Web Server

移植

--- 1.0

1.0

修订记录

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **日期** | **修订人** | **修订内容** |
| 1.0 | 2022/9/5 | 曾毅 | 初稿 |
|  |  |  |  |

目录

[Web Server 1](#_Toc262)

[移植 1](#_Toc1948)

[--- 1.0 1](#_Toc32229)

[1. 目的 4](#_Toc4121)

[2. 移植 5](#_Toc192)

[2.1 移植 5](#_Toc26034)

[2.2 编译 5](#_Toc19483)

[3. Server配置和启动 6](#_Toc27475)

[3.1 http web服务 6](#_Toc8130)

[3.2 https web服务 6](#_Toc28645)

[3.3 证书 6](#_Toc3099)

[3.3.1 根授权证书 6](#_Toc21195)

[3.3.2 自授权证书 12](#_Toc10542)

[4. JSON数据适配 13](#_Toc29023)

[4.1 UBUS 13](#_Toc5814)

[4.2 uhttpd配置ubus解析 13](#_Toc19745)

[4.3 启动服务端 13](#_Toc5628)

[4.4 通过ubus访问服务端 13](#_Toc6341)

[4.5 通过http访问服务端 14](#_Toc21326)

[4.6 通过https访问服务端 14](#_Toc2620)

# 目的

本文档用于描述uhttpd Web Server的移植，安装和调试。

# 移植

uhttpd组件来自于OpenWRT，主要在OpenWRT上使用，由于OpenWRT社区比较活跃并且家庭路由器也在广泛使用uhttpd，使用uhttpd搭建Web Server可以更好的使用公司资源。

要编译uhttpd还依赖json-c，libubox和ubus组件包。

接下来主要是基于Connect平台描述对json-c，libubux，ubus以及uhttpd的编译。

## 移植

首先应该在对应开源社区下载相应的开源包；

然后在Connect系统的package目录建立对应目录，并且将代码放到dl和code两个子目录；

从其他开源目录拷贝一份build.sh和unbuild.sh编译脚本修改其中的组件名字和版本号。

## 编译

在Connect系统的cppf编译平台中新增了CPPF\_OPENSOURCE\_PKG\_COMPILE变量，该变量用于节约开源包的编译时间，默认情况下该变量为0直接引用之前编译好的库和头文件，如果需要重新编译之前已经适配好的组件包需要打开文件：“vi ./tools/compile/cppf-s.ini”，将export CPPF\_OPENSOURCE\_PKG\_COMPILE = 0 改为 export CPPF\_OPENSOURCE\_PKG\_COMPILE = 1。

uhttpd开源包的编译顺序为，

|  |
| --- |
| ./cppf dc json-c  ./cppf dc ncurses  ./cppf dc readline  ./cppf dc libubox  ./cppf dc ubus  ./cppf dc openssl  ./cppf dc uhttpd |

uhttpd编译后的uhttpd进程即为web server。

# Server配置和启动

uhttpd提供了一些配置接口，我们主要关心绑定接口，页面目录和JSON解析字符识别。

## http web服务

通过uhttpd -p可以指定绑定的IP和端口，

|  |
| --- |
| ./uhttpd -p 0.0.0.0:80 |

在浏览器上输入IP地址即可访问http web服务。

## https web服务

使用uhttpd -s可以指定https绑定的IP和端口，

|  |
| --- |
| ./uhttpd -s 0.0.0.0:443 |

要使用https不光要绑定端口，还需要指定证书和密钥，通过-C指定证书，-K指定密钥，

|  |
| --- |
| ./uhttpd -s 192.168.111.87:443 -C /tmp/self.cert -K /tmp/self.key |

## 证书

我们可以通过openssl制作两种证书，

### 根授权证书

#### 根CA授权

根CA授权只需要建立根CA，然后对证书进行授权，

第一步，建立制作证书目录或者直接依托HOST平台openssl默认配置也是可以的，

|  |
| --- |
| mkdir ca  cd ca  mkdir certs crl newcerts private  chmod 700 private  touch index.txt  echo 1000 > serial |

第二步，生成openssl的配置文件openssl.cnf，依托HOST平台也可以，

|  |
| --- |
| vi ./openssl.cnf  # OpenSSL root CA configuration file.  # Copy to `/root/ca/openssl.cnf`.  [ ca ]  # `man ca`  default\_ca = CA\_default  [ CA\_default ]  # Directory and file locations.  dir = /home/zengyi/share/ca  certs = $dir/certs  crl\_dir = $dir/crl  new\_certs\_dir = $dir/newcerts  database = $dir/index.txt  serial = $dir/serial  RANDFILE = $dir/private/.rand  # The root key and root certificate.  private\_key = $dir/private/ca.key.pem  certificate = $dir/certs/ca.cert.pem  # For certificate revocation lists.  crlnumber = $dir/crlnumber  crl = $dir/crl/ca.crl.pem  crl\_extensions = crl\_ext  default\_crl\_days = 30  # SHA-1 is deprecated, so use SHA-2 instead.  default\_md = sha256  name\_opt = ca\_default  cert\_opt = ca\_default  default\_days = 375  preserve = no  policy = policy\_loose  [ policy\_strict ]  # The root CA should only sign intermediate certificates that match.  # See the POLICY FORMAT section of `man ca`.  countryName = match  stateOrProvinceName = optional  organizationName = optional  organizationalUnitName = optional  commonName = supplied  emailAddress = optional  [ policy\_loose ]  # Allow the intermediate CA to sign a more diverse range of certificates.  # See the POLICY FORMAT section of the `ca` man page.  countryName = optional  stateOrProvinceName = optional  localityName = optional  organizationName = optional  organizationalUnitName = optional  commonName = supplied  emailAddress = optional  [ req ]  # Options for the `req` tool (`man req`).  default\_bits = 2048  distinguished\_name = req\_distinguished\_name  string\_mask = utf8only  # SHA-1 is deprecated, so use SHA-2 instead.  default\_md = sha256  # Extension to add when the -x509 option is used.  x509\_extensions = v3\_ca  [ req\_distinguished\_name ]  # See <https://en.wikipedia.org/wiki/Certificate\_signing\_request>.  countryName = Country Name (2 letter code)  stateOrProvinceName = State or Province Name  localityName = Locality Name  0.organizationName = Organization Name  organizationalUnitName = Organizational Unit Name  commonName = Common Name  emailAddress = Email Address  # Optionally, specify some defaults.  countryName\_default = CN  stateOrProvinceName\_default = SICUAN  localityName\_default =  0.organizationName\_default = WWYL  organizationalUnitName\_default =  emailAddress\_default =  [ v3\_ca ]  # Extensions for a typical CA (`man x509v3\_config`).  subjectKeyIdentifier = hash  authorityKeyIdentifier = keyid:always,issuer  basicConstraints = critical, CA:true  keyUsage = critical, digitalSignature, cRLSign, keyCertSign  [ v3\_intermediate\_ca ]  # Extensions for a typical intermediate CA (`man x509v3\_config`).  subjectKeyIdentifier = hash  authorityKeyIdentifier = keyid:always,issuer  basicConstraints = critical, CA:true, pathlen:0  keyUsage = critical, digitalSignature, cRLSign, keyCertSign  [ usr\_cert ]  # Extensions for client certificates (`man x509v3\_config`).  basicConstraints = CA:FALSE  nsCertType = client, email  nsComment = "OpenSSL Generated Client Certificate"  subjectKeyIdentifier = hash  authorityKeyIdentifier = keyid,issuer  keyUsage = critical, nonRepudiation, digitalSignature, keyEncipherment  extendedKeyUsage = clientAuth, emailProtection  [ server\_cert ]  # Extensions for server certificates (`man x509v3\_config`).  basicConstraints = CA:FALSE  nsCertType = server  nsComment = "OpenSSL Generated Server Certificate"  subjectKeyIdentifier = hash  authorityKeyIdentifier = keyid,issuer:always  keyUsage = critical, digitalSignature, keyEncipherment  extendedKeyUsage = serverAuth  [ crl\_ext ]  # Extension for CRLs (`man x509v3\_config`).  authorityKeyIdentifier=keyid:always  [ ocsp ]  # Extension for OCSP signing certificates (`man ocsp`).  basicConstraints = CA:FALSE  subjectKeyIdentifier = hash  authorityKeyIdentifier = keyid,issuer  keyUsage = critical, digitalSignature  extendedKeyUsage = critical, OCSPSigning |

第三步，创建CA密钥，

|  |
| --- |
| openssl genrsa -aes256 -out private/ca.key.pem 4096  #passwd: 12345678 |

第四步，创建CA证书，

|  |
| --- |
| openssl req -config openssl.cnf \  -key private/ca.key.pem \  -new -x509 -days 7300 -sha256 -extensions v3\_ca \  -out certs/ca.cert.pem  #Common Name []:WWYL  #Email Address []:zengyi@micronet.ltd |

至此CA配置完毕，接下来创建自身的密钥，证明以及对证明进行证书签验，

第五步，生成设备密钥，

|  |
| --- |
| openssl genrsa -out self.key 2048 |

该密钥会被放到设备中，使用uhttpd -K指定。

第六步，生成证明文件，证明文件将被CA签核，

|  |
| --- |
| openssl req -new -key self.key -out self.csr |

以上两步生成了设备证书的基本信息，接下来由CA对证明文件进行签核并发放证书，

第七步，

|  |
| --- |
| openssl ca -config ./openssl.cnf \  -extensions server\_cert -days 375 -notext -md sha256 \  -in ./private/self.csr \  -out ./private/self.cert |

Self.cert证书将被放到设备中，使用uhttpd -C指定。

#### 中间CA授权

中间证书授权主要是由CA根证书授权中间证书，再由中间证书对证明文件签核，

|  |
| --- |
| #1. init  mkdir ca  cd ca  mkdir certs crl newcerts private  chmod 700 private  touch index.txt  echo 1000 > serial  #2. get openssl.cnf，参考3.3.1.1第二步  #3. create ca key  openssl genrsa -aes256 -out private/ca.key.pem 4096  #passwd: 12345678  #4. create ca cert  openssl req -config openssl.cnf \  -key private/ca.key.pem \  -new -x509 -days 7300 -sha256 -extensions v3\_ca \  -out certs/ca.cert.pem  #Country Name (2 letter code) [GB]:CN  #State or Province Name [England]:SICUAN  #Locality Name []:CHENGDU  #Organization Name [Alice Ltd]:MU  #Organizational Unit Name []:SW  #Common Name []:MU  #Email Address []:zengyi@micronet.ltd  #5. create middle key and csr  mkdir intermediate  cd intermediate  mkdir certs crl csr newcerts private  chmod 700 private  touch index.txt  echo 1000 > serial  echo 1000 > crlnumber  cd ..  openssl genrsa -aes256 -out intermediate/private/intermediate.key.pem 4096  #passwd:12345678  #get openssl.cnf，注意中间证书同样需要openssl.cnf文件  openssl req -config intermediate/openssl.cnf -new -sha256 \  -key intermediate/private/intermediate.key.pem \  -out intermediate/csr/intermediate.csr.pem  #Country Name (2 letter code) [GB]:CN  #State or Province Name [England]:SICUAN  #Locality Name []:CHENGDU  #Organization Name [Alice Ltd]:MU  #Organizational Unit Name []:SW  #Common Name []:MU  #Email Address []:zengyi@micronet.ltd  #6. middle ca，可以看到中间证书是由ca根证书授权的，  openssl ca -config openssl.cnf -extensions v3\_intermediate\_ca \  -days 3650 -notext -md sha256 \  -in intermediate/csr/intermediate.csr.pem \  -out intermediate/certs/intermediate.cert.pem  #7. web key  openssl genrsa -out intermediate/private/web.key.pem 2048  chmod 400 intermediate/private/web.key.pem  #8. web csr  openssl req -config intermediate/openssl.cnf \  -key intermediate/private/web.key.pem \  -new -sha256 -out intermediate/csr/web.csr.pem  #Country Name (2 letter code) [GB]:CN  #State or Province Name [England]:SICUAN  #Locality Name []:CHENGDU  #Organization Name [Alice Ltd]:MU  #Organizational Unit Name []:SW2  #Common Name []:MU  #Email Address []:zengyi@micronet.ltd  #9. get openssl-switch.cnf，如果有必要需要增加subj章节  #10. web ca  openssl ca -config intermediate/openssl-switch.cnf \  -extensions server\_cert -days 375 -notext -md sha256 \  -in intermediate/csr/web.csr.pem \  -out intermediate/certs/web.cert.pem |

### 自授权证书

自授权不需要根证书，但是有的浏览器禁止使用自授权证书，

|  |
| --- |
| openssl genrsa -out ca.key 2048  openssl req -new -out ca.csr -key ca.key -config openssl.cnf  openssl x509 -req -days 3650 -in ca.csr -signkey ca.key -out ca.crt -extensions v3\_req -extfile openssl.cnf |

# JSON数据适配

uhttpd使用JSON+UBUS进行系统数据访问。

## UBUS

首先要启动UBUS服务，

|  |
| --- |
| ./ubusd & |

## uhttpd配置ubus解析

ubusd服务启动以后，需要配置uhttpd解析特定字段进行ubus通信以及提供json数据格式，

|  |
| --- |
| ./uhttpd -s 192.168.111.87:443 -C /tmp/web.cert -K /tmp/web.key -h /tmp -u /ubus |

## 启动服务端

uhttpd只是作为客户端，还需要启动服务端提供数据的配置和收集，这里假设有一个服务进程app-demo，app-demo会向ubusd注册“对象”和“函数”，

|  |
| --- |
| ./app-demo &  /tmp # ./ubus list test -v  'test' @2e33fd13  "hello":{"id":"Integer","msg":"String"}  "watch":{"id":"Integer","counter":"Integer"}  "count":{"to":"Integer","string":"String"} |

## 通过ubus访问服务端

ubus是ubus提供的调试进程，可以通过它直接调用服务，

|  |
| --- |
| /tmp # ./ubus call test hello '{"id":1,"msg":"zengyi"}'  {  "message": "test received a message: zengyi"  } |

## 通过http访问服务端

通过http访问服务端也是可以的，

|  |
| --- |
| ./uhttpd -p 192.168.111.87:80 -h /tmp -u /ubus |

在浏览器或者主机linux系统上，

|  |
| --- |
| zengyi~/share/ubus$ wget --post-data='{"id":1,"msg":"zengyi"}' http://192.168.111.87:80/ubus/1/test/hello  --2022-09-07 00:08:58-- http://192.168.111.87/ubus/1/test/hello  Connecting to 192.168.111.87:80... connected.  HTTP request sent, awaiting response... 200 OK  Length: 49 [application/json]  Saving to: ‘hello’  hello 100%[==================================================>] 49 --.-KB/s in 0s  2022-09-07 00:08:59 (839 KB/s) - ‘hello’ saved [49/49]  zengyi~/share/ubus$ cat ./hello  {  "message": "test received a message: zengyi"  }zengyi~/share/ubus$ |

## 通过https访问服务端

通过https访问服务端也是可以的，

|  |
| --- |
| ./uhttpd -s 192.168.111.87:443 -C /tmp/web.cert -K /tmp/web.key -h /tmp -u /ubus |

在浏览器或者主机linux系统上，

|  |
| --- |
| zengyi~/share/ubus$ wget --no-check-certificate --post-data='{"id":1,"msg":"zengyi"}' https://192.168.111.87:443/ubus/1/test/hello  --2022-09-07 00:10:45-- https://192.168.111.87/ubus/1/test/hello  Connecting to 192.168.111.87:443... connected.  WARNING: cannot verify 192.168.111.87's certificate, issued by ‘CN=DEBI,O=DEBI,ST=SICUAN,C=CN’:  Unable to locally verify the issuer's authority.  WARNING: certificate common name ‘DEBI’ doesn't match requested host name ‘192.168.111.87’.  HTTP request sent, awaiting response... 200 OK  Length: 49 [application/json]  Saving to: ‘hello’  hello 100%[==================================================>] 49 --.-KB/s in 0s  2022-09-07 00:10:46 (7.11 MB/s) - ‘hello’ saved [49/49]  zengyi~/share/ubus$ cat ./hello  {  "message": "test received a message: zengyi"  }zengyi~/share/ubus$ |

一点儿毛病没有。