scannex

ip.buffer App Note
AN009 : Security Considerations

Date	*Author*	*Release*
2008-12-18	MP	Initial draft

Scannex Electronics Ltd, UK
t: +44(0)8707 48 65 65
f: +44(0)8707 48 67 67

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

Since the ip.buffer often needs to exist in the context of a network, considering the security implications of using the ip.buffer is often a requirement for the installation.

The ip.buffer uses a rugged real-time operating system, called INTEGRITY by Green Hills Software (www.ghs.com). Green Hills Software takes security very seriously, and the operating system is built with security in mind. For example, they have recently announced a version of INTEGRITY that is the world's first operating system to achieve NSA's EAL6+ certification ("certified to protect classified information and other high value resources at risk of attack from hostile and well-funded attackers"). The TCP/IP stacks are also by Green Hills Software.

Scannex are supported by Green Hills Software in maintaining the security of the ip.buffer.

However, given the flexibility of the ip.buffer, there are still issues that should be considered when deploying the ip.buffer. The aim of this document is to outline the main issues[1].

[1] This document covers features that are available in version 2.30. However, only a few features are new to v2.30 – most of the security features are available in earlier versions.

2. Protecting the PBX

Usually the ip.buffer is collecting data from the PBX and forwarding this data to a central system for processing.

Sometimes end users can be nervous about the link between the PBX and their network. They may wonder if people who telephone in can access their network systems. Obviously the PBX itself has a pivotal role to play in terms of security. Here are some general guidelines:

- If the PBX is connected with a serial, RS232, cable to the ip.buffer, no one phoning in can gain access to the network. It is impossible for the serial cable used for data logging to carry network traffic.
- Some PBX models accept programming information through the data logging port.
    - If the PBX is connected with a serial cable to the ip.buffer, then a custom cable can be made that has no connection between the ip.buffer's transmit line and the PBX's receive line – it is then impossible to send anything back to the PBX.
    - Most PBXs can disable this feature. When disabled inside the PBX, there is nothing anyone, or the ip.buffer, can do to change the PBX, or make calls.
    - If the ip.buffer is being used as a gateway to allow network-based "moves and changes" (i.e. reprogramming the PBX remotely), then the ip.buffer provides several mechanisms to increase security. See section 6.
    - If the ip.buffer is <u>not</u> being used as a gateway to allow "moves and changes" to the PBX, there is no link between the ip.buffer's delivery mechanism and the PBX. (e.g. an FTP-Push from the ip.buffer will not provide a means to write back to the PBX).
        - The ip.buffer does have a "pass-through" TCP/IP socket that allows for "moves and changes" connection. This can be <u>disabled</u> completely in the programming of the ip.buffer.

3. Using the modem in the ip.buffer

Some versions of the ip.buffer have a built-in modem that provides access to the ip.buffer configuration web-pages and other services.

The internal firmware design of the ip.buffer was built with security in mind:

- When dialling into the ip.buffer from the outside:
  - The ip.buffer uses CHAP (or PAP) authentication to authenticate the incoming caller. Without a correct login, the caller will be disconnected.
  - The incoming PPP caller has no access to the LAN. The TCP/IP stacks in the ip.buffer have had the packet-forwarding stripped at compilation time. Consequently, it is impossible to dial-in and use the ip.buffer as a router onto the LAN.
  - The only device a caller has access to is the ip.buffer itself.
- When the ip.buffer dials out to deliver data, alerts, or logs via the modem:
  - The ip.buffer is firewalled to prevent traffic from entering the ip.buffer while on a (possibly) untrusted PPP network.
  - The ip.buffer maintains control of the modem link – when the data transfer is complete the modem will be hung-up and reset. It is not possible for anyone of the remote network to keep the modem link active.
- If the above is not sufficient:
  - The ip.buffer can be programmed to abort all incoming calls (by setting a zero second answer time). The ip.buffer will never be able to accept an incoming call, or establish a network link (without being reprogrammed).
  - The outbound telephone numbers can be blanked out in the programming. Then the ip.buffer will never attempt to dial out.
  - The modem lead can simply be unplugged!

4. Protecting the ip.buffer configuration

The ip.buffer includes options to increase security for protecting the configuration parameters in the ip.buffer. The specifics are covered in the ip.buffer manual. However, these are some of the features available:

- There is SSL/TLS firmware available for free download from our website[2]. The SSL/TLS firmware adds high strength AES encryption to all services – including the web-access.
    - You can program the ip.buffer to force the use of "https" connections. All traffic between the PC web-browser and the ip.buffer web-server is completely protected.
    - The ip.buffer can be loaded with an SSL certificate – either generated by the ip.buffer, or by the customer. Any web-browsers that connect to the ip.buffer can then prove whether they are communicating with the correct ip.buffer (this protects against "man-in-the-middle" attacks).
- When authenticating with a web-browser the PC will pass the username across the network to the ip.buffer.
    - If the connection is "https" then neither the username, nor password, can be seen by anyone.
    - The ip.buffer can enforce "Digest" authorization. In this mode, the browser will only pass an MD5 signature of the password – never the actual password. This effectively protects the administration password from sniffing attacks on the network[3].
    - !The user should be careful not to store the username/password combination on the PC they are using – especially if the PC is accessible by others!
    - !When not using RADIUS there is only a single username/password for the administration of the ip.buffer. Obviously this should be protected!
        - The ip.buffer does also include a "WebLock" feature that provides a limited time "window" for the username to work. The user must obtain a temporary unlock code from the person holding a secret that has been previously loaded into the ip.buffer. This allows for safe telephone activation, or temporary engineer access.
- The ip.buffer can authenticate back to a central RADIUS server:

[2] Please note that the country you are using the ip.buffer in may have strict legislation on the use of cryptographic firmware – it is your responsibility to make adequate checks in this regard!

[3] The username is visible.

  - !Note that “Digest” authorization is technically impossible when authenticating back to RADIUS. For maximum security, “https” connections should be enforced by the ip.buffer.
  - The central RADIUS server can maintain a large list of username/password combinations for the ip.buffer(s). Not giving the same username & password to every person is a protection – any password that is compromised can be immediately disabled at the RADIUS server (without affecting the other users).

- The ip.buffer can generate several different forms of notification when the configuration is changed, or when an attempted access has been made:
  - SNMP traps are sent out to the designated SNMP trap agent.
  - A syslog message is sent to the designated syslog server.
  - An email or http alert can be sent.
  - Information is also stored in the RAM-based log (which can be emailed or http pushed).
- !Since the configuration itself could contain sensitive information (e.g. even the password for the web-access itself):
  - There is an option to hide all passwords. When viewing configuration data on the ip.buffer the user will see “********”. Anyone sniffing on the network would thus not be able to extract the passwords. (But again, forcing “https” connects prevents all sniffing activities.)
  - The web-pages that have sensitive configuration information are all protected by password in the web-server.
- There are a few pages that are available to view by anyone on the ip.buffer:
  - The “Status” page. This page contains very limited information. It provides enough information to tell if all connections are working. However, it will never divulge user & password information, etc.
- When the ip.buffer emails, or http pushes, the status information to a central system, that status information contains no sensitive data (like usernames, passwords etc).

5. Protecting pass-through ports

The ip.buffer provides “pass-through” sockets that allow bidirectional access to the PBX when “moves and changes” must be done remotely. However, this feature is completely programmable. The ip.buffer provides the following features:

- You can completely disable the pass-through on a channel by setting the listening port to zero.
- When using the SSL/TLS firmware, the socket can be a TLS socket, i.e. encrypted. Any attempts at connecting non-encrypted will be rejected. This is highly desirable when granting access to a PBX that allows moves-and-changes!
- The ip.buffer can be programmed to accept connections from only one IP address. All other PCs will be rejected when they attempt to connect (even before asking for a password!).
- The ip.buffer can authenticate back to a RADIUS server to validate the incoming socket. The user will need to enter a username & password.
- If not using RADIUS, the pass-through socket can be protected by a simple password.
    - !If the socket connection is not encrypted SSL/TLS, then this password could be found by sniffing the network.
- The pass-through port has various modes that can be configured:
    - “Stored” & “Not stored” - these modes provide bidirectional access to the PBX (whether connected by serial or network).
        - !You should protect pass-through sockets with at least one of the above methods.
    - “Monitor” - the pass-through socket is read-only. Nothing sent by the user is ever sent to the PBX.

6. Protecting delivery sockets

The ip.buffer has a very flexible set of delivery methods. Some comments on security for each mode are outlined here:

6.1. Email push (SMTP client)

There is no real risk to the ip.buffer. However, since the ip.buffer is connecting to an SMTP server there should be adequate security in place, especially if that server provides email forwarding.

- The ip.buffer can authenticate to the SMTP server with a username and password.
- When using the SSL/TLS firmware, the ip.buffer can establish either an implicit or explicit encrypted link with the SMTP server before sending any authentication information[4].
- !If not using an SSL/TLS link to the SMTP server, then the data can be sniffed across the network.
    - Scannex provide a proprietary 40-bit encryption for the data that will provide good obfuscation of the data. Obviously the receiving end has to be programmed to decrypt the data too.
    - You can choose to zlib compress the data. This will provide another layer of obfuscation, and will probably prevent all but the most determined hacker from seeing the data. Again, the receiving end must know how to decompress the data[5].

6.2. HTTP post to web server

There is no real risk to the ip.buffer. Considerations are similar to the Email push.

- The ip.buffer can authenticate to the web server with a username and password.
    - !The ip.buffer will only use “Basic” authorization to the web-server, so the username and password can be sniffed.
    - You can put IP-based restrictions in the web-server to enhance security.
- The SSL/TLS firmware can “https” post, using strong encryption. Make sure the url begins with “https://”, and that the SSL/TLS firmware is loaded.
- If not using SSL/TLS, then the same security issues for the transferred data exist – see above.

6.3. FTP Server

The ip.buffer is behaving as a server:

[4] Obviously, the SMTP server has to support encrypted connections!

[5] !The email itself does disclose that the data is zlib compressed, when this option is used.

- The channel's data is protected by a username and password.
    - !The FTP protocol sends the username and password across the network “in the clear”.
- The SSL/TLS firmware allows:
    - The client to connect to the standard, unencrypted, port 21 and request an “upgrade” in security. The two devices then establish a high strength encrypted channel. Then all usernames, passwords, commands, and data are all protected.
    - The ip.buffer can set the FTP server to force an encrypted connection:
        - Implicit encryption – the conversation begins encrypted.
        - Explicit encryption – the FTP server will not allow username, password or data to be passed across the link until SSL/TLS encryption has been requested by the client.
        - !Since there is a single, common, FTP server, if collecting from a Cisco Call Manager using FTP Server collection, then encryption cannot be forced (since the CCM needs to make an unencrypted FTP link). The connecting FTP client must be relied on to instruct the ip.buffer's FTP server to use SSL/TLS.

6.4. FTP Push

The security considerations for FTP push are the same as for Email push, in section 6.1.

- For the SSL/TLS firmware, the ip.buffer can make a connection to an SSL/TLS-enabled FTP server. This can be either:
    - Implicit SSL/TLS
    - Explicit SSL/TLS
- When not using SSL/TLS, the data can be protected in the ways outline above.

6.5. TCP Server (passive)

The ip.buffer can be programmed to:

- The “Allow” field in the ip.buffer can restrict access to one device by its IP address.
- The ip.buffer can restrict access to: LAN only, Incoming PPP modem only, or both.
- The SSL/TLS firmware version can use an encrypted TCP/IP socket.
- The ip.buffer can be programmed to authenticate back to a RADIUS server.
- When RADIUS is not used, a simple password can be used. Obviously this is sent “in the clear” unless SSL/TLS encryption is used.

- The client cannot write back to the PBX.

6.6. TCP Push

Like the other "push" methods, the data can be protected as outline above. TCP Push also allows the option of using an SSL/TLS encrypted socket.

- The client cannot write back to the PBX.

6.7. Legacy Emulation

When emulating legacy devices in the ip.buffer the amount of additional security that can be provided is limited (since we have to emulate an older device that may not be as secure). However, the ip.buffer does have these facilities:

- The service can be restricted to one device by its IP address (by using the "Allow" field in the ip.buffer)
- The ip.buffer can restrict access to: LAN only, Incoming PPP modem only, or both.
- When using SSL/TLS firmware, the legacy emulation can use an encrypted socket[6].

The legacy emulation script itself dictates what the client can do. This may allow only reading data. However, some scripts provide bi-directional access back to a PBX – however, these modes often only work in tandem with other explicit settings made to the ip.buffer.

- Scannex recommend using the standard TCP/IP methods of connection instead of Legacy Emulation whenever possible! It is more secure!

[6] Note that legacy dial-in modem access is not possible when using SSL/TLS encryption.

7. Network Services

The UDP and TCP ports of most of the ip.buffer services can be configured. All services are enabled by default. However, in a secure environment it is preferable to disable any unused services.

The ip.buffer manual documents how to disable each service (usually by setting the TCP port to zero, or by blanking out the name/IP address field).

This is the list of non-configurable ports in the ip.buffer:

- **TCP** port **80**: used for the web-server ("http").
- **TCP** port **443**: used for secure web-server ("https").
- **UDP** port **161**: SNMP agent.
  - The SNMP agent provides only simple system level responses. This is to allow automatic asset management software to find the ip.buffer on the LAN.
  - It has no "SET" ability, i.e. no changes can ever be made through SNMP – the code just isn't there!
- **UDP** port **56024**: The Scannex "SEDiscover" protocol.
  - Packets from the PC are broadcast to 255.255.255.255
    - The protocol only works across the LAN and will not function across a router/firewall. It is impossible to "discover" ip.buffers on the other side of a firewall or router.
  - The current protocol in the ip.buffer will send back response packets from 0.0.0.0 to 255.255.255.255 (the same behaviour as the DHCP initialisation process).
  - Earlier versions of the ip.buffer sent back responses direct to the PC's IP address, appearing to originate from the same IP address. Many modern switches saw this as a "land attack" packet and dumped it (hence the change to a slightly revised protocol).