ISAPI 协议报警布防介绍

1 流程介绍(以 C++为例)

典型的 HTTP 协议处理过程如下所示：

（1）客户与服务器建立连接；

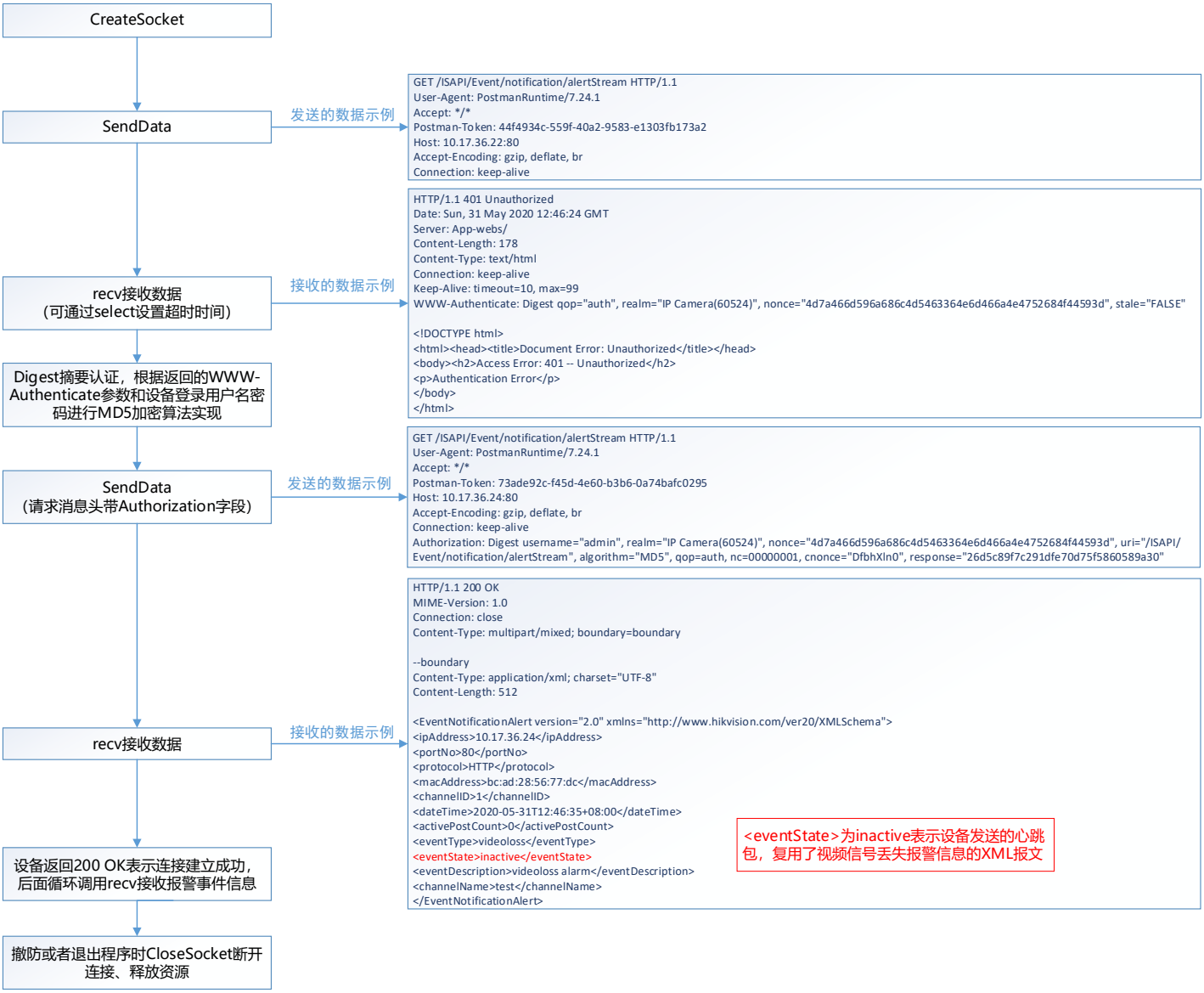
（2）客户向服务器提出请求；

（3）服务器接受请求，并根据请求返回相应的文件作为应答；

（4）客户与服务器关闭连接。

ISAPI 协议基于标准 HTTP REST 架构，而 HTTP 协议使用的是 TCP 协议，因此创建连接也是直接使用套接字（socket），使用 SOCK_STREAM 流式套接字。

1. 报警布防方式：设备作为服务端，程序作为客户端，客户端主动与设备建立 TCP 连接，客户端通过建立的长连接持续接收设备主动上传的报警或事件信息。从 HTTP/1.1 起默认都开启了 Keep-Alive（http 头加入或者不加入"Connection: Keep-Alive"都可以），可以保持连接特性，即我们说的长连接。Keep-Alive 不会永久保持连接，它有一个保持时间，这个时间由服务器即设备决定。另外，我们设备也会自己定时发送应用层的心跳包的方式来判断连接是客户端创建 socket 连接。
2. 客户端向设备发送 GET /ISAPI/Event/notification/alertStream 的布防连接请求。
3. 客户端接收到设备返回 401 认证失败的信息。
4. 客户端需要使用 digest 摘要认证方式，根据设备返回的 WWW-Authenticate 参数和设备登录用户名密码进行 MD5 加密算法实现。
5. 重新发送 GET /ISAPI/Event/notification/alertStream 的布防连接请求，请求消息头需要带正确的 Authorization 认证信息。可以基于前面创建的 Socket 连接发送，也可以 CloseSocket 关闭前面的连接，再重新 CreateSocket。
6. 客户端接收到设备返回 200 OK 的响应，表示连接建立成功，后面客户端一直等待接收设备主动上传的报警事件信息（包含心跳包）即可，接收到报警信息或者心跳包之后客户端需要回复设备一个 ACK 数据。
7. 客户端如果不需要再接收报警信息，直接 CloseSocket 关闭 socket 连接。设备端上传数据之后一直接收不到客户端 ACK 响应（一般是连续 3 次上传没有收到响应）即认为连接断开，从而释放设备端的布防连接资源。



注：流程里面的接口详见 socket 通信系统头文件 WinSock2.h。

2 C#语言实现

C#开发，可以参考我司 C#AppsDemo 的 Demo 示例，直接使用 HttpWebRequest 的类。

```
    public int StartHttpLongLink(string strUserName, string strPassword, string str
,string strparam, ProcessLongLinkData processLongLinkData, ref string strResponse,
    {

        HttpWebRequest request = (HttpWebRequest)HttpWebRequest.Create(strUrl);
        request.Credentials = GetCredentialCache(strUrl, strUserName, strPassword);
        request.Method = strHttpMethod;

        if (!string.IsNullOrEmpty(strparam))
        {
            byte[] bs = Encoding.ASCII.GetBytes(strparam);
            request.ContentType = "application/x-www-form-urlencoded";
            request.ContentLength = bs.Length;
            using (Stream reqStream = request.GetRequestStream())
            {
                reqStream.Write(bs, 0, bs.Length);
            }
        }

        try
        {
            RequestState myRequestState = new RequestState();
            myRequestState.request = request;
            myRequestState.processLongLinkData = processLongLinkData;
            IAsyncResult ret = request.BeginGetResponse(new AsyncCallback(RespCallback), myRequestState);

            if (bBlock)
            {
                int nTimeoutLimit = m_iHttpTimeOut / 100;
                int nTimeoutCount = 0;
                while (!ret.IsCompleted && nTimeoutCount < nTimeoutLimit)
                {
                    Thread.Sleep(100);
                    nTimeoutCount++;
                }

                if (nTimeoutCount == nTimeoutLimit)
                {
                    request.Abort();
                }

                if (myRequestState.response != null && myRequestState.response.StatusCode == HttpStatusCode.OK)
```

strUrl = "http://" + strDeviceIp + ":" + sPort + "/ISAPI/Event/notification/alertStream";

摘要认证

下发布防请求

设置回调函数接收报警事件

摘要认证可以直接使用 CredentialCache 类。

```
private CredentialCache GetCredentialCache(string sUrl, string strUserName, string strPassword)
{
    if (_credentialCache == null)
    {
        _credentialCache = new CredentialCache();
        _credentialCache.Add(new Uri(sUrl), "Digest", new NetworkCredential(strUserName, strPassword));
        strURL = sUrl;
    }
    if (strURL != sUrl)
    {
        _credentialCache.Add(new Uri(sUrl), "Digest", new NetworkCredential(strUserName, strPassword));
        strURL = sUrl;
    }

    return _credentialCache;
}
```

回调函数里面解析接收到的数据：

```
private static void RespCallback(IAsyncResult asynchronousResult)
{
    // State of request is asynchronous.
    RequestState myRequestState = (RequestState)asynchronousResult.AsyncState;
    try
    {
        HttpWebRequest myHttpWebRequest = myRequestState.request;
        myRequestState.response = (HttpWebResponse)myHttpWebRequest.EndGetResponse(asynchronousResult);

        string strBoundary = myRequestState.response.ContentType;
        int nIndex = strBoundary.IndexOf("boundary=");
        if (nIndex >= 0)
        {
            strBoundary = strBoundary.Substring(nIndex + "boundary=".Length);
            myRequestState.strBoundary = strBoundary;
        }

        // Read the response into a Stream object.
        Stream responseStream = myRequestState.response.GetResponseStream();
        myRequestState.streamResponse = responseStream;

        // Begin the Reading of the contents of the HTML page and print it to the console.
        IAsyncResult asynchronousInputRead = responseStream.BeginRead(myRequestState.BufferRead, 0, BUFFER_SIZE, new AsyncCallback(ReceiveData), myRequestState);
        return;
    }
    catch (WebException e)
    {
        myRequestState.eStatus = e;
    }
}
```

3 Java 语言实现

Java 开发，可以参考我司 JFinal 的 JavaDemo，直接使用 CloseableHttpAsyncClient 的类。

```
package communicationCom;

import java.io.IOException;
import java.nio.CharBuffer;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.Future;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.auth.AuthScope;
import org.apache.http.auth.UsernamePasswordCredentials;
import org.apache.http.client.CredentialsProvider;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.concurrent.FutureCallback;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.BasicCredentialsProvider;
import org.apache.http.impl.nio.client.CloseableHttpAsyncClient;
import org.apache.http.impl.nio.client.HttpAsyncClients;
import org.apache.http.nio.IOControl;
import org.apache.http.nio.client.methods.AsyncCharConsumer;
import org.apache.http.nio.client.methods.HttpAsyncMethods;
import org.apache.http.nio.protocol.HttpAsyncRequestProducer;
import org.apache.http.protocol.HttpContext;

import alarm.ParseAlarmData;

public class HttpAysncClientUtil {
```

```
public static int    iPort = 0;
public static String strIP="";
public static CloseableHttpAsyncClient httpAsyncclient;

private static int reconnect=3;
private static int timeout=10000;
private static boolean stoplink=false;

private static boolean DataRecv=false;
private static List<Character>chBuffer=new ArrayList<Character>();
private static ParseAlarmData AlarmData=new ParseAlarmData();

//Initializes a long connection communication object
public static void HttpAysncInit(String user,String password)
{
        //摘要认证
         CredentialsProvider credentialsProvider = new BasicCredentialsProvider();
       credentialsProvider.setCredentials(AuthScope.ANY, new UsernamePasswordCredentials(user, password));
       httpAsyncclient = HttpAsyncClients.custom()
                                   .setDefaultCredentialsProvider(credentialsProvider)
                                   .build();

}
//Long connection function
public static void LonLink(String Url ,String event, boolean subscribe)
{
       stoplink=false;
       chBuffer.clear();
       try {
          //设置回调函数
           FutureCallback<Boolean> callback = new FutureCallback<Boolean>() {
                @Override
                public void cancelled() {
                     // TODO Auto-generated method stub
                     System.out.println("cancelled");
                }
                @Override
                public void completed(Boolean arg0) {
                     // TODO Auto-generated method stub
                     System.out.println("completed");
                }
                @Override
                public void failed(Exception arg0) {
                     // TODO Auto-generated method stub
                     System.out.println("failed");
                }
           };
           //Open the connection
           httpAsyncclient.start();
```

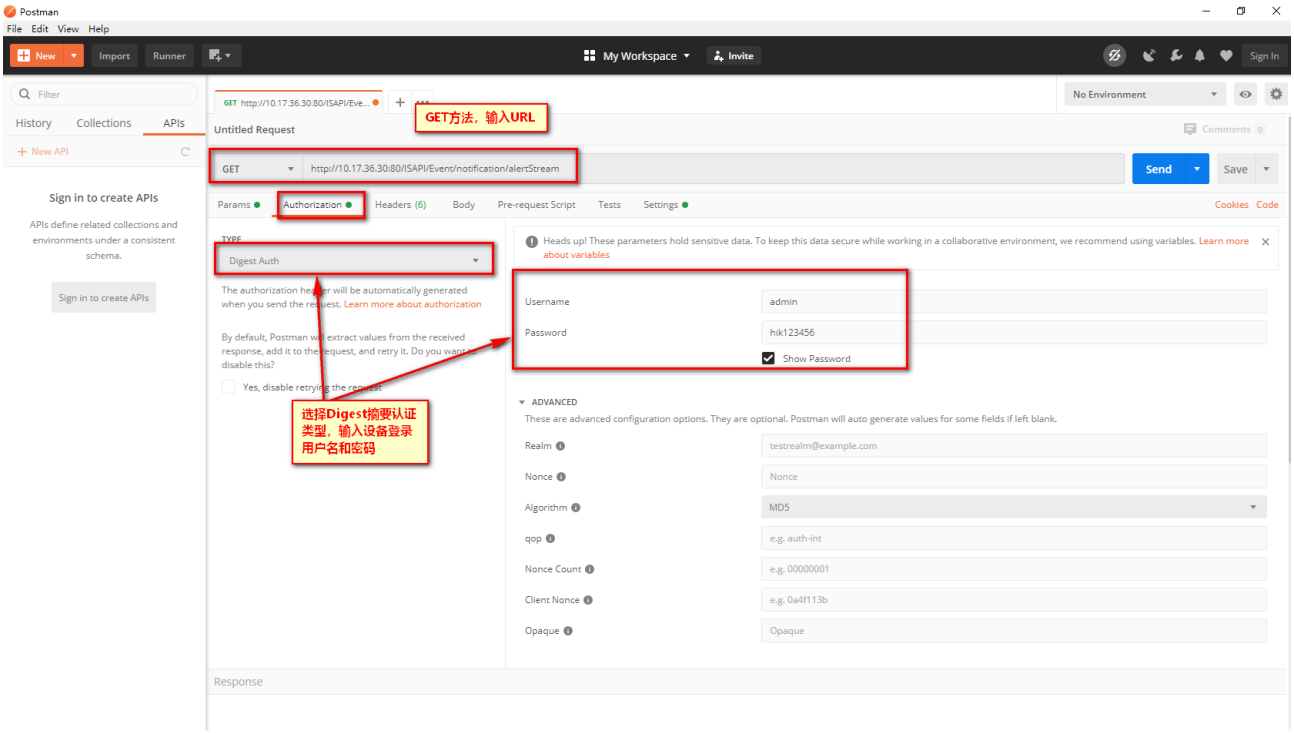
```
            //Reconnect the query thread with a timeout on
            ReConnect recn=new ReConnect();
            Thread Rethread =new Thread(recn);
            Rethread.start();

            //创建连接，设置接收报警事件的回调函数
            // Url="http://"+ip+":"+port+"/ISAPI/Event/notification/alertStream";
            Future<Boolean> future = httpAsyncclient.execute(
                        HttpAsyncMethods.createGet(Url),
                        new ResponseConsumer(), callback);
             Boolean result = future.get();
             if (result != null && result.booleanValue()) {
                    System.out.println("Request successfully executed");
             } else {
                    System.out.println("Request failed");
             }
             System.out.println("Shutting down");

        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
          return;
    }

    public static void StopLink()
    {
        stoplink=true;
        DataRecv=false;
    }

    //接收和解析报警事件信息
    static class ResponseConsumer extends AsyncCharConsumer<Boolean> {

        //Message type
        public String type;
        @Override
        protected void onResponseReceived(final HttpResponse response) {
          //Determine the message type
            System.out.println("onResponseReceived");
            String tbuf=response.toString();
            if(tbuf.contains("multipart"))
            {
               type="multipart";
            }else if(tbuf.contains("xml"))
            {
               type="xml";
            }else if(tbuf.contains("json"))
```

```
        {
            type="json";
        }
    }
    //Callback function to receive a message
    @Override
    protected void onCharReceived(final CharBuffer buf, final IOControl ioctrl) throws IOException {

      DataRecv=true;
      //Parsing by message type
      if(type.equals("multipart"))
      {
            int length=buf.length();
            for(int i=0;i<buf.length();i++)
            {
                  //Fill buffer
                  chBuffer.add(buf.charAt(i));
            }
            //Form data parsing s
            AlarmData.parseMultiData(chBuffer);
      }else if(type.equals("xml"))
      {
           int length=buf.length();
              for(int i=0;i<buf.length();i++)
              {
                    //Fill buffer
                    chBuffer.add(buf.charAt(i));
              }
              //Form data parsing s
              AlarmData.parseMultiData(chBuffer);
      }else if(type.equals("json"))
      {
           int length=buf.length();
              for(int i=0;i<buf.length();i++)
              {
                    //Fill buffer
                    chBuffer.add(buf.charAt(i));
              }
              //Form data parsing s
              AlarmData.parseMultiData(chBuffer);
      }

          if(stoplink)
          {
            buf.clear();
            chBuffer.clear();
              this.close();
              stoplink=false;
          }
```

```
        }
        @Override
        protected Boolean buildResult(final HttpContext context) {
            return Boolean.TRUE;
        }
    }
```

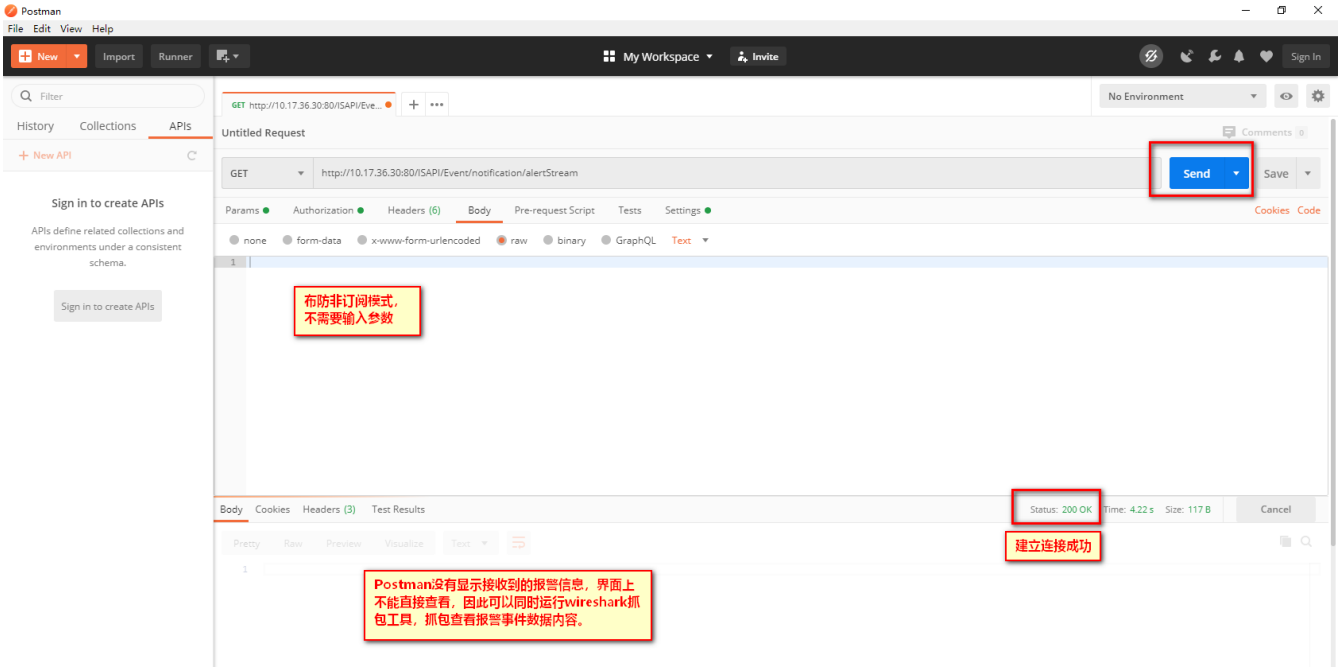
4 Postman 测试

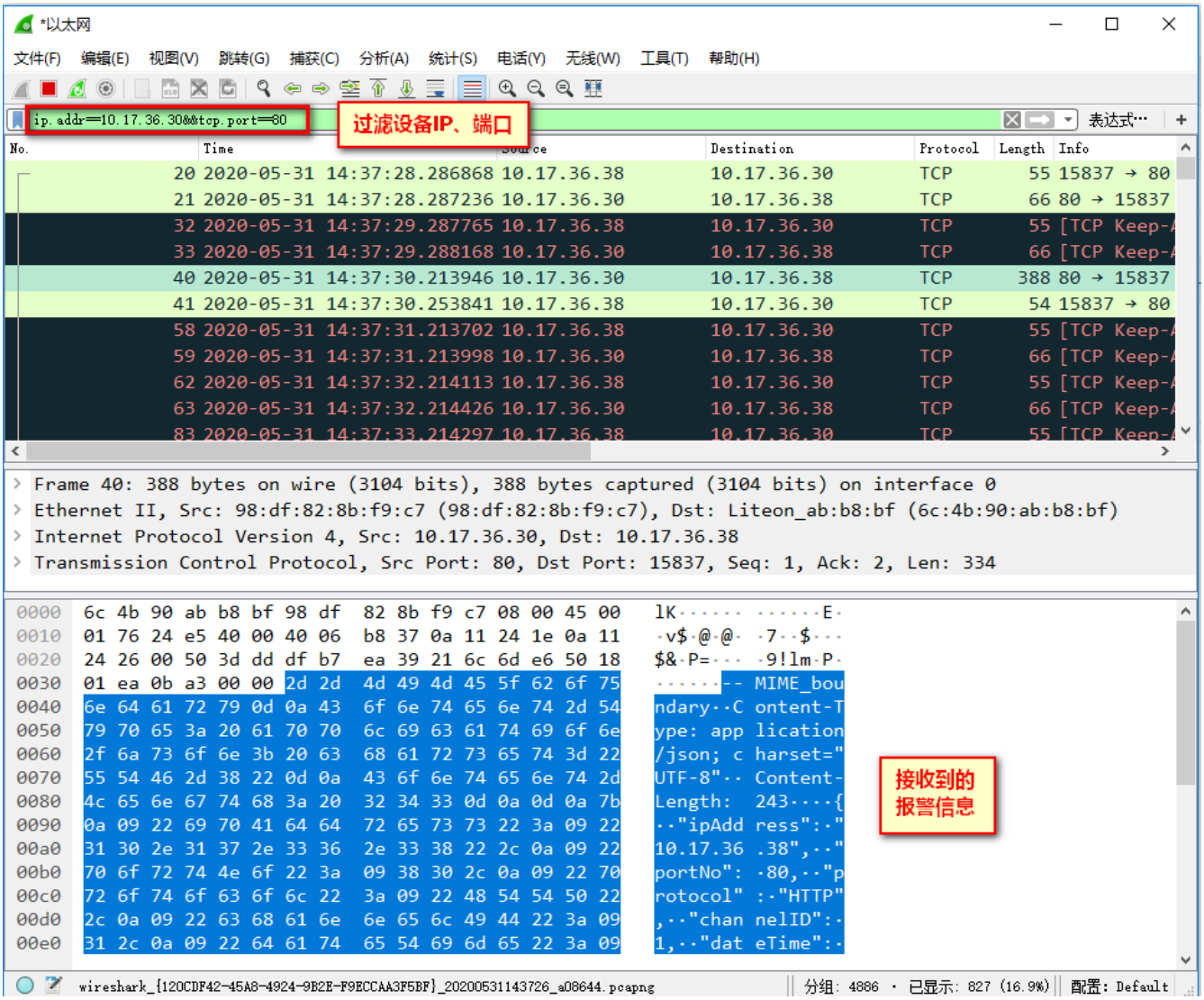
首先，根据布防协议文档说明，输入对接设备的 IP、端口以及布防 URI 命令，Authorization 里面选择 Digest 摘要认证方式，输入设备的用户名、密码。



然后，如果是非订阅方式，URI 为/ISAPI/Event/notification/alertStream，没有输入参数，直接“Send”发送请求即可。

发送请求之后建立成功可以参考如下图的 Status 状态，200 OK 表示建立成功。报警事件信息和心跳包等数据在 Postman 界面查看不到，可以同时运行 Wireshark 抓包，查看具体内容。





Follow TCP 查看报文内容：

Wireshark · 追踪 TCP 流 (tcp.stream eq 13) · 以太网

```
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Authorization: Digest username="admin", realm="DS-2ED2A36F", nonce="NmVmZDIzMGM3MjRlOGZkYmM2NzVlOTFlNDBkYzk2MWQ=",
uri="/ISAPI/Event/notification/alertStream", algorithm="MD5", qop=auth, nc=00000001, cnonce="jTrNBcAF",
response="e336567eff42f918f00e37896a8144f1"

HTTP/1.1 200 OK
MIME-Version: 1.0
Connection: keep-alive
Content-Type: multipart/mixed; boundary=MIME_boundary

Content-Length: 334


--MIME_boundary
Content-Type: application/json; charset="UTF-8"
Content-Length: 243

{
          "ipAddress":        "10.17.36.38",
          "portNo": 80,
          "protocol":         "HTTP",
          "channelID":        1,
          "dateTime":         "2020-05-31T14:52:05+08:00",
          "activePostCount":  1,
          "eventType":        "videoloss",
          "eventState":       "inactive",
          "eventDescription": "videoloss alarm"
}
```

订阅方式报警布防：

POST 方法，URI 是/ISAPI/Event/notification/subscribeEvent，同样选择 Digest 摘要认证，需要输入订阅参数。