис. Процесс загрузки и исполнения кода из секции ресурсов.

Вредоносный код, который был скопирован в блок памяти, осуществляет нестандартное использование регистров, вероятно это указывает на то, что он был написан на ассемблере. Например, для заполнения адресов импортируемых динамически функций используется своя реализация *GetProcAddress* с заранее определенной таблицей хэшей ROR для каждой из имен функций. Такой способ редко используется для программ, скомпилированных на более высоком языке программирования. По этой ссылке можно найти больше информации об использовании ассемблера.

```
dw 22h
dw 0Dh
db 'Kernel32.dll',0
dd 2099AD4Fh
dd 0FF0D6657h
dd 6D4D9176h
dd 94E43293h
dd 6BA6BCC9h
dd 7EB0FB1h
dd 98E63979h
dd 4FD18963h
dd 60CC8F79h
dd 402DC1ABh
dd 80AE9074h
dd 0B4FFAFEDh
dd 8E6902B2h
dd 0E72D6895h
dd 12F461BBh
dd 0EC496A9Eh
dd 95620529h
dd 8FFB3B6Eh
dd 3A2FE6BBh
dd 0BBAFDF85h
dd 0F8262C81h
dd 0C917432h
dd 867D1F70h
dd 90B91F70h
dd 7A8D6770h
dd 30FD5A80h
dd 7EAD1F70h
dd 0F39E8CD6h
dd 130F36B2h
dd 35D1662Ch
dd 0F0B986D7h
dd 0CB9765A0h
dd 1A22F51h
dd 1EDE5967h
```

```
dw 22h                          ; see fct_loading_r
dw 0Dh
db 'Kernel32.dll',0
dd offset kernel32_GetOEMCP
dd offset kernel32_CloseHandle
dd offset kernel32_CreateEventA
dd offset kernel32_CreateFileA
dd offset kernel32_CreateProcessA
dd offset kernel32_CopyFileA
dd offset kernel32_DeleteFileA
dd offset kernel32_ExitProcess
dd offset kernel32_ExitThread
dd offset kernel32_GetComputerNameA
dd offset kernel32_GetProcessHeap
dd offset kernel32_GetModuleFileNameA
dd offset kernel32_GetSystemDirectoryA
dd offset kernel32_GetShortPathNameA
dd offset ntdll_RtlGetLastWin32Error ; repl
dd offset kernel32_GetEnvironmentVariableA
dd offset kernel32_GetVersion
dd offset kernel32_GetCurrentThread
dd offset kernel32_GetCurrentProcess
dd offset kernel32_GetProcAddress
dd offset ntdll_RtlAllocateHeap ; replaced
dd offset kernel32_LoadLibraryA
dd offset kernel32_lstrcat
dd offset kernel32_lstrcpy
dd offset kernel32_lstrlen
dd offset kernel32_lstrcmpi
dd offset kernel32_lstrcmp
dd offset kernel32_ResumeThread
dd offset kernel32_ReadFile
dd offset kernel32_SetPriorityClass
dd offset kernel32_SetThreadPriority
dd offset kernel32_Sleep
dd offset kernel32_WinExec
dd offset kernel32_VirtualAlloc
```

Рис. Таблица хэшей импортируемых функций и их соответствующие названия.

Следующим действием, которое выполняет Protux.NAR, является создание события с именем «sxX5{c4», что позволяет ему фиксировать запуск вредоносного кода для предотвращения активности нескольких его копий. Организация работы с управляющими C&C серверами организована через три домена, которые опрашиваются с 30-ти секундным таймаутом в цикле. Два домена располагаются в данных конфигурации, которые зашифрованы с использованием XTEA, как было показано на скриншоте выше. Третий адрес представляет собой пару: IP скомпрометированной системы и порт 8000 (используется для операций proxy tunneling). Как только соединение с C&C установлено, вредоносный код отправляет туда информацию о скомпрометированной системе в пакете размером 1024 байта. Формат такого пакета представлен ниже.

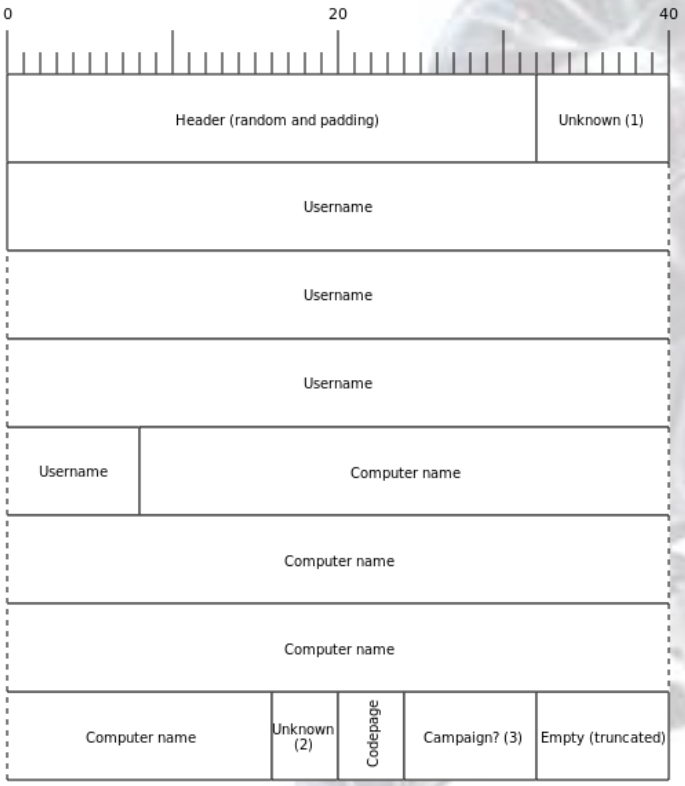

Рис. Отправляемые ботом на удаленный сервер данные о скомпрометированном компьютере.

Основными полями в пакете являются Username и Computer name, размер каждого из которых составляет 128 байт. Другое интересное поле – текущая кодовая страница (Codepage), используемая в системе, размером 4 байта. В пакете также присутствуют еще три целочисленных поля: два из них принимают значения 0x130, 0 (1), а другое принимает значение 0x30005 (2). Эти значения совпадают со значениями аналогичных полей из образцов вредоносного кода, которые анализировались FireEye. Поле 3 содержит строку и, возможно, является идентификатором кампании (campaign ID).

Взаимодействие с C&C зашифровано с использованием простой схемы: к каждому байту из потока применяются байты ключа XOR, а затем применяется операция циклического сдвига вправо (ROR) на три позиции. Ключ является статическим и выглядит как «YHCRA». Алгоритм представлен ниже.

```
def encrypt(pt):
    key = "ARCHY"[::-1]
    ct = ""
    for c in pt:
        p = ord(c)
        for k in key:
            p = p ^ ord(k)
            p = ror(p, 3)
        ct += chr(p)
    return ct
```

Рис. Шифрование при работе с управляющим сервером, используемое в Terminator RAT.

Пакет, представляющий ответ сервера, содержит идентификатор команды в первом четырехбайтовом поле. Вредоносный код использует следующие команды.

ID команды	Описание команды
0x211	Исполнить действие, передаваемое в данных команды
0x212	Переподключиться для приема данных
0x213	Уснуть, закрыть сокет и подключиться заново
0x214	Выйти

Нашим коллегам из Trend Micro удалось зафиксировать сам факт атаки и какие вспомогательные операции управляющий сервер передает боту по команде 0x211. Речь идет о выполнении операций с использованием: командной строки, файлового менеджера, диспетчера процессов, редактора реестра, модуля захвата скриншотов, модуля кражи паролей и модуля загрузки файлов. Как видно, спектр выполняемых действий весьма обширен.

Нами были обнаружены следующие домены C&C в этом вредоносном коде.

Домен	**IP**	**Порт**
"catlovers.25u.com " (1)	отсутствует преобразование	9000
dryboxs.4dq.com	123.51.208.142	9090
localhost	depends (2)	8000

Как видно выше в таблице, домен (1) содержит пробел в конце, это значит, что служба DNS не сможет преобразовать такой домен в соответствующий ему IP-адрес. Домен (2), как мы отмечали выше, был получен с использованием функций *gethostname* и *gethostbyname*. Домены 25u.com и 4dq.com управляются через сервис динамического DNS changeip.com, расположенного в США. Адрес 123.51.208.142 принадлежит Тайваню.

В таблице ниже отмечены различия между исследованными модификациями вредоносной программы, упоминаемыми антивирусными лабораториями.

	Trend Micro	FireEye	ESET
ID кампании	Не указано	zjz1020	wet
Способ распространения	Документы Word или Excel с эксплойтами	Документы Word или Excel с эксплойтами	Методы социальной инженерии
Выживаемость	Параметр реестра Run	Модификация расположения директории Автозагрузки	Модификация расположения директории Автозагрузки
Ключ XTEA	Не используется	0x3c78…	0x9ac9…
Сетевой трафик	Поддельный заголовок в первых 32-х байтах	Повторяющийся шаблон в первых 32-х байтах	Произвольные байты в первых 32-х байтах
Proxy tunnel	Не указано	Компонент для несанкционированной передачи данных (exfiltration) через корпоративный прокси	Компонент для несанкционированной передачи данных (exfiltration) через корпоративный прокси
Имя компонента прокси	Отсутствует	sss.exe	winlogon.ini, затем winnlogon.exe
C&C	• vcvcvcvc.dyndns.org • zjhao.dtdns.net • avira.suroot.com • *.googmail.com • *.yourturbe.org • freeavg.sytes.net	• liumingzhen.zapto.org • liumingzhen.myftp.org • catlovers.25u.com • localhost port 8000	• "catlovers.25u.com**[пробел]**" port 9000 (не рабочий) • dryboxs.4dq.com port 9090 • localhost port 8000 (см proxy tunnel)
IP адреса	Разные	123.51.208.69	123.51.208.142 (same /24)
DDNS Provider	DynDNS, DtDNS, noip.com	noip.com	changeip.com

В то же время между модификациями, которые были проанализированы вышеуказанными антивирусными компаниями, присутствуют и сходства.

- Один и тот же алгоритм шифрования трафика (“ARCHY”[::-1] xor/ror3)
- Такой же набор данных, передаваемых в качестве полезной нагрузки по сети (1024-х байтный блок)
- Одни и те же команды C&C (0x211 и др.)
- Большинство C&C опираются на сервис динамического DNS.

Можно констатировать, что в Terminator RAT отсутствует последовательный дизайн в реализации вредоносного кода. Очевидно, что атакующие многократно изменяли его для достижения выбранных ими целей или использования в соответствующих атаках. Наличие нескольких механизмов шифрования и двух методов загрузки адресов функций, очевидно, оправдывают это предположение. Кроме этого вредоносный код использует шифрование для информации, передаваемой на C&C, но в то же время хранит их в

открытом виде в position-independent коде. Это не свидетельствует о правильности выбранного подхода. Наконец, некоторые функции неловко исправлены через добавление в них вспомогательных возможностей, таких как шифрование/дешифрование, как показано ниже на скриншоте.

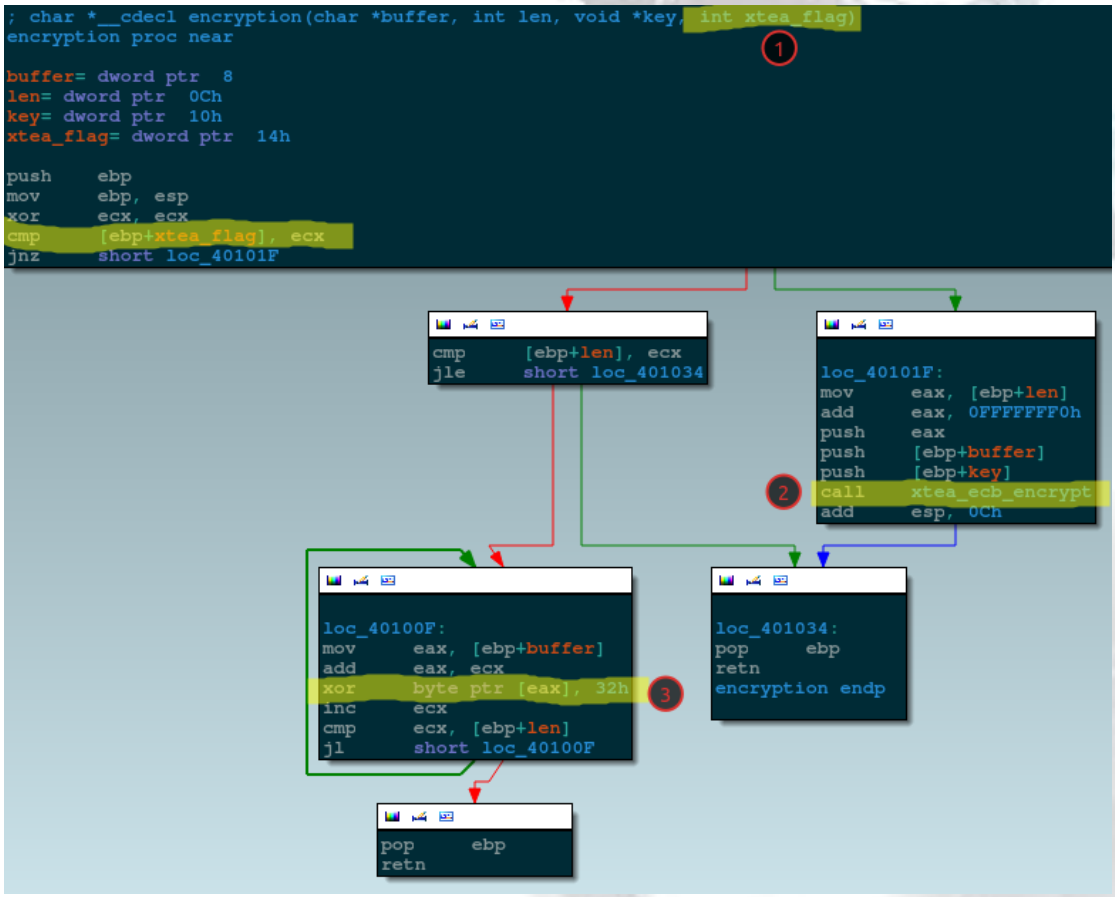

Рис. Функция шифрования в Terminator RAT.

Переменная xtea_flag (1) используется для определения того, с какой целью вызвана сама функция: для выполнения XTEA шифрования (2) или шифрования через XOR с фиксированным однобайтовым ключом (3). Очевидно, что один из фрагментов кода, который реализует данный алгоритм был встроен позднее.

На самом деле полезно иметь несколько анализов одного семейства вредоносного кода, так как мы можем видеть изменения в различных модификациях и в каких кампаниях они были использованы. В случае с Terminator RAR видно, что в различных атаках компоненты вредоносного кода и компоненты инфраструктуры были изменены. Такие компоненты как ключи XTEA, заголовки сетевых пакетов, название исполняемого файла, который отвечает за реализацию proxy tunnel, были изменены. С точки зрения инфраструктуры изменению подверглись провайдер DDNS и IP адреса C&C.

Теперь мы остановимся более подробно на компоненте proxy tunnel, который уже был детально описан FireEye как sss.exe. Он используется в тех случаях, когда вредоносный код оказался в сети, которая не позволяет устанавливать исходящие соединения напрямую с сервером (т. е. в сети используется прокси). В таком случае модуль резервирует порт 8000 и будет перебрасывать через него соединения, которые используют легитимный прокси, сконфигурированный для компьютера. Для

выполнения задачи подключения к C&C через этот прокси, используется метод CONNECT HTTP-протокола.

В исследуемой нами модификации файл этого proxy tunnel модуля называется winslogon.exe. Мы также обнаружили зашифрованный лог-файл, оставляемый этим компонентом по жестко зашитому пути %TEMP%\~DF3bbs.tmp. Файл может быть расшифрован с помощью XOR с ключом 0xAB как показано ниже.

```
key = 0xAB
ct = open("logfile", "rb").read()
pt = "".join([chr(ord(e) ^ key) for e in ct])
print pt
```

Рис. Функция расшифровки лога компонента proxy tunnel.

Для предотвращения запуска нескольких копий этого модуля в памяти, он использует объект события, в имени которого фигурирует непечатный символ с кодом 0x13. Добавление такого компонента к Terminator RAT расширяет возможности этой угрозы в области механизмов похищения данных (exfiltration) и поскольку он является фактически независимым от RAT компонентом, то может быть легко использован повторно в другой модификации вредоносного кода. Видно, что этот компонент не легко сразу обнаружить как вредоносный поскольку он не выполняет явных вредоносных функций. К тому же тот факт, что полезная нагрузка вредоносного кода скрыта в position-independent коде, делает статический RAT анализ довольно проблематичным.

Заключение

В процессе анализа файлов вредоносной программы стало очевидно, что ни один из них не был упакован для того, чтобы помешать анализу вредоносного кода. Мы также не увидели ни одного эксплойта, который мог бы помочь злоумышленникам достичь поставленных целей лучшим образом. Кроме этого, сам вредоносный код написан не самым лучшим образом, что свидетельствует о не совсем профессиональном уровне подготовки самих злоумышленников. Список недочетов можно продолжать: плохо реализованные алгоритмы криптографии, ошибки в командах при их вводе оператором, ошибки в конфигурации доменов C&C. Складывается впечатление, что заказчики подобной атаки выделили весьма малый бюджет на организационные нужды нападения.