НИЯУ «МИФИ»

Кафедра №42 «Криптология и Кибербезопасность»

Информационная безопасность АСУ ТП

Отчет по лабораторной работе

“Добавление модуля анализа промышленного протокола S7 Communication (S7comm) в ПО Suricata.”

Выполнили студенты группы Б17-505:

Иванова Д.С.

Казьмин С.К.

Андрюшин М.А.

2020

СОДЕРЖАНИЕ

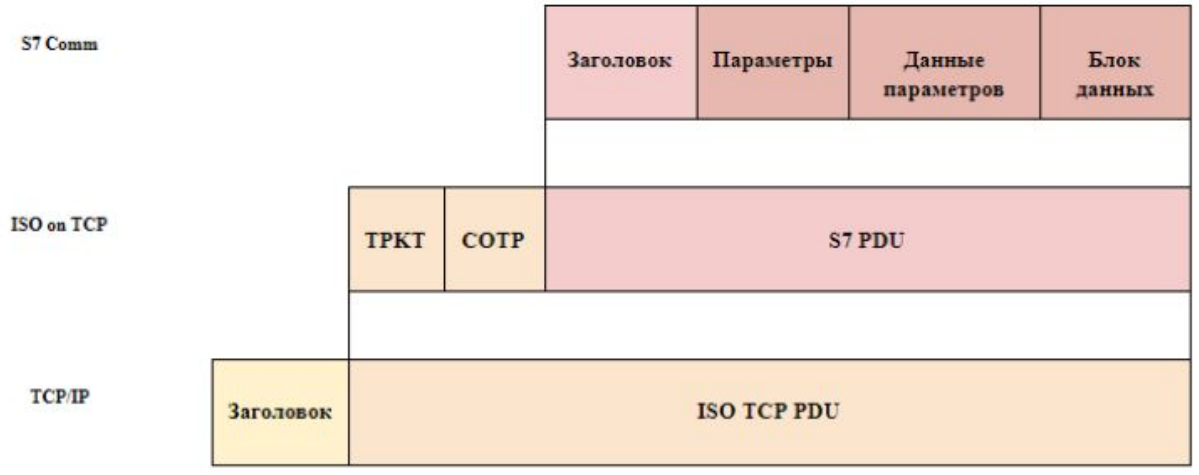
Описание протокола S7 Communication (S7comm)

S7 Communication – проприетарный протокол компании Siemens, разработан в 1994 году, когда появилась серия продуктов SIMATIC S7. Используется для коммуникации между PLC SIMATIC (S-200, S-300, S-400, S-1200, S-1500).

S7comm основан на блочно-ориентированном транспортном протоколе ISO, который базируется на стеке протоколов TCP/IP. Каждый блок (телеграмма) называется PDU (Protocol Data Unit) и состоит из заголовка, набора параметров, данных параметров, блока данных. Длина PDU зависит от CP (communication processor) и согласовывается во время соединения. Если места не хватает, то данные делятся на несколько PDU.

В отличии от других протоколов прикладного уровня, протокол S7comm не инкапсулируется напрямую в TCP/IP. Сначала он инкапсулируется в протокол COTP (Connection Oriented Transport Protocol), а затем в TPKT (ISO transport services on top of the TCP), что позволяет передавать PDU через TCP/IP.

Структура пакета S7 Communication



Уровни OSI для S7 Communication:

1. Физический – Ethernet
2. Канальный – Ethernet
3. Сетевой – IP

4. Транспортный – ISO-on-TCP (RFC 1006)
5. Сеансовый – S7 communication
6. Представления – S7 communication
7. Прикладной – S7 communication

Структура заголовка протокола S7 Communication

Бит	0-7	8-15	16-31	
0	Идентификатор протокола	Тип сообщения	Зарезервировано	
32	Ссылка на PDU		Длина поля параметров	
64	Длина поля данных		Класс ошибки	Код ошибки

Идентификатор протокола S7comm – константа 0x32.

Типы сообщения (или ROSCTR):

- Job request (0x01) – запрос на выполнение типичных функций (например, чтение/запись блоков, запуск/остановка устройства, настройка связи);
- Ack (0x02) – подтверждение запроса без поля данных;
- Ack-data (0x03) – подтверждение запроса с полем данных;
- Userdata (0x07) – функции расширенной версии протокола.

Зарезервировано - по умолчанию 0x0000.

Ссылка на PDU генерируется инициатором соединения и ее значение увеличивается с каждой передачей, ссылка требуется для синхронизации передачи данных.

Класс ошибки и *код ошибки* присутствуют только в сообщениях типа Ack-data.

В приложении А представлены все возможные значения полей протокола.

Структура поля параметров протокола S7 Communication

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 (0xF0) сообщения типа Job request. При этом в пакете используются только заголовок, а также поля параметров, остальные поля не используются. Структура поля параметров функции TCON представлена на рисунке.

Бит	0 - 7	8 - 15	15 - 31	31 - 47	47 - 63
	Код функции	**Зарезервировано**	**Макс. кол-во потоков запросившего**	**Макс. кол-во потоков ответившего**	**Макс. длина PDU**

Описание служб

- PUT / GET – эта служба представляет собой однонаправленную службу чтения/записи для передачи небольших объемов данных на устройство и с него.
- BSEND / BRCV – этот сервис представляет собой двунаправленный и блочно-ориентированный сервис для передачи больших объемов данных между двумя устройствами.
- USEND / URCV – этот сервис представляет собой двунаправленный и нескоординированный сервис для передачи небольших объемов данных между двумя ус-вами.

Протокол S7 позволяет передавать данные от 1 байта до 64 Кбайт. Максимальный размер данных зависит от используемой службы и используемого процессора S7.

Процесс передачи информации с использованием S7comm

Допустим, что PLC состоит из: *центральный процессор (CPU)*; *коммуникационный процессор (CP)*; *память*.

CPU нужен для выполнения команд и функций программной прошивки устройства и для обработки принятых запросов.

CP управляет соединениями, принимает и отправляет запросы и ответы.

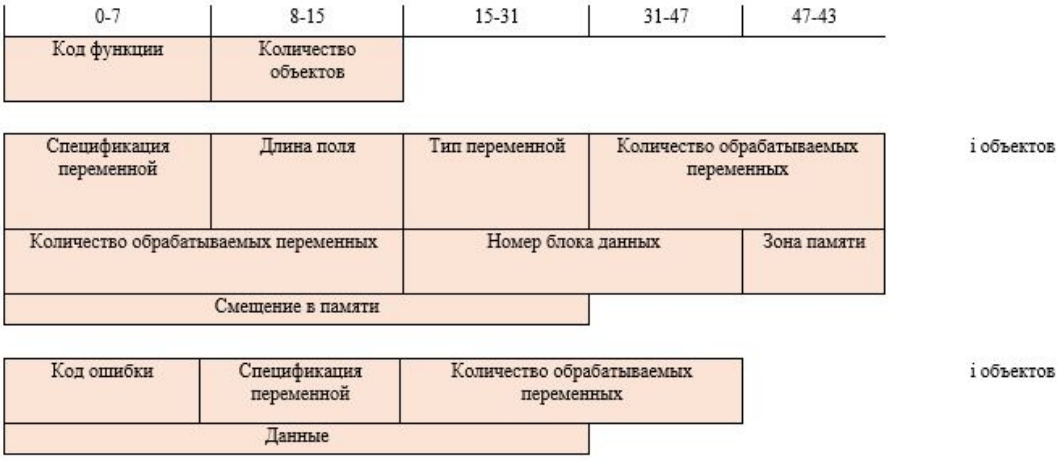
В **памяти** находится код программ пользовательских и системных функций, а также других данных.

Для создания соединения клиент должен 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 CP PLC. Также возможно, но не обязательно указать тип соединения, используя одну из трех констант: PG (консоль программирования), OP (HMI) или S7 Basic (обычное соединение). Константа PG имеет значение 0x01, OP – 0x02, а S7 Basic может иметь значение от 0x03 до 0x10.

Защита записи/чтения и её уязвимости

S7comm поддерживает защиту записи и чтения/записи при помощи установления пароля для соединения. Такая защита устраняет возможность несанкционированного выполнения только некоторых функций чтения и записи. Такие функции, как чтение системной информации, а также чтение и запись в область флагов не защищены системой аутентификации. Максимальная длина пароля составляет 6 байт, причем перед передачей пароля производится побитовый XOR с константой 0x55 и i-2 символом пароля. В таком виде пароль можно без труда восстановить. К тому же потенциальный злоумышленник может украсть коммуникационную сессию и использовать аутентификационные данные даже без их расшифровки.

Структура пакета при операциях чтения/записи



Операции чтения и записи определены для типа сообщения Job request с кодами функции 0x04 и 0x05. После указания кода функции указывается количество объектов, которые будут считаны или записаны (оно ограничено длиной поля (1 байт)). Далее указаны параметры 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 0x12;
- 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).

Поле данных (если оно присутствует) для каждого соответствующего объекта будет обладать следующей структурой:

- код ошибки (1 байт), при успешном чтении/записи - 0xFF, при запросе на запись - 0x00, иначе 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 данных рассчитывается как «длина переменной * количество переменных»

Протокол S7 Communication поддерживает три режима адресации при операциях чтения/записи:

- по умолчанию (0x10) с указанием зоны памяти, длины переменной и смещения;
- по блоку памяти (0xB0), который используется в контроллерах серии S7-400;
- режим символьной адресации (0xB2), который используется в контроллерах серии S7-1200/1500.

При режиме адресации по умолчанию необходимо указать соответствующую зону памяти, тип переменной и ее смещение в памяти. Допустимые зоны памяти являются следующими:

- периферия (0x80);
- входы (0x81);
- выходы (0x82);
- флаги (0x83);
- блоки данных (0x84);
- локальные данные (0x86);
- счетчики (0x1C);
- таймеры (0x1D).

Все перечисленные зоны памяти соответствуют зонам памяти на контроллере. Протокол поддерживает обширное число типов переменных, таких как: бит (0x01), байт (0x02), символ (0x03), слово (0x04), целое число (0x05), действительное число (0x08), дата и время (0x0F) и т.д.

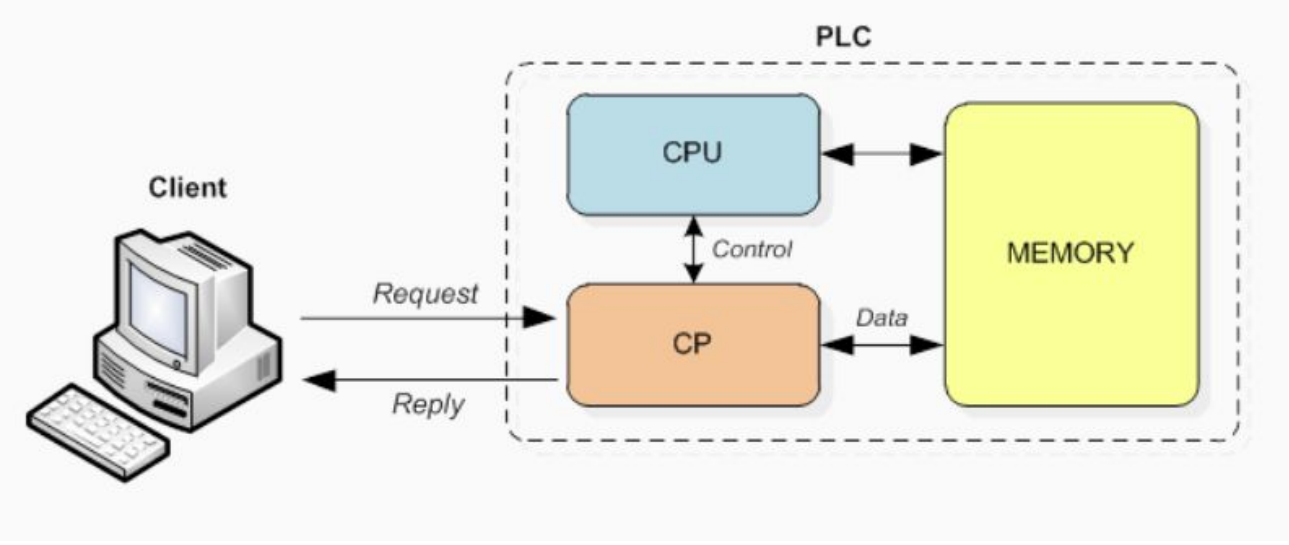
Функции протокола

Кроме чтения/записи доступны такие функции как:

- загрузка и выгрузка отдельных блоков;
- управление PLC с подфункциями запуска и удаления блоков;
- создания дампа оперативной памяти PLC;
- остановка PLC;
- чтение системной информации;
- получение списка блоков в памяти, а также информации о них;
- чтение и изменение времени.

Сценарии использования протокола

1) PLC - сервер, другие устройства являются клиентами. Они подключаются к внутреннему серверу коммуникационного процессора (CP) и делают запрос S7.



Сервер отвечает телеграммой ответа S7.

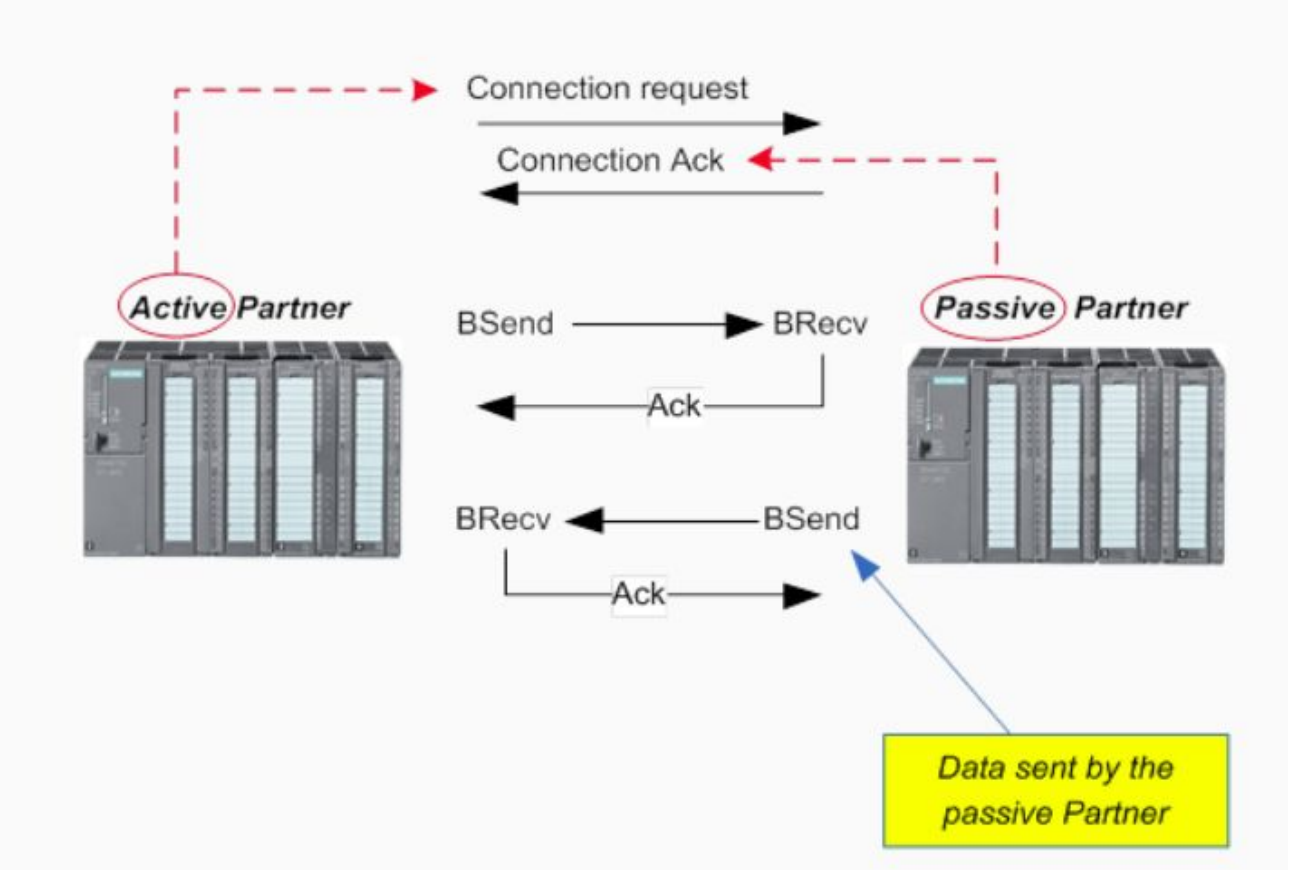
Никакая конфигурация не требуется на стороне сервера. Служба сервера автоматически обрабатывается встроенным программным обеспечением CP.

CP могут быть внешними, такими как CP343/CP443, или внутренними (встроенными в процессоры 3XX-PN или 4XX-PN), они, однако, работают по той же схеме.

2) PLC может работать как клиент, в этом случае запросы на чтение/запись данных выполняются через FB14/FB15 (Get/Put), и требуется соединение S7, созданное с помощью NetPRO.

3) Партнеры могут обмениваться незапрошенными данными, то есть, как только соединение установлено, оба могут отправлять данные другому партнеру.

В своих руководствах Siemens часто называет этот вид связи "клиент-клиент".



Одноранговый узел, запрашивающий соединение, называется активным партнером, а одноранговый узел, принимающий соединение, - пассивным партнером.

Связь осуществляется через FB12/FB13 (S7300) или SFB12/SFB13 (S7400), их символические названия-BSend/BRecv (Block Send / Block Recv).

Важным замечанием является то, что, когда PLC A вызывает BSend, BRecv должен быть вызван в PLC B в то же время, чтобы завершить транзакцию.

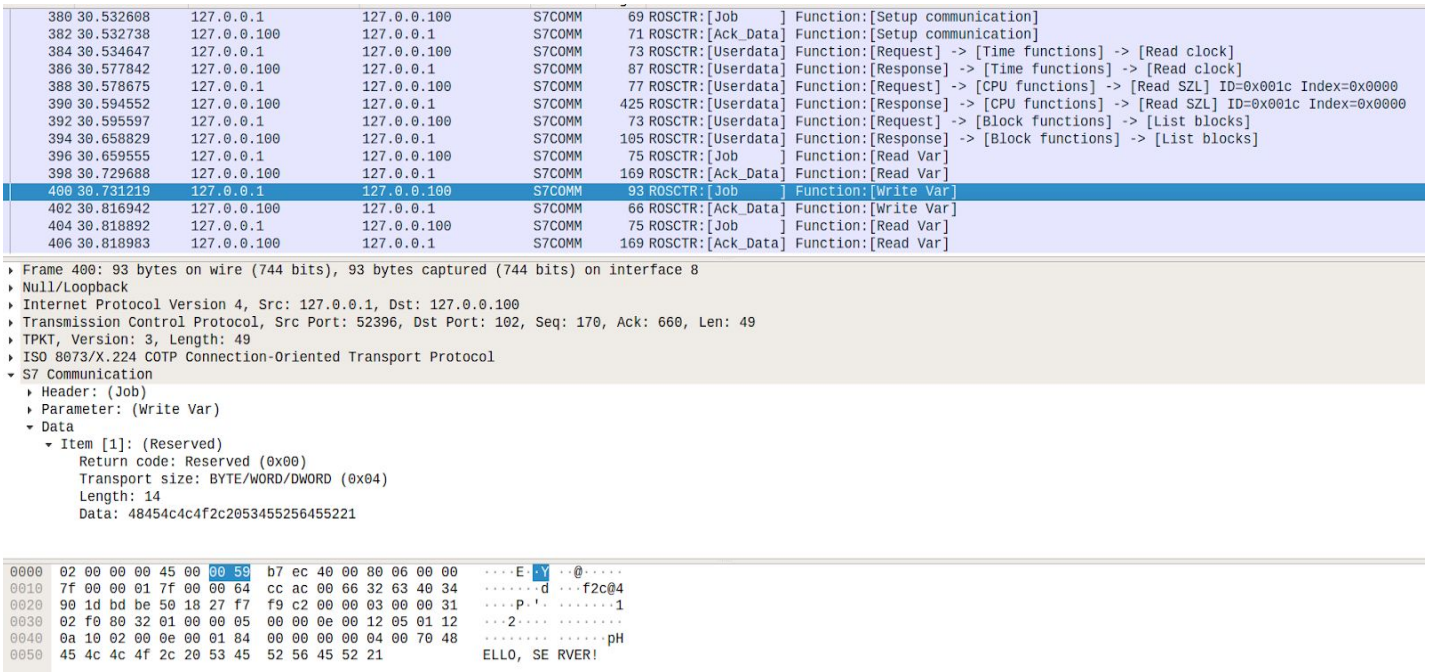
4) Существуют и другие сценарии использования протокола

Описание лабораторного стенда

Клиент – сервер

Для выполнения лабораторной работы была реализована архитектура “клиент-сервер” на языке программирования Python. Для передачи данных с помощью протокола S7comm была использована библиотека snap7. Клиент и сервер могут работать на одной машине. В текущей реализации используется 127.0.0.100 — следующий “внутренний” IP - адрес, передача осуществляется по 102 порту. Изначально сервер выделяет некоторые области памяти для использования клиентом. Затем начинает прослушивать порт. Клиент после прохождения регистрации, запрашивает некоторые данные у сервера (текущее время и описание CPU), после этого делает запросы на чтение и запись, и наконец отключается.

Трафик, генерируемый стендом, отслеживался в Wireshark. Из логов можно наблюдать, что все функции работают корректно. На скриншоте представлен пример пакета при запросе на запись от клиента.



Модуль для протокола S7comm для ПО Suricata

Система предотвращения вторжений (англ. Intrusion Prevention System) — программная или аппаратная система сетевой и компьютерной безопасности, обнаруживающая вторжения или нарушения безопасности и автоматически защищающая от них.

Систем обнаружения вторжений (Intrusion Detection System) - программное или аппаратное средство, предназначенное для выявления фактов неавторизованного доступа в компьютерную систему или сеть либо несанкционированного управления ими в основном через Интернет.

Suricata — open source IPS/IDS система. Основана разработчиками, которые трудились над IPS версией Snort. Основное отличие Suricata от Snort — возможность использования GPU в режиме IDS, более продвинутая система IPS, многозадачность, как следствие высокая производительность, позволяющая обрабатывать трафик до 10Gbit на обычном оборудовании, и многое другое, в том числе полная поддержка формата правил Snort.

Данное ПО является легко расширяемым, имеется возможность добавлять собственные модули. В рамках лабораторной работы был разработан модуль для протокола S7comm.

Файлы detect-s7comm-s7commbuf

В этих файлах описаны функции, которые отвечают за парсинг правил и обнаружение пакетов, соответствующих правилам. Описана структура `struct DetectS7comm_`, поля которой хранят признаки, используемые в правилах для детектирования пакетов.

Создано регулярное выражение для парсинга правил, связанных со значением функции пакета:

```
"^\\s*\"?\\s*function\\s*([0-9]+)\\s*\"?\\s*$"
```

Описание функций:

`static DetectS7comm *DetectS7commFunctionParse(DetectEngineCtx *de_ctx, const char *s7commstr);`

Функция вызывается в начале работы сурикаты, парсит правила и достает нужные значения полей. Затем эти значения помещаются в структуру `DetectS7comm` для удобства использования. После этого значение функции извлекается и проверяется на допустимость с помощью следующего условия

`(StringParseUint8(&s7comm->function, 10, 0, (const char *)ptr) < 0)`

в случае выполнения условия выводится сообщение об ошибке, иначе в структуре выставляется флаг, сигнализирующий о том, что поле function имеет валидное значение.

`static DetectS7comm *DetectS7commTypeParse(DetectEngineCtx *de_ctx, const char *s7commstr)`

Та же самая логика только для правил, относящихся к типу пакета.

`int DetectS7commMatch(DetectEngineThreadCtx *det_ctx, Packet *p, const Signature *s, const SigMatchCtx *ctx)`

Функция вызывается во время непосредственной работы suricat'ы. В качестве параметров принимает пакет, а также структуру, которая заполнялась в Parse-функциях. В ней содержится логика, которая определяет является ли пакет S7comm пакетом, и в случае успеха извлекает проверяемые поля из пакета и сравнивает их со значениями в структуре.

`int DetectS7commSetup(DetectEngineCtx *de_ctx, Signature *s, const char *s7commstr)`

Функция вызывается сурикатой для настройки модуля при запуске. В ней вызываются функции парсинга правил.

```
void DetectS7commS7commbufRegister(void)
```

Эта функция зовётся сурикатой в первую очередь. Единственная функция, которая является внешней по отношению к модулю, так как объявлена в хедере(.h). Она предназначена для регистрации функций-обработчиков, а также для регистрации ключевого слова. Все эти данные заносятся в sigmatch_table, с помощью которой они и будут вызываться / использоваться во время работы. Также здесь происходит конвертирование регулярных выражений из обычных строк в удобные для использования структуры типа `DetectParseRegex`

```
DetectSetupParseRegexes(PARSE_REGEX_TYPE, &type_parse_regex);

DetectSetupParseRegexes(PARSE_REGEX_FUNCTION, &function_parse_regex);
```

Остальные файлы не были изменены вручную, все дополнения сделаны автоматически скриптом, создающим новый модуль.

Установка и запуск ПО Suricata

Для работы архитектуры “клиент-сервер” требуется установить библиотеку snap7:

- pip install snap7
- download snap7.dll from snap7 opensource lib and copy it to directory with python files

Процесс установки состоит из нескольких этапов:

- Загрузка необходимых пакетов
- Конфигурация проекта
- Билд модуля для Suricat’ы

Последовательность установки ПО Suricata:

1. `cd ~`
2. `sudo apt-get update && sudo apt-get upgrade -y`
3. `sudo 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 cargo python3-pip cbindgen`
4. `sudo pip install python-snap7`
5. `sudo pip install --upgrade suricata-update`
6. `git clone https://github.com/yerseg/suricata.git`
7. `cd suricata/`
8. `git checkout yerseg/s7comm_investigation`
9. `git clone https://github.com/OISF/libhtp.git`
10. `sudo ./autogen.sh`
11. `sudo ./configure --prefix=/usr --sysconfdir=/etc --localstatedir=/var`
12. `sudo mkdir /var/log/suricata`
13. `sudo mkdir /etc/suricata`
14. `sudo make && sudo make install && sudo make install-conf`
15. `sudo cp suricata.yaml /etc/suricata`
16. `sudo suricata-update -D /etc/suricata`
17. `sudo ifconfig lo mtu 1522`

После любого изменения в файлах проекта (.c, .h), проект нужно пересобирать `sudo make install`

Установка лабораторного стенда

1. `cd ~`
2. `git clone https://github.com/yerseg/s7comm_investigation.git`
3. `cd s7comm_investigation/`
4. `sudo cp ./libsnap7.so /usr/lib`
5. `sudo ldconfig`
6. После этого можно запускать клиент и сервер в привилегированном режиме

Изменение правил и конфигураций

- `sudo gedit /etc/suricata/suricata.yaml` -- set interface to `lo`
- Изменять правила можно с помощью следующей команды:

`sudo gedit /etc/suricata/rules/suricata.rules`

Наконец можно запустить Suricata с помощью команды:

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

Отследить трафик можно с помощью ПО Wireshark.

Используемое правило:

`alert tcp 127.0.0.1 any -> 127.0.0.100 any (s7comm: function 4;)`

Проверить работоспособность модуля можно с помощью логов, где отображаются alert’ы, которые обозначают, что Suricata обнаружила пакет, соответствующий правилу.

`sudo cat /var/log/suricata/eve.json | grep "\"event_type\":\"s7comm\""`

Использованные источники

1. Описание протокола S7comm и устройств, которые его используют(поддерживают) — https://cache.industry.siemens.com/dl/files/423/1172423/att_39879/v1/iethb_e.pdf
2. Документация библиотеки, которая была использована для создания “клиент-сервер” архитектуры — https://python-snap7.readthedocs.io/en/latest/
3. Исходный код клиента и сервера, а также дамп пакетов, содержащий S7comm — https://github.com/yerseg/s7comm_investigation
4. Suricata с дополнительным модулем для протокола S7Comm — https://github.com/yerseg/suricata/tree/yerseg/s7comm_investigation

Приложение А

```
#Protocol ID:
0x32 - Protocol ID

#Message Types:
0x01 - Job Request
0x02 - Ack
0x03 - Ack-Data
0x07 - Userdata

#Header Error Class:
0x00 - No error
0x81 - Application relationship error
0x82 - Object definition error
0x83 - No resources available error
0x84 - Error on service processing
0x85 - Error on supplies
0x87 - Access error

#Parameter Error Codes:
0x0000 - No error
0x0110 - Invalid block type number
0x0112 - Invalid parameter
0x011A - PG ressource error
0x011B - PLC resource error
0x011C - Protocol error
0x011F - User buffer too short
0x0141 - Request error
0x01C0 - Version mismatch
0x01F0 - Not implemented
0x8001 - L7 invalid CPU state
0x8500 - L7 PDU size error
0xD401 - L7 invalid SZL ID
0xD402 - L7 invalid index
0xD403 - L7 DGS Connection already announced
0xD404 - L7 Max user NB
```

0xD405 - L7 DGS function parameter syntax error

0xD406 - L7 no info

0xD601 - L7 PRT function parameter syntax error

0xD801 - L7 invalid variable address

0xD802 - L7 unknown request

0xD803 - L7 invalid request status

#Return value of item response

0x00 - Reserved

0x01 - Hardware fault

0x03 - Accessing the object not allowed

0x05 - Address out of range

0x06 - Data type not supported

0x07 - Data type inconsistent

0x0a - Object does not exist

0xff - Success

#Job Request/Ack-Data function codes

0x00 - CPU services

0xF0 - Setup communication

0x04 - Read Variable

0x05 - Write Variable

0x1A - Request download

0x1B - Download block

0x1C - Download ended

0x1D - Start upload

0x1E - Upload

0x1F - End upload

0x28 - PLC Control

0x29 - PLC Stop

#Memory Areas

0x03 - System info of S200 family

0x05 - System flags of S200 family

0x06 - Analog inputs of S200 family

0x07 - Analog outputs of S200 family

0x1C - S7 counters (C)

0x1D - S7 timers (T)

0x1E - IEC counters (200 family)

0x1F - IEC timers (200 family)

0x80 - Direct peripheral access (P)

0x81 - Inputs (I)

0x82 - Outputs (Q)

0x83 - Flags (M) (Merker)

0x84 - Data blocks (DB)

0x85 - Instance data blocks (DI)

0x86 - Local data (L)

0x87 - Unknown yet (V)

#Transport size (variable Type) in Item data

0x01 - BIT

0x02 - BYTE

0x03 - CHAR

0x04 - WORD

0x05 - INT

0x06 - DWORD

0x07 - DINT

0x08 - REAL

0x09 - DATE

0x0A - TOD

0x0B - TIME

0x0C - S5TIME

0x0F - DATE AND TIME

0x1C - COUNTER

0x1D - TIMER

0x1E - IEC TIMER

0x1F - IEC COUNTER

0x20 - HS COUNTER

#Variable addressing mode

0x10 - S7-Any pointer (regular addressing) memory+variable length+offset

0xa2 - Drive-ES-Any seen on Drive ES Starter with routing over S7

0xb2 - S1200/S1500? Symbolic addressing mode

0xb0 - Special DB addressing for S400 (subitem read/write)

#Transport size in data

0x00 - NULL

0x03 - BIT

0x04 - BYTE/WORD/DWORD

0x05 - INTEGER

0x07 - REAL

0x09 - OCTET STRING

#Block type constants

'08' - OB

'0A' - DB

'0B' - SDB

'0C' - FC

'0D' - SFC

'0E' - FB

'0F' - SFB

#Sub block types

0x08 - OB

0x0a - DB

0x0b - SDB

0x0c - FC

0x0d - SFC

0x0e - FB

0x0f - SFB

#Block security mode

0 - None

3 - Know How Protect

#Block Language

0x00 - Not defined

0x01 - AWL

0x02 - KOP

0x03 - FUP

0x04 - SCL

0x05 - DB

0x06 - GRAPH

0x07 - SDB

0x08 - CPU-DB DB was created from Plc program (CREAT_DB)

0x11 - SDB (after overall reset) another SDB, don't know what it means, in SDB 1 and SDB 2, uncertain

0x12 - SDB (Routing) another SDB, in SDB 999 and SDB 1000 (routing information), uncertain

0x29 - ENCRYPT block is encrypted (encoded?) with S7-Block-Privacy

#Userdata transmission type

0x0 - Push cyclic data push by the PLC

0x4 - Request by the master

0x8 - Response by the slave

#Userdata last PDU

0x00 - Yes

0x01 - No

#Userdata Functions

0x1 - Programmer commands

0x2 - Cyclic data

0x3 - Block functions

0x4 - CPU functions

0x5 - Security

0x7 - Time functions

#Variable table type of data

0x14 - Request

0x04 - Response

#VAT area and length type

0x01 - MB

0x02 - MW

0x03 - MD

0x11 - IB

0x12 - IW

0x13 - ID

0x21 - QB

0x22 - QW

0x23 - QD

0x31 - PIB

```
0x32 - PIW

0x33 - PID

0x71 - DBB

0x72 - DBW

0x73 - DBD

0x54 - TIMER

0x64 - COUNTER


#Userdata programmer subfunctions

0x01 - Request diag data (Type 1)

0x02 - VarTab

0x0c - Erase

0x0e - Read diag data

0x0f - Remove diag data

0x10 - Forces

0x13 - Request diag data (Type2)


#Userdata cyclic data subfunctions

0x01 - Memory

0x04 - Unsubscribe


#Userdata block subfunctions

0x01 - List blocks

0x02 - List blocks of type

0x03 - Get block info


#Userdata CPU subfunctions

0x01 - Read SZL

0x02 - Message service

0x03 - Transition to stop

0x0b - Alarm was acknowledged in HMI/SCADA 1

0x0c - Alarm was acknowledged in HMI/SCADA 2

0x11 - PLC is indicating a ALARM message

0x13 - HMI/SCADA initiating ALARM subscription



#Userdata security subfunctions

0x01 - PLC password
```

#Userdata time subfunctions

0x01 - Read clock

0x02 - Set clock

0x03 - Read clock (following)

0x04 - Set clock

#Flags for LID access

0x2 - Encapsulated LID

0x3 - Encapsulated Index

0x4 - Obtain by LID

0x5 - Obtain by Index

0x6 - Part Start Address

0x7 - Part Length

#TIA 1200 area names

0x8a0e - DB

0x0000 - IQMCT

0x50 - Inputs (I)

0x51 - Outputs (Q)

0x52 - Flags (M)

0x53 - Counter (C)

0x54 - Timer (T)