ip.buffer App Note
AN012 : SSL Certificates

Date	*Author*	*Release*
2010-10-08	MP	Initial draft

Scannex Electronics Ltd, UK
t: +44(0)1273 715460
f: +44(0)1273 715469

http://www.scannex.co.uk
info@scannex.co.uk

Scannex LLC, USA
t: 1-866-4BUFFER
(1-866-428-3337)

http://www.scannex.com
info@scannex.com

Table of Contents

1. Introduction

The TLS/SSL security protocol provides a mechanism for checking the identity of both servers and clients.

The ip.buffer can check the identity of remote servers it is 'pushing' data to, check the identity of clients that are 'pulling' data, as well as provide certificates so that remote servers and clients can check the identity of the ip.buffer.

The usage of the certificates usually falls under the term “PKI” - Public Key Infrastructure, although the certificates can be used within a private environment as well.

- The TLS/SSL protocol, and the corresponding certificate features are only available in the “SSL” firmware version of the ip.buffer. The ip.buffer is shipped with “CF” (Crypto-Free) firmware, but the SSL firmware is available for download from our web-site. See Application Note “AN004 – SSL/TLS Security”

1.1. How SSL certificates work

Essentially, SSL certificates are split into two parts: a private key, and a public certificate.

The private key allows for decryption of the data and should be closely guarded. It is never revealed to third parties[1], and is usually kept in encrypted form on the server.

The public certificate can be given out freely and only contains the information needed to encrypt data, and to validate the user of the certificate. Only the owner of the private key can successfully decrypt the data stream.

When a client connects to an SSL-enabled server, the server will always present the client with one or more public certificates[2]. The client will normally use these certificates to check that:

- The server is recognised by a party that is trusted by the client
- The server hasn't been 'spoofed' or replaced by another machine (that may be trying to dupe the client into revealing secret information)

The server can optionally request the client for a certificate, or certificate chain. The server can therefore verify the identity of the client.

Both the client and server can then decide whether to continue with the communication, or to abort the connection.

Assuming that both decide to continue, the two parties will agree on an encryption algorithm (cipher suite) and a session key. The session key is agreed on without exchanging the key itself (otherwise others could decrypt the data) by using some sophisticated mathematics. The data across the TLS/SSL link is then completely encrypted using that session key.

[1] If the private key is released then security is compromised and new certificates should be generated.

[2] If the server's certificate is part of a chain-of-trust there will be multiple certificates.

1.2. Certificates for public web servers

Most people will see certificates when visiting online shopping web-sites, or banking web-sites. When there is a single server with a wide number of unknown clients (i.e. a "public facing web server") it is important that the certificate can be linked to some form of public trust.

All web-browsers ship with a large set of publicly recognised Certificate Authority (CA) certificates. For example, commercial companies such as Thawte and Verisign will have their CAs included in the browser distribution.

To illustrate how this works, consider a fictitious bank "Acme Banking":

- Acme approach a public certificate authority, such as Verisign:
  - Acme pay a large fee for the certificate.
  - The certificate will be 'signed' by the Verisign CA certificate
  - The certificate will have a very limited life-span, typically one or two years.
    - Before the certificate expires, Acme will need to pay more money for a new certificate.
- Once Verisign supply the certificate and private key to Acme, Acme install the certificate and key on their web-server
- A user now visits the Acme Banking web site:
  - The Acme web-server sends the public certificate to the user's web-browser
  - The web-browser will examine the certificate and ask itself the following questions:
    - Does the URL typed match the certificate's Common Name (CN) field?
    - Is the certificate within date?
    - Do I know the signing authority name? (In this example Verisign)
    - Does the digital signature on the certificate match the signature that I have in Verisign's CA certificate?[3]

This scenario is common, but expensive. Thankfully for the ip.buffer there are several other options available.

[3] Remember the Verisign certificate will be issued with the web-browser distribution.

1.3. ip.buffer certificates

Generally you will not have many unknown clients accessing a single ip.buffer (as is the case with a public web-server). The ip.buffers will normally not be "public facing". In fact, the model is usually reversed – you will have a very few known clients accessing a larger number of ip.buffers. Consequently, the use of certificates is quite different to the 'Acme Banking' example.

It is fairly simple to create a 'self-signed' certificate. A self-signed certificate is generated and controlled by you (or an IT department) without having to pay a commercial company to generate the certificate. Self-signed certificates can have much longer life-spans, and are far more flexible.

You can use a 'self-signed' certificate to:

- Verify that the ip.buffer is genuine (and has not been tampered with)
- Optionally to verify that the ip.buffer came through an approved channel (and not from some other supply route)

Each ip.buffer should have an individual "Device Certificate". The Common Name (CN) field of the certificate should match the URL you use to access the buffer.

The ip.buffers ship with a chain of Scannex generated self-signed certificates, in the following chain-of-trust:

1. "**`Scannex Root CA`**". This is Scannex's self-signed root certificate. We can generate child certificates that have the ability to sign device certificates.
2. "**`Scannex ip.buffer`**". This is the certificate used for all ip.buffers. Other products in the future will have a different certificate (but still descended from, and signed by, the "Scannex Root CA")
3. **ip.buffer Device certificate**. This is the certificate for the individual ip.buffer.

Depending on the software you are using on the PC and the level of security you wish to maintain, there are two options available:

- Make use of the pre-installed "`Scannex Root CA`" and "`Scannex ip.buffer`" certificates to generate the ip.buffer's device certificate. See 3.Using the Scannex supplied CA certificate
- Use your own root CA and/or signing certificate to generate the ip.buffer's device certificate. See 4.Using your own CA certificate

These options will be discussed in the following sections.

2. Creating the device certificate

When shipped, the ip.buffer will include a dummy certificate that is for IP address "`192.168.0.235`" (i.e. the Common Name CN field for the certificate will be 192.168.0.235, and a web-browser will always complain!)

Once the ip.buffer is installed you should always create a proper device certificate.

The ip.buffer includes the ability to create its own device certificate through its web-interface.

Use the "`Setup`", "`Advanced: Configuration, Scripts, Upgrade...`" links on the ip.buffer web interface, and choose "`Server Certificate: Generate`"[4].

You will be presented with a form to fill in:

- **Hostname**: This will default to the browser's URL that you used. It should always match, otherwise the web-browser will give a warning when you access the buffer using SSL.
- **Organisation Name, Organisation Unit, City, State or Province**: Fill in details about your company. This information is stored within the public certificate that the browser will see.
- **Country**: Enter a two digit country code. e.g. "`GB`", or "`US`". This field must be present.
- **Email**: An optional field.

Once you click "`Save`" the ip.buffer will work away and create a certificate[5]. The private key is stored within the ip.buffer and cannot be extracted – it remains internal to the ip.buffer.

- If you are currently accessing the ip.buffer through an HTTPS link then you will need to restart your browser to see the newly generated certificate.

[4] If you don't see the menu options, then check that the SSL-enabled firmware is installed. The footer of each ip.buffer web-page includes the firmware version. It should look something like "Version IPBSSL2.60.159 2010-07-20 / i5.0.10".

[5] Calculating the certificate can take quite a time!

3. Using the Scannex supplied CA certificate

The simplest approach is to use the Scannex-supplied CA certificates chain. You simply generate the device certificate as outlined in "2.Creating the device certificate".

Depending on the client and server software you are using on the PC, you have three options for approving the ip.buffers:

- Approve everything from Scannex. In this case you should import and approve the "`Scannex Root CA`" as a trusted certificate[6].
  - All ip.buffers, and any future products will be automatically approved.
- Approve all ip.buffers from Scannex. In this case you should import the "Scannex ip.buffer" certificate as a trusted certificate.
  - All ip.buffers will be approved.
  - If you do not import and trust the "`Scannex Root CA`" then future products will require authorisation and will not automatically be approved.
- Approve each device certificate one-at-a-time.
  - This approach could potentially involve a lot of certificates.
  - Any new ip.buffers, or changes to certificates will require the individual certificates to be added to your PC software.

[6] Microsoft security models appear to assume a "public-facing" scenario and only have this option. IE7 and Google Chrome on the PC need to have the Root CA approved to stop warning the user.

4. Using your own CA certificate

The ip.buffer includes the ability to install a signing certificate (usually a self-signed signing certificate), so that the ip.buffer can create a device certificate descended and signed by your CA.

For example, let's assume you have an 'Acme' company-wide root CA. You could proceed as follows:

- Create a new certificate with signing ability for your ip.buffers.
  - For example, create an “Acme ip.buffers” certificate.
  - This certificate should be signed by your root CA.
  - If your browsers, servers and other clients are already programmed to trust the company-wide root CA then anything signed by this certificate will be automatically trusted as well
    - Alternatively, install the public certificate into your browsers so that the ip.buffers are trusted.
- Upload the public certificate and the private key to the ip.buffer
  - Use the ip.buffer page[7]:
    https://192.168.0.235**/oem/loadcer.htm**
  - The private key should be closely guarded!
    - For example, only the owner of the certificate (e.g. the IT department) should load the certificate into each buffer before it is deployed.
    - Compromising the private key would mean that anyone could create another device certificate that would appear to be approved by the root CA.
  - For maximum security, the certificate and private key should be uploaded using an HTTPS session on a trusted network (so that no one can eaves-drop on the conversation and potentially obtain the private secret from a Wireshark tap).
- Once the public certificate and private key are loaded into the ip.buffer they *cannot* be read out from the ip.buffer
- Finally, create the device certificate as outlined in “2.Creating the device certificate”

[7] Obviously, use the appropriate IP address or name for your buffer!

5. Installing certificates within your browser

The specific instructions for installing and trusting a certificate on your browser vary between operating systems and individual browsers.

Before accepting the certificate into your OS or browser (especially a Root CA certificate), you should check that the fingerprint matches the actual certificate you expect. The default Scannex-supplied fingerprints are listed in "8.Scannex Certificate Fingerprints". If you are using your own CA then you should check with the department that issued your CA and get them to confirm the fingerprint via telephone or email (or some other 'out-of-band' route)[8].

However, here are some hints:

5.1. Firefox 3

You will get a "`This Connection is Untrusted`" warning box on first visiting the ip.buffer.

- Click "`I Undestand the Risks`"
- Click "`Add Exception...`" button
- A dialog appears. Click "`Get Certificate`".
  - If you want to trust just this ip.buffer then simply click on "`Confirm Security Exception`"
  - If you want to trust one of the signing certificates you can:
    - Click on "`View`"
    - Click the "`Details`" tab where you can see the Certificate Heirarchy
    - Select the certificate you wish to trust
    - Click the "`Export`" button and export the certificate to a file.
    - You can then import the certificate using
      - The "`Tools`", "`Options`" menu item
      - Then click on the "`Advanced`" tab, and select "`Encryption`" sub-tab
      - Click the "`View Certificates`" button, and click on the "`Import`" button to import into Firefox.

You can view and manage Firefox's certificates by using the "Tools", "Options" menu and clicking on the "Advanced" tab, selecting "Encryption" sub-tab and clicking on "View Certificates" button.

- Firefox has the ability to accept and approve just the device certificate, rather than having to explicity accept the Root CA.

[8] This is required so you don't inadvertently install a counterfeit Root CA!

5.2. Internet Explorer 7

On first connection you will see a "There is a problem with this website's security certificate" appear.

- Click "Continue to this website (not recommended)"
- On the top bar you will see a red "Certificate Error" box.
  - Click on this box
  - Click the "View certificates" link that appears
  - Click on the "Certification Path" tab
    - You will see the chain of certificates, and one or more will be marked with a red cross
    - Click on the highest certificate you wish to trust, and click "View Certificate"
      - Now click on "Install Certificate"
      - Follow the instructions to import and install the certificate.
      - You will be presented with a dialog box headed "Security Warning". Double check the SHA1 thumbprint/fingerprint and click "Yes" (it is warning you since IE will then trust all certificates signed by that CA[9])
    - Now the ip.buffer will be approved.

Installed certificates can be viewed and managed by:

- Using the "Tools", "Internet Options" menu item.
- Select the "Content" tab
- Click the "Certificates" button
  - You will find the ip.buffer certificates under the "Other People" tab.
  - Any root CAs that are imported will by under the "Trusted Root Certificate Autorities" tab.

- Unlike Firefox, Internet Explorer **must** have the Scannex Root CA (or your Root CA) installed and approved before it will stop complaining about the ip.buffer certificate! There is no apparent way to approve just the device certificate or the "Scannex ip.buffer" certificate.

[9] If a potential attacker can get you to approve a counterfeit CA then their counterfeit servers will be automatically approved by your IE browser!

5.3. Google Chrome

On first connection you will see a "`The site's security certificate is not trusted!`" red background screen.

- Click the "`Proceed anyway`" button
- When viewing the ip.buffer the "https://" part of the URL will be crossed with a red line.
- Click the padlock icon to the left (it has a red cross)
  - You will be shown a "`Security Information`" dialog
  - Click on the "`Certificate Information`" button
  - The dialog is the same as the Internet Explorer 7 dialog, but does not have the "Install Certificate" option. You will either have to:
    - Approve the certificate CA through IE7
    - Use the "`Copy to File...`" button and manually import by double clicking on the exported file
  - (Chrome uses the same certificate store as IE7)

- Like Internet Explorer, Google Chrome uses the Microsoft certificate method, and requires approval of the "Scannex Root CA".

6. Getting the ip.buffer to validate servers and clients

The ip.buffer also includes facilities to check remote servers have correct certificates, and options to check connecting clients are approved.

The options are fully documented in the ip.buffer Manual.

They can be found in the ip.buffer web-pages: "`Setup`", "`Global: Settings`", "`Certificates`".

The options are:

- **Local Fingerprint**: This is the SHA1 fingerprint of the ip.buffer's server certificate. You can use this value to double check againt the certificate you import or use.
- **Approved Fingerprints**: A list of approved SHA1 fingerprints for server and client certificates.
- **Verify Servers**: You can choose whether remote servers are validated against the list of approved fingerprints. By default the ip.buffer will accept any server certificate.
- **Verify Date**: For a server or client, the ip.buffer can check that the certificate date range is current. By default the ip.buffer will not check the date validity.
- **Verify Name**: For a server the ip.buffer can check whether the URL, name, or IP address matches the CN (Common Name) field of the certificate given by the server. By default the ip.buffer does not check.
- **Verify Clients**: When clients connect to the ip.buffer (e.g. TCP clients, or FTP clients[10]) you can tell the ip.buffer to check the client's certificate and whether it is in the list of approved fingerprints. By default the ip.buffer does not check.

[10] Be aware that many FTP clients and TCP clients do not support client-side certificates, even if they support TLS/SSL! If your client software does not support client certificates then they will be refused connection to the ip.buffer.

7. Authenticating the ip.buffers on your server

When the ip.buffers are 'pushing' data to a central server – through FTP-push, Email-push, or HTTP-post – you can program your server to accept only approved devices[11].

For example, a Microsoft IIS resource can be protected by allowing only certain SSL certificates. You should be able to tell the server to accept either the specific ip.buffer device certificates, or one of the higher level signing certificates, e.g. the “Scannex ip.buffer” certificate, or the “Scannex Root CA” certificate.

This type of authentication is in addition to the username/password style authentication.

[11] This is separate to the checks that the ip.buffer makes of the server. By adding this check on your server you are creating a *mutual* authentication.

8. Scannex Certificate Fingerprints

The Scannex supplied certificates have the following SHA1 fingerprints:

Scannex Root CA	eb 31 f8 dd 5c e1 ea ec 41 97 9e 3a 9d fc 87 b9 f3 d8 dc 17
Scannex ip.buffer	88 b4 de ae d5 45 77 ed ef 01 a7 a3 c7 e3 47 90 34 c2 3f 45
Default certificate “192.168.0.235”	8e 52 81 63 7b 06 a6 d4 8b ef d1 0a 03 05 be 2d 54 0d 74 88

9. References

- Wikipedia information:
  - PKI: http://en.wikipedia.org/wiki/Public_key_infrastructure
  - TLS/SSL: http://en.wikipedia.org/wiki/Transport_Layer_Security
  - Self-signed: http://en.wikipedia.org/wiki/Self-signed_certificate
- Open SSL:
  - http://openssl.org/
- Creating self-signed certificates:
  - For Microsoft IIS6: http://www.somacon.com/p42.php
  - With OpenSSL: http://sial.org/howto/openssl/self-signed/
  - With OpenSSL: http://www.madboa.com/geek/openssl/#cert-self

- Creating certificates can be fiddly and time-consuming. There are also various scripts available for easing the command-line burden! (e.g. I have found that generating certificates on Mac OS X much easier than on Windows)