基于 TR069 协议的 CPE 安全认证机制①

张 红[1], 赵 云[2], 陶 然[1], 赵伟真[1]

[1](东华大学 计算机科学与技术学院, 上海 201620)
[2](上海剑桥科技股份有限公司, 上海 201620)

摘 要: TR069(CPE 广域网管理协议)提供了对下一代网络中的家庭网络设备进行管理配置的通用框架和协议, 其协议栈中包括 SOAP, HTTP, SSL / TLS, TCP/IP 等标准协议. 当自动配置服务器(ACS)和网络终端设备(CPE)建立连接时, 可以选择使用 SSL/TLS 层增强通信的安全性. 本文设计并实现了使用 OpenSSL 开发包对 ACS 与 CPE 的相互认证过程进行加密, 包括生成自签证认证证书的方法, 以及证书验证过程. 最后, 通过实验验证了其有效性, 并抓包分析了关键帧.

关键词: TR069; OpenSSL; 认证

CPE Security Authentication Mechanism Based on TR069 Protocol

ZHANG Hong[1], ZHAO Yun[2], TAO Ran[2], ZHAO Wei-Zhen[1]

[1](School of Computer Science and Technology, Donghua University, Shanghai 201620, China)
[2](Cambridge Industries Group, Shanghai 201620, China)

Abstract: TR069 protocol is called the “CPE WAN Management Protocol”. It provides a common framework and protocol which manage and configure for next-generation home network devices. Its protocol stack include some standard protocols, such as SOAP, HTTP, SSL / TLS, TCP/IP and so on. When Auto-Configuration Server(ACS) starts to establish connection with the Customer Premises Equipment (CPE), it may choose the layer of SSL/TLS to increase security. In this paper, the process of mutual authentication between CPE and ACS is encrypted, which is designed and implemented using OpenSSL development package, include the method of generate a self-signed certificate, and the process of certificate is verified. Finally, this paper verified its validity by experiment, and analyzes the key frame by capture package.

Key words: TR069; OpenSSL; authentication

1 概述

随着网络技术的不断发展, 网络终端设备在各行各业被广泛应用, 但是网络也是十分不安全的, 隐匿在网络中的攻击者可能窃听、篡改网络中的数据, 甚至伪装成他人发送信息. 基于 TR069 协议的自动配置服务器(ACS[1])和网络终端设备(CPE[1])进行通信时, 通信内容为明文, 且整个通信过程暴露在大网环境中, 通信内容容易被劫持或篡改, 势必引起不必要的损失, 而 TR069 协议栈中的 SOAP, HTTP, TCP/IP 等不能对服务器的身份进行验证, 也不能保证数据传输的私密性, 无法提供安全性保证.

CA 认证技术[2]和 SSL 加密通信被认为是抵抗网络攻击的最有效手段, 本文将 CA 认证技术和 SSL 加密通信应用到 ACS 与 CPE 通讯中, 通过 OpenSSL 生成认证证书并且对通信双方进行身份认证, 从而实现了对 ACS 与 CPE 通讯的安全管理.

2 相关知识

2.1 TR069 协议

TR069 协议[3]全称为“CPE 广域网管理协议”, 是 DSL 论坛开发的技术规范之一. 它提供了对网络终端设备进行管理配置的通用框架和协议, 是为实现网络

① 收稿时间:2014-05-08;收到修改稿时间:2014-06-20

终端设备(CPE)和自动配置服务器(ACS)之间的通讯而设计的. 其主要目的是对支持 TR069 协议的终端设备, 如网络中的路由器、网关、机顶盒等支持 TR069 协议的终端设备进行集中管理.

TR069 协议除了包含一些协议特有的组件外, 同时还采纳了其它的几种标准协议, 协议栈的描述如表 1 所示.

表 1 TR069 协议栈

CPE/ACS Management Application	该应用程序分别用于 CPE 广域网管理协议的 CPE 和 ACS
RPC Methods	CPE 广域网管理协议定义的特殊的远程过程调用方法
SOAP	标准的基于 XML 的语法
HTTP	要求支持 HTTP1.1
SSL/TLS	标准的 Internet 传输层安全协议. 特指 SSL3.0 或 TLS1.0
TCP/IP	标准的 TCP/IP

2.2 SSL 协议及其实现 OpenSSL

SSL(Secure Sockets Layer)安全套接层协议, 位于 TCP/IP 协议与各种应用层协议之间, 是一个分层协议, 由两层组成, 上层是握手协议层, 下层是记录协议层. SSL 记录协议(SSL Record Protocol)建立在可靠的传输协议(如 TCP)之上, 为高层协议提供数据封装、压缩、加密等基本功能的支持. SSL 握手协议(SSL Handshake Protocol)建立在 SSL 记录协议之上, 用于通信实体在交换应用数据前, 进行身份认证、协商加密算法、交换加密密钥等, 使客户端和服务器端建立并保持用于安全通信的状态信息. 其中 SSL 握手协议允许自动配置服务器 ACS 和用户终端 CPE 互相认证, 并在双方应用层协议传输数据之前协商出一个加密算法和会话密钥, 为 CPE 与 ACS 之间的通信提供了可靠的安全保证.

SSL 只是一个协议, 要在实际应用中使用这个协议, 还需要支持此协议的软件包. 本文选用了 OpenSSL 软件包, OpenSSL 是一个开放源代码的 SSL 协议及相关加密技术的产品实现. 它采用 C 语言作为开发语言, 具有跨平台开发性能. 它提供了加密算法库并封装了具体的交互过程, 因此只需要调用相应的 API 即可实现 SSL 通信. 另外, OpenSSL 还提供了生成证书[4]的应用, 方便进行测试验证工作. SSL/TLS 协议可以直接在 TCP 链接上使用, 特别的 SSL/TSL 提供了机密性和数据完整性, 同时允许基于证书的认证代替共享密钥的认证, 如果 ACS URL 已经被指示为一个 HTTPS URL, 则 CPE 必须使用 SSL/TLS 跟 ACS 建立链接. 如果使用了 SSL/TLS, CPE 必须使用 ACS 提供的证书来鉴权这个 ACS.

3 认证模型

认证模型[5]是 SSL 协议规定的通信双方如何通过证书实现身份验证的框架, 常用的认证模型有客户端对服务器的身份认证和客户端与服务器互相身份认证两种模型.

3.1 客户端对服务器的身份认证模型

客户端对服务器的身份认证模[6]型在 web 浏览器和服务器之间建立 SSL 链接时使用, 该模型结构图如下图 1. 初始化时, SSL Server 加载一个由 SSL Client 信任的 CA 证书签发的证书, 当 Client 认证 Server 是否可信时, Server 将此证书发送给 Client 与 Client 信任的 CA 证书做验证,验证通过, 将建立可信的 SSL 链接, 验证不通过, 将阻止建立可信的 SSL 链接.

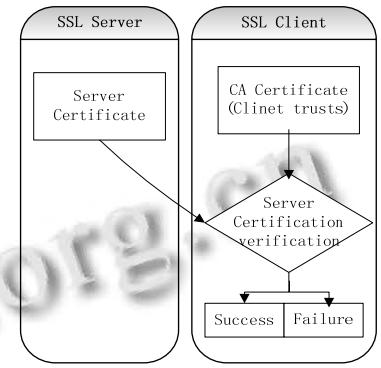


图 1 客户端对服务器的身份认证模型结构图

3.2 客户端/服务器互相身份认证模型

一般情况下只要客户端对服务器身份认证即可, 但是出于实际安全需要, 为防止未经认证许可客户端直接接入服务器, 影响服务器的安全稳定, 也需要服务器对客户端进行身份认证, 因此提出了客户/服务器互相身份认证模型, 该模型结构图如下图 2. 实际使用中客户端和服务器信任的 CA 证书可以是同一个证书, 初始化时, SSL Server 加载一个由双方共同信任的 CA 证书签发的证书, SSL Client 也加载一个由双方共同信任的 CA 证书签发的证书, Server 将此证书发送给 Client 与 CA 证书做验证, 同时 Client 将此证书发送

给 Server 与 CA 证书做验证, 双方验证通过, 将建立可信的 SSL 链接, 双方验证不通过, 将阻止建立可信的 SSL 链接, 下文设计并实现了这种模型.

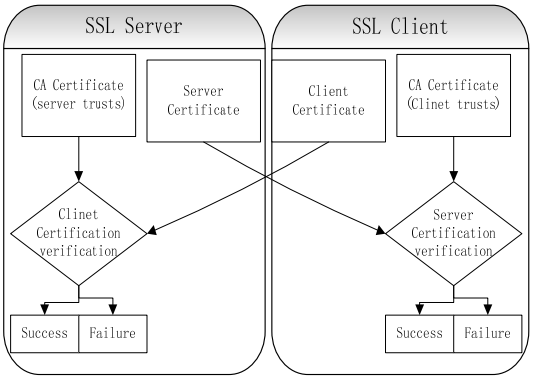


图 2　客户端与服务器互相身份认证模型结构图

4　设计实现CPE-ACS安全认证

根据 TR069 协议, ACS Server 与 CPE client 之间建立会话有两种方式[7], ACS 发起异步会话或者 CPE 发起会话. ACS 发起一步会话的链接请求必须是一个到 CPE 指定的 URL 的 HTTP1.1GET, 链接请求必须使用 HTTP, 而不是 HTTPS, 所以 ACS 向 CPE 发起的链接请求不需要加密的, 而 CPE 发起会话的链接请求必须是一个到 ACS 指定的 URL 的 HTTP1.1 POST, 该链接请求可以使用 HTTP, 也可以使用 HTTPS, 通讯过程需要安全认证机制, 因此认证过程的设计主要集中在这部分.

本文使用文献[8]中的方式, 基于 gSoap 来生成客户端和服务端的基础代码, 然后在此基础上, 调用 OpenSSL 提供的 API 来实现 CPE-ACS 互相身份认证, 使用的证书由 OpenSSL 生成. CPE-ACS 具体认证过程实现流程如下图 3.

1) 初始化阶段. 客户端和服务器端的初始化是类似的, 只有加载证书时有所区别.

SSL_library_init(); /*在所有的 OpenSSL API 之前调用*/

OpenSSL_add_all_algorithms(); /*加载加密和 hash 算法*/

OpenSSL_add_all_digests();

ctx = SSL_CTX_new(SSLv23_method());/*选择使用 SSL/TLS 的版本, 生成 SSL 上下文句柄 ctx*/

SSL_CTX_load_verify_locations(ctx, cafile,

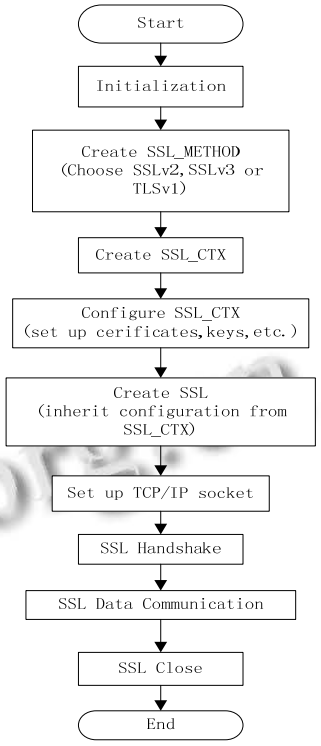


图 3　CPE-ACS 互相身份认证流程图

NULL);/* 为了能够基于根证书进行验证, 必须加载对端发布的 CA 证书, 如果是客户端调用, 则加载服务器 发布的 CA 证书; cafile 是证书的完整路径及证书名. */

SSL_CTX_use_certificate_chain_file(ctx, keyfile); /* 加载私有证书, keyfile 是证书的完整路径及证书名. */

SSL_CTX_use_PrivateKey_file(ctx, keyfile, SSL_FILETYPE_PEM); /* 加载私有 key 文件, 私有证书和私有 key 文件可以存放到一个文件中, 在证书生成一节将详细解释, 因此参数 keyfile 可以和上一个函数调用中的 keyfile 一致. */

SSL_CTX_set_options(ctx, SSL_OP_ALL | SSL_OP_NO_SSLv2 |SSL_OP_NO_TLSv1); /* 本例只使用 SSL v3, 所以需要 disable SSL v2 和 TLS v1. */

SSL_CTX_set_verify(ctx, SSL_VERIFY_PEER, fsslverify);/* 设置需要验证标志和验证回调函数, fsslverify 函数主要是调用 X509_NAME_oneline 对证书进行验证. */

2) 握手阶段. 客户端和服务器端在此阶段的实现是有差别的, 和 socket 编程类似, 客户端应该等到有

绝对时间时才和服务器端链接. 如果客户端选择了在没有绝对时间之前和服务器端进行链接, 则必须忽略服务器端证书中包含绝对时间的内容.

```
/*服务器端*/
master = (int)socket(AF_INET, SOCK_STREAM, 0);
bind(master, (struct sockaddr*)& peer, (int) peerlen);
listen(master, backlog);
sock = accept(master, (struct sockaddr*)& peer, (SOAP_SOCKLEN_T*)&peerlen);
ssl = SSL_new(ctx);/* 创建 SSL 结构*/
SSL_set_fd(ssl, sock); /*关联 socket 进 SSL 结构*/
SSL_accept(ssl);/* 执行 SSL 握手*/
/*客户端*/
sock = (int)socket(AF_INET, SOCK_STREAM, 0);
connect(sock, (struct sockaddr*)& peer, sizeof(peer));
ssl = SSL_new(ctx);/* 创建 SSL 结构*/
r = SSL_connect(ssl));
```

3) 证书验证. 证书验证时, 需要对证书进行可信度和有效期验证, 如果证书的通用名与该服务器的主机名不匹配, 此证书被标记为值得怀疑的, 如果这个证书已经超期, 或者包含一个不可信的签名, 那么这个证书就会被标记为无效.

由于证书验证不是 SSL 标准的一部分, OpenSSL 不会根据主机名对该证书的“名字”进行检查, 所以需要手动添加验证代码, 对通用名和主机名进行比较, 确保这个证书是来自于相同的主机. 证书的“名字”是证书中的 Common Name (通用名)字段, 首先使用 X509_get_subject_name() 从证书中检索 X509_NAME 结构, 返回一个指向 X509_NAME 的对象, 然后使用 X509_NAME_get_text_by_NID() 来检索通用名, 并保存到一个字符串中, 最后将通用名与主机名进行匹配.

```
SSL_get_verify_result(ssl);/*对对端提供的证书进行内部验证*/
X509 *peer = SSL_get_peer_certificate(ssl);/* 获取指向对端证书的指针*/
char commonName [512];
X509_NAME*name=X509_get_subject_name(peerCertificate);
X509_NAME_get_text_by_NID(name, NID_commonName, commonName, 512);
if(stricmp(commonName, hostname) != 0)
{
    /* 证书验证不通过处理*/
}
```

4) 数据交换.

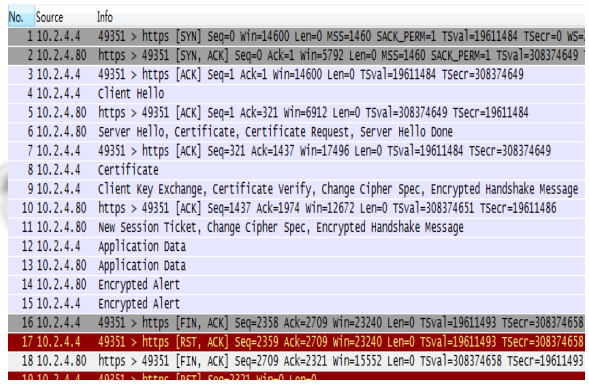
```
SSL_write(ssl, buf, (int)n);
SSL_read (ssl, buf, (int)n);
```

5) 关闭链接. 关闭 SSL 链接和 socket, 并释放资源, 顺序不可颠倒.

```
SSL_shutdown(ssl);
close(sock);
SSL_free(ssl);
SSL_CTX_free(ctx);
```

5 实验及分析

本实验实现了客户端与服务器双向身份验证模型, 服务器安装操作系统为 ubuntu 10.04, 其 ip 地址设置为 10.2.4.80, 客户机安装操作系统为 opensusue 12.2 , 其 ip 地址设置为 10.2.4.4, 在客服机使用 tcpdump 抓取交互数据包. Server 设置验证客户端证书, Client 设置验证服务器证书. 抓包截图如下图 4 所示, 下文将对此内容进行详细分析.



No.	Source	Info
1	10.2.4.4	49351 > https [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=19611484 TSecr=0 WS=
2	10.2.4.80	https > 49351 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=308374649
3	10.2.4.4	49351 > https [ACK] Seq=1 Ack=1 Win=14600 Len=0 TSval=19611484 TSecr=308374649
4	10.2.4.4	Client Hello
5	10.2.4.80	https > 49351 [ACK] Seq=1 Ack=321 Win=6912 Len=0 TSval=308374649 TSecr=19611484
6	10.2.4.80	Server Hello, Certificate, Certificate Request, Server Hello Done
7	10.2.4.4	49351 > https [ACK] Seq=321 Ack=1437 Win=17496 Len=0 TSval=19611484 TSecr=308374649
8	10.2.4.4	Certificate
9	10.2.4.4	Client Key Exchange, Certificate Verify, Change Cipher Spec, Encrypted Handshake Message
10	10.2.4.80	https > 49351 [ACK] Seq=1437 Ack=1974 Win=12672 Len=0 TSval=308374651 TSecr=19611486
11	10.2.4.80	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
12	10.2.4.4	Application Data
13	10.2.4.80	Application Data
14	10.2.4.80	Encrypted Alert
15	10.2.4.4	Encrypted Alert
16	10.2.4.4	49351 > https [FIN, ACK] Seq=2358 Ack=2709 Win=23240 Len=0 TSval=19611493 TSecr=308374658
17	10.2.4.4	49351 > https [RST, ACK] Seq=2359 Ack=2709 Win=23240 Len=0 TSval=19611493 TSecr=308374658
18	10.2.4.80	https > 49351 [FIN, ACK] Seq=2709 Ack=2321 Win=15552 Len=0 TSval=308374658 TSecr=19611493

图 4 抓包截图

前 3 帧为三次握手, 第 4 帧为客户端首先发出问候消息(client hello), 第 6 帧为服务器收到客户端问候消息之后发出的问候消息(server hello), 客户端和服务器问候消息(client hello)被用来在客户端和服务器之间建立安全性能协商, 客户端和服务器的问候消息(client hello)将产生下列属性: 协议版本号、对话标识符、加密套接字及压缩方法, 同时产生两个随机数 ClientHello.random 和 ServerHello.random, 并且相互交

换. 在客户端和服务器问候消息发出之后, 要求验证身份, 服务器将发出其证书, 同时发出一个 certificate request 消息, 然后服务器将发出结束问候消息(server hello done), 并等待客户端的应答.

第 8 帧为客户端发出的证书消息(certificate), 同时发送其证书, 第 9 帧为客户端密钥交换消息(client key exchange)准备发送, 而且消息的内容将取决于在客户端问候消息(client hello)和服务器问候消息(server hello)之间所选择的公钥算法. 如果客户端已经发出了一个具备签名能力的证书, 数字签名后的证书验证消息(certificate verify)将被发送以确认此证书的合法性. 客户端发送一个改变加密说明消息(change cipher spec), 同时将未决的 Cipher Spec 复制到当前 Cipher Spec. 然后, 客户端立用协商的新的算法、密钥、和共享信息发出结束消息(Encrypted Handshake Message).

第 10 帧为服务器发出自己的改变 cipher spec 消息(change cipher spec)作为回应, 将未决(pending) cipher spec 复制到当前 Cipher Spec, 并用新的 cipher spec 发出其结束消息(Encrypted Handshake Message). 此时, 握手结束, 客户端和服务器开始交换应用层数据.

6 结语

本文使用 OpenSSL 实现了对 ACS 和 CPE 的相互认证过程进行加密, 可以防止中间人攻击, 阐述了生成自签证认证证书的方法, 以及认证过程的设计与实现, 通过实验进行了验证, 并分析了 SSL 的交互报文, 达到了预期的效果.

在实际应用中, 还需要考虑 SSL 的不足之处, SSL 协议需要在握手之前建立 TCP 连接, 因此不能对 UDP 应用进行保护; 为了避免由于安全协议的使用而导致网络性能大幅下降, SSL 协议并不是默认地要求进行客户鉴别. 今后可以针对 SSL 的上述不足之处进一步深入研究.

参考文献

1 DSL forum TR-069 issue 1 amendment 2 CPE WAN management protocol[Technical Report]. DSL Home-Technical Working Group. 2007, 12.

2 章晓明,王则林,陆建德.基于 OpenSSL 的嵌入式网络安全通信设计与实现.计算机技术与发展,2006,16(1).

3 HP Open Source Security for OpenVMS Volume 2: HP SSL for OpenVMS. http://h71000.www7. hp.com/doc/83final/ba554_90007/ ch04.html. 2012.

4 殷停.gSOAP 在基于 TR069 协议的网络视频监控系统中的应用.工业控制计算机,2010,23:61–65.

5 SSL 协议 Version 3.0.

6 郭靖,王营冠.基于 openssl 的 CA 认真及 SSL 加密通信.现代电子技术,2012,35(3).

7 殷婷.基于 TR069 协议的网络视频监控系统的关键技术研究[硕士学位论文].杭州:浙江理工大学,2010.

8 龚少麟.OpenSSL 密码安全平台实现机制的研究.计算机与数字工程,2011,11:118–121.