

# **Data**tek

## **Solutions**

### **IP-COMMKIT® SECURITY ANALYSIS**

721 Route 202-206  
Bridgewater, NJ 08807  
fax: 908.218.1736  
phone: 908.218.0500  
email: [sales@datatekcorp.com](mailto:sales@datatekcorp.com)  
<http://www.datatekcorp.com>



## INTRODUCTION

IP-CommKit<sup>®</sup> uses an IP network to connect a host computer to a BNS-2000 or BNS-2000 VCS (a.k.a. Datakit<sup>®</sup> II VCS), and allows the host to communicate with Datakit endpoints. This document examines the effect of IP-CommKit on the vulnerability of a Datakit network to unauthorized access, i.e., whether an unauthorized host computer could gain access to a Datakit network through the IP network. It does not address general Datakit network or host security issues. We conclude that IP-CommKit does not make the Datakit network more vulnerable.

## BACKGROUND

### *Datakit II VCS*

The Datakit II Virtual Circuit Switch is the progenitor of modern ATM switches. It is built around a proprietary 180-bit cell format. Each cell is divided into 18 ten-bit envelopes. Two envelopes carry the virtual-channel identifier, and the remaining sixteen carry the payload. In addition to their smaller size, Datakit cell payloads are unlike standard ATM cell payload in that they support a 9-bit code space. The value of the 9<sup>th</sup> bit selects one 8-bit space for data and one 8-bit space for control and signaling. This makes it easy to interleave signaling and control information with user data on the same virtual channel.

---

<sup>®</sup> CommKit and IP-CommKit are registered trademarks of Lucent Technologies, Inc., licensed to Datatek Applications, Inc., a company independent of Lucent Technologies, Inc.

<sup>®</sup> Datakit is a registered trademark of Lucent Technologies, Inc., licensed to Datatek Applications, Inc., a company independent of Lucent Technologies, Inc.



Datakit uses a proprietary transport protocol above the cell layer. This protocol supports simple octet streams like asynchronous terminal traffic, and packet streams like X.25 traffic. It provides negotiable error detection, error correction and flow control. This protocol exploits the special 9-bit code space available in the Datakit proprietary cell format, and makes a portion of the control space available to higher-layer protocols.

Datakit II VCS uses a variety of proprietary signaling and data-transport protocols above the transport protocol. As with the transport layer, these protocols often take advantage of the 9-bit code space.

### ***CommKit Host Interface***

The CommKit Host Interface has both hardware and software components. The CommKit Host Interface hardware consists of a CPM-HS card in the Datakit node that is connected by an optical fiber to an interface card in the host computer. The CPM-HS card accepts Datakit cells from the switch that are destined for the host, and converts them to a serial format that is suitable for transmission on an optical fiber.

The host-interface card takes cells from the fiber and delivers them to the device-driver software. In the other direction, the host-interface card accepts Datakit cells from the device-driver software and transmits them through the optical fiber to the CPM-HS card in the Datakit node. The CPM-HS card takes the cells from the optical fiber and delivers them to the switch.

The device-driver software on the host does the transport and signaling-layer protocol processing. To establish a virtual-circuit connection with another Datakit endpoint, the

---



host communicates with the Datakit node controller, using a proprietary signaling protocol. This protocol rides above the proprietary transport protocol, which itself rides above the proprietary cell layer.

### ***IP-CommKit***

IP-CommKit replaces the CommKit Host Interface. It also has hardware and software components. IP-CommKit hardware consists of a UTM card in the Datakit node that is connected to an IP network via a 10Base-T interface. The host computer typically connects to the IP network through a LAN interface card.

The UTM card collects a group of Datakit cells from the switch that are destined for the host and converts them to an octet stream, using a proprietary algorithm. This algorithm efficiently maps the 9-bit code space of Datakit cell payloads into the 8-bit code space of an octet stream. Next, the UTM card encapsulates the octet stream in a proprietary protocol that guarantees in-sequence transmission. Each packet is transmitted to the host in a UDP packet via the IP network. The UDP protocol is used instead of the more common TCP protocol because the error correction provided by TCP is undesirable in this application. The unbounded delays that can occur with TCP retransmission are incompatible with the Datakit transport protocol, which provides its own error correction.

When the UDP packet arrives at the host, the LAN interface delivers the packet to the IP-CommKit device-driver software, which re-sequences the packets in case the IP network has delivered them out-of-sequence. Next, it converts the UDP packet payload back to a group of Datakit cells, using the inverse of the algorithm running on the UTM. After the UDP packets are converted back into Datakit cells, the device-driver software



does the same proprietary transport and signaling-layer processing that is done by the CommKit Host Interface software.

To establish a virtual-circuit connection with another Datakit endpoint, the host communicates with the Datakit node controller, which involves the following protocol layers:

- 1) Datakit signaling-layer protocol
- 2) Datakit transport-layer protocol
- 3) Datakit cell to octet stream mapping protocol
- 4) Sequence-guarantee protocol
- 5) UDP
- 6) IP
- 7) MAC

The host computer also maintains a proprietary *heartbeat* protocol with the UTM card. This protocol allows IP-CommKit to emulate the behavior of the optical fiber used with the CommKit Host Interface. This allows the host to notify the applications when communications with the BNS node are lost, as would happen when the optical fiber connection is broken.



## SECURITY ANALYSIS

The UTM and host driver software enforce basic security at the IP layer. The UTM card is configured with the IP address of the host computer, and the host computer is configured with the IP address of the UTM. The UTM only processes packets whose IP source address matches the configured host address. Similarly, the host driver software only processes packets whose source address matches the configured UTM address. At the UDP layer, port  $N$  is used for the heartbeat protocol and port  $N+1$  is used for the sequence-guarantee protocol, where  $N$  is a configured value. The UTM and host only process packets whose source and destination port numbers are  $N$  or  $N+1$ , as shown in the table below. Suppose that the host IP address is  $H$  and the UTM address is  $U$ . All datagrams that the host and UTM exchange will pass through the following filter:

IP Destination Address	IP Source Address	Protocol	UDP Destination Port	UDP Source Port
$H$	$U$	$UDP$	$N$	$N$
$H$	$U$	$UDP$	$N+1$	$N+1$
$U$	$H$	$UDP$	$N$	$N$
$U$	$H$	$UDP$	$N+1$	$N+1$

An unauthorized host computer could assume the IP address of an authorized host, and attempt to gain entry to the BNS network through the UTM card. For this to succeed, several barriers must be overcome. There are at least four proprietary protocol layers running above UDP:



- 1) Datakit signaling-layer protocol
- 2) Datakit transport-layer protocol
- 3) Datakit cell to octet stream mapping protocol
- 4) Sequence-guarantee protocol

Unlike UDP and IP, these protocols are not in the public domain. Since these protocols are part of the core intellectual property of Datatek Applications, Inc., their specifications are not made available to our customers. As a result, an unauthorized user would need to capture packets transmitted between the host and UTM, examine those packets and *reverse engineer* the protocols.

It's difficult to estimate the effort required to reverse engineer the required protocols, but it took approximately five staff-years to develop the protocol processing software in IP-CommKit using programmers experienced with BNS, complete documentation, and a well-equipped testing environment.

Supposing that the proprietary protocols could be reverse engineered, several design features of the protocols create additional barriers to unauthorized access. For example, each packet in the Sequence-Guarantee protocol carries a 16-bit sequence number. It would be impossible for the authorized and unauthorized hosts to simultaneously exchange packets with the UTM, because they would be competing for the same sequence-number space. Since the proprietary transport protocol also uses sequence numbering, a similar barrier exists at this level. Since the authorized host and Datakit node exchange heartbeat packets at regular intervals, the authorized host would have to be removed from service for an attempt at unauthorized access to be successful.



## CONCLUSIONS

IP-CommKit does not make a Datakit network more vulnerable to unauthorized access. While the host-to-UTM communications uses public domain protocols at the lower layers, the higher layers use proprietary protocols. The requirement that unauthorized users gain access to legitimate packets and reverse engineer the protocol stack presents a major barrier. Additional barriers are presented by the design of the protocols.

