Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский ядерный университет «МИФИ»

Институт интеллектуальных кибернетических систем

Кафедра №42 «Криптология и кибербезопасность»

Отчёт о БДЗ

**S7 Protocol**

Выполнили студенты группы Б17-565:

Волянский Юлиан

Нурматов Салим

Калинкин Артем

Чыонг Тхи Ан Хай

**Технический разбор протокола Siemens S7 Communication**

Протокол Siemens S7 Communication очень тесно связан с продуктами компании Siemens и используется в большинстве случаев для коммуникации внутри сетей с промышленными контроллерами, ЧМИ, ПО и прочим оборудованием компании Siemens.

S7 Communication – функционально-(командно-) ориентированный протокол, который подразумевает то, что каждый переданный пакет содержит в себе команду или ответ на нее.

Данные S7comm поступают как полезная нагрузка пакетов данных COTP. Первый байт всегда 0x32 как идентификатор протокола. Специальные коммуникационные процессоры для серии S7-400 (CP 443) могут использовать этот протокол без уровней TCP / IP.

|  |  |  |
| --- | --- | --- |
|  | **Слой OSI** | **Протокол** |
| 7 | Уровень приложения | S7-связь |
| 6 | Слой презентации | S7-связь |
| 5 | Сессионный уровень | S7-связь |
| 4 | Транспортный уровень | ISO-на-TCP (RFC 1006) |
| 3 | Сетевой уровень | IP |
| 2 | Уровень канала передачи данных | Ethernet |
| 1 | Физический слой | Ethernet |

Сам пакет представляет из себя «матрешку» из различных протоколов – исходный TCP/IP-пакет несет в себе PDU протокола ISO-on-TCP [8], который состоит из двух заголовков протокола TPKT и COTP и инкапсулирует телеграмму протокола S7 Communication. Каждая телеграмма в этом случае будет состоять из следующих частей:

− заголовок;

− набор параметров;

− данные параметров;

− блок данных

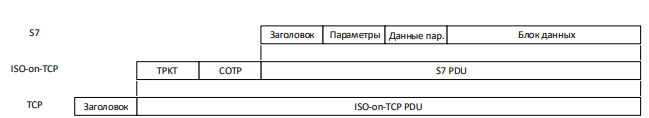


Рисунок 1 – Структура пакета S7 в общем виде

Протокол Siemens S7 обычно используется в системах SCADA для связи между HMI и PLC. Она представляет из себя основанную на модели систему обнаружения вторжений (IDS), разработанную для сетей S7. Подход основан на ключевом наблюдении так, что трафик S7 к и от конкретного ПЛК очень периодичен;

В результате каждый канал HMI-PLC может быть смоделирован с использованием его собственного уникального детерминированного конечного автомата. Полученная в результате IDS на основе DFA очень чувствительна и может выявлять аномалии, например, сообщение появляется не на своем месте в нормальной последовательности. Подход к обнаружению вторжений оценивался на трафике двух производственные системы. Несмотря на высокую чувствительность, у системы был очень низкий уровень ложных срабатываний – более 99,82% трафика было определено как нормальное.

Протокол S7 Communication разделяет все устройства на три типа: клиент, сервер и партнер (соединение клиент-клиент). Отличие партнера от клиента или сервера в том, что партнер может как отправлять запросы, так и обрабатывать их от других партнеров в сети. Обычно в промышленной сети сервером становится ПЛК, клиентами – станции оператора и ЧМИ, а партнерами – ПЛК, которым необходимо передавать и обрабатывать запросы от других ПЛК. В случае партнеров различают: активных партнеров и пассивных партнеров. Активным партнером называют клиента, который инициирует соединение, а пассивным – клиент, который принимает это соединение.

Чтобы установить соединение с S7 PLC, необходимо выполнить 3 шага:

1. Подключиться к ПЛК через TCP-порт 102
2. Подключение на уровне ISO (запрос на подключение COTP)
3. Подключиться на уровне S7comm (s7comm.param.func = 0xf0, установить связь)

Шаг 1) использует IP-адрес PLC / CP.

Шаг 2) использует в качестве получателя TSAP длиной два байта. Первый байт TSAP назначения кодирует тип связи (1 = PG, 2 = OP). Второй байт TSAP назначения кодирует номер стойки и слота: это позиция ЦП ПЛК. Номер слота кодируется битами 0-4, номер стойки кодируется битами 5-7.

Шаг 3) предназначен для согласования конкретных деталей S7comm (например, размера PDU).

**Процесс передачи информации с использованием протокола S7 Communication.**

В данном случае клиентом будет выступать станция оператора технологического процесса, а сервером будет являться ПЛК S7-300 компании Siemens. Контроллер при этом будет обладать следующей аппаратной структурой: − центральный процессор (ЦП, CPU); − коммуникационный процессор (КП, CP); − память

Центральный процессор требуется для выполнения команд и функций программной прошивки устройства, а также обрабатывает принятые запросы, коммуникационный процессор выполняет функцию управления соединениями, принимает и отправляет запросы и ответы, в памяти же находится код программ пользовательских и системных функций, а также других данных. Компанией Siemens предусмотрена возможность использования как совмещенного коммуникационного и центрального процессора, так и раздельная реализация в виде отдельного модуля коммуникационного процессора, подключаемого к контроллеру

Для создания соединения клиенту требуется сначала указать IP-адрес сервера, локальную точку доступа транспортных сервисов (TSAP) и удаленную TSAP. Локальной и удаленной TSAP в данном случае будет являться стойка и слот, в котором содержится коммуникационный процессор ПЛК соответственно

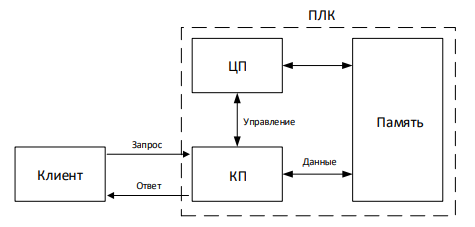


Рисунок 2 –модель коммуникации станции оператора и ПЛК в общем виде.

**Структура пакета S7 Communication.**

Как было упомянуто ранее, PDU пакета S7 Communication начинается с заголовка. Заголовок имеет длину от 10 до 12 байт (при получении подтверждения задействуются 2 последних байта), а также следующую структуру: − идентификатор протокола (1 байт); − тип сообщения (1 байт); − зарезервировано (2 байта); − ссылка на PDU (2 байта); − длина поля набора параметров и данных параметров (2 байта); − длина поля данных (2 байта); − класс ошибки (1 байт); − код ошибки (1 байт).

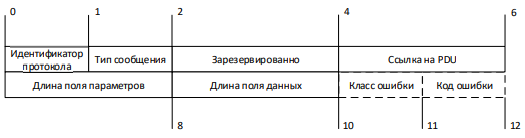


Рисунок 3 – структура пакета S7 Communication.

Идентификатор протокола служит для того, чтобы явно идентифицировать протокол S7 Communication. Ссылка на PDU генерируется инициатором соединения и ее значение увеличивается с каждой передачей, ссылка требуется для синхронизации передачи данных.

Тип сообщения отвечает за структуру пакета далее. Существует четыре типа сообщения:

− Job request (0x01) – запрос на выполнение типичных функций;

− Ack (0x02) – подтверждение (результат) запроса без поля данных;

− Ack-data (0x03) – подтверждение (результат) запроса с полем данных;

− Userdata (0x07) – функции расширенной версии протокола.

В начале каждой коммуникационной сессии обе стороны соединения договариваются о максимальной длине PDU, а также о максимальном количестве потоков обработки. Данная операция производится при помощи функции TCON (0хF0) сообщения типа Job request. При этом в пакете используются только заголовок, а также поля параметров, остальные поля не используются.



Рисунок 4 – структура поля параметров функции TCON

Протоколом S7 Communication также поддерживаются функции защиты записи и чтения/записи при помощи установления пароля для соединения. Стоит отметить, что максимальной длиной пароля является 6 байт. Причем перед передачей пароля производится побитовая операция исключающего ИЛИ с константой 0х55 и i-2 символом пароля. В таком виде пароль без труда поддается дешифровки без применения каких-либо серьезных технических средств. Данные недостатки, а также факт того, что потенциальный злоумышленник может украсть коммуникационную сессию и использовать аутентификационные данные даже без их расшифровки, наводит на вывод о недостаточности и неэффективности защитных мер самом протоколе передачи данных S7 Communication.

**Строение пакета при типичных операциях чтения и записи**.

Операции чтения и записи определены для типа сообщения Job request. После указания кода функции необходимо указать количество объектов, которые будут считаны или записаны. Количество таких объектов ограничено длиной поля (1 байт), однако обычно количество таких объектов на практике редко превышает 20 штук. Далее идут указания параметров для каждого объекта, после чего пакет завершается (если это запрос на чтение) либо идет поле данных (если это ответ сервера, либо запрос на запись), в котором указываются параметры данных для каждого объекта. Поле параметров для каждого объекта имеет следующую структуру:

− спецификация переменной (1 байт), определяет структуру объекта, для функций чтения/записи равен 0х12;

− длина поля (1 байт);

− режим адресации (1 байт);

− тип переменной (1 байт);

− количество обрабатываемых переменных (2 байта);

− номер блока данных (2 байта), указывается при выборе соответствующей зоны памяти, иначе игнорируется;

− зона памяти (1 байт);

− смещение в памяти (3 байта). Если присутствует, то поле данных для каждого соответствующего объекта будет обладать следующей структурой:

− код ошибки (1 байт), при успешном чтении/записи равен 0хFF, при запросе на запись равен 0х00, в иных случаях равен коду соответствующей ошибки;

− спецификация переменной (1 байт), равна соответствующему значению из поля параметров;

− количество обрабатываемых переменных (2 байта), равно соответствующему значению из поля параметров;

− данные, длина поля рассчитывается как «длинна переменной \* количество переменных» Протокол S7 Communication поддерживает три режима адресации при операциях чтения/записи:

− режим адресации по умолчанию (0x10) с указанием зоны памяти, длины переменной и смещения;

− режим адресации по блоку памяти (0xB0), который используется в контроллерах серии S7-400;

− режим символьной адресации (0xB2), который используется в контроллерах серии S7-1200/1500. Для контроллера серии S7-300 рекомендуется использовать режим по умолчанию, соответственно далее будут указаны параметры, которые необходимы для запроса с таким режимом адресации. При таком режиме необходимо указать соответствующую зону памяти, тип переменной и ее смещение в памяти. Допустимые зоны памяти являются следующими:

− периферия;

− входы;

− выходы

− флаги;

− блоки данных;

− локальные данные;

− счетчики;

− таймеры.

Все перечисленные зоны памяти соответствуют зонам памяти на контроллере, указанных ранее. Протокол поддерживает обширное число типов переменных, таких как: бит (0х01), байт (0х02), символ (0х03), слово (0х04), целое число (0х05), действительное число (0х08), дата и время (0х0F) т.д.

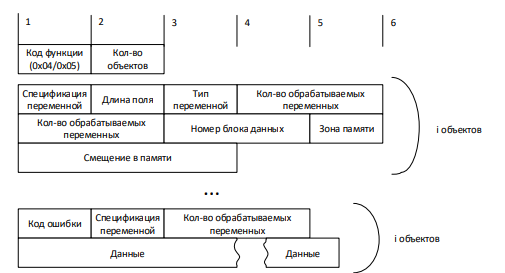


Рисунок 5 – структура пакета S7 Communication для команд чтения/записи

Спецификацией протокола также определены такие функции как:

− загрузка и выгрузка отдельных блоков;

− управление ПЛК с подфункциями запуска и удаления блоков, а также создания дампа оперативной памяти ПЛК;

− остановка ПЛК;

− чтение системной информации;

− получение списка блоков в памяти, а также информации о них;

− чтение и изменение времени. Все перечисленные функции также влияют на структуру пакета, но они не будут подробно описаны далее, так как это не входит в задачи данной работы.

Обобщая описанные ранее технические характеристики протокола S7 Communication, можно сказать, что данный протокол качественно отличается от протокола Modbus более усложненной структурой пакета, а также тем, что он фактически привязан ко всей системе продуктов промышленной автоматизации от компании Siemens, так как является проприетарным протоколом. Стоит отметить, что протокол S7 Communication позволяет использовать обширный функционал ПЛК от компании Siemens, например, выполнение встроенных системных функций, загрузка и выгрузка блоков в памяти, что упрощает диагностику и отладку ПО контроллера. Также стоит отметить, что протоколом предусмотрены некоторые функции безопасности, в отличие от протокола Modbus, однако, данные функции не являются серьезным препятствием для потенциального нарушителя, что делает их крайне неэффективными решениями.

# Разработка системы клиент-сервер

Для проектирования системы клиент-сервер с использованием протокола S7comm использовалась библиотека snap7.

Перед установкой модуля следует провести операции:

$ sudo add-apt-repository ppa:gijzelaar/snap7

$ sudo apt-get update

$ sudo apt-get install libsnap7-1 libsnap7-dev

Для того, чтобы установить модуль:

$ pip install python-snap7

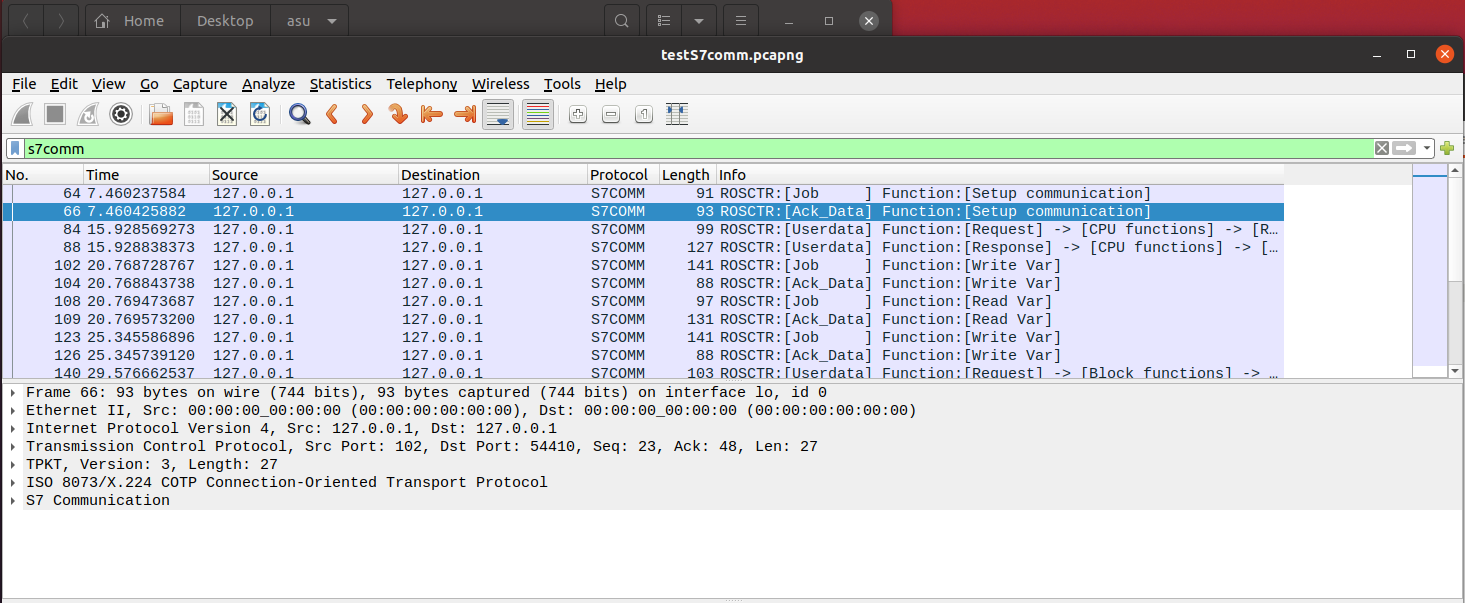
Далее надо запустить сервер:

$ sudo python3 server.py

И потом запускать клиента:

$ python3 client\_test.py

У клиента есть несколько опций, которые позволяют протестировать работу протокола, а также снять дамп процесса обмены с помощью wireshark.



В примере представлен набор пакетов, которые посылает клиент серверу и обратно. В первой части видно, что происходит подключение к серверу и начало процесса обмена данными.

В консоли клиента есть следующие возможности:

1. Start client
2. Get cpu state
3. Test DB read
4. Test DB write
5. Test DB get
6. Test list blocks
7. Test get block\_info

Они позволяют эмулировать работу асу, в которой используется данный протокол. Например, можно узнать состояние процессоров или считать или записать данные. Так же можно список блоков и информацию о них.

# Описание созданных правил для Suricata.

В начале разработки было принято решение написать модуль для работы с протоколом S7comm с помощью ПО Suricata. Однако в силу того, что возникли сложности с установкой модуля, то было принято решение написать набор правил для детектирования протокола и его проблемных точек.

1. Правило:

alert tcp 127.0.0.1 any -> 127.0.0.1 any (msg: "Request for CPU parametres"; content: "|00 01 12|"; offset: 17; depth: 3; rawbytes; sid: 121;)

Оно позволяет определить, что были запрошены параметры процессоров.

1. Правило:

alert tcp 127.0.0.1 102 -> 127.0.0.1 any (msg: "Ack\_Data and function WriteVar"; content: "|32 01|"; distance: 4; rawbytes; sid: 123;)

Данное правило позволяет определить, когда происходит запрос на запись со стороны клиента.

1. Правило:

alert tcp 127.0.0.1 any -> 127.0.0.1 any (msg: "Detected S7Comm with Ack\_Data"; content: "|32 03|"; offset: 8; sid: 124;)

Данное правило позволяет определить, когда был отправлен пакет с идентификатором Ack\_Data.

1. Правило:

alert tcp 127.0.0.1 any -> 127.0.0.1 any (msg: "Setup communication"; content: "|f0|"; offset: 19; sid: 125;)

Данное правило позволяет увидеть, когда начинает клиент хочет установить связь с сервером.

1. Правило:

alert tcp 127.0.0.1 any <> 127.0.0.1 any (msg: "Test"; sid:122;)

Данное правило является тестовым для проверки работы правил Suricata

1. Правило:

alert tcp 127.0.0.1 any -> 127.0.0.1 any (msg: "Setup communication 2"; content: "|03 00 00 1f|"; sid: 127;)

Оно позволяет увидеть, когда приходит ответ от сервера на соединение с клиентом.

1. Правило:

alert tcp 127.0.0.1 any -> 127.0.0.1 any (msg: "Big length"; dsize: >100; sid: 128;)

Оно позволяет определить, когда данные превышают размер в 100 байт.

1. Правило:

alert tcp 127.0.0.1 any -> 127.0.0.1 any (msg: "Write data"; content: "|05|"; offset: 18; sid: 129; classtype: not-suspicious;)

Данное правило позволяет определить, когда происходит запрос на запись данных со стороны клиента.

1. Правило:

alert tcp 127.0.0.1 any -> 127.0.0.1 any (msg: "User data"; content: "|32 07|"; offset: 7; content: "|00 01 12|"; distance: 8; classtype: not-suspicious; sid: 130;)

Позволяет определить, когда пакет несет в себе пользовательские данные.

1. Правило:

alert tcp 127.0.0.1 any -> 127.0.0.1 any (msg: "Reading var"; content: "|32 01|"; offset: 7; content: "|04|"; distance: 8; classtype: not-suspicious; sid: 131;)

Данное правило позволяет определить, когда происходит запрос на чтение данных.

1. Правило:

alert tcp 127.0.0.1 any -> 127.0.0.1 any (msg: "Success returning"; content: "|ff|"; offset: 30; sid: 132;)

Позволяет определить, когда прошло успешное считывание или запись.

1. Правило:

alert tcp 127.0.0.1 any -> 127.0.0.1 any (msg: "Detected S7Comm with Job"; content: "|32 01|"; offset: 8; sid: 133;)

Данное правило позволяет определить, когда был отправлен пакет с идентификатором Job.

1. Правило:

alert tcp 127.0.0.1 any -> 127.0.0.1 any (msg: "Testing the size of setup communication"; dsize: 25; sid: 134;)

Данное правило позволяет определить, когда пакеты имеют длину 25 байт в случае установления связи между клиентом и сервером.

В приложении А показано, как Suricata производит логирование и детектирование по пакетам, которыми обмениваются клиент и сервер. Данный пример основан на наборе пакетов, которые были получены в течении тестирования клиента – сервера.

# Настройка правил

Для того, чтобы настроить правила, следует установить пакет Suricata:

1. sudo su

2. apt-get update && apt-get upgrade -y

3. apt-get install libpcre3 libpcre3-dbg libpcre3-dev build-essential libpcap-dev \

libnet1-dev libyaml-0-2 libyaml-dev pkg-config zlib1g zlib1g-dev \

libcap-ng-dev libcap-ng0 make libmagic-dev libjansson-dev \

libnss3-dev libgeoip-dev liblua5.1-dev libhiredis-dev libevent-dev liblz4-dev \

m4 autoconf autogen

4. git clone https://github.com/OISF/suricata.git

5. cd suricata

6. git clone https://github.com/OISF/libhtp

7. ./autogen.sh

8. ./configure --prefix=/usr --sysconfdir=/etc --localstatedir=/var

9. python3 scripts/setup-app-layer.py --detect --logger --parser S7 Commbuf

10. make

Далее требуется прописать правила в файле local.rules, которые находятся в: /etc/suricata/rules

После этого следует запустить клиент-сервер и Suricata:

sudo suricata -c /etc/suricata/suricata.yaml -i lo --set capture.disable-offloading=false

В файле /var/log/suricata/fast.log можно проверить детектирование пакетов и работу правил.

ПРИЛОЖЕНИЕ А.

11/09/2020-11:42:51.657111 [\*\*] [1:122:0] Test [\*\*] [Classification: (null)] [Priority: 3] {TCP} 127.0.0.1:34564 -> 127.0.0.1:35627

11/09/2020-11:42:51.671937 [\*\*] [1:122:0] Test [\*\*] [Classification: (null)] [Priority: 3] {TCP} 127.0.0.1:38316 -> 127.0.0.1:43303

11/09/2020-11:42:51.672009 [\*\*] [1:122:0] Test [\*\*] [Classification: (null)] [Priority: 3] {TCP} 127.0.0.1:43303 -> 127.0.0.1:38316

11/09/2020-11:42:51.657179 [\*\*] [1:122:0] Test [\*\*] [Classification: (null)] [Priority: 3] {TCP} 127.0.0.1:35627 -> 127.0.0.1:34564

11/09/2020-11:42:54.887243 [\*\*] [1:128:0] Big length [\*\*] [Classification: (null)] [Priority: 3] {TCP} 127.0.0.1:38316 -> 127.0.0.1:43303

11/09/2020-11:42:59.115588 [\*\*] [1:122:0] Test [\*\*] [Classification: (null)] [Priority: 3] {TCP} 127.0.0.1:54410 -> 127.0.0.1:102

11/09/2020-11:42:59.115647 [\*\*] [1:122:0] Test [\*\*] [Classification: (null)] [Priority: 3] {TCP} 127.0.0.1:102 -> 127.0.0.1:54410

11/09/2020-11:42:54.922982 [\*\*] [1:128:0] Big length [\*\*] [Classification: (null)] [Priority: 3] {TCP} 127.0.0.1:38316 -> 127.0.0.1:43303

11/09/2020-11:42:59.117349 [\*\*] [1:134:0] Testing the size of setup communication [\*\*] [Classification: (null)] [Priority: 3] {TCP} 127.0.0.1:54410 -> 127.0.0.1:102

11/09/2020-11:42:59.117537 [\*\*] [1:125:0] Setup communication [\*\*] [Classification: (null)] [Priority: 3] {TCP} 127.0.0.1:102 -> 127.0.0.1:54410

11/09/2020-11:43:07.585680 [\*\*] [1:121:0] Request for CPU parametres [\*\*] [Classification: (null)] [Priority: 3] {TCP} 127.0.0.1:54410 -> 127.0.0.1:102

11/09/2020-11:43:07.585680 [\*\*] [1:130:0] User data [\*\*] [Classification: Not Suspicious Traffic] [Priority: 3] {TCP} 127.0.0.1:54410 -> 127.0.0.1:102

11/09/2020-11:43:07.585949 [\*\*] [1:121:0] Request for CPU parametres [\*\*] [Classification: (null)] [Priority: 3] {TCP} 127.0.0.1:102 -> 127.0.0.1:54410

11/09/2020-11:43:07.585949 [\*\*] [1:130:0] User data [\*\*] [Classification: Not Suspicious Traffic] [Priority: 3] {TCP} 127.0.0.1:102 -> 127.0.0.1:54410

11/09/2020-11:43:07.585949 [\*\*] [1:132:0] Success returning [\*\*] [Classification: (null)] [Priority: 3] {TCP} 127.0.0.1:102 -> 127.0.0.1:54410

11/09/2020-11:42:54.937897 [\*\*] [1:128:0] Big length [\*\*] [Classification: (null)] [Priority: 3] {TCP} 127.0.0.1:38316 -> 127.0.0.1:43303

11/09/2020-11:43:12.425840 [\*\*] [1:131:0] Reading var [\*\*] [Classification: Not Suspicious Traffic] [Priority: 3] {TCP} 127.0.0.1:54410 -> 127.0.0.1:102

11/09/2020-11:43:48.621798 [\*\*] [1:128:0] Big length [\*\*] [Classification: (null)] [Priority: 3] {TCP} 127.0.0.1:38316 -> 127.0.0.1:43303

11/09/2020-11:43:12.425955 [\*\*] [1:129:0] Write data [\*\*] [Classification: Not Suspicious Traffic] [Priority: 3] {TCP} 127.0.0.1:102 -> 127.0.0.1:54410

11/09/2020-11:43:12.426585 [\*\*] [1:131:0] Reading var [\*\*] [Classification: Not Suspicious Traffic] [Priority: 3] {TCP} 127.0.0.1:54410 -> 127.0.0.1:102

11/09/2020-11:43:48.634278 [\*\*] [1:128:0] Big length [\*\*] [Classification: (null)] [Priority: 3] {TCP} 127.0.0.1:38316 -> 127.0.0.1:43303

11/09/2020-11:43:17.002698 [\*\*] [1:131:0] Reading var [\*\*] [Classification: Not Suspicious Traffic] [Priority: 3] {TCP} 127.0.0.1:54410 -> 127.0.0.1:102

11/09/2020-11:43:17.002850 [\*\*] [1:129:0] Write data [\*\*] [Classification: Not Suspicious Traffic] [Priority: 3] {TCP} 127.0.0.1:102 -> 127.0.0.1:54410

11/09/2020-11:43:21.233774 [\*\*] [1:121:0] Request for CPU parametres [\*\*] [Classification: (null)] [Priority: 3] {TCP} 127.0.0.1:54410 -> 127.0.0.1:102

11/09/2020-11:43:21.233774 [\*\*] [1:130:0] User data [\*\*] [Classification: Not Suspicious Traffic] [Priority: 3] {TCP} 127.0.0.1:54410 -> 127.0.0.1:102

11/09/2020-11:43:21.233847 [\*\*] [1:121:0] Request for CPU parametres [\*\*] [Classification: (null)] [Priority: 3] {TCP} 127.0.0.1:102 -> 127.0.0.1:54410

11/09/2020-11:43:21.233847 [\*\*] [1:128:0] Big length [\*\*] [Classification: (null)] [Priority: 3] {TCP} 127.0.0.1:102 -> 127.0.0.1:54410

11/09/2020-11:43:21.233847 [\*\*] [1:129:0] Write data [\*\*] [Classification: Not Suspicious Traffic] [Priority: 3] {TCP} 127.0.0.1:102 -> 127.0.0.1:54410

11/09/2020-11:43:21.233847 [\*\*] [1:130:0] User data [\*\*] [Classification: Not Suspicious Traffic] [Priority: 3] {TCP} 127.0.0.1:102 -> 127.0.0.1:54410

11/09/2020-11:43:48.646693 [\*\*] [1:128:0] Big length [\*\*] [Classification: (null)] [Priority: 3] {TCP} 127.0.0.1:38316 -> 127.0.0.1:43303

11/09/2020-11:43:48.646708 [\*\*] [1:128:0] Big length [\*\*] [Classification: (null)] [Priority: 3] {TCP} 127.0.0.1:43303 -> 127.0.0.1:38316

11/09/2020-11:43:21.234454 [\*\*] [1:131:0] Reading var [\*\*] [Classification: Not Suspicious Traffic] [Priority: 3] {TCP} 127.0.0.1:54410 -> 127.0.0.1:102

11/09/2020-11:43:48.659772 [\*\*] [1:128:0] Big length [\*\*] [Classification: (null)] [Priority: 3] {TCP} 127.0.0.1:38316 -> 127.0.0.1:43303

11/09/2020-11:43:21.234550 [\*\*] [1:128:0] Big length [\*\*] [Classification: (null)] [Priority: 3] {TCP} 127.0.0.1:102 -> 127.0.0.1:54410

11/09/2020-11:43:48.700416 [\*\*] [1:128:0] Big length [\*\*] [Classification: (null)] [Priority: 3] {TCP} 127.0.0.1:38316 -> 127.0.0.1:43303

11/09/2020-11:43:26.706149 [\*\*] [1:121:0] Request for CPU parametres [\*\*] [Classification: (null)] [Priority: 3] {TCP} 127.0.0.1:54410 -> 127.0.0.1:102

11/09/2020-11:43:26.706149 [\*\*] [1:130:0] User data [\*\*] [Classification: Not Suspicious Traffic] [Priority: 3] {TCP} 127.0.0.1:54410 -> 127.0.0.1:102

11/09/2020-11:43:26.706302 [\*\*] [1:121:0] Request for CPU parametres [\*\*] [Classification: (null)] [Priority: 3] {TCP} 127.0.0.1:102 -> 127.0.0.1:54410

11/09/2020-11:43:26.706302 [\*\*] [1:130:0] User data [\*\*] [Classification: Not Suspicious Traffic] [Priority: 3] {TCP} 127.0.0.1:102 -> 127.0.0.1:54410

11/09/2020-11:43:31.833690 [\*\*] [1:121:0] Request for CPU parametres [\*\*] [Classification: (null)] [Priority: 3] {TCP} 127.0.0.1:54410 -> 127.0.0.1:102

11/09/2020-11:43:31.833690 [\*\*] [1:130:0] User data [\*\*] [Classification: Not Suspicious Traffic] [Priority: 3] {TCP} 127.0.0.1:54410 -> 127.0.0.1:102

11/09/2020-11:43:31.833769 [\*\*] [1:121:0] Request for CPU parametres [\*\*] [Classification: (null)] [Priority: 3] {TCP} 127.0.0.1:102 -> 127.0.0.1:54410

11/09/2020-11:43:31.833769 [\*\*] [1:128:0] Big length [\*\*] [Classification: (null)] [Priority: 3] {TCP} 127.0.0.1:102 -> 127.0.0.1:54410

11/09/2020-11:43:31.833769 [\*\*] [1:129:0] Write data [\*\*] [Classification: Not Suspicious Traffic] [Priority: 3] {TCP} 127.0.0.1:102 -> 127.0.0.1:54410

11/09/2020-11:43:31.833769 [\*\*] [1:130:0] User data [\*\*] [Classification: Not Suspicious Traffic] [Priority: 3] {TCP} 127.0.0.1:102 -> 127.0.0.1:54410

11/09/2020-11:43:51.974143 [\*\*] [1:123:0] Ack\_Data and function WriteVar [\*\*] [Classification: (null)] [Priority: 3] {TCP} 127.0.0.1:54410 -> 127.0.0.1:102

11/09/2020-11:43:51.974143 [\*\*] [1:125:0] Setup communication [\*\*] [Classification: (null)] [Priority: 3] {TCP} 127.0.0.1:54410 -> 127.0.0.1:102

11/09/2020-11:43:51.974143 [\*\*] [1:127:0] Setup communication 2 [\*\*] [Classification: (null)] [Priority: 3] {TCP} 127.0.0.1:54410 -> 127.0.0.1:102

11/09/2020-11:43:51.974143 [\*\*] [1:129:0] Write data [\*\*] [Classification: Not Suspicious Traffic] [Priority: 3] {TCP} 127.0.0.1:54410 -> 127.0.0.1:102

11/09/2020-11:43:51.974143 [\*\*] [1:130:0] User data [\*\*] [Classification: Not Suspicious Traffic] [Priority: 3] {TCP} 127.0.0.1:54410 -> 127.0.0.1:102

11/09/2020-11:43:51.974143 [\*\*] [1:131:0] Reading var [\*\*] [Classification: Not Suspicious Traffic] [Priority: 3] {TCP} 127.0.0.1:54410 -> 127.0.0.1:102

11/09/2020-11:43:51.974143 [\*\*] [1:132:0] Success returning [\*\*] [Classification: (null)] [Priority: 3] {TCP} 127.0.0.1:54410 -> 127.0.0.1:102

11/09/2020-11:43:51.974143 [\*\*] [1:133:0] Detected S7Comm with Job [\*\*] [Classification: (null)] [Priority: 3] {TCP} 127.0.0.1:54410 -> 127.0.0.1:102

11/09/2020-11:43:51.974956 [\*\*] [1:124:0] Detected S7Comm with Ack\_Data [\*\*] [Classification: (null)] [Priority: 3] {TCP} 127.0.0.1:102 -> 127.0.0.1:54410

11/09/2020-11:43:51.974956 [\*\*] [1:125:0] Setup communication [\*\*] [Classification: (null)] [Priority: 3] {TCP} 127.0.0.1:102 -> 127.0.0.1:54410

11/09/2020-11:43:51.974956 [\*\*] [1:129:0] Write data [\*\*] [Classification: Not Suspicious Traffic] [Priority: 3] {TCP} 127.0.0.1:102 -> 127.0.0.1:54410

11/09/2020-11:43:51.974956 [\*\*] [1:130:0] User data [\*\*] [Classification: Not Suspicious Traffic] [Priority: 3] {TCP} 127.0.0.1:102 -> 127.0.0.1:54410

11/09/2020-11:43:51.974956 [\*\*] [1:132:0] Success returning [\*\*] [Classification: (null)] [Priority: 3] {TCP} 127.0.0.1:102 -> 127.0.0.1:54410