NAME

APRECV − APSR network testing tool designed to receive arbitrary packets

SYNOPSIS

aprecv [*options*]

DESCRIPTION

APRECV is designed to *receive* and *analyze* arbitrary network packets or complete protocols. A complete list of currently supported protocols can be found in the *PROTOCOLS* section.

APRECV supports alot of command line options to control the *packet sniffing engine, BPF filters, the output format, statistics* and *various error checks.* Please have a look at the *OPTIONS* section.

Every non-printable characters in the payload field of the different protocols will be replaced by a ".".

OPTIONS

−−bt <device-number>

Receive HCI Frames from hci device <device-number>. These option can only be used together with --statistic.

−−calc-checksums

Recalculate the checksums of the sniffed packets (if any) and print the correct one if it's wrong. Currently only IPv4, ICMPv4, TCP/UDP over IPv4 are implemented.

−c | −−count

Count the size of the received packets.

−d | −−device <device>

Listen on device. If no interface is given, *APRECV* tries to autoselect one (excluding loopback devices). On some systems you can use "any" to listen on all interfaces at the same time.

−D | −−daemon

Run *APRECV* in daemon mode (fork in background and exit).

−−expand-<protocol>

Expand the protocol header and display all protocol header fields verbosely. Following options are available: *--expand-all, --expand-ethernet, --expand-arp, --expand-token, --expand-llc* and *--expand-pppoe.*

−f | −−filter filter rule

Set up a *BPF* filter rule.

−F | −−filter−nopt filter rule

Set up a non-optimized *BPF* filter rule.

−h | −−help

Display a help message and exit.

−I | −−iptrace-file <file>

Read an iptrace dump file, can be gzipped.

−l | −−logfile <file>

Write all received packets wich are normally printed to STDOUT to a file. If the file doesn't exist, *APRECV* will create a new one, otherwise it will append the information.

−−max-packets <num>

Maximum number of packets to receive, default is a endless loop.

−−module <file>

Load a shared module.

−−modules <directory>
Load all shared modules in this directory.

−−module-ignore
Ignore the builtin BPF filter of the module.

−−no-dissection
Don't dissect anything starting from link or network layer. Only print when frames received. This option should be used together with --logfile-raw or --logfile for better performance.

−−print-<protocol>-hex
Print the payload of a packet in hex. The following options are available: *--print-ip-hex*, *--print-pppoe-hex*, *--print-tcp-hex* and *--print-udp-hex*.

−−print-<protocol>-text
Print the payload of a packet in text. The following options are available: *--print-tcp-text* and *--print-udp-text*.

−−print-tcp-stream
Print captured TCP streams out.

−p | −−promisc <0|1>
Enable or disable promiscuous mode for the device (0=off, 1=on). The promiscuous mode is useful in ethernet networks to receive packets which are not sent to the *MAC address* of your network interface.

−P | −−pcap-file <file>
Read a raw pcap dump from a file.

−−quiet
Close the stdout filehandle, useful with raw logfile mode, but print out errors.

−r | −−logfile-raw <file>
Log all packets in raw format to a file.

−−really-quiet
Close the stdout and stderr filehandle, useful with raw logfile mode.

−R | −−print-raw-hex
Print raw packets in hex.

−−server-addr <ip>
IP Address of the APSR-Server, to connect to.

−−server-port <port>
Port number of the APSR-Server.

−s | −−snaplen <up to 65535>
Set the snaplen for pcap, default is 65535.

−−statistic
Print a statistic of all counted packets and protocols before terminating. Format: NORMAL_COUNT (TRUNCATED_PACKETS|PROTOCOL_ERRORS|CHECKSUM_ERRORS).

−S | −−snoop-file <file>
Read an snoop dump file, can be gzipped.

−−use-apsrlib
Use the APSRLib instead of Libpcap. Some options may be disabled or not useable.

−v | −−verbose
Print more information about the protocols, the interface, the localnet and the netmask address and activate resolving of IP and MAC addresses.

−V | −−version
Display the *APRECV* version and the compiled libpcap version.

PROTOCOLS

Currently the following protocols and operating system specifi c headers are supported: Ethernet II, LLC/SNAP(802.3), 802.2, VLAN(802.1p/1q), Linux SLL header, Linux ATM CLIP, ATM rfc1483 PDU's, SunATM header(Solaris), LUNI 2.0, 802.11b, Prism monitor mode frames, TokenRing(802.5), FDDI, Cisco HDLC, CDP, CGMP, VTP, GSMP, BOFL, PPPoE with Tags, PPP, ARP/RevARP/InvARP, MPLS, RSVP, IPv4/IPv6/IPv7, IPv6 Routing Header, IPv6 Hop-by-Hop Header, IPv4 over IPv4, EtherIP, IPX, ICMPv4/ICMPv6, IGMPv0/IGMPv1/IGMPv2/RGMP and a little bit of IGMPv3, TCP, UDP, RDPv1/RDPv2, SPX/SPX2, IPComp, IPAuth, MTP, IFMP, ESP, SCTP, EGP, BGP, GGP, CBT, IRTP, GREv0/GREv1, OSPFv2/OSPFv3, NARP, IGRP, EIGRP, VRRP, PIM, RIPv1/RIPv2, IPXRIP, RIPng, MSDP, TMux, NETBLK, SDRP and PGM.

PROTOCOLS PPP-Suite

Control Protocols: PPP with LCP, IPCP, IPV6CP, IPXCP, ATCP, ECP, CCP, BCP, DNCP(Decnet Phase IV), OSINLCP and SNACP(SNACP and SNA over LLC CP).

Additional PPP Protocols: PAP, CHAP, EAP and LQR.

SEE ALSO

apsrserver (1), apsend (1), apsrlib (3), http://www.tcpdump.org or pcap (3) for *BPF* fi lter logic.

AUTHOR

APSR development team (http://www.aa-security.de).

REPORTING BUGS

Report bugs to <bugs@aa-security.de>.