ADAM 4570 Series Networking Notes

By: Frantisek Rysanek <rysanek@fccps.cz>

Summary

Configuration traffic (the config utility)

1) the PC scanning for ADAM modules:

```
Request:   UDP  PC_IP:client   ->  IP_broadcast:5048
Response:  UDP  ADAM_IP:5048   ->  PC_IP:client
```

2) the PC communicating with a particular ADAM module (still in configuration mode):

```
Request:   UDP  PC_IP:client2 ->  IP_broadcast:5048
Response:  UDP  ADAM_IP:5048   ->  PC_IP:client2
Request:   UDP  PC_IP:client2 ->  IP_broadcast:5048
Response:  UDP  ADAM_IP:5048   ->  PC_IP:client2
```

...etc

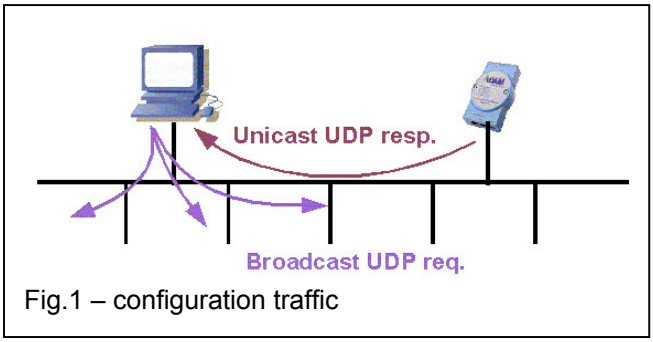

Fig.1 – configuration traffic

Payload traffic (the virtual COM port driver)

```
RS232 PORT1 = TCP  PC_IP:client -> ADAM_IP:5202
RS232 PORT2 = TCP  PC_IP:client -> ADAM_IP:5202
```

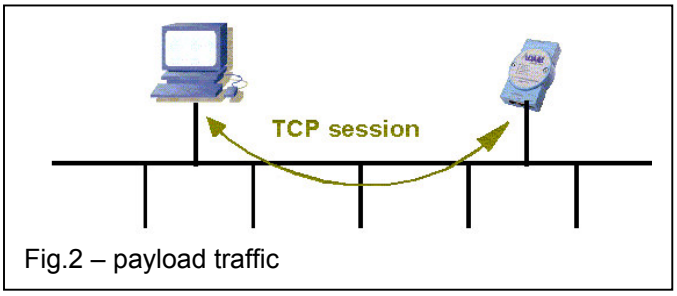

Fig.2 – payload traffic

Abbreviations used

PC_IP: the IP address of the PC workstation running Windows, where the virtual COM ports are established.
ADAM_IP: the IP address of the ADAM converter. Initially, this is a default or some previously set IP address – likely a nonsense in the context of a particular production network. Needs to be set/changed, using the configuration util, before production deployment, to make the payload tunneling work on a particular production network.
:client: a particular TCP/UDP port number from the „client“ range, i.e. greater than or equal to 1024, corresponding to a particular client socket open in the operating system. When a client opens a TCP connection or starts sending UDP packets, this is the source port on these outgoing TCP/UDP packets. The particular port number is chosen essentially at random - the lowest port number available is allocated.
IP_broadcast: the special IP address meaning „everyone“. Please note that the configuration utility is using the „global“ broadcast address of 255.255.255.255, rather than the local broadcast (derived from the local network address and netmask).

Windows registry keys involved

HKLM\Software\Advantech
HKLM\SYSTEM\CurrentControlSet\Services\AESPV1X
HKLM\Software\Microsoft\COM3
HKLM\HARDWARE\DEVICEMAP\SERIALCOMM
HKLM\SYSTEM\CurrentControlSet\Services\AESPV1X\Parameters
KLM\HARDWARE\DEVICEMAP\SERIALCOMM\AESPV1XP5 ="COM5"
HKLM\SYSTEM\CurrentControlSet\Services\AESPV1X\Parameters\AESPV1XP5
HKLM\HARDWARE\DEVICEMAP\SERIALCOMM\AESPV1XP6 ="COM6"
HKLM\SYSTEM\CurrentControlSet\Services\AESPV1X\Parameters\AESPV1XP6

Recommended debugging software

- **`tcpdump`** from http://www.tcpdump.org
  (available for Linux, Free/Open/NetBSD and other UNIXes)
- **`filemon`** and **`regmon`** from http://www.sysinternals.com
  (available for Windows 9x/ME and Windows NT/2k/XP)

Firewalling considerations

In general

The very basic setup as per the ADAM's manual relies on having the PC and the ADAM on a single Ethernet.
In reality, most customers will want to deploy the ADAM converters in a more complex intranet environment – they will need to deal with routers and firewalls. The payload traffic and especially the configuration traffic present IP networking issues that need to be addressed in order to make the ADAM converters work against the corresponding Windows-based software.

Example scenario

In a typical scenario, there's a protected private network, there's the wild public internet, and there's perhaps a separate plant automation network with the ADAM converters attached.

The Plant automation network will be more on the private/internal/protected side, but the network admins may still have it set up as a separate subnet (hence separated by a router/firewall) – e.g., to protect the office network from security breaches and to protect the automation network from being swamped by assorted mess originating from Windows.

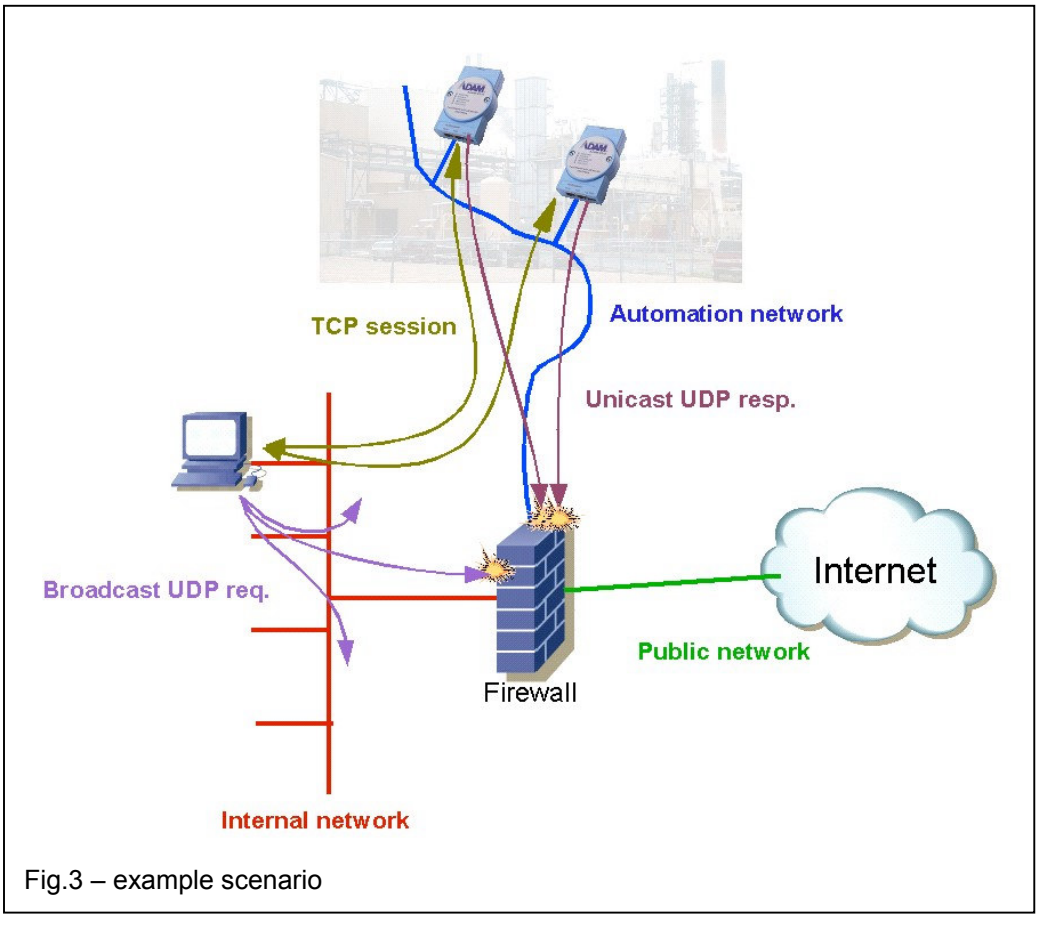

Fig.3 – example scenario

The payload traffic running over TCP is easy to firewall. The firewall admin will need to permit TCP sessions initiated from the internal network and destined for the automation network to TCP port 5202.

The configuration traffic (if required) may prove more troublesome. Firewalls typically don't let broadcast packets through by default, but a decent IP router or packet-based firewall should be able to do that. The unicast response packets will need a separate rule. In order to keep at least a basic level of anti-spoofing security, the user deploying the ADAM converters will need to pre-set their IP address in an "on-desk" setup so that once the ADAM gets connected to the automation network, the configuration-mode responses will already come from a sane IP address (i.e., one that belongs to the automation network's subnet). In other words, if remote configuration is required, the firewall admin will need to set up two more rules:

1) permit udp from internal network/mask port >1023 to IP_broadcast port 5048.
   And, only permit propagation of the broadcast from the internal network to the automation network. We don't need to propagate them to the public internet. Such a setup may require more than just one rule.
2) permit udp from automation network/mask port 5048 to internal network/mask port >1023

All of the above is only true for packet-level firewalls - these are inherently transparent to the traffic if correctly configured. Proxy-based firewalls are most likely non-transparent to general TCP/UDP traffic and hence will not work for the ADAM modules, as the ADAMs' PC software is not ready to work with them.

In packet-level firewalls, stateful inspection is probably only viable for the payload traffic (regular TCP). The configuration traffic uses a custom UDP-based communication scheme, that will go unnoticed by a general stateful inspection firewall (the response traffic won't get through). Hence the need for two state-less rules on part of just the configuration traffic.
Even with the payload traffic (TCP to port 5202) it may prove appropriate to disable stateful inspection if the user traffic is sparse (RS232 events occuring at potentially long time intervals) – to prevent communication errors due to the stateful sessions timing out at the firewall.

MS ISA Server – always something special

The MS ISA Server is firewall – a machine providing access from an internal office network to the public internet. It can be generally categorized as a Socks work-alike. It does not use the standard Socks 4/5 protocol, but its proprietary scheme does resemble it to some extent.

Specifically, the MS ISA Server uses client stubs installed on the client PC's. The client PC becomes a slave of the ISA Server – without the server's permission and cooperation, it won't communicate at all, not even within the local Ethernet (IP subnet). Nevertheless, the whole complex client/server firewall setup is almost completely transparent to software running on the client PC's – as part of the "client stub", the winsock.dll on the client PC's is "socksified", effectively doing all the magic behind the scenes, without the legacy client applications ever knowing.
And that's exactly how it's possible that the virtual COM port driver can work through the MS ISA Server, despite it being a proxy-based firewall.

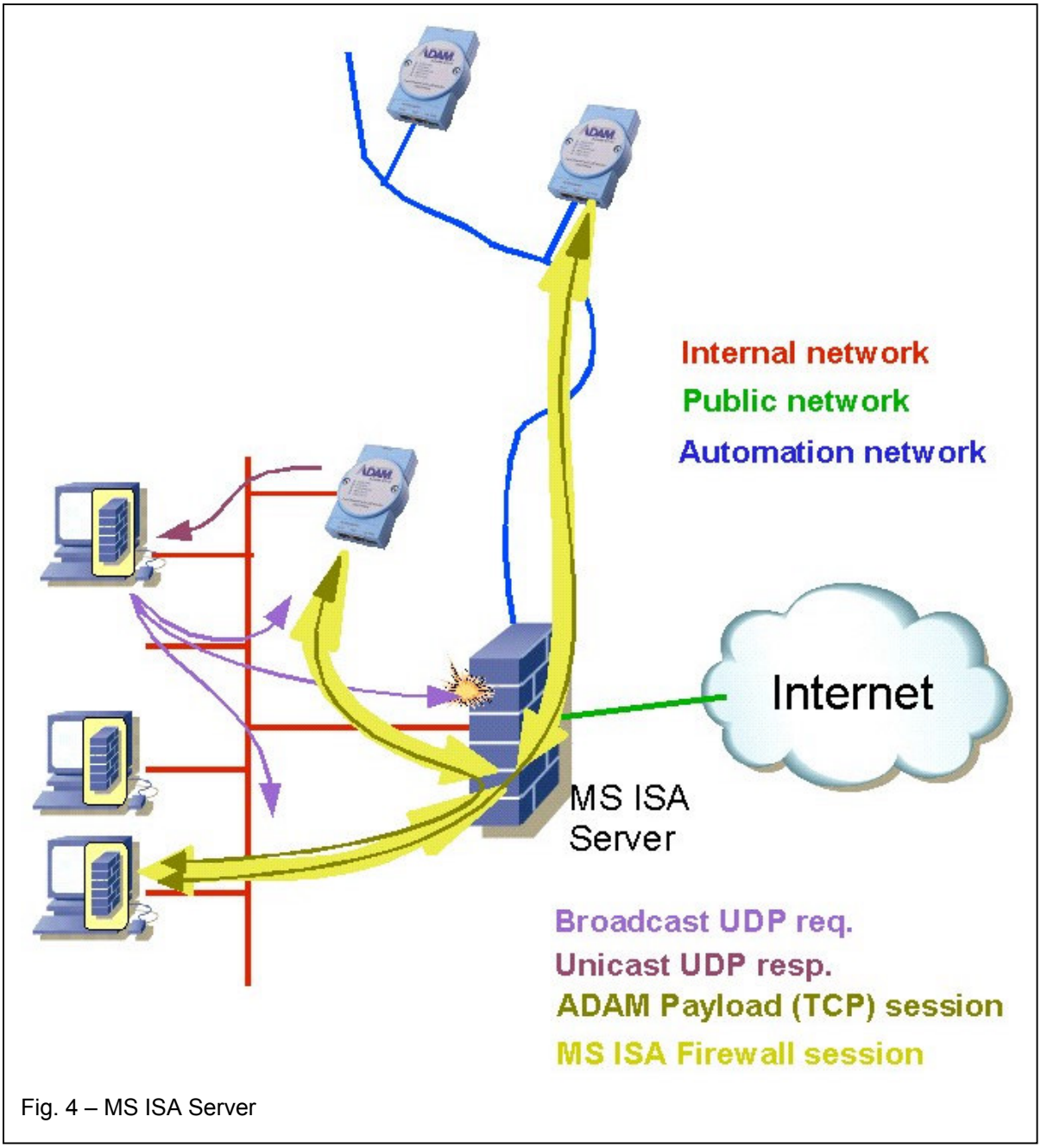

Fig. 4 – MS ISA Server

A typical "ADAM payload" TCP Session looks like this:

- the client software (the virtual COM port driver) opens a client TCP socket and tries to connect to a particular ADAM's IP address.
- the socksified winsock.dll first performs a kerberos (authentication) query to the local Windows PDC (a flurry of packets)
- the socksified winsock.dll performs some other query against the MS ISA Server (a few more packets)
- the socksified winsock.dll opens a TCP session to the MS ISA Server to a high TCP port (a TCP SYN/ACK sequence of packets)
- the MS ISA Server opens a TCP session to the target ADAM to TCP port 5202
- the virtual port at the client PC becomes functional

If you have a packet sniffer such as tcpdump attached to the local Ethernet (via a hub, rather than a switch), you'll see a great number of packets, most of them seemingly irrelevant to the desired "ADAM payload" session.

For comparison, without the MS ISA Server, the "ADAM payload" session looks like this:

- the client software (the virtual COM port driver) opens a client TCP socket and tries to connect to a particular ADAM's IP address.
- the winsock.dll opens a TCP session to the MS ISA Server to a high TCP port = a single TCP SYN/ACK sequence of packets
- the virtual port at the client PC becomes functional

I.e., the packet sniffer can see a total of about two or four packets, all of them clearly belonging to the "ADAM payload" TCP session.

If the MS ISA Server is not configured to permit the "ADAM payload" traffic, the session fails halfway through:

- the client software (the virtual COM port driver) opens a client TCP socket and tries to connect to a particular ADAM's IP address.
- the socksified winsock.dll first performs a kerberos (authentication) query to the local Windows PDC (a flurry of packets)
- the socksified winsock.dll performs some other query against the MS ISA Server (a few more packets)

No TCP session is ever opened (perhaps not even attempted) and the virtual port never becomes functional.

Again, there are differences between the payload traffic and the configuration traffic.
The payload traffic is a regular TCP session and, as such, it is hijacked by the MS ISA client stub and processed by the firewall's authentication/authorization rules.
The configuration traffic **is not** processed by the firewall at all – it appears straight on the local ethernet, without any authentication ever attempted. The "ADAM configuration" traffic cannot be permitted on the MS ISA Server.
Consequently, in the configuration util, ADAMs on the local ethernet are visible and configurable, but ADAMs on networks connected via the ISA Server are not.

In a practical case, a customer was trying to configure and test his ADAM on a local network. He was able to configure it but the virtual port would never become alive (the built-in test option would always fail). It turned out that there was an ISA server installed in the corporate network and the developers' workstations were "socksified". The MS ISA server was not configured for the "ADAM payload". Thus, the MS ISA server was refusing ADAM payload sessions even on the local Ethernet. Once the ISA server got configured, the ADAM started to work. The configuration traffic seems to bypass the socksified part of winsock.dll, hence configuration worked regardless of how the MS ISA Server was set up.

To configure the MS ISA Server, define a protocol called something like "ADAM payload", which is TCP to destination port 5202. Then you need to apply this rule on the interfaces involved (local/internal network only and/or the local network against the automation network).
You will still only be able to configure your ADAMs on the local ethernet where your workstation is connected, or using a cross-over Ethernet cable, or using a Windows PC attached straight to the target (automation) network.