Digest 摘要认证

1 摘要认证介绍

ISAPI 协议基于 HTTP REST 架构，协议交互需要安全认证，Digest 摘要认证比 Basic 基础认证的安全级别更高：

1）通过传递用户名、密码等计算出来的摘要来解决明文方式在网络上发送密码的问题。

2）通过服务产生随机数 nonce 的方式可以防止恶意用户捕获并重放认证的握手过程。

1.1 认证握手过程

1. 客户端发出一个没有认证证书的请求。

```
GET /ISAPI/Security/userCheck HTTP/1.1
Accept: text/html, application/xhtml+xml, */*
Accept-Language: zh-CN
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
Accept-Encoding: gzip, deflate
Host: 10.18.37.12
Connection: Keep-Alive
```

注：此处示例为用户名密码校验的ISAPI协议命令（GET方法），每次下发新的命令都需要重新认证。

2. 服务器产生一个随机数nonce，并且将该随机数放在WWW-Authenticate响应头，与服务器支持的认证算法列表，认证的域realm一起发送给客户端。

```
HTTP/1.1 401 Unauthorized
Date: Wed, 30 May 2018 19:16:52 GMT
Server: App-webs/
Content-Length: 178
Content-Type: text/html
Connection: keep-alive
Keep-Alive: timeout=10, max=99
WWW-Authenticate: Digest qop="auth", realm="IP Camera(12345)",
nonce="4e5749344e7a4d794e544936596a4933596a51784e44553d", stale="FALSE"
```

注：401 Unauthorized表示认证失败、未授权。返回的WWW-Authenticate表示设备支持的认证方式，此处设备只支持Digest摘要认证方式。

```
HTTP/1.1 401 Unauthorized
Date: Wed, 30 May 2018 19:23:32 GMT
Server: App-webs/
Content-Length: 178
Content-Type: text/html
```

```
Connection: keep-alive
Keep-Alive: timeout=10, max=99
WWW-Authenticate: Digest qop="auth", realm="IP Camera(12345)",
nonce="4f5455784e4452684f544136596a49344d54566a4f57553d", stale="FALSE"
WWW-Authenticate: Basic realm="IP Camera(12345)"
```

注：401 Unauthorized表示认证失败、未授权。返回的WWW-Authenticate表示设备支持的认证方式，此处设备同时支持Digest摘要认证和Basic认证两种方式。stale表示nonce值是否过期，如果过期会生成新的随机数。

3. 客户端接收到401响应表示需要进行认证，选择一个算法（目前只支持MD5）生成一个消息摘要（message digest，该摘要包含用户名、密码、给定的nonce值、HTTP方法以及所请求的URL），将摘要放到Authorization的请求头中重新发送命令给服务器。如果客户端要对服务器也进行认证，可以同时发送客户端随机数cnonce，客户端是否需要认证，通过报文里面的qop值进行判断，详见1.2章节介绍。

```
GET /ISAPI/Security/userCheck HTTP/1.1
Accept: text/html, application/xhtml+xml, */*
Accept-Language: zh-CN
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
Accept-Encoding: gzip, deflate
Host: 10.18.37.12
Connection: Keep-Alive
Authorization: Digest username="admin",realm="IP
Camera(12345)",nonce="595463314d5755354d7a4936596a49344f475a6a5a44453d",uri="/ISAPI/Security/userCheck",cnonc
e="011e08f6c9d5b3e13acfa810ede73ecc",nc=00000001,response="82091ef5aaf9b54118b4887f8720ae06",qop="auth"
```

4. 服务接收到摘要，选择算法以及掌握的数据，重新计算新的摘要跟客户端传输的摘要进行比较，验证是否匹配，若客户端反过来用客户端随机数对服务器进行质询，就会创建客户端摘要，服务可以预先将下一个随机数计算出来，提前传递给客户端，通过 Authentication-Info 发送下一个随机数。该步骤选择实现。

```
HTTP/1.1 200 OK
Date: Wed, 30 May 2018 19:32:49 GMT
Server: App-webs/
Content-Length: 132
Connection: keep-alive
Keep-Alive: timeout=10, max=98
Content-Type: text/xml

<?xml version="1.0" encoding="UTF-8"?>
<userCheck>
<statusValue>200</statusValue>
<statusString>OK</statusString>
</userCheck>
```

注：响应200 OK表示认证成功。

详细说明请参考RFC 2617规范文档。

1.2 摘要计算过程

在说明如何计算摘要之前，先说明参加摘要计算的信息块。信息块主要有两种：

1. 表示与安全相关的数据的A1

A1中的数据时密码和受保护信息的产物，它包括用户名、密码、保护域和随机数等内容，A1只涉及安全信息，与底层报文自身无关。

若算法是：MD5

则 A1=<user>:<realm>:<password>

若算法是：MD5-sess

则 A1=MD5(<user>:<realm>:<password>):<nonce>:<cnonce>

2. 表示与报文相关的数据的A2

A2表示是与报文自身相关的信息，比如URL，请求反复和报文实体的主体部分，A2加入摘要计算主要目的是有助于防止反复，资源或者报文被篡改。

若 qop 未定义或者 auth：

A2=<request-method>:<uri-directive-value>

若 qop 为 auth:-int

A2=<request-method>:<uri-directive-value>:MD5(<request-entity-body>)

注：<uri-directive-value>为完整的协议命令 URI，比如“/ISAPI/Security/userCheck”。

下面定义摘要的计算规则：

若 qop 没有定义：

摘要 response=MD5(MD5(A1):<nonce>:MD5(A2))

若 qop 为 auth:

摘要 response=MD5(MD5(A1):<nonce>:<nc>:<cnonce>:<qop>:MD5(A2))

若 qop 为 auth-int:

摘要 response= MD5(MD5(A1):<nonce>:<nc>:<cnonce>:<qop>:MD5(A2))

1.3 随机数的生成

RFC2617建议采用这个假想的随机数公式：

nonce = BASE64(time-stamp MD5(time-stamp ":" ETag ":" private-key))

其中:

time-stamp是服务器产生的时间戳或者其他不会重复的序列号，ETag是与所请求实体有关的HTTP ETag首部的值，priviate-key是只有服务器知道的数据。

这样，服务器就可以收到客户端的认证首部之后重新计算散列部分，如果结果与那个首部的随机数不符，或者是时间戳的值不够新，就可以拒绝请求，服务器可以通过这种方式来限制随机数的有效持续时间。

包括了ETag可以防止对已经更新资源版本的重放请求。注意：在随机数中包含客户端IP，服务器好像就可以限制原来获取此随机数的客户端重用这个随机数了，但这会破坏代理集群的工作，使用代理集群时候，来自单个用户的多条请求通常会经过不同的代理进行传输，而且IP地址欺骗实现起来也不复杂。

对于我司设备，认证的随机数超时时间如下所示：
认证返回的 nonce 是 3s 超时；
非认证返回的 none 是 30s 超时。

2 摘要认证开发实现方法

目前以 HTTP 协议为例。

2.1 C++实现

MD5 算法：

```
void __stdcall mprGetMD5Hash(unsigned char *buf, int length, unsigned char *digest)
{

    HMAC_MD5_CTX          context;
    unsigned char         hash[CRYPT_HASH_SIZE];
    const char            *hex = "0123456789abcdef";
    char                  *r;
    char                  result[(CRYPT_HASH_SIZE * 2) + 1];
    int                   i;
    /*
     *    Take the MD5 hash of the string argument.
     */
    hmac_MD5Init(&context);
    hmac_MD5Update(&context, (unsigned char*) buf, (unsigned int) length);
    hmac_MD5Final(hash, &context);
    for (i = 0, r = result; i < 16; i++) {
        *r++ = hex[hash[i] >> 4];
        *r++ = hex[hash[i] & 0xF];
    }
    *r = '\0';

    strcpy(digest, result);
}
```

摘要认证实现：

```
char szSrc[HTTP_DEGIST_SRC_LEN];
memset(szSrc, 0, HTTP_DEGIST_SRC_LEN);
```

```
sprintf(szSrc, "%s:%s:%s", m_strUserName, m_szHttpRealm, m_strUserPwsd);
char szHA1[HTTP_HA_LEN] = {0};    //与安全相关的数据的A1
mprGetMD5Hash((unsigned char *)szSrc, min(HTTP_DEGIST_SRC_LEN, strlen(szSrc)),(unsigned char *)szHA1);

memset(szSrc, 0, HTTP_DEGIST_SRC_LEN);
if (m_iProtocolCommand == ISAPI_GET)
{
        sprintf(szSrc, "GET:%s", m_szProtocolUrl);
}
else if (m_iProtocolCommand == ISAPI_PUT)
{
        sprintf(szSrc, "PUT:%s", m_szProtocolUrl);
}
else if (m_iProtocolCommand == ISAPI_POST)
{
        sprintf(szSrc, "POST:%s", m_szProtocolUrl);
}
else
{
        return;
}

char szHA2[HTTP_HA_LEN] = {0}; //与报文相关的数据的A2
mprGetMD5Hash((unsigned char *)szSrc, min(HTTP_DEGIST_SRC_LEN, strlen(szSrc)),(unsigned char *)szHA2);
memset(szSrc, 0, HTTP_DEGIST_SRC_LEN);
sprintf(szSrc, "%s:%s:%s", szHA1, m_szHttpNonce, szHA2);

char szResponse[HTTP_HA_LEN] = {0};    //Authorization请求头里面的response内容
mprGetMD5Hash((unsigned char *)szSrc, min(HTTP_DEGIST_SRC_LEN, strlen(szSrc)),(unsigned char *)szResponse);

//下发命令并且带Authorization摘要认证内容
const char *pFormat = "%s HTTP/1.1\r\nHost:%s\r\nAccept: text/html, application/xhtml+xml,
*/*\r\nUser-Agent: %s\r\nAuthorization: Digest username=\"%s\", realm=\"%s\", nonce=\"%s\", uri=\"%s\",
response=\"%s\"\r\n\r\n";
sprintf(szSendBuf, pFormat, m_strUrl.GetBuffer(0), m_szDeviceIP, g_pUserAgent, m_strUserName, m_szHttpRealm,
m_szHttpNonce, m_szProtocolUrl, szResponse);
```

2.2 C#实现

C#语言开发有现成的 WebClient 类可以直接使用，所属命名空间为 System.Net。

```
using System.Net;

public static WebClient m_webClient = new WebClient();
```

```
m_webClient.Credentials = new NetworkCredential(sUserName, sPassword);   //设备登录用户名密码
m_webClient.BaseAddress = "http://" + sDeviceAddress + ":" + iPort;   //设备 IP 地址和端口
strUrl =    "/ISAPI/Security/userCheck";
string loginResult = "";
loginResult = m_webClient.DownloadString(strUrl);   //WebClient 类中方法,以 String 形式下载指定资源
```

DownloadString 返回的结果是字符串，数据可能会异常，使用 UploadData 返回数据可以在 Byte 数组缓冲区里面，如下所示：

```
public bool Upload(ICredentials Credentials, String requestUrl, out String responseText)
{
        WebClient webClient = new WebClient();
        webClient.Credentials = Credentials;
        webClient.Headers.Add("Content-Type", "multipart/form-data; boundary=" + boundary);
        byte[] responseBytes;
        byte[] bytes = MergeContent();
        try
        {
            responseBytes = webClient.UploadData(requestUrl, bytes);
            responseText = System.Text.Encoding.UTF8.GetString(responseBytes);
            return true;
        }
        catch (WebException ex)
        {
            Stream responseStream = ex.Response.GetResponseStream();
            responseBytes = new byte[ex.Response.ContentLength];
            responseStream.Read(responseBytes, 0, responseBytes.Length);
        }
        responseText = System.Text.Encoding.UTF8.GetString(responseBytes);
        return false;
}
```

```
WebClient client = new WebClient();
client.Credentials = new NetworkCredential(strUserName, strPassword, struDeviceInfo.chDeviceIP);
client.Headers.Add("Content-Type", "application/x-www-form-urlencoded; charset=UTF-8");
client.Headers.Add("ContentLength", bytes.Length.ToString());
byte[] responseData = client.UploadData(strUrl, "POST", bytes);

string strFD = System.Text.Encoding.UTF8.GetString(responseData);
//获取json参数
string[] szjsontemp = Regex.Split(strFD, "\r\n", RegexOptions.IgnoreCase);
//解析json数据
m_jsonTaskID = JsonConvert.DeserializeObject<jsonTaskID>(strFD);
```

2.3 Java 实现

Java 语言开发有现成的 httpclient 工具类可以直接使用。

```
import org.apache.commons.httpclient.UsernamePasswordCredentials;
import org.apache.commons.httpclient.auth.AuthScope;

strDevicePort = "80";
UsernamePasswordCredentials creds = new UsernamePasswordCredentials(UsertextField.getText(),
PasstextField.getText());     //设备登录用户名密码，摘要认证方式
HTTPClientUtil.client.getState().setCredentials(AuthScope.ANY, creds);

//登录校验代码待补充
String strUrl = "/ISAPI/Security/userCheck";
strOut= HTTPClientUtil.doGet("http://"+IPtextField.getText()+":"+strUrl, null);
```

```
import org.apache.commons.httpclient.HttpClient;
import org.apache.commons.httpclient.methods.GetMethod;
import org.apache.commons.httpclient.methods.PutMethod;
import org.apache.commons.httpclient.methods.PostMethod;
import org.apache.commons.httpclient.methods.DeleteMethod;

public static HttpClient client = new HttpClient();
public class HTTPClientUtil{
    public static String doGet(String url, String charset) throws Exception
    {
        GetMethod method = new GetMethod(url);
        method.setDoAuthentication(true);

        int statusCode = client.executeMethod(method);

        byte[] responseBody =
        method.getResponseBodyAsString().getBytes(method.getResponseCharSet());
        //在返回响应消息使用编码(utf-8 或 gb2312)
        String response = new String(responseBody, "utf-8");
        //释放连接
        method.releaseConnection();
        return response;
    }
}
```