Руководство пользователя

Библиотека маршрутизатора верификации вызовов
“HexCVS SDK”

Москва, Россия 15.09.2022

1. Назначение

Библиотека маршрутизатора верификации вызовов “HexCVS SDK” предназначена для использования в качестве библиотек разработчика решений (SDK), предназначенных для реализации связности между Центрами Верификации Операторов связи с целями обеспечения проверки вызовов в рамках исполнения закона 319-ФЗ.

2. Минимальные требования

Библиотека маршрутизатора верификации вызовов “HexCVS SDK” написана на языке программирования «c», которая оттестирована и работает на следующих операционных системах:

- Astra Linux SE v.1.7

- Debian 10 и выше

Минимальные требования к аппаратной части: 8 CPU, 4G RAM, 20G HDD

3. Интеграция библиотеки в проект

Для написания приложения с использованием библиотеки рекомендуется добавить файл Makefile в аналогичный файл Makefile существующей библиотеки. В качестве альтернативы, в собственный файл сборки может быть добавлена статическая библиотека libhexcvs.a.

3.1 Файл заголовок

В исходный код программы библиотека подключается одним единственным include:

```
#include "cvs_lib.h"
```

3.2 Сборка программы

При компиляции вашей собственной программы необходимо добавить статическую библиотеку во время линковки. Это можно сделать в файле Makefile, где необходимо добавить libhexcvs.a после объектных файлов приложения:

```
пример: libhexcvs $(EXAMPLE_OBJS)
        $(CC) -o $@ example.o libhexcvs.a $(LDFLAGS)
```

4. Реализованные методы

Ниже приведены различные примеры реализованных методов.

4.1 Инициализация

Библиотека может быть инициализирована с помощью одного вызова, который возвращает «-1» в случае сбоя:

```
int ret = InitCVSLib(CFG_FILENAME, SS7_FILENAME);
Trace(0, "InitCVSLib returns %d\n", ret);
if (ret != 0)
{
  Trace(0, "InitCSVLib() returned error, exiting...\n");
  printf("InitCSVLib() returned error, exiting...\n");
  return 1;
}
```

Вызов функции выполняет инициализацию всех частей библиотеки, чтение файлов настроек, выделение необходимой памяти и ресурсов и запуск необходимых потоков. Если результат выполнения равен -1, следует проверить журнал или стандартный вывод на наличие сообщений об ошибках. Типовые причины сбоя это недостаточно свободной памяти и недостаточные права пользователя.

4.2 Чтение INI файлов

Для чтения INI файлов реализованы следующие методы:

```
void GetIniString(const char *sFile, const char *sSect, const char *sKey, const char
*sDefault, char *sValue, size_t nValueLength);
void GetIniStringInd(const char *sFile, const char *sSect, const char *sKey, int
nInd, const char *sDefault, char *sValue, size_t nValueLength);
void GetIniInt(const char *sFile, const char *sSect, const char *sKey, int nDefault,
int *pValue);
void GetIniDouble(const char *sFile, const char *sSect, const char *sKey, double
dbDefault, double *pValue);
```

Пример использования:

```
GetIniString(CFG_FILENAME, "GLB", "EXAMPLE_PAREMETER", "test_value",
glb_sExampleParameter, sizeof(glb_sExampleParameter));

GetIniInt(CFG_FILENAME, "GLB", "EXAMPLE_PAREMETER2", 100, &glb_nExampleParameter2);

GetIniDouble(CFG_FILENAME, "GLB", "EXAMPLE_PAREMETER3", 5.0,
&glb_dExampleParameter3);
```

4.3 Методы логирования

Логирование событий поддерживается двумя вызовами: Track() и TRACE_FUNCTION(). Оба принимают одни и те же аргументы, но макрос TRACE_FUNCTION() добавляет к сообщению имя текущих функций.

Первый аргумент определяет категорию трассировки. Остальные аргументы следуют шаблону printf().

Система логирования может быть инициирована до инициализации библиотеки вызовом:

```
InitLogging(CFG_FILENAME);
```

Это позволяет вести журнал до начала основной инициализации.

Вызовы функций логирования приведены ниже:

```
Trace(0, "This is a test for GENERAL logging!\n");
Trace(1, "This is a test for SCTP logging!\n");
Trace(2, "This is a test for M3UA logging!\n");
Trace(3, "This is a test for SCCP logging!\n");
Trace(4, "This is a test for RULE logging!\n");
Trace(5, "This is a test for TCAP logging!\n");

TRACE_FUNCTION(0, "1 + 1 = %d\n", 1 + 1);
```

Состав выводимых параметров логирования определяется INI файлом.

4.4 Методы мониторинга

Отправка SNMP сообщений конфигурируется в INI файле. Программа при отправке SNMP сообщений использует вызов:

```
SendSNMPTrap("Application", 0, 4, "test event text");
```

4.5 Методы SCTP

Конфигурация стека и настройка SCTP ассоциаций выполняются через INI-файл. При установлении SCTP ассоциаций программа всегда выступает как клиент (отправляет начальный пакет инициализации).

При обработке IDPs запросов программа должна иметь функцию обратного вызова. Это реализуется с помощью вызова функции:

```
SetTCAPCallbacks(&TCAPIdpCallbackFunction, &TCAPResponseCallbackFunction);
```

Пример использования функции:

```
void TCAPIdpCallbackFunction(int64_t nID, int nAssocIndex, unsigned int nOTid, int
nServiceKey, int nANOA, const char *sANum, int nBNOA, const char *sBNum, int nCNOA,
const char *sCNum, int nLNNOA, const char *sLNNum)
{
  Trace(0,"Received IDP (ID %016lx) assoc. %d, Originating TID %08x, SK %d, A number
NOA %d <%s>, B number NOA %d <%s>, ""C number NOA %d <%s>, Location number NOA %d
<%s>\n",
        nID, nAssocIndex, nOTid, nServiceKey, nANOA, sANum, nBNOA, sBNum, nCNOA,
sCNum, nLNNOA, sLNNum);
}
void TCAPResponseCallbackFunction(int nAssocIndex, unsigned int nDTid, int nOpCode)
{
  if (nOpCode == TCAP_OPCODE_CONTINUE)
  {
```

```
    Trace(0, "Received Continue assoc. %d, Destination TID %08x\n", nAssocIndex,
nDTid);
  }
  else if (nOpCode == TCAP_OPCODE_RELEASE)
  {
    Trace(0, "Received Release assoc. %d, Destination TID %08x\n", nAssocIndex,
nDTid);
  }
}
```

Программа может отправлять IDP запросы с помощью функции SendIDP();

```
int SendIDP(int nAssocIndex, const char *sCalledGT, uint32_t nOTid, int nNOAA, const
char *sANum, int nNOAB, const char *sBNum, int nNOAC, const char *sCNum, uint32_t
nServiceKey, const char *sLocationNumber);
```

4.6 Методы HTTP

Методы HTTP поддерживают прием входящих и отправку исходящих HTTP-запросов. Прием HTTP-запросов осуществляется через HTTP сервер, который слушает порт, настроенный в файле настроек INI. Любой полученный запрос передается программе с помощью функции обратного вызова.

При приеме входящих запросов необходимо вызвать функцию:

```
  SetHTTPRequestCallback(&HTTPRequestFunction);
```

Функция получает HTTP запрос (библиотека имеет внутреннее ограничение буфера на общую поддерживаемую длину, после чего данные игнорируются) и его содержимое (которое уже декодировано, если оно было отправлено с фрагментированной кодировкой) в отдельном буфере. Пример запросов приведен ниже:

```
void HTTPRequestFunction(const char *sSrcAddr, uint16_t nSrcPort, int64_t nID, const
char *sRequest, int nRequestLen, const char *sContent)
{
  TRACE_FUNCTION(0, "Got HTTP request from %s:%d ID %016lx, whole request:\n%.*s\n",
sSrcAddr, nSrcPort, nID,nRequestLen, sRequest);
  TRACE_FUNCTION(0, "Content:\n%s\n", sContent);

  int nMethod = GetHTTPMethod(sRequest);

  switch (nMethod)
  {
    case HTTP_POST:
      {
        char sUrl[1024];
        if (GetHTTPUrl(sRequest, sUrl, sizeof(sUrl)) == 0)
        {
          TRACE_FUNCTION(0, "HTTP URL: %s\n", sUrl);
          if ((strcasecmp(sUrl, "/verify") == 0) || (strcasecmp(sUrl, "/verify/") ==
0))
          {
```

```
            /* process the request in the main thread */
            glb_nHTTPRequestID = nID;
            glb_nHTTPRequestInProgress = 1;
          }
          else
          {
            TRACE_FUNCTION(0, "Unsupported URL: %s\n", sUrl);
            CloseHTTPRequest(nID);
          }
        }
        else
        {
          TRACE_FUNCTION(0, "Error reading HTTP request URL\n");
          CloseHTTPRequest(nID);
        }
      }
      break;
    case HTTP_ERROR:
      TRACE_FUNCTION(0, "Error reading HTTP request method\n");
      CloseHTTPRequest(nID);
      break;
    default:
      TRACE_FUNCTION(0, "HTTP method %d not supported\n", nMethod);
      CloseHTTPRequest(nID);
      break;
  }
}
```

Если функция сохраняет идентификатор входящих запросов, она затем может отправить отложенный ответ на запрос в другой части программы. В противном случае она должна вызвать метод CloseHTTPRequest(nID); чтобы закрыть сокет и связанные с ним ресурсы запроса. Если программа не вызывает CloseHTTPRequest() или ReplyHTTPRequest() для определенного входящего запроса, библиотека автоматически закроет сокет и освободит ресурсы по истечении времени ожидания, указанного в INI-файле.

Для инициации HTTP запросов используется метод StartHTTPRequest().

```
int64_t StartHTTPRequest(const char *sDstAddr, uint16_t nDstPort, const char *sBuffer, int nBufferLength, HTTPResponseCallbackType pCallback, int nTimeoutMs, void *pUserData);
```

Программа может использовать персональную функцию обратного вызова и тайм-аут для каждого запроса.

Пример функции показан ниже:

```
void HttpResponseCallbackFunction(int nEventCode, int64_t nID, const char *sResponse, int nResponseLen, const char *sContent, void *pUserData)
{
  int nUserCode = (int)(intptr_t)pUserData;
```

```
  if (nEventCode == EC_TIMEOUT)
  {
    TRACE_FUNCTION(0, "HTTP ID %016lx, timeouted\n", nID);
  }
  else if (nEventCode == EC_ERROR_SENDING)
  {
    TRACE_FUNCTION(0, "HTTP ID %016lx, error sending\n", nID);
  }
  else if (nEventCode == EC_RESPONSE_RECEIVED)
  {
    TRACE_FUNCTION(0, "Got HTTP response ID %016lx, whole response:\n%.*s\n", nID,
nResponseLen, sResponse);
    TRACE_FUNCTION(0, "Content:\n%s\n", sContent);

    /* process response */
  }
  else
  {
    TRACE_FUNCTION(0, "Unknown EventCode %d\n", nEventCode);
  }
}
```

После завершения выполнения функции сокет будет закрыт, а ресурсы освобождены библиотекой.

5. Пример использования библиотеки

Пример программы, показывающий использование большинства функций библиотеки HexCVS SDK, находится в папке библиотеки в файле с именем example.c. Его можно использовать во время разработки в качестве примера.

6. Описание конфигурационных файлов (INI)

Конфигурационные файлы библиотеки включают два файла. При использовании библиотеки необходимо указать имена файлов. В примере использования библиотеки используются следующие имена файлов EXAMPLE_CFG.INI и EXAMPLE_SS7.INI. EXAMPLE_CFG.INI содержит общие настройки библиотеки, EXAMPLE_SS7.INI содержит настройки, которые связаны с конфигурацией стека протокола SCTP и определениями ассоциаций SCTP.

Пользователь библиотеки также может считывать свои собственные параметры из этих двух файлов или использовать свой собственный.

Все конфигурационные параметры библиотеки описаны ниже в том же порядке, в котором они отображаются в INI-файлах. Порядок параметров не влияет на поведение программы, в случае если INI-файлы отсутствуют или в них отсутствуют определенные записи, они автоматически генерируются (или добавляются в конец файла) в предопределенном

порядке. Данный порядок параметров соответствует тому порядку, в котором они считываются программой.

Описание полей конфигурационных файлов:

Параметр – имя параметра и его значение по умолчанию,
Описание – описание параметра,
Перезагрузка - если «✓» изменение параметра изменяет поведение приложения только после перезапуска приложения, в противном случае параметр может быть обновлен во время выполнения (при повторном чтении из INI-файлов).

ПРИМЕЧАНИЕ: «✓ (assoc.)» означает, что будет достаточно либо перезапуска приложения, либо перезапуска SCTP ассоциации.

EXAMPLE_CFG.INI

Параметр	Описание	Перезагрузка
GLB:LOG_FILE=ifo_log.csv	Имя файла логов M3UA & SCCP	✓
GLB:APP_LOG=ifo_app.csv	Имя основного файла логов	✓
GLB:SCTP_LOG=sctp_log.csv	Имя файла логов SCTP	✓
GLB:LOGGING=2	Уровень логирования (0 = выключено, 1 = только LOG_GENERAL, 2 = включено)	
GLB:LOG_GENERAL=2	Тип логирования для уровня GENERAL (0 = выключено, 1 = только в файл, 2 = стандартный вывод (stdout) и в файл)	
GLB:LOG_SCTP=0	Тип логирования для уровня SCTP (0 = выключено, 1 = только в файл, 2 = стандартный вывод (stdout) и в файл)	
GLB:LOG_M3UA=0	Тип логирования для уровня M3UA (0 = выключено, 1 = только в файл, 2 = стандартный вывод (stdout) и в файл)	
GLB:LOG_SCCP=0	Тип логирования для уровня SCCP (0 = выключено, 1 = только в файл, 2 = стандартный вывод (stdout) и в файл)	

GLB:LOG_RULE=2	Тип логирования для уровня RULE (0 = выключено, 1 = только в файл, 2 = стандартный вывод (stdout) и в файл)	
GLB:LOG_TCAP=2	Тип логирования для уровня TCAP (0 = выключено, 1 = только в файл, 2 = стандартный вывод (stdout) и в файл)	
GLB:LOG_MAX_SIZE_MB=1024	Максимальный размер лог файла в мегабайтах до ротации, отрицательное значение = неограниченно	✓
GLB:HTTP_SERVER_PORT=8081	HTTP порт сервера	✓
GLB:HTTP_CONNECTIONS_MAX=10000	Максимальное количество HTTP соединений	✓
GLB:HTTP_REQUEST_RECEIVING_TIMEOUT_MS=2000	Таймаут для входящих HTTP запросов в миллисекундах	
SNMP:SERVER_NAME=hexRT-1	Имя сервера (используется в SNMP авариях)	
SNMP:ENTERPRISE_NUMBER=100	Параметр Enterprise Number используемый в SNMP	
SNMP:SNMP_COMMUNITY=public	Параметр Community используемый в SNMP	
SNMP:ALERT_CLEAR_MODE=1	2 = альтернативный формат SNMP	
SNMP:HOST_COUNT=0	Количество конечных точек (endpoint) SNMP	✓
SNMP:HOST_ADDRESS(_2,...)=127.0.0.1	SNMP endpoint IP адрес	✓
SNMP:HOST_PORT(_2,...)=162	SNMP endpoint порт	✓
GLB:STATUS_FILE=cvs_stat.csv	Имя файла “stat”	
GLB:WORKER_COUNT=7	Количество рабочих потоков	✓
GLB:TCAP_QUERY_REQUESTS_MAX_SIZE=10000	Максимальное количество TCAP запросов	✓

GLB:TCAP_QUERY_TIMEOUT_MS=2000	Таймаут для TCAP запросов в миллисекундах	

EXAMPLE_SS7.INI

Параметр	Описание	Перезагрузка
GLB:TIMER_ACTIVITY=10	Если нет входящих пакетов (RX) по ассоциации в течении указанного количества секунд, пере-инициализация SCTP ассоциации	
GLB:TIMER_HB=5	Если нет входящих пакетов (RX) по ассоциации в течении указанного количества секунд, отправка heartbeat	
GLB:OWN_GT=0123456789	GT который используется как CallingGT в исходящих инициированных узлом пакетах или ответах на пакеты в зависимости от GLB:GT_OVERRIDE	
GLB:GT_OVERRIDE=1	Если 1 SCCP заголовок в ответах будет иметь CallingGT и OPC из входящего пакета (from CalledGT, DPC), если 0 будут использоваться данные из конфигурации (GLB:OWN_GT and ASSOC#n:OPC)	
GLB:DPC_OVERRIDE=1	Если 1 SCCP заголовок в ответах будет иметь DPC из входящего пакета (from OPC), если 0 если 0 будут использоваться данные из конфигурации (ASSOC#n:DPC)	
GLB:SCTP_ASSOC_COUNT=1	Количество SCTP ассоциаций	✓
ASSOC#n:DISPLAY_NAME=ASSOC	Имя для данной ассоциации	
ASSOC#n:ASSOCIATION_ACTIVE=0	1 = активировать ассоциацию	✓ (assoc.)
ASSOC#n:PORT_LOCAL=0	Номер порта на локальной стороне для ассоциации	✓ (assoc.)

ASSOC#n:PORT_REMOTE=0	Номер порта на удаленной стороне для ассоциации	✓ (assoc.)
ASSOC#n:IP_LOCAL1=0.0.0.0	Первый IP адрес на локальной стороне	✓ (assoc.)
ASSOC#n:IP_LOCAL2=0.0.0.0	Второй IP адрес на локальной стороне (или 0.0.0.0 чтобы запретить мултихоминг)	✓ (assoc.)
ASSOC#n:IP_PEER1=0.0.0.0	Первый IP адрес на удаленной стороне	✓ (assoc.)
ASSOC#n:IP_PEER2=0.0.0.0	Второй IP адрес на удаленной стороне (или 0.0.0.0 чтобы запретить мултихоминг)	✓ (assoc.)
ASSOC#n:ROUTING_CONTEXT=0	Значение Routing Context если активировано	
ASSOC#n:TRAFFIC_MODE=2	Значение Traffic Mode в ASPAC и ASPAC_ACK сообщениях	
ASSOC#n:NETWORK_APPEARANCE=0	Если > 0, добавляется Network Appearance параметр в ответах с установленных значением	
ASSOC#n:ADDRESS_IND=67	Не используется	
ASSOC#n:SSN=252	Calling (Own) SSN если GT_OVERRIDE=0	
ASSOC#n:OPC=1	Код пункта сигнализации - Originating Point Code	
ASSOC#n:DPC=15033	Код пункта сигнализации - Destination Point Code	
ASSOC#n:SEND_DAUD=1	Если 1 отправляется DAUD во время инициализации ассоциации	
ASSOC#n:SEND_DAVA=1	Если 1 отправляется DAVA во время инициализации ассоциации	
ASSOC#n:SEND_ASPUP=1	Не используется	
ASSOC#n:SEND_NTFY=1	Если 1 отправляется NTFY во время инициализации ассоциации после ASPAC_ACK	

ASSOC#n:SEND_NTFY_DOWN=1	Не используется	
ASSOC#n:SEND_RC=0	Если 1 добавляется Routing Context параметр в данные ответа и DAVA	
ASSOC#n:NETWORK_INDICATOR=3	Значение Network Indicator в заголовке M3UA	
ASSOC#n:NP_CALLING=1	Значение CallingGT Numbering Plan в заголовке SCCP (if GT_OVERRIDE=0) - 1 = ISDN	
ASSOC#n:NP_CALLED=1	Значение CalledGT Numbering Plan в заголовке SCCP	
ASSOC#n:IW_GROUPS=1	0 или более групп, разделенных точкой с запятой IW_GROUPS для данной ассоциации (к которой IW группы принадлежат, значения 0-19)	
ASSOC#n:HISTORY_PACKET_MAX_SIZE=500	Максимальный размер буфера истории сообщений для повторной отправки исходящих сообщений	✓
ASSOC#n:HISTORY_PACKET_COUNT=2000	Количество буферов истории сообщений	✓
ASSOC#n:QUIT_ON_SEND_HISTORY_FAILURE=1	Если 1, остановить обработку если отправка из буфера истории сообщений невозможна (пакет не найден в буфере)	

7. Ссылки

1. RFC 2616 – описание протокола HTTP
2. RFC 4960 - описание протокола SCTP
3. RFC 3332 – описание протокола M3UA
4. ITU-T Q.711-Q.714 – описание протокола SCCP
5. ITI-T Q.771-Q.775 – описание протокола TCAP
6. ETSI TS 101 046 – описание протокола CAMEL