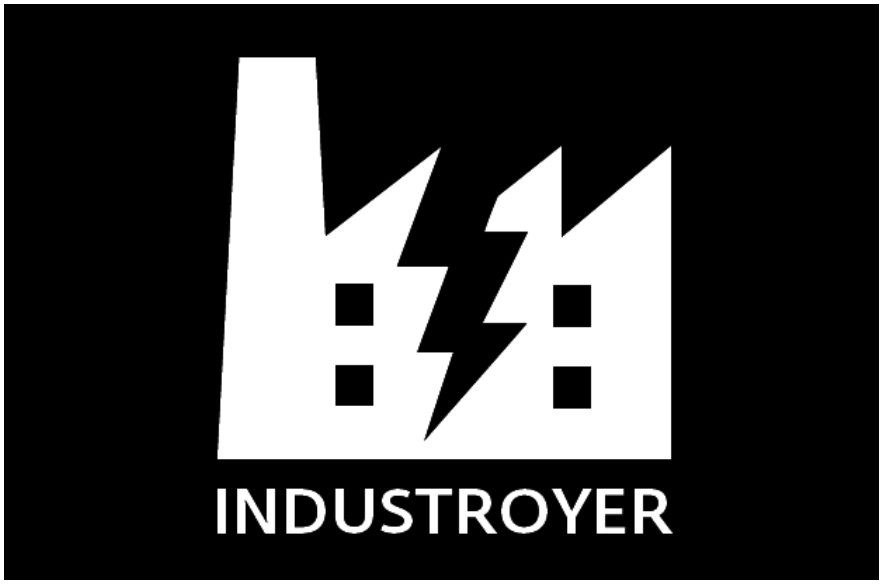
АНТИВИРУСНАЯ ЗАЩИТА БИЗНЕС-КЛАССА

Win32/Industroyer: новая угроза для промышленных систем управления

13 июня 2017 года

Win32/Industroyer – сложная вредоносная программа, предназначенная для нарушения рабочих процессов в промышленных системах управления (ICS), в частности, на электрических подстанциях.

Создателей Win32/Industroyer отличает высокая квалификация и глубокое понимание промышленных систем управления и протоколов связи в электроэнергетике. Маловероятно, чтобы кто-либо мог написать и протестировать подобное ПО без доступа к специализированному оборудованию, которое используется в целевой среде.



Авторы вредоносной программы реализовали поддержку четырех промышленных протоколов, описанных в следующих стандартах:

- IEC 60870-5-101 (IEC 101)
- IEC 60870-5-104 (IEC 104)
- IEC 61850
- OLE for Process Control Data Access (OPC DA)

В дополнение к этому авторы Industroyer разработали инструмент для выполнения DoS-атак (denial-of-service – отказ в обслуживании), нацеленных на определенные устройства релейной защиты, в частности, линейку Siemens SIPROTEC.

Возможности Win32/Industroyer впечатляют. Если сравнить его с инструментами, которые использовались в атаках на украинскую энергосистему в 2015 году и привели к массовым отключениям электроэнергии 23 декабря 2015 года (BlackEnergy, KillDisk и другие компоненты, включая легитимное ПО для удаленного доступа), можно утверждать, что за Industroyer стоит кибергруппа более высокого уровня. Авторы создали вредоносную программу, которая позволяет напрямую управлять выключателями и прерывателями цепи в сети электрических подстанций. По

некоторым признакам мы предполагаем, что Industroyer может быть связан со сбоем энергоснабжения в Киеве в декабре 2016 года. Тем не менее, на момент написания отчета это не было подтверждено, и расследование продолжается. Вектор заражения пока не установлен.

Industroyer состоит из нескольких модулей, описание и анализ которых представлены в следующих разделах отчета. Прежде чем перейти к деталям, предлагаем упрощенную схему, которая показывает связь между компонентами вредоносной программы.

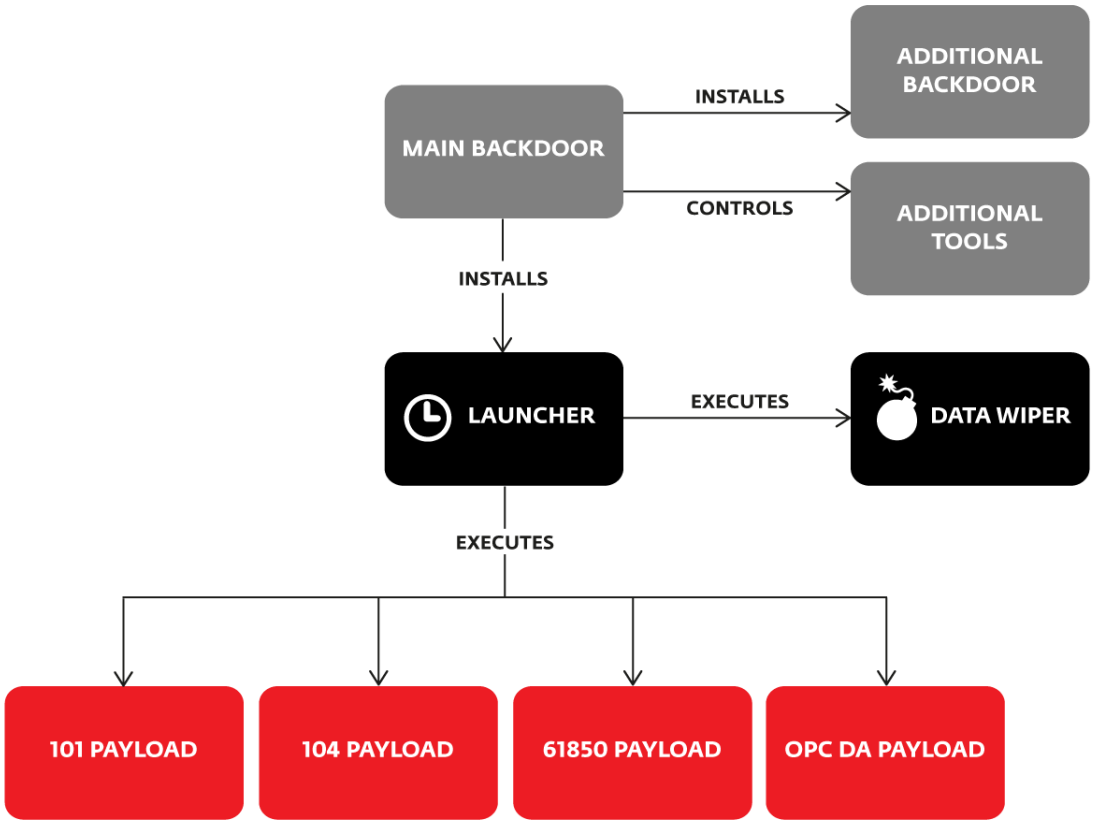


Рисунок 1. Упрощенная схема компонентов Win32/Industroyer.

Некоторые компоненты (включая стиратель данных) аналогичны по своей концепции инструментам, использовавшимся в атаках BlackEnergy на украинские энергокомпании в 2015 году. Тем не менее, мы не видим связи между прошлыми атаками и кодом нового вредоносного ПО.

Основной бэкдор

Главный компонент Industroyer – основной бэкдор – используется атакующими для управления остальными компонентами программы.

Как и положено бэкдору, этот компонент довольно прост. Он подключается к удаленному C&C-серверу через HTTPS и получает команды от атакующих. Все изученные образцы жестко запрограммированы на использование одного и того же адреса прокси, расположенного в локальной сети. Таким образом, бэкдор явно предназначен для работы в определенной организации. Также стоит упомянуть, что большинство C&C-серверов бэкдора используют Tor.

Возможно, наиболее интересная особенность бэкдора в том, что атакующие могут задать конкретный час, когда малварь будет активна. Например, можно модифицировать бэкдор таким

образом, чтобы он обращался к C&C-серверу в нерабочее время. Это усложняет обнаружение, основанное только на проверке сетевого трафика. Тем не менее, все образцы, изученные до настоящего времени, работали круглосуточно.

```
int main_loop()
{
  struct _SYSTEMTIME SystemTime; // [esp+0h] [ebp-14h]@4
  DWORD dwMilliseconds; // [esp+10h] [ebp-4h]@2

  SetLastError(0);
  if ( GetLastError() != ERROR_ALREADY_EXISTS )
  {
    dwMilliseconds = 5000;
    SetUnhandledExceptionFilter(TopLevelExceptionFilter);
    if ( !GetSystemMetrics(SM_CLEANBOOT) )
    {
      if ( create_imapi_handle() )
      {
        while ( 1 )
        {
          do
          {
            Sleep(dwMilliseconds);
            GetLocalTime(&SystemTime);
          }
          while ( SystemTime.wHour >= 24u );
          c2_connect_and_execute_cmd(&dwMilliseconds);
        }
      }
    }
  }
  return 0;
}
```

Рисунок 2. Декомпилированный код основного бэкдора с возможностью задать время.

После подключения к удаленному C&C-серверу основной бэкдор отправляет в POST-запрос следующие данные:

- строка GUID (глобально-уникальный идентификатор) для текущего профиля оборудования, полученная при помощи функции GetCurrentHwProfile
- версия вредоносного ПО – 1.1e
- жестко запрограммированный ID образца
- результат любой ранее полученной команды

Жестко запрограммированный ID используется атакующими в качестве идентификатора зараженной машины. Среди всех изученных образцов мы обнаружили следующие значения ID:

- DEF
- DEF-C
- DEF-WS
- DEF-EP
- DC-2-TEMP
- DC-2
- CES-McA-TEMP
- CES
- SRV_WSUS
- SRV_DC-2
- SCE-WSUS01

Основной бэкдор поддерживает следующие команды:

ID команды	Цель
0	Выполнить процесс.
1	Выполнить процесс под определенной учетной записью пользователя. Аутентификационные данные предоставляются атакующими.
2	Загрузить файл с C&C-сервера.
3	Копировать файл.
4	Выполнить shell-команду.
5	Выполнить shell-команду под определенной учетной записью пользователя. Аутентификационные данные предоставляются атакующими.
6	Прекратить работу.
7	Остановить службу.
8	Остановить службу под определенной учетной записью пользователя. Аутентификационные данные предоставляются атакующими.
9	Запустить службу под определенной учетной записью пользователя. Аутентификационные данные предоставляются атакующими.
10	Заменить параметр реестра ImagePath для процесса.

Получив права администратора, атакующие могут обновить установленный бэкдор до более привилегированной версии, которая выполняется как Windows service program. Для этого им нужно выбрать существующий некритический процесс Windows и заменить параметр реестра ImagePath на путь к новому бинарному файлу бэкдора.

Функциональность основного бэкдора, который работает как служба Windows, идентична описанной. Но есть два небольших различия: название версии бэкдора (1.1s вместо 1.1e) и обфускация кода. Код этой версии бэкдора смешан с ненужными командами ассемблера.

```
.text:00403FD2 main_func proc near                     ; CODE XREF: WinMain(x,x,x,x)+14↑p
.text:00403FD2                                         ; .text:004038C4↑p
.text:00403FD2         call    $+5
.text:00403FD7
.text:00403FD7 loc_403FD7:                             ; CODE XREF: main_func+57↓j
.text:00403FD7                                         ; main_func+5F↓j
.text:00403FD7         add     esp, 4
.text:00403FDA         push    ebp                     ; lpOverlapped
.text:00403FDB         mov     ebp, esp
.text:00403FDD         cmp     edx, 142F9F9Ah
.text:00403FE3         jz      short loc_404023
.text:00403FE5         push    ecx
.text:00403FE6         push    ecx
.text:00403FE7         mov     eax, [ebp+10h]
.text:00403FEA         mov     dword_416190, eax
.text:00403FEF         mov     eax, [ebp+8]
.text:00403FF2         mov     dword_416194, eax
.text:00403FF7         mov     eax, [ebp+0Ch]
.text:00403FFA         cmp     edx, 0B5B93EF3h
.text:00404000         jz      short loc_404023
.text:00404002         mov     lpOverlapped, eax
.text:00404007         mov     [ebp-8], eax
.text:0040400A         lea     eax, [ebp-8]
.text:0040400D         push    eax                     ; lpServiceStartTable
.text:0040400E         mov     dword ptr [ebp-4], offset ServiceMain
.text:00404015         call    ds:StartServiceCtrlDispatcherW
.text:0040401B         xor     al, al
.text:0040401D         mov     esp, ebp
.text:0040401F         pop     ebp
.text:00404020         retn
```

Рисунок 3. Обфусцированный ассемблерный код основного бэкдора, работающего как служба Windows.

Дополнительный бэкдор

Дополнительный бэкдор обеспечивает альтернативный механизм устойчивости, что позволяет

атакующим восстановить доступ к целевой сети, если основной бэкдор будет обнаружен и/или деактивирован.

Бэкдор представляет собой троянизированную версию приложения Windows Notepad. Это полнофункциональное приложение, но вирусописатели добавили в него вредоносный код, который выполняется при каждом запуске. Далее, получив права администратора, атакующие могут вручную заменить вредоносный Notepad легитимным.

Добавленный вредоносный код сильно обфусцирован. После расшифровки он подключается к удаленному C&C-серверу (отличается от C&C основного бэкдора) и загружает полезную нагрузку. Она имеет формат шелл-кода, который загружается непосредственно в память и выполняется. Кроме того, добавленный код расшифровывает исходный код Windows Notepad, хранящийся в конце файла, и затем передает ему выполнение – так приложение работает должным образом.

```
.text:01004AD5     lea     eax, [ebp+var_50]
.text:01004AD8     push    eax
.text:01004AD9     lea     eax, [ebp+h]
.text:01004ADC     push    eax
.text:01004ADD     push    0B0h
.text:01004AE2     push    hWnd
.text:01004AE8     mov     stru_100A680.lStructSize, 58h
.text:01004AF2     mov     stru_100A680.hwndOwner, edx
.text:01004AF8     mov     stru_100A680.nMaxFile, 104h
.text:01004B02     mov     stru_100A500.lStructSize, 28h
.text:01004B0C     mov     stru_100A500.hwndOwner, edx
.text:01004B12     call    esi ; SendMessageW
.text:01004B14     push    [ebp+var_50]
.text:01004B17     push    [ebp+h]
.text:01004B1A     push    0B1h
.text:01004B1F     push    hWnd
.text:01004B25     call    esi ; SendMessageW
.text:01004B27     push    ebx
.text:01004B28     push    ebx
.text:01004B29     push    0B7h
.text:01004B2E     push    hWnd
.text:01004B34     call    esi ; SendMessageW
.text:01004B36     push    ebx
.text:01004B37     call    ds:GetKeyboardLayout
.text:01004B3D     and     ax, 3FFh
.text:01004B41     cmp     ax, 11h
.text:01004B45     jnz     short loc_1004B58
.text:01004B47     push    1
.text:01004B49     push    1
.text:01004B4B     push    0D8h
.text:01004B50     push    hWnd
.text:01004B56     call    esi ; SendMessageW
```

```
.text:01004AD5     lea     eax, [ebp+var_50]
.text:01004AD8     push    eax
.text:01004AD9     lea     eax, [ebp+h]
.text:01004ADC     push    eax
.text:01004ADD     push    0B0h
.text:01004AE2     push    hWnd
.text:01004AE8     mov     stru_100A680.lStructSize, 58h
.text:01004AF2     mov     stru_100A680.hwndOwner, edx
.text:01004AF8     mov     stru_100A680.nMaxFile, 104h
.text:01004B02     mov     stru_100A500.lStructSize, 28h
.text:01004B0C     mov     stru_100A500.hwndOwner, edx
.text:01004B12     call    esi ; SendMessageW
.text:01004B14     pusha
.text:01004B15     pushf
.text:01004B16     neg     ebx
.text:01004B18     shr     eax, 1
.text:01004B1B     dec     ebx
.text:01004B1C     mov     eax, 17B200AFh
.text:01004B21     mov     edi, 71CFC28h
.text:01004B26     or      edi, dword_10095C7
.text:01004B2C     xor     esi, 1C779E91h
.text:01004B32     xor     eax, eax
.text:01004B34     dec     edi
.text:01004B35     rol     esi, 5
.text:01004B38     and     esi, edi
.text:01004B3A     and     esi, edi
.text:01004B3C     rol     edx, 6
.text:01004B3F     neg     eax
.text:01004B41     xor     esi, eax
.text:01004B43     neg     ebx
.text:01004B45     shr     ebx, 5
.text:01004B48     mov     ecx, 5E95422h
```

Рисунок 4. Сравнение оригинального бинарного кода Notepad (слева) и бэкдора.

Компонент запуска (Launcher)

Компонент представляет собой отдельный исполняемый файл, предназначенный для запуска полезной нагрузки и компонента стирателя данных.

Компонент запуска содержит определенное время и дату. В изученных образцах было задано две даты: 17 и 20 декабря 2016 года. Как только одна из двух дат наступает, компонент создает два потока. Первый пытается загрузить вредоносную DLL, второй ждет один или два часа (в зависимости от версии компонента), а затем пытается загрузить компонент стирателя данных. Оба потока имеют высший приоритет THREAD_PRIORITY_HIGHEST, что означает, что они получают более высокую долю ресурсов центрального процессора.

Имя вредоносной DLL добавляется атакующими через параметр командной строки, предоставленный в одной из команд основного бэкдора («Выполнить shell-команду»). Имя файла стирателя данных – всегда haslo.dat. Ожидаемые командные строки имеют вид:

%LAUNCHER%.exe %WORKING_DIRECTORY% %PAYLOAD%.dll %CONFIGURATION%.ini

Каждый аргумент командной строки обозначает следующее:

• %LAUNCHER%.exe – имя файла компонента запуска
• %WORKING_DIRECTORY% – каталог, в котором хранится вредоносная DLL и конфигурация
• %PAYLOAD%.dll – имя файла вредоносной DLL
• %CONFIGURATION%.ini – файл, в котором хранятся данные конфигурации конкретной полезной нагрузки. Путь к этому файлу передается вредоносной DLL компонентом запуска

Полезная нагрузка и компонент стирателя данных – стандартные файлы DLL Windows. Чтобы быть загруженными компонентом запуска, они должны экспортировать функцию Crash как показано на рис. 5.

```
;
; Export directory for Crash101.dll
;
                dd 0                    ; Characteristics
                dd 5855F8EDh            ; TimeDateStamp: Sun Dec 18 02:48:13 2016
                dw 0                    ; MajorVersion
                dw 0                    ; MinorVersion
                dd rva aCrash101_dll    ; Name
                dd 1                    ; Base
                dd 1                    ; NumberOfFunctions
                dd 1                    ; NumberOfNames
                dd rva off_100355F8     ; AddressOfFunctions
                dd rva off_100355FC     ; AddressOfNames
                dd rva word_10035600    ; AddressOfNameOrdinals
;
; Export Address Table for Crash101.dll
;
off_100355F8    dd rva Crash            ; DATA XREF: .rdata:100355EC↑o
;
; Export Names Table for Crash101.dll
;
off_100355FC    dd rva aCrash           ; DATA XREF: .rdata:100355F0↑o
                                        ; "Crash"
;
; Export Ordinals Table for Crash101.dll
;
word_10035600   dw 0                    ; DATA XREF: .rdata:100355F4↑o
aCrash101_dll   db 'Crash101.dll',0     ; DATA XREF: .rdata:100355DC↑o
aCrash          db 'Crash',0            ; DATA XREF: .rdata:off_100355FC↑o
```

Рисунок 5. Пример вредоносной DLL с внутренним именем Crash101.dll и экспортированной функцией Crash.

Компонент 101

Вредоносная DLL с именем файла 101.dll названа в честь международного стандарта IEC 101 (он же IEC 60870-5-101), который описывает протокол мониторинга и управления электроэнергетическими системами. Протокол обеспечивает коммуникацию между промышленными системами управления и удаленными терминальными блоками (Remote Terminal Units – RTUs). Обмен данными осуществляется через последовательное соединение.

Компонент 101 частично реализует протокол, описанный в стандарте IEC 101. Он способен связываться с RTU и любым другим устройством, поддерживающим этот протокол.

После выполнения компонент 101 анализирует конфигурацию, хранящуюся в INI-файле. Конфигурация может содержать несколько записей: имя процесса, имена устройств Windows (обычно COM-порты), количество диапазонов и значения диапазонов для адресов информационных объектов (Information Object Address – IOA). IOA – номер, который идентифицирует конкретный элемент данных в устройстве. На рис. 6 показан файл конфигурации компонента 101 с двумя определенными диапазонами IOA: 10-15 и 20-25.

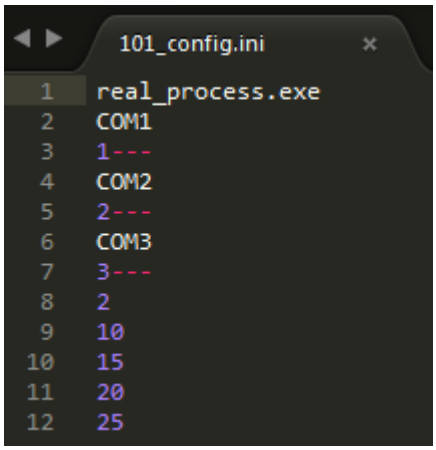


Рисунок 6. Пример конфигурации компонента 101.

Имя процесса, указанного в конфигурации, принадлежит приложению, которое, как предполагают атакующие, работает в системе жертвы. Это должно быть приложение, которое машина использует для коммуникаций с RTU через последовательное соединение. Компонент 101 пытается завершить этот процесс и обращается к указанному устройству, используя функции Windows API CreateFile, WriteFileи ReadFile. Первый COM-порт из файла конфигурации используется для фактической связи, два остальных – открыты, чтобы предотвратить обращение других процессов. Таким образом, компонент 101 может взять на себя управление устройством RTU.

Компонент выполняет итерацию по всем диапазонам IOA. Для каждого IOA он создает пакеты команд выбора и исполнения с однопозиционной (C_SC_NA_1) и двухпозиционной командой (C_DC_NA_1) и отправляет на устройство RTU. Основная цель компонента – изменить значение выключателя On/Off для однопозиционного и двухпозиционного командного типа IOA. Так, на первом этапе компонент пытается переключить IOA в состояние Off, на второй – On, на заключительном этапе – вернуть в значение Off.

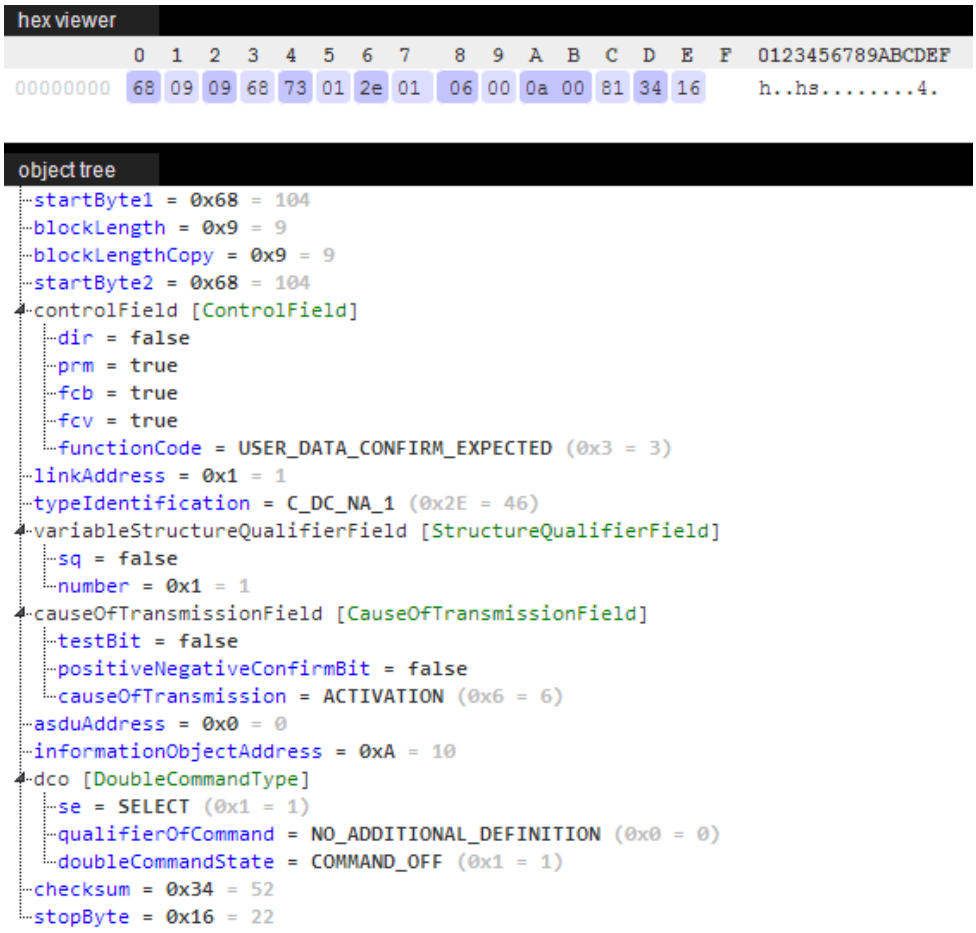


Рисунок 7. Пример разобранного пакета полезной нагрузки в Kaitai Struct WebIDE.

Компонент 104

DLL с именем файла 104.dll названа в честь стандарта IEC 104 (IEC 60870-5-104). Протокол IEC 104 дополняет IEC 101 так, чтобы передавать данные по TCP/IP. Благодаря гибко изменяемой конфигурации, компонент может быть настроен атакующими под различное оборудование. Рис. 8 показывает, как может выглядеть файл конфигурации.

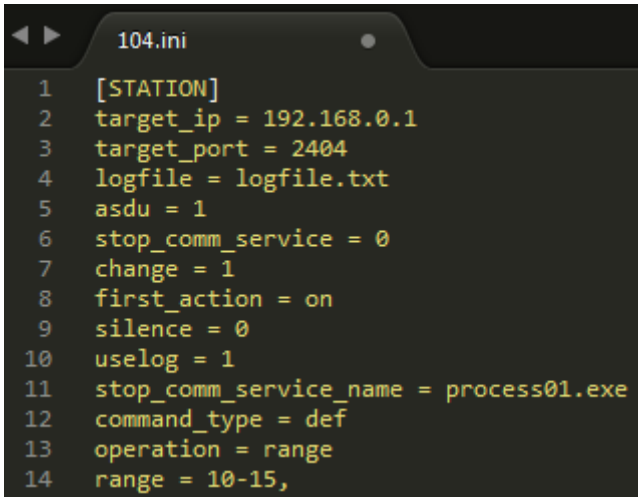


Рис. 8. Образец файла DLL конфигурации компонента 104.

После исполнения DLL компонента 104 произведет попытку прочитать файл конфигурации. Путь к

файлу конфигурации берется из компонента загрузчика.

Конфигурация содержит раздел STATION, за которым следуют свойства, определяющие работу компонента 104. Конфигурация может содержать множество записей STATION.

Наш анализ этого компонента показывает следующие возможные параметры конфигурации:

Свойства	Ожидаемое значение	Цель
target_ip	IP-адрес	IP-адрес для передачи данных по протоколу стандарта IEC 104
target_port	Номер порта	Понятно из названия
uselog	1 или 0	Включает или выключает запись логов в файл
logfile	Имя файла	Определяет имя файла для логов, если они включены
stop_comm_service	1 или 0	Включает или выключает завершение процесса
stop_comm_service_name	Имя процесса	Определяет имя процесса, который будет завершен
timeout	Время ожидания, миллисекунды	Определяет время ожидания между `send` и `recv`. Значение по умолчанию: 15000
socket_timeout	Время ожидания, миллисекунды	Определяет время ожидания ответа. Значение по умолчанию: 15000
silence	1 или 0	Включает или выключает консольный вывод
asdu	целочисленная переменная	Определяет адрес ASDU (блок данных прикладных услуг), также известный как сектор
first_action	on или off	Определяет значение выключателя в пакете ASDU для первого цикла
change	1 или 0	Определяет, нужно ли инвертировать значение выключателя в пакете ASDU во время циклов
command_type	def или short или long	Определяет продолжительность управляющего импульса для управляющего параметра команды (QOC)
operation	range или sequence или shift	Определяет тип цикла для адресов информационных объектов (IOA)
range	Определенный формат IOAs	Определяет диапазон для адресов информационных объектов (IOA)

После прочтения файла конфигурации компонент 104 создает процесс для каждого фрагмента STATION, по одному на каждый. В каждом таком процессе компонент 104 попытается связаться с указанным адресом IP при помощи протокола, описанного в стандарте IEC 104. До установления соединения он попытается завершить легитимный процесс, который должен отвечать за обмен данными с устройством. Это происходит только в том случае, если свойство stop_comm_serviceпрописано в конфигурации. По умолчанию компонент 104 завершает процесс с именем D2MultiCommService.exe, либо с тем именем, которое прописано в его конфигурации.

Основная идея использования компонента 104 относительно проста. Он связывается с указанным IP и начинает передачу пакетов с адресом ASDU, прописанным в его конфигурации. Цель этого обмена данных — связаться с IOA однопозиционного типа команд. В файле конфигурации атакующий может определить свойство operation, чтобы указать, как будут опрошены IOA адреса однопозиционного типа.

Первый такой режим operation — range. Хакеры используют его для обнаружения возможных IOA в интересующем устройстве. Им приходится применять данный подход, потому что протокол, описанный в стандарте IEC 104, не дает определенного метода получения этой информации.

range работает в два этапа. Во время первого, после получения диапазона IOA из файла конфигурации, компонент 104 подключается к целевому адресу IP и выполняет опрос указанных IOA. На каждый из этих IOA компонент 104 передает пакеты команд выбора и исполнения для изменения состояния и проверки, относится ли данный IOA к однопозиционному командному типу.

```
▷ Internet Protocol Version 4, Src: 192.168.0.1, Dst: 192.168.0.2
▷ Transmission Control Protocol, Src Port: 2404, Dst Port: 49168, Seq: 39, Ack: 45, Len: 16
▷ IEC 60870-5-104-Apci: -> I (2,2)
◢ IEC 60870-5-104-Asdu: ASDU=1 C_SC_NA_1 ActTerm IOA=10 'single command'
    TypeId: C_SC_NA_1 (45)
    0... .... = SQ: False
    .000 0001 = NumIx: 1
    ..00 1010 = CauseTx: ActTerm (10)
    .0.. .... = Negative: False
    0... .... = Test: False
    OA: 0
    Addr: 1
  ◢ IOA: 10
      IOA: 10
    ◢ SCO: 0x01
        .... ...1 = ON/OFF: On
        .000 00.. = QU: No pulse defined (0)
        0... .... = S/E: Execute
```

Рис. 9. Пример декодированного в Wireshark пакета компонента 104.

Как только все возможные IOA из определенного диапазона опрошены, компонент 104 переходит ко второму этапу режима range. Если возможность записи в логи включена, компонент запишет в него Starting only success. Остальная часть второго этапа состоит в бесконечном цикле использования ранее обнаруженных IOA однопозиционного типа. В цикле компонент непрерывно передает пакеты команд выбора и исполнения. Вдобавок, если опция change определена, компонент переключает состояние On/Off между этапами цикла.

На рис. 10 показан файл лога, который выдал компонент 104 во время нашего анализа. Из него мы видим, что компонент опросил IOA от 10 до 15 и, когда IOA однопозиционного типа были обнаружены, начал их использование в цикле. В конфигурации опция change была включена, так что между циклами компонент переключал значение выключателя с On на Off и писал это в лог.

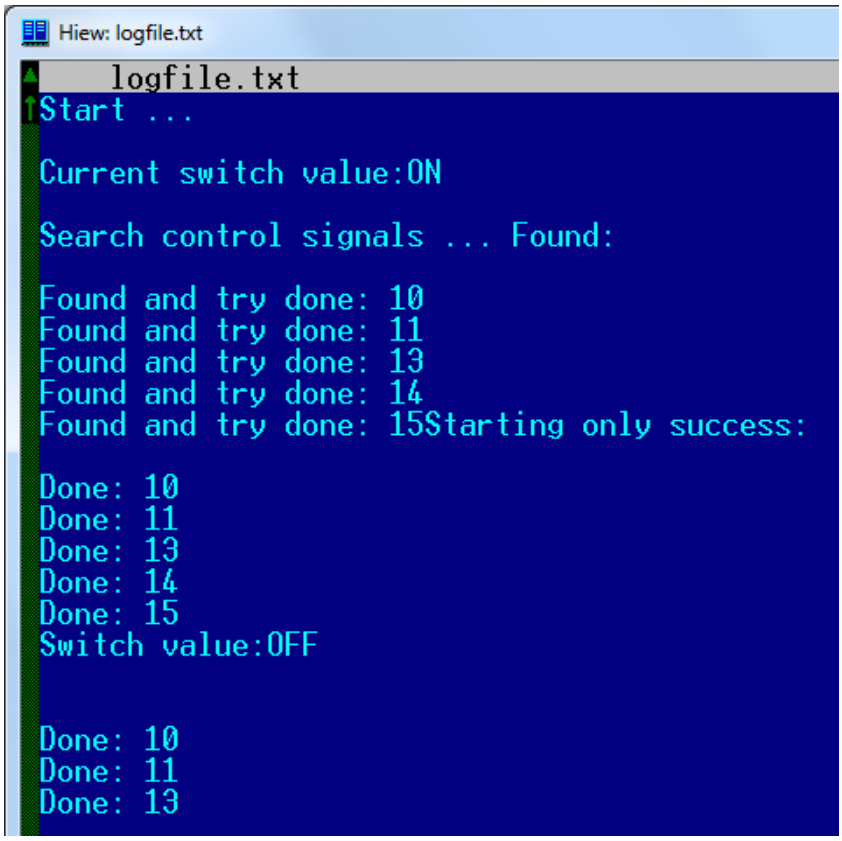


Рис. 10. Пример логов компонента 104.

Второй режим operation — shift. Он очень похож на режим range. Атакующий определяет в файле конфигурации диапазон IOA и изменяемые значения. После активации компонента 104 все происходит точно так же, как и в режиме range; однако, как только все IOA в определенном диапазоне будут опрошены, он начинает опрос нового диапазона. Новый диапазон высчитывается путем сложения дефолтного диапазона и значений сдвига.

Третий режим operation — sequence. Может применяться хакерами после того, как они узнают значения всех IOA однопозиционного типа команд, поддерживаемых подключенным устройством. Этот компонент незамедлительно начнет исполнение бесконечного цикла, передавая пакеты выбора и исполнения на IOA, указанные в файле конфигурации.

Кроме способности писать в лог, компонент 104 может выводить отладочную информацию в консоль, как это показано на рис. 11.

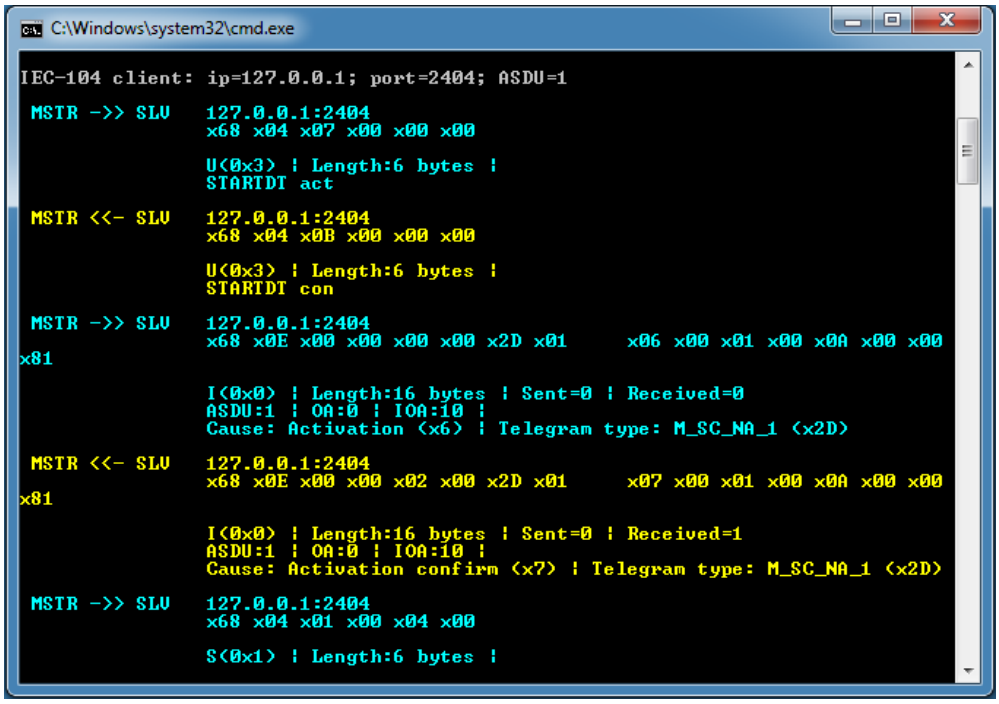


Рис. 11. Вывод на консоль компонента 104.

Компонент 61850

В отличие от компонентов 101 и 104, существует в виде отдельного вредоносного инструмента, состоящего из исполняемого файла под названием 61850.exe и DLL 61850.dll. Назван в честь стандарта IEC 61850. Этот стандарт описывает протокол, используемый для обмена данными между устройствами различных производителей, которые выполняют функции защиты, автоматизации, измерения, мониторинга и управления систем автоматики электрических подстанций. Это сложный и надежный протокол, но компонент 61850 использует только небольшой ряд его параметров, чтобы привести к разрушительным последствиям.

После исполнения DLL компонента 61850 пробует прочесть файл конфигурации, чей путь предоставляется компонентом запуска. Отдельная версия по умолчанию берет конфигурацию из i.ini. Ожидается, что файл конфигурации содержит список IP-адресов устройств, которые имеют возможность обмена данными по протоколу, описанному в стандарте IEC 61850.

Если файл конфигурации не находится, компонент 61850 нумерует все подключенные адаптеры сети для определения их масок подсети TCP/IP. Затем компонент нумерует все возможные IP-адреса для каждой из масок подсети и пробует подключиться к порту 102 каждого адреса. Таким образом, компонент умеет автоматически обнаруживать подходящие устройства в сети.

В другом случае, если файл конфигурации найден и содержит целевые адреса IP, компонент подключается к порту 102 по этим адресам и найденным автоматически.
Как только этот компонент подключается к целевому хосту, он передает пакет запроса на подключение при помощи транспортного протокола, ориентированного на соединение, как показано на рис. 12.

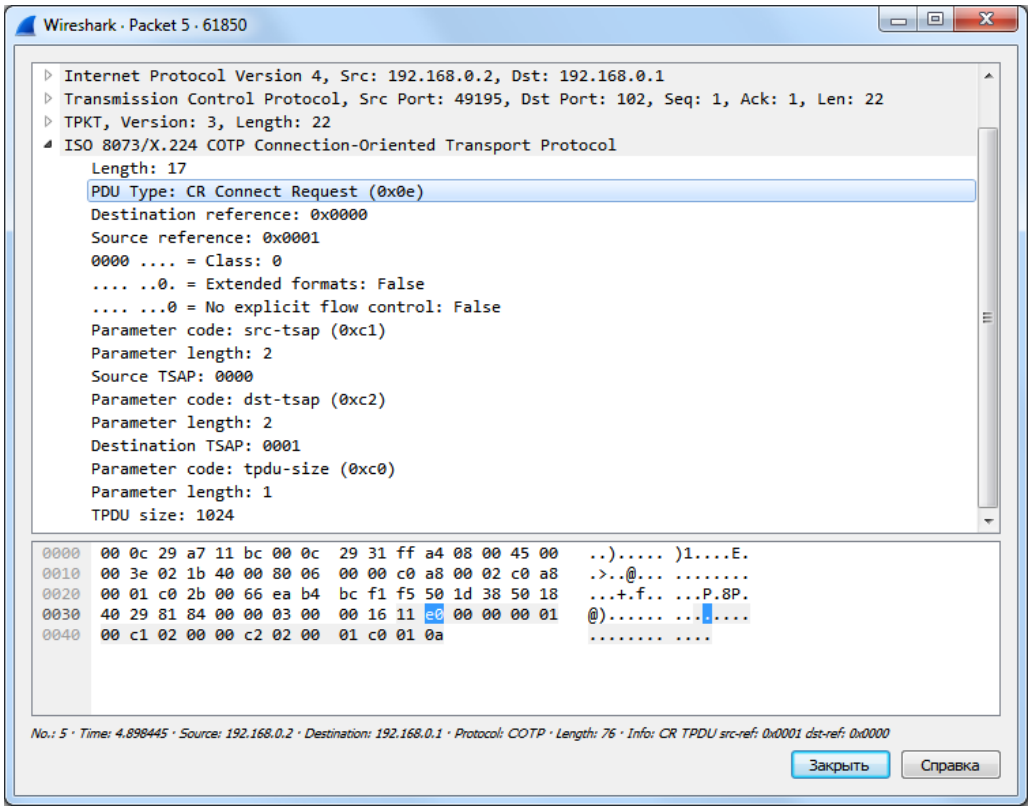


Рис. 12 Декодированный пакет запроса на подключение в Wireshark.

Если целевое устройство отвечает должным образом, компонент 61850 передает пакет InitiateRequest при помощи Спецификации производственных сообщений (MMS). Если ожидаемый ответ получен, он передает MMS запрос getNameList. Таким образом, компонент составляет список имен объектов в виртуальном производственном устройстве (VMD).

Затем этот компонент 61850 нумерует компоненты, обнаруженные на предыдущем этапе, и отправляет предметно-ориентированные запросы getNameList с каждым именем объекта. Таким образом, компонент нумерует названные переменные в определенном домене.

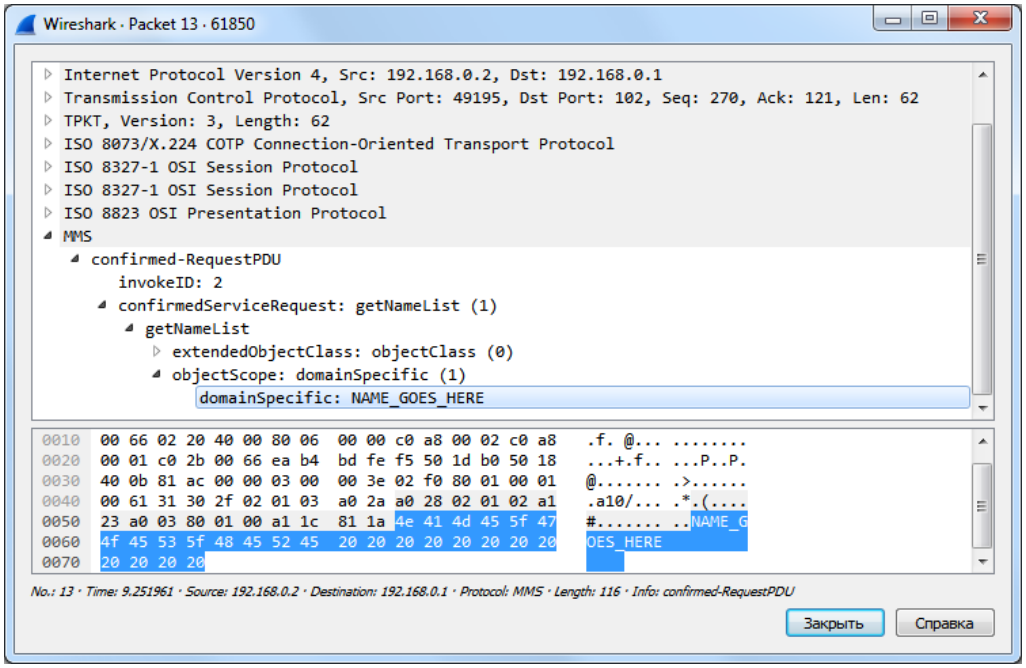


Рис. 13. Декодированный запрос MMS getNameList в Wireshark.

Затем компонент 61850 парсит информацию, полученную в ответ на эти запросы, и ищет переменные, которые содержат следующие строковые последовательности:

- CSW, CF, Pos и Model
- CSW, ST, Pos и stVal
- CSW, CO, Pos, Oper, но не $T
- CSW, CO, Pos, SBO, но не $T

Последовательность CSW — это название логического узла, который используется для управления прерывателями цепи и выключателями.

Для переменных, которые содержат последовательности Model или stVal, компонент 61850 передает дополнительный MMS запрос Read. Для некоторых из этих переменных компонент также может передать MMS запрос Write, который изменит их текущее состояние.

Компонент 61850 создаст файл с логами его работы, содержащий IP-адреса, MMS домены, названные переменные и состояние узлов (открытые или закрытые) его целей.

Компонент OPC DA

Компонент OPC DA внедряет клиент для протокола, описанного в спецификации OPC Data Access. OPC (OLE for Process Control) — это стандарт ПО и спецификация, основанная на технологиях Microsoft, таких как OLE, COM и DCOM. Часть, относящаяся к доступу к данным (DA) спецификации OPC, позволяет производить обмен данными в реальном времени между распространяемыми компонентами по принципу модели «клиент-сервер».

Этот компонент существует в качестве отдельного вредоносного инструмента с именем файла OPC.exeи DLL, который использует функционал и 61850, и OPC DA компонентов. Внутреннее название DLL в экспортированной таблице PE OPCClientDemo.dll, что дает основание предположить, что этот компонент может быть основан на проекте с открытым исходным кодом OPC Client.

```
;
; Export Address Table for OPCClientDemo.dll
;
off_10039678    dd rva Crash            ; DATA XREF: .rdata:1003966C↑o
;
; Export Names Table for OPCClientDemo.dll
;
off_1003967C    dd rva aCrash           ; DATA XREF: .rdata:10039670↑o
                                        ; "Crash"
```

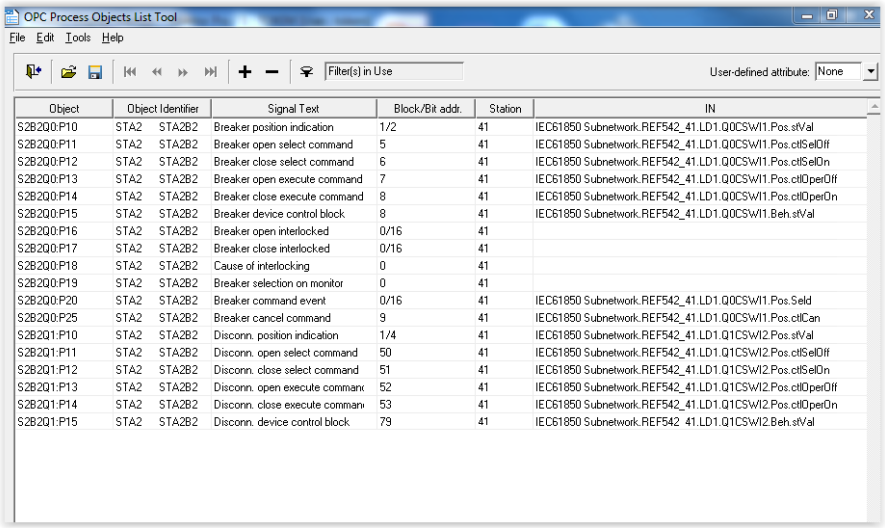
Рис. 14. Таблица PE показывает внутреннее имя DLL компонента OPC DA.

Компонент OPC DA не требует какого-либо файла с конфигурацией. После исполнения атакующим он номерует все OPC-серверы при помощи метода ICatInformation::EnumClassesOfCategories с идентификатором категории CATID_OPCDAServer20 и IOPCServer::GetStatus для определения тех, которые запущены.

Следующий компонент использует интерфейс IOPCBrowseServerAddressSpace для нумерации всех серверных элементов OPC. Особым образом он ищет элементы, содержащие следующие последовательности в имени:

- ctlSelOn
- ctlOperOn
- ctlSelOff
- ctlOperOff
- \Pos и stVal

Имена элементов дают основание предполагать, что атакующих интересует элементы OPC, предоставленные серверами OPC, которые относятся к решениям ABB, такие как линейка MicroSCADA. На рис. 15 показан образец элементов OPC, который содержат имена с похожими строковыми последовательностями. Этот список элементов OPC получен инструментом списков объектов процесса OPC от ABB.



Object	Object Identifier	Signal Text	Block/Bit addr.	Station	IN
S2B2Q0:P10	STA2 STA2B2	Breaker position indication	1/2	41	IEC61850 Subnetwork.REF542_41.LD1.Q0CSWI1.Pos.stVal
S2B2Q0:P11	STA2 STA2B2	Breaker open select command	5	41	IEC61850 Subnetwork.REF542_41.LD1.Q0CSWI1.Pos.ctlSelOff
S2B2Q0:P12	STA2 STA2B2	Breaker close select command	6	41	IEC61850 Subnetwork.REF542_41.LD1.Q0CSWI1.Pos.ctlSelOn
S2B2Q0:P13	STA2 STA2B2	Breaker open execute command	7	41	IEC61850 Subnetwork.REF542_41.LD1.Q0CSWI1.Pos.ctlOperOff
S2B2Q0:P14	STA2 STA2B2	Breaker close execute command	8	41	IEC61850 Subnetwork.REF542_41.LD1.Q0CSWI1.Pos.ctlOperOn
S2B2Q0:P15	STA2 STA2B2	Breaker device control block	8	41	IEC61850 Subnetwork.REF542_41.LD1.Q0CSWI1.Beh.stVal
S2B2Q0:P16	STA2 STA2B2	Breaker open interlocked	0/16	41	
S2B2Q0:P17	STA2 STA2B2	Breaker close interlocked	0/16	41	
S2B2Q0:P18	STA2 STA2B2	Cause of interlocking	0	41	
S2B2Q0:P19	STA2 STA2B2	Breaker selection on monitor	0	41	
S2B2Q0:P20	STA2 STA2B2	Breaker command event	0/16	41	IEC61850 Subnetwork.REF542_41.LD1.Q0CSWI1.Pos.SeId
S2B2Q0:P25	STA2 STA2B2	Breaker cancel command	9	41	IEC61850 Subnetwork.REF542_41.LD1.Q0CSWI1.Pos.ctlCan
S2B2Q1:P10	STA2 STA2B2	Disconn. position indication	1/4	41	IEC61850 Subnetwork.REF542_41.LD1.Q1CSWI2.Pos.stVal
S2B2Q1:P11	STA2 STA2B2	Disconn. open select command	50	41	IEC61850 Subnetwork.REF542_41.LD1.Q1CSWI2.Pos.ctlSelOff
S2B2Q1:P12	STA2 STA2B2	Disconn. close select command	51	41	IEC61850 Subnetwork.REF542_41.LD1.Q1CSWI2.Pos.ctlSelOn
S2B2Q1:P13	STA2 STA2B2	Disconn. open execute comman	52	41	IEC61850 Subnetwork.REF542_41.LD1.Q1CSWI2.Pos.ctlOperOff
S2B2Q1:P14	STA2 STA2B2	Disconn. close execute comman	53	41	IEC61850 Subnetwork.REF542_41.LD1.Q1CSWI2.Pos.ctlOperOn
S2B2Q1:P15	STA2 STA2B2	Disconn. device control block	79	41	IEC61850 Subnetwork.REF542_41.LD1.Q1CSWI2.Beh.stVal

Рис. 15. Пример имен элементов OPC в поле IN, полученным при помощи инструмента списков объектов процесса OPC.

Атакующие используют строку Abdul при добавлении новой группы OPC. Возможно эта строка используется атакующими в качестве сленгового названия решений ABB.

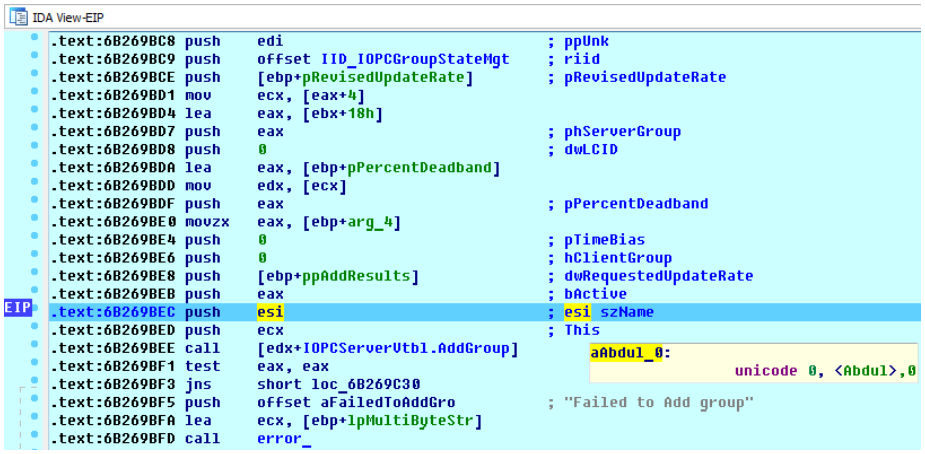


Рис. 16. Дизассемблированный код компонента OPC DA, который использует строку Abdul.

На финальном этапе компонент OPC DA пробует изменить состояние обнаруженных элементов OPC при помощи интерфейса IOPCSyncIO, прописывая значение 0x01 дважды.

```
.text:004034FE          mov     eax, VT_I2
.text:00403503          mov     word ptr [ebp+pItemValues.anonymous_0], ax
.text:0040350A          mov     eax, 1
.text:0040350F          mov     word ptr [ebp+pItemValues.anonymous_0+8], ax
.text:00403516          lea     eax, [ebp+pItemValues]
.text:0040351C          push    eax                     ; pItemValues
.text:0040351D          mov     eax, [ebp+OPC_items]
.text:00403523          mov     ecx, [eax+esi*4]
.text:00403526          call    IOPCSyncIO_Write
.text:0040352B          cmp     esi, edi
.text:0040352D          jb      short loc_403539
.text:0040352F          push    80070057h
.text:00403534          call    throw_exception
```

Рис. 17. Дизассемблированный код компонента OPC DA, который использует интерфейс IOPCSyncIO.

Компонент прописывает имя сервера OPC, состояние имени элемента OPC, код качества и значение в лог-файле. Записанные значения разделяются следующими строками заголовка:

- [*ServerName: %SERVERNAME%] [State: Before]
- [*ServerName: %SERVERNAME%] [State: After ON]
- [*ServerName: %SERVERNAME%] [State: After OFF]

Компонент стирателя данных

Компонент стирателя данных (Data Wiper) — это деструктивный компонент, который используется на финальном этапе атаки. Хакеры используют этот компонент, чтобы замести следы и усложнить процесс восстановления.

Имя файла этого компонента haslo.dat или haslo.exe, он может быть исполнен компонентом запуска, либо использоваться в качестве отдельного вредоносного инструмента.

После исполнения компонент пробует пронумеровать все ключи в реестре, который перечисляет службы Windows.

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services

Он пытается установить значение реестра ImagePath с пустой строкой в каждой обнаруженной записи. Это приведет к тому, что операционная система перестанет загружаться.

Следующий этап — удаление содержимого файла. Компонент нумерует файлы с определенными расширениями на всех приводах, подключенных к компьютеру, от C:\ до Z:\. Нужно отметить, что во время нумерации компонент пропускает файлы, расположенные в подкаталоге, которые содержат слово Windows в названии.

Компонент подменяет содержимое файла бессмысленной информацией, получаемой из содержимого новой выделенной памяти. Чтобы исполнить эту операцию должным образом, компонент пробует переписать файлы дважды. Первая попытка происходит, когда файл обнаруживается на приводе. Если первая попытка безуспешна, компонент совершает вторую попытку, но перед этим завершает все процессы, кроме включенных в список критических системных процессов. Список процессов показан на рис. 18.

Чтобы ускорить процесс стирания, компонент переписывает только часть содержимого файла в его начале. Объем информации, который будет переписан, зависит от размера файла. Наименьший объем информации будет переписан у файлов размером 1 Mb (4096 байтов) и

меньше. Наибольший объем информации будет переписан у файлов размером 10 Mb (32768 байтов) и меньше.

Наконец, этот компонент пробует завершить все процессы, включая системные, кроме собственного. Это приведет к тому, что система перестанет отвечать и в итоге выйдет из строя.

```
off_10010E88    dd offset aAudiodg_exe  ; DATA XREF: _terminate_processes:loc_10001470↑r
                                        ; "audiodg.exe"
                dd offset aConhost_exe  ; "conhost.exe"
                dd offset aCsrss_exe    ; "csrss.exe"
                dd offset aDwm_exe      ; "dwm.exe"
                dd offset aExplorer_exe ; "explorer.exe"
                dd offset aLsass_exe    ; "lsass.exe"
                dd offset aLsm_exe      ; "lsm.exe"
                dd offset aServices_exe ; "services.exe"
                dd offset aShutdown_exe ; "shutdown.exe"
                dd offset aSmss_exe     ; "smss.exe"
                dd offset aSpoolss_exe  ; "spoolss.exe"
                dd offset aSpoolsv_exe  ; "spoolsv.exe"
                dd offset aSvchost_exe  ; "svchost.exe"
                dd offset aTaskhost_exe ; "taskhost.exe"
                dd offset aWininit_exe  ; "wininit.exe"
                dd offset aWinlogon_exe ; "winlogon.exe"
                dd offset aWuauclt_exe  ; "wuauclt.exe"
```

Рис. 18. Список процессов, которые не завершаются во время второй попытки.

Маски имен файлов, которые переписывает компонент стирателя:

- SYS_BASCON.COM
- *.v
- *.PL
- *.paf
- *.v
- *.XRF
- *.trc
- *.SCL
- *.bak
- *.cid
- *.scd
- *.pcmp
- *.pcmi
- *.pcmt
- *.xml
- *.CIN
- *.ini
- *.prj
- *.cxm
- *.elb
- *.epl
- *.mdf
- *.ldf
- *.bak
- *.bk
- *.bkp
- *.log
- *.zip
- *.rar
- *.tar
- *.7z

- *.exe
- *.dll

Этот список содержит расширения имен файлов, которые используются в стандартной среде, такие как бинарники Windows (.exe/.dll), архивы (.7z /.tar/.rar/.zip), файлы бэкапа (.bak/.bk/.bkp), файлы сервера Microsoft SQL (.mdf/.ldf) и различные файлы конфигурации (.ini/.xml). Вдобавок компонент стирает файлы, которые могут использоваться в промышленных системах управления, например, написанные при помощи языка описания конфигурации подстанций (.scl/.cid/.scd), а также файлы и расширения, которые используются продуктами ABB.

Например, файл под названием SYS_BASCON.COM используется решениями ABB для хранения информации с конфигурацией, а файлы с расширением .paf (Product Authorization File) – для хранения информации по лицензиям для продуктов ABB MicroSCADA.

Дополнительные инструменты: сканер портов

Арсенал атакующих включает сканер портов, который может быть использован для составления карты сети и поиска компьютеров, относящихся к их атаке. Что интересно, вместо использования уже существующего ПО, хакеры создали собственный сканер портов. Как видно из рис. 19, они могут назначить диапазон IP-адресов и диапазон портов сети, которые будут просканированы этим инструментом.

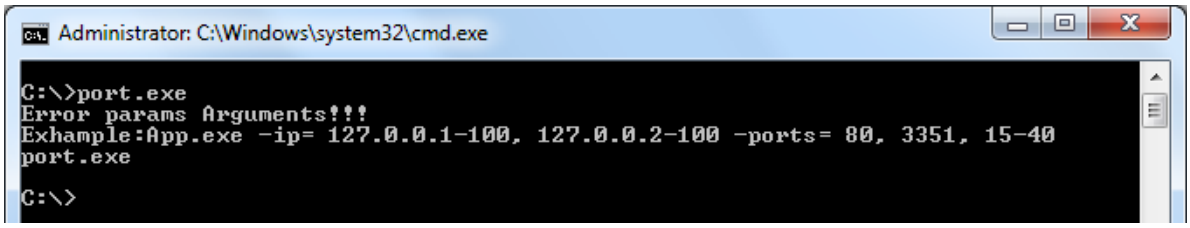


Рис. 19. Пример использования сканера портов.

Дополнительные инструменты: DoS Tool

Еще один инструмент из арсенала хакеров — инструмент отказа в обслуживании (DoS), нацеленный на устройства Siemens SIPROTEC. Инструмент эксплуатирует уязвимость CVE-2015-5374, чтобы вызвать зависание устройства. Как только эта уязвимость будет использована, устройство перестанет отвечать на какие-либо команды до тех пор, пока не будет перезагружено вручную.

Чтобы использовать эту уязвимость, атакующие жестко запрограммировали в DoS-инструменте IP-адреса устройств. При использовании инструмент посылает специально созданные пакеты на порт 50000 на целевой IP-адрес при помощи UDP (User Datagram Protocol). Пакет UDP содержит всего 18 байтов.

Рис. 20. Содержимое пакета UDP, задействованного при использовании уязвимости CVE-2015-5374.

Вывод

Расследование отключения электроэнергии в Киеве в декабре 2016 года пока не завершено. На

данный момент нет подтверждений тому, что Industroyer стал прямой причиной сбоя. Тем не менее, мы считаем эту версию вполне вероятной, поскольку вредоносная программа позволяет напрямую управлять выключателями и прерывателями цепи в сети электрических подстанций при помощи протоколов ICS, а также содержит метку времени активации 17 декабря 2016 года – в день отключения электроэнергии.

Мы можем с уверенностью сказать, что семейство Win32/Industroyer — это продвинутая и сложная вредоносная программа, нацеленная на промышленные системы управления. Проблема в том, что малварь – лишь инструмент в руках еще более продвинутых и высококвалифицированных злоумышленников. Кибергруппа, стоящая за Industroyer, может адаптировать программу для любой подобной среды.

Широко распространенные промышленные протоколы, которые использует Industroyer, создавались десятилетия назад без учета вопросов безопасности. Таким образом, любое вмешательство хакеров в работу промышленной сети, где применяются эти протоколы, заведомо приведет к серьезным последствиям.

Индикаторы заражения (IoCs)

Хеши SHA-1:
F6C21F8189CED6AE150F9EF2E82A3A57843B587D
CCCCE62996D578B984984426A024D9B250237533
8E39ECA1E48240C01EE570631AE8F0C9A9637187
2CB8230281B86FA944D3043AE906016C8B5984D9
79CA89711CDAEDB16B0CCCCFDCFBD6AA7E57120A
94488F214B165512D2FC0438A581F5C9E3BD4D4C
5A5FAFBC3FEC8D36FD57B075EBF34119BA3BFF04
B92149F046F00BB69DE329B8457D32C24726EE00
B335163E6EB854DF5E08E85026B2C3518891EDA8

IP-адреса C&C серверов:
195.16.88[.]6
46.28.200[.]132
188.42.253[.]43
5.39.218[.]152
93.115.27[.]57

Внимание! Большинство серверов с этими IP-адресами были частью сети Tor, что означает, что использование индикаторов может дать ложноположительное совпадение.

Задать дополнительные вопросы или передать образцы вредоносного ПО, имеющие отношение к Win32/Industroyer, можно по электронной почте threatintel@eset.com.