0pt

Snoop IDS Reference Manual

1.0

Generated by Doxygen 1.2.18

Thu Aug 19 23:57:16 2004

Contents

Chapter 1

Snoop IDS Data Structure Index

1.1 Snoop IDS Data Structures

Here are the data structures with brief descriptions:

Chapter 2

Snoop IDS File Index

2.1 Snoop IDS File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Snoop IDS Data Structure Documentation

3.1 hashtable Struct Reference

Structure for a HT.

`#include <hashtable.h>`

Data Fields

- ht_bucket * table
  Table of pointers to buckets.
- unsigned int size
  Maximum size.
- unsigned int count
  how many entries

3.1.1 Detailed Description

Structure for a HT.

The documentation for this struct was generated from the following file:

- hashtable.h

3.2 ht_bucket Struct Reference

Structure for a HT bucket.

`#include <hashtable.h>`

Data Fields

- u_char * packet

  Pointer to raw packet.

- ht_state state

  Whether the entry is valid, invalid, deleted or late.

- time_t timestamp

  timestamp when received

3.2.1 Detailed Description

Structure for a HT bucket.

The documentation for this struct was generated from the following file:

- hashtable.h

3.3 intruder Struct Reference

The intruder struct holds the mac, ipv6 addresses, the time when it was classified as intrusive and the dropcount for that period.

`#include <snoop.h>`

Data Fields

- u_int8_t ether_addr [6]
  ethernet address
- u_int8_t ip6_addr [16]
  ip6 address
- time_t when_detected
  end of sampling period is recorded
- u_int32_t dropcount
  right now there is just one type, need (type,count) for each type of attack

3.3.1 Detailed Description

The intruder struct holds the mac, ipv6 addresses, the time when it was classified as intrusive and the dropcount for that period.

The documentation for this struct was generated from the following file:

- snoop.h

3.4 neighbor Struct Reference

The neighbor struct stores that mac address and corresponding IPv6 address of its neighbors.

`#include <snoop.h>`

Data Fields

- u_int8_t src_ether [6]
  source ethernet address
- u_int8_t src_ip6 [16]
  corresponding ip6 address
- r_state route_state
  live, bad or expired
- u_int32_t dropcount
  count of dropped packets since last timeout
- clock_t expiry
  when this route will expire

3.4.1 Detailed Description

The neighbor struct stores that mac address and corresponding IPv6 address of its neighbors.

The documentation for this struct was generated from the following file:

- snoop.h

Chapter 4

Snoop IDS File Documentation

4.1 common.h File Reference

Common header files required by all other modules.

```
#include <pcap.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netinet/if_ether.h>
#include <net/ethernet.h>
#include <netinet/ether.h>
#include <netinet/ip.h>
#include <netinet/ip6.h>
#include <netinet/udp.h>
#include <netinet/tcp.h>
#include <time.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <signal.h>
#include <unistd.h>
#include "aodv.h"
```

Defines

- #define AODV6_RREQ 1

  The AODV6 message type values, from RFC 3561.

- #define ETHER_HDRLEN 14

  Length of the ethernet header in bytes.

- #define ETHERTYPE_IPV6 0x86dd

  The protocol field value for IPv6.

- #define IP_PROTO_TCP 6

  Value for protocol field for TCP in IPv6 datagrams is 6.

- #define IP_PROTO_UDP 17

  Value for protocol field for UDP in IPv6 datagrams is 17.

- #define IP_PROTO_ICMPV6 58

  Value for protocol field for ICMPv6 in IPv6 datagrams is 58.

4.1.1 Detailed Description

Common header files required by all other modules.

```
Author: Anand Patwardhan
email:  anand.patwardhan@umbc.edu
Date : 30 April 2004


The SNOOP program is an intrusion detection mechanism to detect local
intrusions in a Mobile Ad Hoc Network.

 Copyright (C) 2004  Anand Patwardhan
 E-mail:  anand.patwardhan@umbc.edu

 eBiquity Research Group
 University of Maryland, Baltimore County
 1000 Hilltop Circle, Baltimore, MD 21250, USA.

  http://research.ebiquity.org

 This program is free software; you can redistribute it and/or
 modify it under the terms of the GNU General Public License
 as published by the Free Software Foundation; either version 2
 of the License, or (at your option) any later version.

 This program is distributed in the hope that it will be useful,
 but WITHOUT ANY WARRANTY; without even the implied warranty of
 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 GNU General Public License for more details.

 You should have received a copy of the GNU General Public License
 along with this program; if not, write to the Free Software
```

```
  Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.
```

4.1.2 Define Documentation

4.1.2.1 #define AODV6_RREQ 1

The AODV6 message type values, from RFC 3561.

One of the RFC has values 16, 17, 18, 19 and one has 1, 2, 3, and 4

4.1.2.2 #define ETHER_HDRLEN 14

Length of the ethernet header in bytes.

This is usually already defined in ether.h

4.1.2.3 #define ETHERTYPE_IPV6 0x86dd

The protocol field value for IPv6.

The protocol field value for IPv6 over ethernet is 0x86dd

4.2 hashtable.c File Reference

Implements a hashtable with quadratic probing.

```
#include "common.h"
#include "hashtable.h"
```

Functions

- hashtable * createHashtable (unsigned int size)
  Create a hashtable of specified size.
- int makeEntry (hashtable *ht, u_char *raw, clock_t time)
  Add a new raw packet to the hashtable (inserts new entry into HT).
- int performID (hashtable *ht, u_char *raw, clock_t time)
  Performs comparison to see if a forwarded packet matches a received packet.
- void print (hashtable *ht)
  Prints current contents of hashtable.
- void dump_packet (u_char *packet)
  Prints the contents of a raw packet in human readable form.
- void hex_dump_packet (u_char *packet)
  Prints the hex contents of a raw packet.

4.2.1 Detailed Description

Implements a hashtable with quadratic probing.

```
Author: Anand Patwardhan
email:  anand.patwardhan@umbc.edu
Date : 30 April 2004

Implements a hashtable with quadratic probing, TCP sequence nos.
are used as keys for hashing packets, the hashtable itself does
not contain the raw packets, but pointer to raw packets.
TCP sequence nos. were chosen to be the keys since we intend to
perform intrusion detection on forwarded packets amongst other
things. If the HT is full, no more packets can be watched, this
however can be controlled by using an appropriate timeout period
for the timer which will flush packets deeming them to be dropped
after the timeout.

The functions makeEntry and PerformID are the HT equivalent of
insert and remove.

The size for the hashtable should be a sufficiently large prime
no. to minimize collisions and reducing the chances of overflow
```

```
Descriptions of each of the functions can be found in hashtable.h.


The SNOOP program is an intrusion detection mechanism to detect local
intrusions in a Mobile Ad Hoc Network.

 Copyright (C) 2004  Anand Patwardhan
 E-mail:  anand.patwardhan@umbc.edu

 eBiquity Research Group
 University of Maryland, Baltimore County
 1000 Hilltop Circle, Baltimore, MD 21250, USA.

  http://research.ebiquity.org

 This program is free software; you can redistribute it and/or
 modify it under the terms of the GNU General Public License
 as published by the Free Software Foundation; either version 2
 of the License, or (at your option) any later version.

 This program is distributed in the hope that it will be useful,
 but WITHOUT ANY WARRANTY; without even the implied warranty of
 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 GNU General Public License for more details.

 You should have received a copy of the GNU General Public License
 along with this program; if not, write to the Free Software
 Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.
```

4.2.2 Function Documentation

4.2.2.1 hashtable* createHashtable (unsigned int *size*)

Create a hashtable of specified size.

Parameters:

size Size of the hashtable

Returns:

address of allocated HT

4.2.2.2 void dump_packet (u_char * *packet*)

Prints the contents of a raw packet in human readable form.

Parameters:

packet The raw packet to be printed

4.2.2.3 void hex_dump_packet (u_char * *packet*)

Prints the hex contents of a raw packet.

Parameters:

packet The raw packet to be printed in hexadecimal format

4.2.2.4 int makeEntry (hashtable * *ht*, u_char * *raw*, clock_t *time*)

Add a new raw packet to the hashtable (inserts new entry into HT).

Uses quadratic probing to make the new entry, uses TCP sequence number. Involves parsing the TCP header to get the sequence no. Thus currently only TCP packets can be watched, though any valid sized packet could still be entered, but not recommended. In the case of TCP packets, seq. nos. are ideal candidates for keys in the HT, especially when searching the HT to match an identical packet.

Parameters:

ht Hashtable to make entry in

raw Raw packet

time Timestamp when packet was received

4.2.2.5 int performID (hashtable * *ht*, u_char * *raw*, clock_t *time*)

Performs comparison to see if a forwarded packet matches a received packet.

In the comparison, the hop limit field is ignored. The hop limit should be expected to decrease by one, if not, should be considered as a malicious modification, though not currently done in this code.

Parameters:

ht Specifies the hashtable

raw The raw packet contents

time The time when this packet was received

4.2.2.6 void print (hashtable * *ht*)

Prints current contents of hashtable.

For debugging purposes, prints current contents of HT

Parameters:

ht Hashtable to print

4.3 hashtable.h File Reference

Implements a hashtable with quadratic probing.

`#include "common.h"`

Data Structures

- struct hashtable

  Structure for a HT.

- struct ht_bucket

  Structure for a HT bucket.

Enumerations

- enum ht_state

  Enumeration of states of a HT entry.

Functions

- hashtable * createHashtable (unsigned int size)

  Create a hashtable of specified size.

- int makeEntry (hashtable *ht, u_char *raw, clock_t time)

  Add a new raw packet to the hashtable (inserts new entry into HT).

- int performID (hashtable *ht, u_char *raw, clock_t time)

  Performs comparison to see if a forwarded packet matches a received packet.

- void print (hashtable *ht)

  Prints current contents of hashtable.

- void dump_packet (u_char *packet)

  Prints the contents of a raw packet in human readable form.

- void hex_dump_packet (u_char *packet)

  Prints the hex contents of a raw packet.

4.3.1 Detailed Description

Implements a hashtable with quadratic probing.

```
Author: Anand Patwardhan
email:  anand.patwardhan@umbc.edu
Date : 30 April 2004
```

```
Implements a hashtable with quadratic probing, TCP sequence nos.
are used as keys for hashing packets, the hashtable itself does
not contain the raw packets, but pointer to raw packets.
TCP sequence nos. were chosen to be the keys since we intend to
perform intrusion detection on forwarded packets amongst other
things. If the HT is full, no more packets can be watched, this
however can be controlled by using an appropriate timeout period
for the timer which will flush packets deeming them to be dropped
after the timeout.

The functions makeEntry and PerformID are the HT equivalent of
insert and remove.

The size for the hashtable should be a sufficiently large prime
no. to minimize collisions and reducing the chances of overflow


The SNOOP program is an intrusion detection mechanism to detect local
intrusions in a Mobile Ad Hoc Network.

 Copyright (C) 2004  Anand Patwardhan
 E-mail:  anand.patwardhan@umbc.edu

 eBiquity Research Group
 University of Maryland, Baltimore County
 1000 Hilltop Circle, Baltimore, MD 21250, USA.

  http://research.ebiquity.org

 This program is free software; you can redistribute it and/or
 modify it under the terms of the GNU General Public License
 as published by the Free Software Foundation; either version 2
 of the License, or (at your option) any later version.

 This program is distributed in the hope that it will be useful,
 but WITHOUT ANY WARRANTY; without even the implied warranty of
 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 GNU General Public License for more details.

 You should have received a copy of the GNU General Public License
 along with this program; if not, write to the Free Software
 Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.
```

4.3.2 Function Documentation

4.3.2.1 hashtable* createHashtable (unsigned int *size*)

Create a hashtable of specified size.

Parameters:

size Size of the hashtable

Returns:

address of allocated HT

4.3.2.2 void dump_packet (u_char * *packet*)

Prints the contents of a raw packet in human readable form.

Parameters:

packet The raw packet to be printed

4.3.2.3 void hex_dump_packet (u_char * *packet*)

Prints the hex contents of a raw packet.

Parameters:

packet The raw packet to be printed in hexadecimal format

4.3.2.4 int makeEntry (hashtable * *ht*, u_char * *raw*, clock_t *time*)

Add a new raw packet to the hashtable (inserts new entry into HT).

Uses quadratic probing to make the new entry, uses TCP sequence number. Involves parsing the TCP header to get the sequence no. Thus currently only TCP packets can be watched, though any valid sized packet could still be entered, but not recommended. In the case of TCP packets, seq. nos. are ideal candidates for keys in the HT, especially when searching the HT to match an identical packet.

Parameters:

ht Hashtable to make entry in

raw Raw packet

time Timestamp when packet was received

4.3.2.5 int performID (hashtable * *ht*, u_char * *raw*, clock_t *time*)

Performs comparison to see if a forwarded packet matches a received packet.

In the comparison, the hop limit field is ignored. The hop limit should be expected to decrease by one, if not, should be considered as a malicious modification, though not currently done in this code.

Parameters:

ht Specifies the hashtable

raw The raw packet contents

time The time when this packet was received

4.3.2.6 void print (hashtable * *ht*)

Prints current contents of hashtable.

For debugging purposes, prints current contents of HT

Parameters:

ht Hashtable to print

4.4 snoop.c File Reference

Uses the pcap library, to listen on any available network interface in promiscuous mode. Raw packets are captured and further processed for filtering, IPv6 packets are of interest. TCP streams over IPv6 are monitored. AODV6 packets are in UDP datagrams, again in IPv6 packets. AODVD hello messages are used to populate the neighbor table. The IDS itself is independent of the routing protocol is in use. In this instance we used SecAODV, so neighbors are discovered using AODV hello messages.

```
#include "common.h"
#include "hashtable.h"
#include "snoop.h"
```

Functions

- void SIGALRM_handler (int sig)

  Alarm handler, updates drop counts.

- void timer_update_state (void)

  Examines the hashtable and clears up packets, updates drop counts.

- void pkt_callback (u_char *args, const struct pcap_pkthdr *pkthdr, const u_char *packet)

  The callback function for the packet capture.

- u_int16_t handle_ethernet (u_char *args, const struct pcap_pkthdr *pkthdr, const u_char *packet)

  Returns the type of packet contained within the ethernet frame.

- int handle_IPv6 (u_char *args, const struct pcap_pkthdr *pkthdr, const u_char *packet)

  Examines IPv6 packets for AODV6 and TCP payloads.

- int handle_AODV (const u_char *packet)

  Examines and handles AODV packets.

- int find_neighbor (neighbor *pair)

  Helper function that returns the index of the provided neighbor in the neighbor table.

- int add_neighbor (neighbor *pair)

  Helper function that adds a neighbor to the neighbor table.

- int incr_dropcount (struct ether_addr *ether_src)

  Increments the dropcount of the entry in the neigbor table corresponding to the provided mac address.

- void print_neighbors (void)

  Helper function that prints contents of neighbor table.

- void log_intrusions (void)

  Reads the neighbor table, logs non-zero dropcount entries as potential intrusions.

4.4.1 Detailed Description

Uses the pcap library, to listen on any available network interface in promiscuous mode. Raw packets are captured and further processed for filtering, IPv6 packets are of interest. TCP streams over IPv6 are monitored. AODV6 packets are in UDP datagrams, again in IPv6 packets. AODVD hello messages are used to populate the neighbor table. The IDS itself is independent of the routing protocol is in use. In this instance we used SecAODV, so neighbors are discovered using AODV hello messages.

See snoop.h documentation for function descriptions.

```
Author: Anand Patwardhan
email:  anand.patwardhan@umbc.edu
Date : 30 April 2004


The SNOOP program is an intrusion detection mechanism to detect local
intrusions in a Mobile Ad Hoc Network.

 Copyright (C) 2004  Anand Patwardhan
 E-mail:  anand.patwardhan@umbc.edu

 eBiquity Research Group
 University of Maryland, Baltimore County
 1000 Hilltop Circle, Baltimore, MD 21250, USA.

  http://research.ebiquity.org

 This program is free software; you can redistribute it and/or
 modify it under the terms of the GNU General Public License
 as published by the Free Software Foundation; either version 2
 of the License, or (at your option) any later version.

 This program is distributed in the hope that it will be useful,
 but WITHOUT ANY WARRANTY; without even the implied warranty of
 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 GNU General Public License for more details.

 You should have received a copy of the GNU General Public License
 along with this program; if not, write to the Free Software
 Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.


We referred to Tim Carstens' example code and Martin Casado's
tutorial. We also looked at the ipgrab source code and tcpdump source.
Source code seems to be the only documentation on pcap.
```

4.4.2 Function Documentation

4.4.2.1 int find_neighbor (neighbor *)

Helper function that returns the index of the provided neighbor in the neighbor table.

Provided a neighbor instance, looks up and returns the index of the neighbor in the neighbor table.

4.4.2.2 int handle_AODV (const u_char *)

Examines and handles AODV packets.

Looks for RREP messages to build the neigbor tables. We can potentially remove entries from the neigbor table if hello messages are missed.

4.4.2.3 u_int16_t handle_ethernet (u_char *, const struct pcap_pkthdr *, const u_char *)

Returns the type of packet contained within the ethernet frame.

We are interested only in IPv6, but can be easily modified to handle other protocols

4.4.2.4 int handle_IPv6 (u_char *, const struct pcap_pkthdr *, const u_char *)

Examines IPv6 packets for AODV6 and TCP payloads.

Once IPv6 packets are filtered out, further filtering of AODV6 and TCP protocols is done here. For AODV6 we watch for ”Hello” messages i.e. special RREP messages, to identify neigbhors and populate the neigbor table.

4.4.2.5 void log_intrusions (void)

Reads the neighbor table, logs non-zero dropcount entries as potential intrusions.

This function reads the neighbor table, logs non-zero entries to the suspects file and resets dropcounts for the next time period.

4.4.2.6 void pkt_callback (u_char *, const struct pcap_pkthdr *, const u_char *)

The callback function for the packet capture.

The callback function opens the device in promiscuous mode and listens for packets. The raw packets captured (1500 bytes) are then passed on for further filtering. We deal with only IPv6 packets, others are ignored.

4.4.2.7 void SIGALRM_handler (int)

Alarm handler, updates drop counts.

Calls the timer_update_state function

4.4.2.8 void timer_update_state (void)

Examines the hashtable and clears up packets, updates drop counts.

Will remove entries from the hashtable that are old, gives newer ones a second chance, if entries are not cleared by a second timeout, the packet is assumed to have been dropped.

4.5 snoop.h File Reference

Uses the pcap library, to listen on any available network interface in promiscuous mode. Raw packets are captured and further processed for filtering, IPv6 packets are of interest. TCP streams over IPv6 are monitored. AODV6 packets are in UDP datagrams, again in IPv6 packets. AODVD hello messages are used to populate the neighbor table.

```
#include "common.h"
```

Data Structures

- struct intruder

  The intruder struct holds the mac, ipv6 addresses, the time when it was classified as intrusive and the dropcount for that period.

- struct neighbor

  The neighbor struct stores that mac address and corresponding IPv6 address of its neighbors.

Defines

- #define MAXNEIGHBORS 10

  MAXNEIGHBORS is used to limit the number of neigbors to watch for intrusions.

- #define ALARM_TIMEOUT 3

  ALARM_TIMEOUT is the value in seconds for the timeout to examine the hashtable.

Enumerations

- enum r_state

  Three states of a neighbor within the neighbor table.

Functions

- void SIGALRM_handler (int)

  Alarm handler, updates drop counts.

- void timer_update_state (void)

  Examines the hashtable and clears up packets, updates drop counts.

- void pkt_callback (u_char *, const struct pcap_pkthdr *, const u_char *)

  The callback function for the packet capture.

- u_int16_t handle_ethernet (u_char *, const struct pcap_pkthdr *, const u_char *)

  Returns the type of packet contained within the ethernet frame.

- int handle_IPv6 (u_char *, const struct pcap_pkthdr *, const u_char *)

Examines IPv6 packets for AODV6 and TCP payloads.

- int handle_AODV (const u_char *)

  Examines and handles AODV packets.

- int find_neighbor (neighbor *)

  Helper function that returns the index of the provided neighbor in the neighbor table.

- int add_neighbor (neighbor *)

  Helper function that adds a neighbor to the neighbor table.

- void print_neighbors (void)

  Helper function that prints contents of neighbor table.

- int incr_dropcount (struct ether_addr *)

  Increments the dropcount of the entry in the neigbor table corresponding to the provided mac address.

- void log_intrusions (void)

  Reads the neighbor table, logs non-zero dropcount entries as potential intrusions.

4.5.1 Detailed Description

Uses the pcap library, to listen on any available network interface in promiscuous mode. Raw packets are captured and further processed for filtering, IPv6 packets are of interest. TCP streams over IPv6 are monitored. AODV6 packets are in UDP datagrams, again in IPv6 packets. AODVD hello messages are used to populate the neighbor table.

```
Author: Anand Patwardhan
email:  anand.patwardhan@umbc.edu
Date : 30 April 2004


The SNOOP program is an intrusion detection mechanism to detect local
intrusions in a Mobile Ad Hoc Network.

 Copyright (C) 2004  Anand Patwardhan
 E-mail:  anand.patwardhan@umbc.edu

 eBiquity Research Group
 University of Maryland, Baltimore County
 1000 Hilltop Circle, Baltimore, MD 21250, USA.

  http://research.ebiquity.org

 This program is free software; you can redistribute it and/or
 modify it under the terms of the GNU General Public License
 as published by the Free Software Foundation; either version 2
 of the License, or (at your option) any later version.

 This program is distributed in the hope that it will be useful,
 but WITHOUT ANY WARRANTY; without even the implied warranty of
 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
```

```
  GNU General Public License for more details.

  You should have received a copy of the GNU General Public License
  along with this program; if not, write to the Free Software
  Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.


 We referred to Tim Carstens' example code and Martin Casado's
 tutorial. We also looked at the ipgrab source code and tcpdump source.
 Source code seems to be the only documentation on pcap.
```

4.5.2 Define Documentation

4.5.2.1 #define ALARM_TIMEOUT 3

ALARM_TIMEOUT is the value in seconds for the timeout to examine the hashtable.

The hashtable is examined every ALARM_TIMEOUT seconds; a second chance algorithm is used to classify packet drops as intrusions. Basically if a packet is not retransmitted by the time of a second timeout, it is considered to be dropped and removed from the hashtable.

4.5.2.2 #define MAXNEIGHBORS 10

MAXNEIGHBORS is used to limit the number of neigbors to watch for intrusions.

To be scalable, it is necessary to watch only a bounded number of neighbors depending on the memory and cpu capacity of the device, or else it can be quickly overwhelmed and effectiveness of the IDS will decrease.

4.5.3 Function Documentation

4.5.3.1 int find_neighbor (neighbor *)

Helper function that returns the index of the provided neighbor in the neighbor table.

Provided a neighbor instance, looks up and returns the index of the neighbor in the neighbor table.

4.5.3.2 int handle_AODV (const u_char *)

Examines and handles AODV packets.

Looks for RREP messages to build the neigbor tables. We can potentially remove entries from the neigbor table if hello messages are missed.

4.5.3.3 u_int16_t handle_ethernet (u_char *, const struct pcap_pkthdr *, const u_char *)

Returns the type of packet contained within the ethernet frame.

We are interested only in IPv6, but can be easily modified to handle other protocols

4.5.3.4 int handle_IPv6 (u_char *, const struct pcap_pkthdr *, const u_char *)

Examines IPv6 packets for AODV6 and TCP payloads.

Once IPv6 packets are filtered out, further filtering of AODV6 and TCP protocols is done here. For AODV6 we watch for "Hello" messages i.e. special RREP messages, to identify neigbhors and populate the neigbor table.

4.5.3.5 void log_intrusions (void)

Reads the neighbor table, logs non-zero dropcount entries as potential intrusions.

This function reads the neighbor table, logs non-zero entries to the suspects file and resets dropcounts for the next time period.

4.5.3.6 void pkt_callback (u_char *, const struct pcap_pkthdr *, const u_char *)

The callback function for the packet capture.

The callback function opens the device in promiscuous mode and listens for packets. The raw packets captured (1500 bytes) are then passed on for further filtering. We deal with only IPv6 packets, others are ignored.

4.5.3.7 void SIGALRM_handler (int)

Alarm handler, updates drop counts.

Calls the timer_update_state function

4.5.3.8 void timer_update_state (void)

Examines the hashtable and clears up packets, updates drop counts.

Will remove entries from the hashtable that are old, gives newer ones a second chance, if entries are not cleared by a second timeout, the packet is assumed to have been dropped.