CITRIC

Generated by Doxygen 1.7.6.1

Thu Oct 24 2013 09:38:07

Contents

Chapter 1

CITRIC introduction

CITRIC is an experimental causal detector that analyzes PCAP-data from a device.It aggregates packets in bidirectional flows and it organizes the flows in causal trees.

A causal tree is a group of flows with a causal relationship. Causal relationship is determined by time-intervals and in some cases content.

To start CITRIC: *sudo CITRIC x.x.x.x y*

with *x.x.x.x* is the address of the observed computer

and *y* is the name of the observed interface.

If y ends with a number it is assumed to be a device e.g. eth0.

If y ends with something else it assumed to be a file in PCAP-format.

In both cases CITRIC is terminated with a Control-C sequence.

This is an experimental version for research.

Author

Pieter Burghouwt

Version

Revision 2.0

Date

Tuesday, May 7, 2013

Chapter 2

Class Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

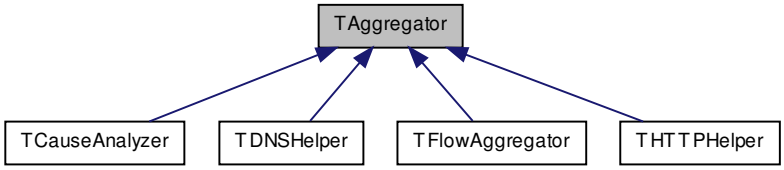
Class Documentation

5.1 TAggregator Class Reference

Baseclass for aggregators. An aggragator is an object dat aggregates multiple data in a combined datastructure or element of a small size. Examples of aggregating elements are Flows, Tree, and DNS-records.

`#include <Aggregator.h>`

Inheritance diagram for TAggregator:



Public Member Functions

- virtual uint8_t add (void)
- virtual void dump (void)

Public Attributes

- int32_t Index

  Index of the current element.

Protected Attributes

- int32_t Size

  Total number of used elements.
- int32_t WriteIndex

  Index to the first element to write in an array.

5.1.1 Detailed Description

Baseclass for aggregators. An aggragator is an object dat aggregates multiple data in a combined datastructure or element of a small size. Examples of aggregating elements are Flows, Tree, and DNS-records.

<

5.1.2 Member Function Documentation

5.1.2.1 virtual uint8_t TAggregator::add (void) [inline, virtual]

Reimplemented in TFlowAggregator, TDNSHelper, and THTTPHelper.

5.1.2.2 virtual void TAggregator::dump (void) [inline, virtual]

adds data to the Aggregator. @ return the result of the addition:

- 1 = succesful addition in existing element(s)
- 2 = new element(s) created
- other = error in processing, data not added

5.1.3 Member Data Documentation

5.1.3.1 int32_t TAggregator::Index

Index of the current element.

5.1.3.2 int32_t TAggregator::Size [protected]

Total number of used elements.

5.1.3.3 int32_t TAggregator::WriteIndex [protected]

Index to the first element to write in an array.

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

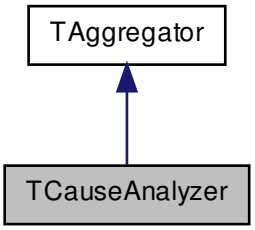
- src/Aggregator.h

5.2 TCauseAnalyzer Class Reference

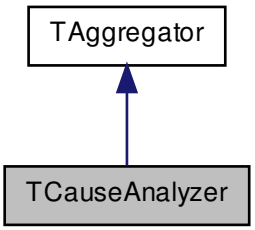
Class that manages causal trees and sorts new flows in it.

`#include <CauseAnalyzer.h>`

Inheritance diagram for TCauseAnalyzer:



Collaboration diagram for TCauseAnalyzer:



Public Member Functions

- TCauseAnalyzer (void)
- uint8_t add (char ∗name)
- void dump (char ∗dm, int dest)

Public Attributes

- char ID [256]
- char BestID [256]

Private Attributes

- int32_t CauseCounter
- int32_t DNSCauseCounter
- int32_t DELDNSCauseCounter
- int32_t RPTDNSCauseCounter
- int32_t URLCauseCounter
- int32_t DELURLCauseCounter
- int32_t SURLCauseCounter
- int32_t DELSURLCauseCounter
- int32_t HTTPCauseCounter
- int32_t HTTPSCauseCounter
- int32_t USERCauseCounter
- int32_t SERVERCauseCounter
- int32_t WhiteListCauseCounter
- int32_t AlreadyOpenCauseCounter
- int32_t UTreeCauseCounter
- int32_t UnknownCauseCounter
- int32_t DNSUnknownCauseCounter
- int32_t ResolvedDNSUnknownCauseCounter
- int32_t IDLOverLengthCounter
- int32_t IDLOverTokenCounter
- int64_t LastUserTimeStamp
- int64_t StatDNSTimes [100000]
- int32_t StatDNSTimesCounter
- int64_t StatURLTimes [100000]
- int32_t StatURLTimesCounter
- int32_t NoNameFlowCounter
- int32_t DNSNoNameFlowCounter
- int32_t CauseCallCounter
- int32_t DNSCounter
- int32_t FoundDNSIndex [10]
- int8_t FoundDNSCount
- int64_t WorstDNSTime
- int8_t WorstDNSPlace
- int8_t BestDNSPlace
- int64_t BestDNSTime
- int64_t DNSDelay
- int32_t BestURLEventIndex
- int16_t BestURLHash
- int64_t BestURLDelay

- int32_t URLEventIndex
- int32_t BestHTTPEventIndex
- int32_t BestUserEventIndex
- int64_t UserDelay

5.2.1 Detailed Description

Class that manages causal trees and sorts new flows in it.

< An instance of TCauseAnalyzer creates a static array of TREE_BUFFER_SIZE (65535) empty trees. The FlowAggregator will call this object in case of a new flow.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 TCauseAnalyzer::TCauseAnalyzer (void)

Constructor. During creation a fixed array of tree is created for fast storage.

5.2.3 Member Function Documentation

5.2.3.1 uint8_t TCauseAnalyzer::add (char * *name*)

Adds a new flow to the appropiate tree.

Returns

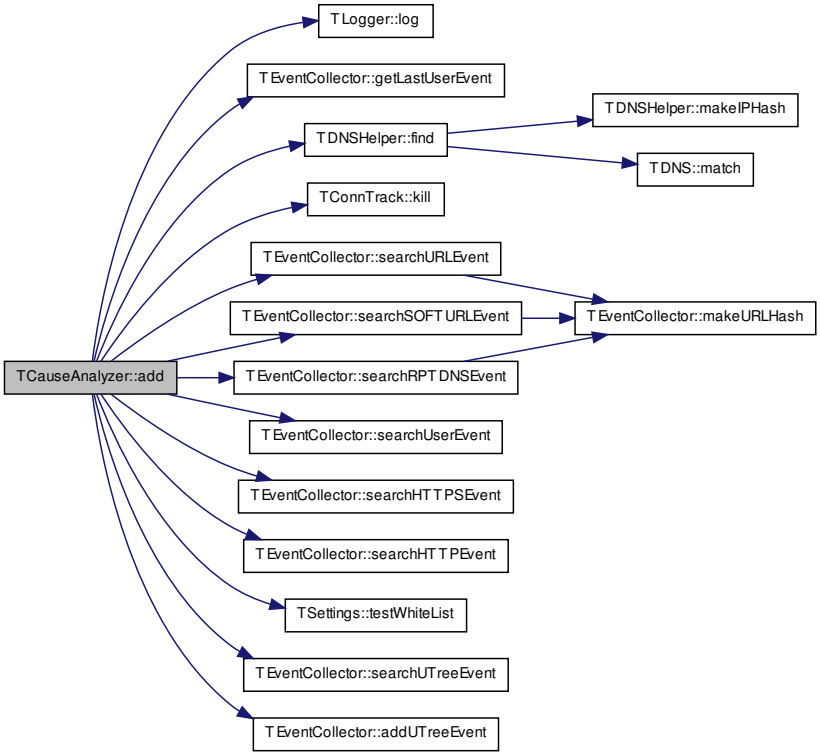
the result of the addition:

- 1 = succesful update of existing tree, Index points to this tree
- 2 = new tree created, Index points to this tree
- other = error in processing, flow not added

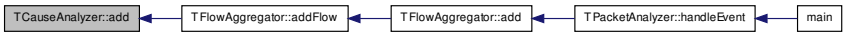
Parameters

*name	Pointer to name string in case of a DNS-reply

Here is the call graph for this function:



Here is the caller graph for this function:



5.2.3.2 void TCauseAnalyzer::dump (char * *dm*, int *dest*)

Dumps the content of all non-empty trees to stdout or logfile

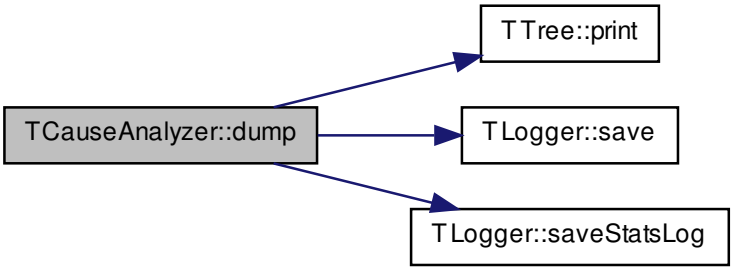
Returns

void

Parameters

∗dm	pointer to content that must be printed. NULL is print to stdout
dest	destination of the log: 0= to stdout, 1=to logfile

Here is the call graph for this function:



Here is the caller graph for this function:

5.2.4 Member Data Documentation

5.2.4.1 int32_t TCauseAnalyzer::AlreadyOpenCauseCounter [private]

5.2.4.2 int8_t TCauseAnalyzer::BestDNSPlace [private]

5.2.4.3 int64_t TCauseAnalyzer::BestDNSTime [private]

5.2.4.4 int32_t TCauseAnalyzer::BestHTTPEventIndex [private]

5.2.4.5 char TCauseAnalyzer::BestID[256]

String with most recent resolved name of last analyzed flow

5.2.4.6 int64_t TCauseAnalyzer::BestURLDelay [private]

5.2.4.7 int32_t TCauseAnalyzer::BestURLEventIndex [private]

5.2.4.8 int16_t TCauseAnalyzer::BestURLHash [private]

5.2.4.9 int32_t TCauseAnalyzer::BestUserEventIndex [private]

5.2.4.10 int32_t TCauseAnalyzer::CauseCallCounter [private]

5.2.4.11 int32_t TCauseAnalyzer::CauseCounter [private]

5.2.4.12 int32_t TCauseAnalyzer::DELDNSCauseCounter [private]

5.2.4.13 int32_t TCauseAnalyzer::DELSURLCauseCounter [private]

5.2.4.14 int32_t TCauseAnalyzer::DELURLCauseCounter [private]

5.2.4.15 int32_t TCauseAnalyzer::DNSCauseCounter [private]

5.2.4.16 int32_t TCauseAnalyzer::DNSCounter [private]

5.2.4.17 int64_t TCauseAnalyzer::DNSDelay [private]

5.2.4.18 int32_t TCauseAnalyzer::DNSNoNameFlowCounter [private]

5.2.4.19 int32_t TCauseAnalyzer::DNSUnknownCauseCounter [private]

5.2.4.20 int8_t TCauseAnalyzer::FoundDNSCount [private]

Number of valid DNS records

5.2.4.21 int32_t TCauseAnalyzer::FoundDNSIndex[10] [private]

DNS-indices that could match

5.2.4.22 int32_t TCauseAnalyzer::HTTPCauseCounter [private]

5.2.4.23 int32_t TCauseAnalyzer::HTTPSCauseCounter [private]

5.2.4.24 char TCauseAnalyzer::ID[256]

String with IP or name of last analyzed flow

5.2.4.25 int32_t TCauseAnalyzer::IDLOverLengthCounter [private]

5.2.4.26 int32_t TCauseAnalyzer::IDLOverTokenCounter [private]

5.2.4.27 int64_t TCauseAnalyzer::LastUserTimeStamp [private]

5.2.4.28 int32_t TCauseAnalyzer::NoNameFlowCounter [private]

For registration of response times

5.2.4.29 int32_t TCauseAnalyzer::ResolvedDNSUnknownCauseCounter [private]

5.2.4.30 int32_t TCauseAnalyzer::RPTDNSCauseCounter [private]

5.2.4.31 int32_t TCauseAnalyzer::SERVERCauseCounter [private]

5.2.4.32 int64_t TCauseAnalyzer::StatDNSTimes[100000] [private]

For registration of response times

5.2.4.33 int32_t TCauseAnalyzer::StatDNSTimesCounter [private]

For registration of response times

5.2.4.34 int64_t TCauseAnalyzer::StatURLTimes[100000] [private]

For registration of response times

5.2.4.35 int32_t TCauseAnalyzer::StatURLTimesCounter [private]

For registration of response times

5.2.4.36 int32_t TCauseAnalyzer::SURLCauseCounter [private]

5.2.4.37 int32_t TCauseAnalyzer::UnknownCauseCounter [private]

5.2.4.38 int32_t TCauseAnalyzer::URLCauseCounter [private]

5.2.4.39 int32_t TCauseAnalyzer::URLEventIndex [private]

5.2.4.40 int32_t TCauseAnalyzer::USERCauseCounter [private]

5.2.4.41 int64_t TCauseAnalyzer::UserDelay [private]

5.2.4.42 int32_t TCauseAnalyzer::UTreeCauseCounter [private]

5.2.4.43 int32_t TCauseAnalyzer::WhiteListCauseCounter [private]

5.2.4.44 int8_t TCauseAnalyzer::WorstDNSPlace [private]

5.2.4.45 int64_t TCauseAnalyzer::WorstDNSTime [private]

The documentation for this class was generated from the following files:

- src/CauseAnalyzer.h
- src/CauseAnalyzer.cpp

5.3 TConnTrack Class Reference

Class that can kill an establshed connection.

`#include <ConnTrack.h>`

Public Member Functions

- TConnTrack (char *ip)
- int kill (uint32_t src, uint32_t dst, uint8_t prot, uint16_t psrc, uint16_t pdst)
- int kill (uint32_t flowindex)

5.3.1 Detailed Description

Class that can kill an establshed connection.

<

5.3.2 Constructor & Destructor Documentation

5.3.2.1 TConnTrack::TConnTrack (char * *ip*)

Constructor. Initialises Firewall

Parameters

ip	local IP-address

5.3.3 Member Function Documentation

5.3.3.1 int TConnTrack::kill (uint32_t *src*, uint32_t *dst*, uint8_t *prot*, uint16_t *psrc*, uint16_t *pdst*)

Kills a flow by specified 5-tuple

Returns

0=success, -1=error

Parameters

src	source IP-address (=local IP-address)
dst	dest IP-address (=remote IP-address)
prot	protocol (only 6=TCP or 17=UDP)
psrc	source port (=local port)
pdst	dest port (=remote port)

Here is the caller graph for this function:

5.3.3.2 int TConnTrack::kill (uint32_t *flowindex*)

Kills a flow by specified flowindex

Returns

0=success, -1=error

Parameters

flowindex	

The documentation for this class was generated from the following files:

- src/ConnTrack.h
- src/ConnTrack.cpp

5.4 TDNS Class Reference

Storage class for DNS-data. Objects of this class are managed by DNSHelper.

`#include <DNS.h>`

Public Member Functions

- TDNS (void)

- void clear (void)
- void set (int32_t f, int64_t ftime, uint32_t ipaddr, uint32_t t, char ∗nm, char ∗cm)
- int match (uint32_t ip, char ∗name)
- void print (char ∗content)

Public Attributes

- int64_t TimeStamp

  Timestamp of the recieved data.
- uint32_t IP

  IP-address.
- char NAME [MAXDOMAINNAMELENGTH]

  Domain name of the original Query.
- char CNAME [MAXDOMAINNAMELENGTH]

  Domain name in the RR (possibly a CNAME)
- uint32_t TTL

  Time To Live in seconds.
- uint8_t Resolved

  Boolean 0= not used, 1=at least 1 received IP is used.
- int32_t FlowIndex

  Index to the most recent DNS Flow that delivered this information.
- int32_t NextDNSIndex

  Links to a next DNS-record by an index (unidirectional linked list).

5.4.1 Detailed Description

Storage class for DNS-data. Objects of this class are managed by DNSHelper.

<

5.4.2 Constructor & Destructor Documentation

5.4.2.1 TDNS::TDNS (void)

The constructor. Resets all the data-members

Here is the call graph for this function:

5.4.3 Member Function Documentation

5.4.3.1 void TDNS::clear (void)

Clears all the data-members, except NextDNSIndex

Returns

void

Here is the caller graph for this function:

5.4.3.2 int TDNS::match (uint32_t *ip*, char * *name*)

Matches IP and Name of DNS-record. If *name is NULL only the IP-address is used in the search

Returns

0=no match, 1=IP match, 2=NAME match, 3=CNAME match

Parameters

ip	IP-address
**name*	Pointer to Name

Here is the caller graph for this function:

5.4.3.3 void TDNS::print (char * *content*)

Prints a flow in readable format to stdout

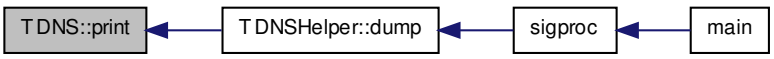
Returns

void

Parameters

**content*	pointer to content that must be printed. NULL is print to stdout

Here is the caller graph for this function:



5.4.3.4 void TDNS::set (int32_t *f*, int64_t *ftime*, uint32_t *ipaddr*, uint32_t *t*, char * *nm*, char * *cm*)

Sets a DNS-object except for the linked list pointer

Returns

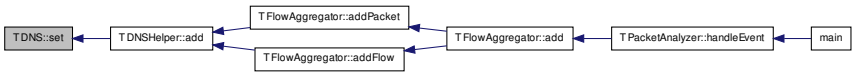
void

Parameters

f	Index to the DNS Flow that delivered this information
ftime	First time the IP-address was referenced in a DNS A record
ipaddr	IP-address of the RR
t	TTL in the RR
nm	Name in the Query

cm	Name in the Answer RR (probably a CNAME or the same as the Query Name)

Here is the caller graph for this function:



5.4.4 Member Data Documentation

5.4.4.1 char TDNS::CNAME[MAXDOMAINNAMELENGTH]

Domain name in the RR (possibly a CNAME)

5.4.4.2 int32_t TDNS::FlowIndex

Index to the most recent DNS Flow that delivered this information.

5.4.4.3 uint32_t TDNS::IP

IP-address.

5.4.4.4 char TDNS::NAME[MAXDOMAINNAMELENGTH]

Domain name of the original Query.

5.4.4.5 int32_t TDNS::NextDNSIndex

Links to a next DNS-record by an index (unidirectional linked list).

5.4.4.6 uint8_t TDNS::Resolved

Boolean 0= not used, 1=at least 1 received IP is used.

5.4.4.7 int64_t TDNS::TimeStamp

Timestamp of the recieved data.

5.4.4.8 uint32_t TDNS::TTL

Time To Live in seconds.

The documentation for this class was generated from the following files:

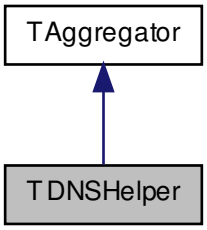
- src/DNS.h
- src/DNS.cpp

5.5 TDNSHelper Class Reference

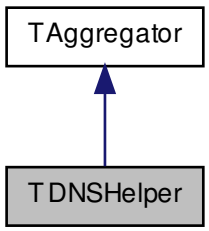
Class specialized in aggregating DNS events. It only processes replies from egress UDP DNS-flows. DNS objects are stored as a chained hashtable of unidirectional linked lists. The hashtable is based on a modulo 256 sum of the IP-address. This results in a fast looukup by IP-adress. Newer records with the same name and IP-adress are replaced. TODO: cleanup-routine that checks one record at a time on expiry.

`#include <DNSHelper.h>`

Inheritance diagram for TDNSHelper:



Collaboration diagram for TDNSHelper:



Public Member Functions

- TDNSHelper (void)
- uint8_t add (void)
- char * getQueryName (void)
- uint8_t deleteRecord (int32_t i)
- int find (uint32_t ip, char *name)
- uint16_t makeIPHash (uint32_t ip)
- void dump (char *content)

Private Member Functions

- uint16_t parseName (uint16_t offset)
- void clearQueryName (void)
- uint8_t addToQueryName (char s)
- void clearAnswerName (void)
- uint8_t addToAnswerName (char s)

Private Attributes

- const uint8_t * Packet

  packet that starts from DNS start
- char QueryName [256]

  string for parsed query name
- char AnswerName [256]

  string for parsed answer name
- uint16_t QueryNameIndex

  write-pointer in QueryString

- uint16_t AnswerNameIndex

  write-pointer in AnswerString

- uint16_t Length

  Length of DNS-header and payload.

- int32_t DNSTable [DNSHASHSIZE]

  Hashtable: Array of indexes to the buckets that belong to the hashtable.

5.5.1 Detailed Description

Class specialized in aggregating DNS events. It only processes replies from egress UDP DNS-flows. DNS objects are stored as a chained hashtable of unidirectional linked lists. The hashtable is based on a modulo 256 sum of the IP-address. This results in a fast looukup by IP-adress. Newer records with the same name and IP-adress are replaced. TODO: cleanup-routine that checks one record at a time on expiry.

<

5.5.2 Constructor & Destructor Documentation

5.5.2.1 TDNSHelper::TDNSHelper (void)

The constructor. Resets all the data-members, including NextFlow

5.5.3 Member Function Documentation

5.5.3.1 uint8_t TDNSHelper::add (void) [virtual]

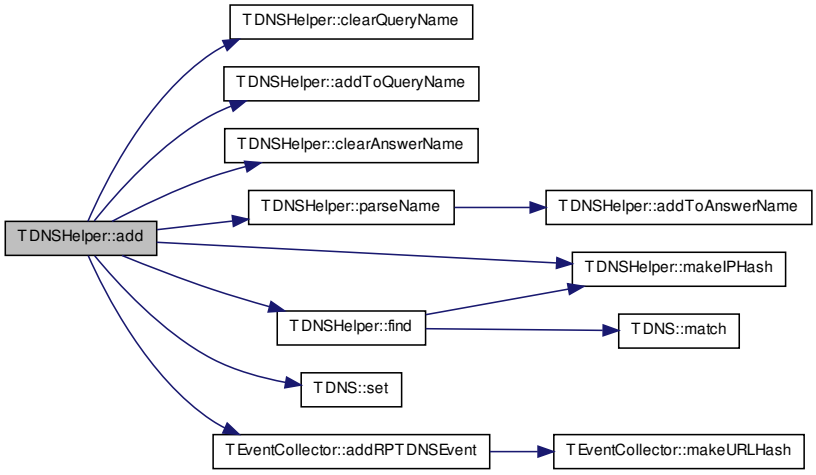
Processes a DNS-response.

Returns

1=success: data updated in existing records, 2=succcess: new DNS-record(s) added, 3+ = error, 0=ingress

Reimplemented from TAggregator.

Here is the call graph for this function:



Here is the caller graph for this function:

5.5.3.2 uint8_t TDNSHelper::addToAnswerName (char *s*) [private]

Copies a character to the AnswerName array.

Returns

1=success, 0=array full

Parameters

s	character

Here is the caller graph for this function:

5.5.3.3 uint8_t TDNSHelper::addToQueryName (char *s*) [private]

Copies a character to the QueryName array.

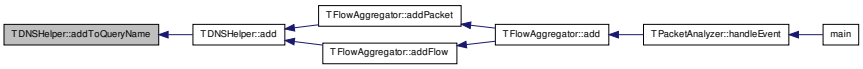
Returns

1=success, 0=array full

Parameters

s	character

Here is the caller graph for this function:



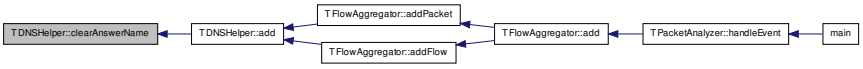
5.5.3.4 void TDNSHelper::clearAnswerName (void) [private]

Clears AnswerName array.

Returns

void

Here is the caller graph for this function:



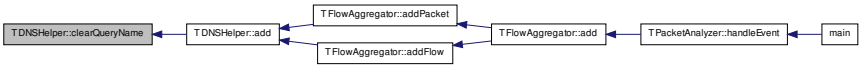
5.5.3.5 void TDNSHelper::clearQueryName (void) [private]

Clears QueryName array.

Returns

void

Here is the caller graph for this function:



5.5.3.6 uint8_t TDNSHelper::deleteRecord (int32_t *i*)

Frees a DNS-record.

Returns

1=ok

Parameters

i	Index of the DNS-record in the static array of DNS-records.

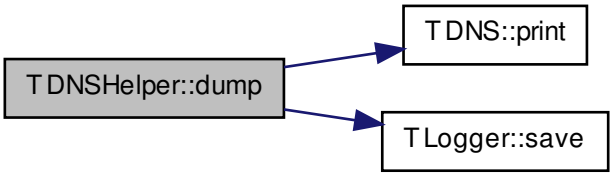
5.5.3.7 void TDNSHelper::dump (char * *content*)

Dumps the content of all DNS-records to stdout return void

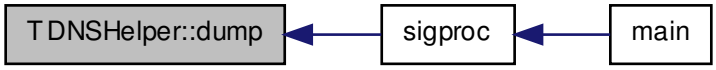
Parameters

**content*	pointer to content that must be printed. NULL is print to stdout

Here is the call graph for this function:



Here is the caller graph for this function:



5.5.3.8 int TDNSHelper::find (uint32_t *ip*, char * *name*)

Finds a DNS-record by specified IP-address and name. If name=NULL only IP-match.

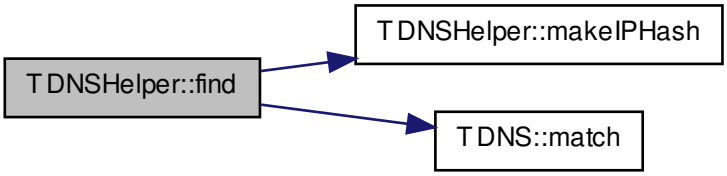
Returns

status, 0=no match-no items left, 1=IP-match, 2=name match, 3=cname match, Index points to the last queried item

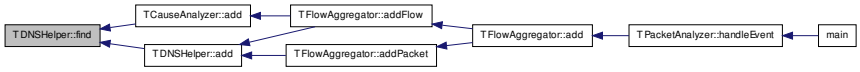
Parameters

ip	IP-address
name	Name of the Query, if NULL only the ip-adress is used in the search

Here is the call graph for this function:



Here is the caller graph for this function:



5.5.3.9 char * TDNSHelper::getQueryName (void)

Gets the Queryname. Ony to be called immediatly after add.

Returns

Pointer to string with most recent queryname

Here is the caller graph for this function:

5.5.3.10 uint16_t TDNSHelper::makeIPHash (uint32_t *ip*)

Calculates a modulo HASHSIZE of the IP-address

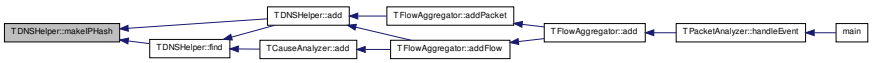
This can be used for organising a fast lookup by a hash-table.

The hashspace is set by DNSHASHSIZE (default=1024).

Returns

HashValue

Here is the caller graph for this function:



5.5.3.11 uint16_t TDNSHelper::parseName (uint16_t *offset*) [private]

Parses an name in an answer from the DNS payload. In case of a pointer it is recursively called.

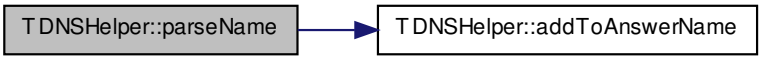
Returns

0=overflow, in all other cases it points to the next field in the packet after the name.

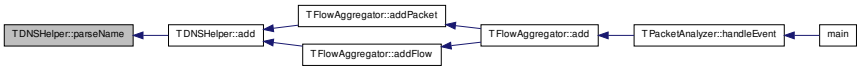
Parameters

offset	present index in the packet

Here is the call graph for this function:



Here is the caller graph for this function:



5.5.4 Member Data Documentation

5.5.4.1 char TDNSHelper::AnswerName[256] [private]

string for parsed answer name

5.5.4.2 uint16_t TDNSHelper::AnswerNameIndex [private]

write-pointer in AnswerString

5.5.4.3 int32_t TDNSHelper::DNSTable[DNSHASHSIZE] [private]

Hashtable: Array of indexes to the buckets that belong to the hashtable.

5.5.4.4 uint16_t TDNSHelper::Length [private]

Length of DNS-header and payload.

5.5.4.5 const uint8_t* TDNSHelper::Packet [private]

packet that starts from DNS start

5.5.4.6 char TDNSHelper::QueryName[256] [private]

string for parsed query name

5.5.4.7 uint16_t TDNSHelper::QueryNameIndex [private]

write-pointer in QueryString

The documentation for this class was generated from the following files:

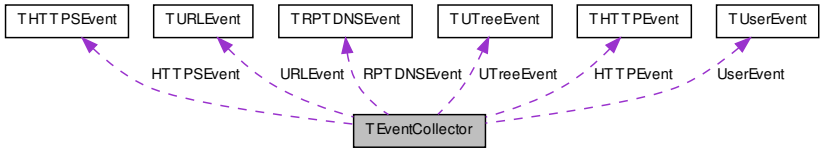
- src/DNSHelper.h
- src/DNSHelper.cpp

5.6 TEventCollector Class Reference

Class specialized in aggregating events that can cause new traffic. Every potential event is buffered here. Hashing is used for fast lookup. FIFO's are used to prevent overflow.

`#include <EventCollector.h>`

Collaboration diagram for TEventCollector:



Public Member Functions

- TEventCollector (void)
- void addUserEvent (int64_t timestamp, uint8_t eventcode, char ∗process)
- void addHTTPEvent (int64_t timestamp, uint8_t eventcode, uint32_t flowindex)
- void removeHTTPEvent (int64_t timestamp, int32_t flowindex)
- void addHTTPSEvent (int64_t timestamp, uint8_t eventcode, uint32_t flowindex)
- void addUTreeEvent (int64_t timestamp, uint32_t treeindex)
- void removeHTTPSEvent (int64_t timestamp, int32_t flowindex)
- void addURLEvent (int64_t timestamp, char ∗url, int32_t flowindex)
- void addRPTDNSEvent (int64_t timestamp, char ∗name, int32_t flowindex)
- int32_t searchUserEvent (int64_t ∗delay)
- int32_t searchHTTPEvent (int64_t ∗delay, int prot)
- int32_t searchHTTPSEvent (int64_t ∗delay, int prot)
- int32_t searchURLEvent (char ∗url, int64_t ∗delay, int32_t index)
- int32_t searchUTreeEvent (int64_t ∗delay)
- int32_t searchSOFTURLEvent (char ∗url, int64_t ∗delay, int32_t index)
- int32_t searchRPTDNSEvent (char ∗name, int64_t ∗delay)
- int64_t getLastUserEvent (void)
- void makeIPHash (uint32_t ip)
- void makeURLHash (char ∗url)
- void dump (char ∗content, int dest)

Public Attributes

- TUserEvent UserEvent [EVENTBUFFERSIZE]

  FIFO of user events.
- THTTPEvent HTTPEvent [EVENTBUFFERSIZE]

  FIFO of generic HTTP events.
- THTTPSEvent HTTPSEvent [EVENTBUFFERSIZE]

  FIFO of generic HTTPS events.
- TUTreeEvent UTreeEvent [EVENTBUFFERSIZE]

  FIFO of Trees with Unknown cause.

- TURLEvent URLEvent [URLEVENTHASHSIZE][URLEVENTBUFFERSIZE]

  Hastable with URLEVENTHASSIZE buckets. Each bucket is a FIFO.
- TRPTDNSEvent RPTDNSEvent [URLEVENTHASHSIZE][RPTDNSEVENTBUFFERSIZE]

  Hastable with URLEVENTHASSIZE buckets. Each bucket is a FIFO.
- int16_t Hash
- int64_t AggregatedWindowTime
- int64_t WindowEndTime
- int32_t UserEventCount
- int32_t HTTPEventCount
- int32_t HTTPSEventCount
- int32_t UTreeEventCount

Private Attributes

- int32_t UserEventIndex

  index to most recent event or -1 if empty
- int32_t HTTPEventIndex

  index to most recent event or -1 if empty
- int32_t HTTPSEventIndex

  index to most recent event or -1 if empty
- int32_t UTreeEventIndex

  index to most recent event or -1 if empty
- int32_t URLEventIndex [URLEVENTHASHSIZE]

  array of indices to most recent URL events or -1 if empty
- int32_t RPTDNSEventIndex [URLEVENTHASHSIZE]

  array of indices to most recent REPEAT DNS events or -1 if empty

5.6.1 Detailed Description

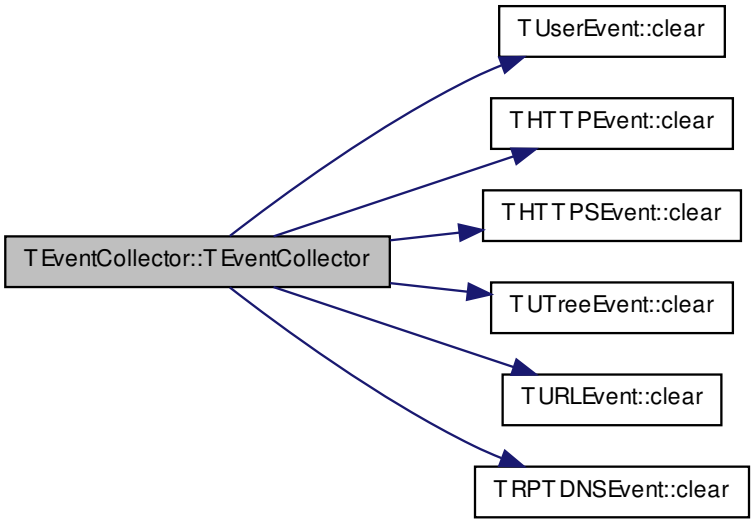
Class specialized in aggregating events that can cause new traffic. Every potential event is buffered here. Hashing is used for fast lookup. FIFO's are used to prevent overflow.

<

5.6.2 Constructor & Destructor Documentation

5.6.2.1 TEventCollector::TEventCollector (void)

Here is the call graph for this function:



5.6.3 Member Function Documentation

5.6.3.1 void TEventCollector::addHTTPEvent (int64_t *timestamp*, uint8_t *eventcode*, uint32_t *flowindex*)

Adds a HTTP event

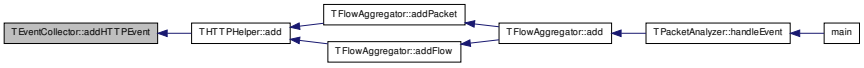
Returns

void

Parameters

timestamp	
eventcode	Describes the type of event: 0=non trivial data received, 1=push flag received
flowindex	Index to the HTTP flow

Here is the caller graph for this function:



5.6.3.2 void TEventCollector::addHTTPSEvent (int64_t *timestamp*, uint8_t *eventcode*, uint32_t *flowindex*)

Adds a HTTP event

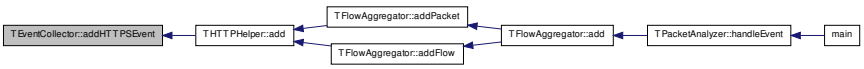
Returns

void

Parameters

timestamp	
eventcode	Describes the type of event: 0=non trivial data received, 1=push flag received
flowindex	Index to the HTTP flow

Here is the caller graph for this function:



5.6.3.3 void TEventCollector::addRPTDNSEvent (int64_t *timestamp*, char * *name*, int32_t *flowindex*)

Adds a REPEAT DNS event

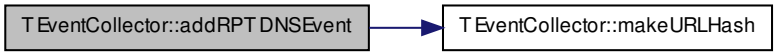
Returns

void

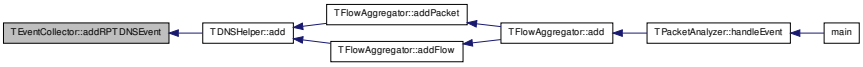
Parameters

timestamp	
**name*	Pointer to the string that contains the query name
flowindex	Index to the DNS flow

Here is the call graph for this function:



Here is the caller graph for this function:



5.6.3.4 void TEventCollector::addURLEvent (int64_t *timestamp*, char * *url*, int32_t *flowindex*)

Adds a HTTP URL event

Returns

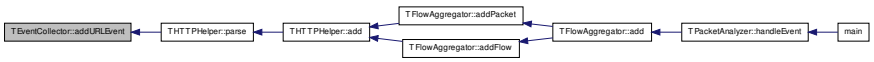
void

Parameters

timestamp	
**url*	Pointer to the string that contains the received URL
flowindex	Index to the HTTP flow

Here is the call graph for this function:

Here is the caller graph for this function:



5.6.3.5 void TEventCollector::addUserEvent (int64_t *timestamp*, uint8_t *eventcode*, char * *process*)

Adds a user event

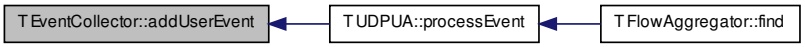
Returns

void

Parameters

timestamp	
eventcode	Describes the type of event: 0=other, 1=F5, 2=LMB, 3=Enter
**process*	Pointer to a string that describes the geerating process (if supported by the agent)

Here is the caller graph for this function:



5.6.3.6 void TEventCollector::addUTreeEvent (int64_t *timestamp*, uint32_t *treeindex*)

Adds an Unknown cause Tree event

Returns

void

Parameters

timestamp	
treeindex	Index to the Tree

Here is the caller graph for this function:

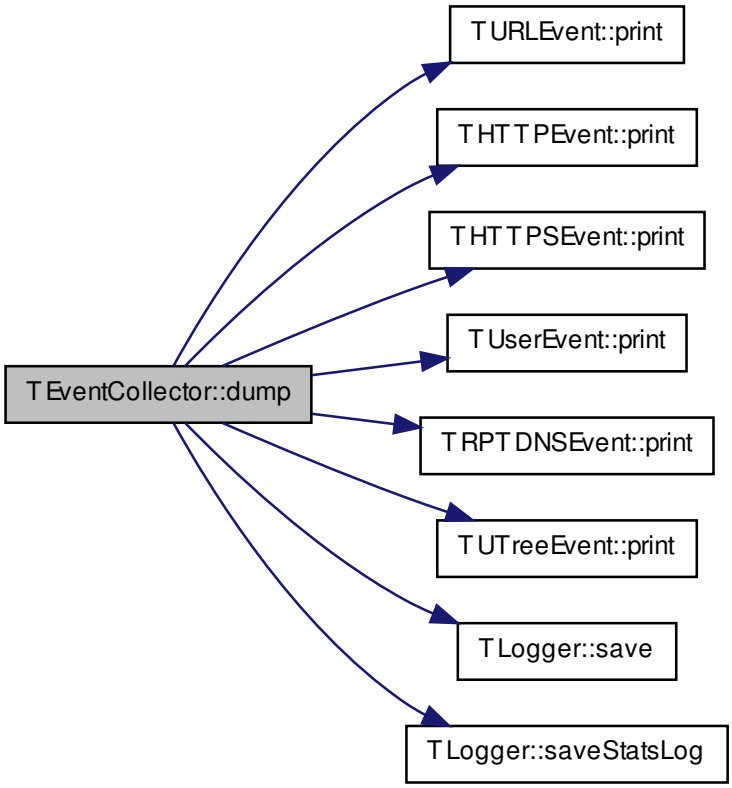
5.6.3.7 void TEventCollector::dump (char * *content*, int *dest*)

Dumps the content of all Events to stdout return void

Parameters

**content*	pointer to content that must be printed. NULL is print to stdout
dest	destination of the log: 0= to stdout, 1=to logfile

Here is the call graph for this function:



Here is the caller graph for this function:

5.6.3.8 int64_t TEventCollector::getLastUserEvent (void)

Gives timestamp of last enter/mousclick/F5 event or zero if none.

Returns

timestamp of last enter/mousclick/F5 or 0 if none

Here is the caller graph for this function:

5.6.3.9 void TEventCollector::makeIPHash (uint32_t *ip*)

Calculates the hash of an IP-address

Returns

hash

Parameters

ip	IP-adress

5.6.3.10 void TEventCollector::makeURLHash (char * *url*)

Calculates the hash of an URL or name

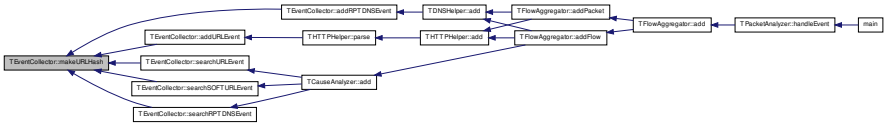
Returns

hash

Parameters

**url*	Pointer to name string

Here is the caller graph for this function:



5.6.3.11 void TEventCollector::removeHTTPEvent (int64_t *timestamp,* int32_t *flowindex*)

Removes a HTTP event, by setting the Eventcode to CAUSE_WITHDRAWN

Returns

void

Parameters

timestamp	
flowindex	Index to the HTTP flow

5.6.3.12 void TEventCollector::removeHTTPSEvent (int64_t *timestamp,* int32_t *flowindex*)

Removes a HTTP event, by setting the Eventcode to CAUSE_WITHDRAWN

Returns

void

Parameters

timestamp	
flowindex	Index to the HTTP flow

5.6.3.13 int32_t TEventCollector::searchHTTPEvent (int64_t * *delay,* int *prot*)

Delivers a HTTP event, starting with the most recent one and counting back on every call.

Returns

index to most Event or -1 if empty

Parameters

∗*delay*	max search time to lookup back in history, returns the time between found event and present time
prot	80=http, 443=https

Here is the caller graph for this function:

5.6.3.14 int32_t TEventCollector::searchHTTPSEvent (int64_t ∗ *delay,* int *prot*)

Delivers a HTTP event, starting with the most recent one and counting back on every call.

Returns

index to most Event or -1 if empty

Parameters

∗*delay*	max search time to lookup back in history, returns the time between found event and present time
prot	80=http, 443=https

Here is the caller graph for this function:

5.6.3.15 int32_t TEventCollector::searchRPTDNSEvent (char ∗ *name,* int64_t ∗ *delay*)

Delivers the flowindex of a REPEAT DNS event, including the name and delay, by walking back in history.

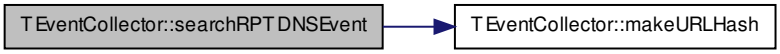
Returns

index to FlowIndex that created the event

Parameters

∗*name*	Pointer to the string that must match the query name
∗*delay*	max search time to lookup back in history, returns the time between found event and present time

Here is the call graph for this function:



Here is the caller graph for this function:

5.6.3.16 int32_t TEventCollector::searchSOFTURLEvent (char ∗ *url*, int64_t ∗ *delay*, int32_t *index*)

Delivers a SOFTHTTP event by URL substring, by walking back in history.

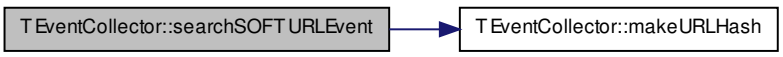
Returns

index to most Event or -1 if empty

Parameters

∗*url*	Pointer to the string that must match the URL
∗*delay*	max search time to lookup back in history, returns the time between found event and present time
index	Last found event, used for search continuation, -1 is start from the most recent entry

Here is the call graph for this function:



Here is the caller graph for this function:

5.6.3.17 int32_t TEventCollector::searchURLEvent (char * *url*, int64_t * *delay*, int32_t *index*)

Delivers a HTTP event by URL, by walking back in history.

Returns

index to most Event or -1 if empty

Parameters

**url*	Pointer to the string that must match the URL
**delay*	max search time to lookup back in history, returns the time between found event and present time
index	Pointer to Index of Last found event, -1=start new search or as return: nothing found

Here is the call graph for this function:

Here is the caller graph for this function:

5.6.3.18 int32_t TEventCollector::searchUserEvent (int64_t * *delay*)

Delivers a DNS event, starting with the most recent one and counting back on every call.

Returns

index to most Event or -1 if empty

Parameters

**delay*	max search time to lookup back in history, returns the time between found event and present time

Here is the caller graph for this function:

5.6.3.19 int32_t TEventCollector::searchUTreeEvent (int64_t * *delay*)

Delivers the most recent unknown cause tree event, by walking back in history.

Returns

index to most Event or -1 if empty

Parameters

**delay*	pointer to max search time to lookup back in history, returns the time between found event and present time

Here is the caller graph for this function:

5.6.4 Member Data Documentation

5.6.4.1 int64_t TEventCollector::AggregatedWindowTime

5.6.4.2 int16_t TEventCollector::Hash

5.6.4.3 THTTPEvent TEventCollector::HTTPEvent[EVENTBUFFERSIZE]

FIFO of generic HTTP events.

5.6.4.4 int32_t TEventCollector::HTTPEventCount

5.6.4.5 int32_t TEventCollector::HTTPEventIndex [private]

index to most recent event or -1 if empty

5.6.4.6 THTTPSEvent TEventCollector::HTTPSEvent[EVENTBUFFERSIZE]

FIFO of generic HTTPS events.

5.6.4.7 int32_t TEventCollector::HTTPSEventCount

5.6.4.8 int32_t TEventCollector::HTTPSEventIndex [private]

index to most recent event or -1 if empty

5.6.4.9 TRPTDNSEvent TEventCollector::RPTDNSEvent[URLEVENTHASHSIZE][-RPTDNSEVENTBUFFERSIZE]

Hastable with URLEVENTHASSIZE buckets. Each bucket is a FIFO.

5.6.4.10 int32_t TEventCollector::RPTDNSEventIndex[URLEVENTHASHSIZE] [private]

array of indices to most recent REPEAT DNS events or -1 if empty

5.6.4.11 TURLEvent TEventCollector::URLEvent[URLEVENTHASHSIZE][URLEVENTBUFFERSIZE]

Hastable with URLEVENTHASSIZE buckets. Each bucket is a FIFO.

5.6.4.12 int32_t TEventCollector::URLEventIndex[URLEVENTHASHSIZE] [private]

array of indices to most recent URL events or -1 if empty

5.6.4.13 TUserEvent TEventCollector::UserEvent[EVENTBUFFERSIZE]

FIFO of user events.

5.6.4.14 int32_t TEventCollector::UserEventCount

5.6.4.15 int32_t TEventCollector::UserEventIndex [private]

index to most recent event or -1 if empty

5.6.4.16 TUTreeEvent TEventCollector::UTreeEvent[EVENTBUFFERSIZE]

FIFO of Trees with Unknown cause.

5.6.4.17 int32_t TEventCollector::UTreeEventCount

5.6.4.18 int32_t TEventCollector::UTreeEventIndex [private]

index to most recent event or -1 if empty

5.6.4.19 int64_t TEventCollector::WindowEndTime

The documentation for this class was generated from the following files:

- src/EventCollector.h
- src/EventCollector.cpp

5.7 TFlow Class Reference

Storage class for a bidirectional flow.

`#include <Flow.h>`

Public Member Functions

- TFlow (void)
- void clear (void)
- uint16_t getHash (void)
- uint8_t match (uint8_t p, uint32_t lIP, uint32_t rIP, uint16_t lP, uint16_t rP, uint16_t id)
- void print (char *content)

Public Attributes

- int32_t NextFlowIndex

  Links to a next flow by an index (a kind of linked list) -1 = no next flow.
- uint8_t Status

  Status is the current status of the flow: 0=empty, 1=aggregated from 1 packet, 2=aggregated from more packets.
- int64_t StartTime

  Time of first packet in Flow in useconds sinze 1970.
- int64_t StopTime

  Time of last packet in Flow in useconds sinze 1970.
- int32_t NumberOfTransmittedBytes

  Aggregated number of transmitted bytes.
- int32_t NumberOfReceivedBytes

  Aggregated number of transmitted bytes.
- int32_t NumberOfTransmittedPackets

  Aggregated number of transmitted bytes.
- int32_t NumberOfReceivedPackets

  Aggregated number of transmitted bytes.
- uint8_t Direction

  Flow direction: UNDEFINED(0), INGRESS(1), EGRESS(2)
- uint8_t Protocol

  1=ICMP, 6=TCP, 17=UDP, 255=IP-other
- uint32_t LocalIP

  IP-address of the observed local computer.
- uint32_t RemoteIP

  IP-address of the remote computer.
- uint16_t LocalPort

  UDP/TCP=Port, ICMP=identifier, Other=0.
- uint16_t RemotePort

  UDP/TCP=Port ICMP=sequence number, Other=0.
- uint16_t Identification

  Identification field (only defined for DNS, 0 for other protocols)
- uint8_t TCPFlag

  Aggregated TCP-Flags if TCP-Flow.

- uint32_t LocalSEQ

  Local SEQ (0 if undefined)
- uint32_t RemoteSEQ

  Remote expected SEQ (0 if undefined)
- int32_t DNSIndex

  points to DNS-name in a STRING-table, -1 = no DNS
- int32_t HTTPIndex

  if HTTP-flow, this index points to a special object that identifies triggers of new flows, 0 = no HTTP-traffic
- int32_t TreeIndex

  id of the flow tree (tree id)
- int32_t ParentFlow

  id of parent flow in the tree, 0=no parents (root flow)
- uint8_t Resolver

  bit 0..3=resolver protocol (0=nothing, 1=DNS, 2..15=spare, bit 7=1=resolved)
- uint8_t Cause

  Cause of the Flow.
- uint8_t CauseReliability

  Reliability of the cause.
- uint64_t CausalTime

  Timestamp of the cause that created this flow.

5.7.1 Detailed Description

Storage class for a bidirectional flow.

<

5.7.2 Constructor & Destructor Documentation

5.7.2.1 TFlow::TFlow (void)

The constructor. Resets all the data-members, including NextFlow

Here is the call graph for this function:

5.7.3 Member Function Documentation

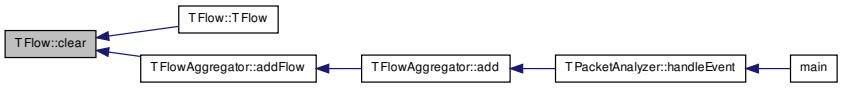
5.7.3.1 void TFlow::clear (void)

Clears all the data-members, except NextFlow

Returns

void

Here is the caller graph for this function:



5.7.3.2 uint16_t TFlow::getHash (void)

Calculates a 10-bit hash by the XOR-sum of SourceIP, DestIP, Sourceport, DestPort and Identification.

This can be used for organising a fast lookup by a hash-table.

The value of hashes is set by FLOWHASHSIZE (default=1024).

Returns

HashValue

Here is the caller graph for this function:

5.7.3.3 uint8_t TFlow::match (uint8_t *p*, uint32_t *lIP*, uint32_t *rIP*, uint16_t *lP*, uint16_t *rP*, uint16_t *id*)

Matches a flow by its addresses and port-numbers and protocol.

Returns

1 if matches or 251-255 if it does not match (completely).

Parameters

p	Protocol number: 1=ICMP, 6=TCP, 17=UDP, 255=IP-other
lIP	Local Ip-address
rIP	Remote Ip-address
lP	Local Port
rP	Remote Port
id	Identication (DNS defined, 0 for other protocols)

5.7.3.4 void TFlow::print (char * *content*)

Prints a flow in readable format to a string

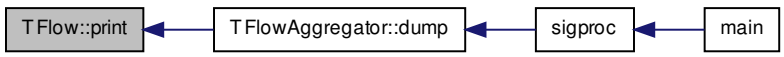
Returns

void

Parameters

**content*	pointer to content that must be printed.

Here is the caller graph for this function:



5.7.4 Member Data Documentation

5.7.4.1 uint64_t TFlow::CausalTime

Timestamp of the cause that created this flow.

5.7.4.2 uint8_t TFlow::Cause

Cause of the Flow.

5.7.4.3 uint8_t TFlow::CauseReliability

Reliability of the cause.

5.7.4.4 uint8_t TFlow::Direction

Flow direction: UNDEFINED(0), INGRESS(1), EGRESS(2)

5.7.4.5 int32_t TFlow::DNSIndex

points to DNS-name in a STRING-table, -1 = no DNS

5.7.4.6 int32_t TFlow::HTTPIndex

if HTTP-flow, this index points to a special object that identifies triggers of new flows, 0 = no HTTP-traffic

5.7.4.7 uint16_t TFlow::Identification

Identification field (only defined for DNS, 0 for other protocols)

5.7.4.8 uint32_t TFlow::LocalIP

IP-address of the observed local computer.

5.7.4.9 uint16_t TFlow::LocalPort

UDP/TCP=Port, ICMP=identifier, Other=0.

5.7.4.10 uint32_t TFlow::LocalSEQ

Local SEQ (0 if undefined)

5.7.4.11 int32_t TFlow::NextFlowIndex

Links to a next flow by an index (a kind of linked list) -1 = no next flow.

5.7.4.12 int32_t TFlow::NumberOfReceivedBytes

Aggregated number of transmitted bytes.

5.7.4.13 int32_t TFlow::NumberOfReceivedPackets

Aggregated number of transmitted bytes.

5.7.4.14 int32_t TFlow::NumberOfTransmittedBytes

Aggregated number of transmitted bytes.

5.7.4.15 int32_t TFlow::NumberOfTransmittedPackets

Aggregated number of transmitted bytes.

5.7.4.16 int32_t TFlow::ParentFlow

id of parent flow in the tree, 0=no parents (root flow)

5.7.4.17 uint8_t TFlow::Protocol

1=ICMP, 6=TCP, 17=UDP, 255=IP-other

5.7.4.18 uint32_t TFlow::RemoteIP

IP-address of the remote computer.

5.7.4.19 uint16_t TFlow::RemotePort

UDP/TCP=Port ICMP=sequence number, Other=0.

5.7.4.20 uint32_t TFlow::RemoteSEQ

Remote expected SEQ (0 if undefined)

5.7.4.21 uint8_t TFlow::Resolver

bit 0..3=resolver protocol (0=nothing, 1=DNS, 2..15=spare, bit 7=1=resolved)

5.7.4.22 int64_t TFlow::StartTime

Time of first packet in Flow in useconds sinze 1970.

5.7.4.23 uint8_t TFlow::Status

Status is the current status of the flow: 0=empty, 1=aggregated from 1 packet, 2=aggregated from more packets.

5.7.4.24 int64_t TFlow::StopTime

Time of last packet in Flow in useconds sinze 1970.

5.7.4.25 uint8_t TFlow::TCPFlag

Aggregated TCP-Flags if TCP-Flow.

5.7.4.26 int32_t TFlow::TreeIndex

id of the flow tree (tree id)

The documentation for this class was generated from the following files:

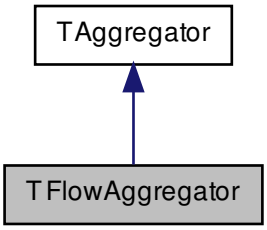
- src/Flow.h
- src/Flow.cpp

5.8 TFlowAggregator Class Reference

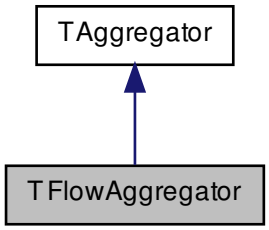
Important class that manages the flows.

`#include <FlowAggregator.h>`

Inheritance diagram for TFlowAggregator:



Collaboration diagram for TFlowAggregator:



Public Member Functions

- TFlowAggregator (uint32_t lIP)
- uint8_t add (void)
- void dump (char ∗content, int dest)

Public Attributes

- uint8_t Direction

  Flow direction of last analyzed packet: INGRESS, EGRESS or UNDEFINED.

Private Member Functions

- uint8_t addPacket (void)
- uint8_t addFlow (void)
- uint8_t deleteFlow (int32_t i)
- uint8_t find (void)
- uint8_t find (uint8_t p, uint32_t sIP, uint32_t dIP, uint16_t sP, uint16_t dP, uint16_t id)

Private Attributes

- uint32_t LocalIP

  IP-address of local as a reference for the flow direction.
- int32_t TotalPacketCounter
- int32_t InPacketCounter
- int32_t OutPacketCounter

- int32_t NoLocalPacketCounter
- int32_t EgressFlowCounter
- int32_t IngressFlowCounter
- int32_t AlreadyOpenCounter
- int32_t EgressUDPFlowCounter
- int32_t NonEmptyRRUDPCounter
- int32_t IngressUDPFlowCounter
- int32_t EgressTCPFlowCounter
- int32_t IngressTCPFlowCounter
- int32_t EgressICMPFlowCounter
- int32_t IngressICMPFlowCounter
- int32_t EgressOtherFlowCounter
- int32_t IngressOtherFlowCounter
- int32_t EgressHTTPFlowCounter
- int32_t EgressHTTPSFlowCounter
- int32_t EgressDNSFlowCounter
- int64_t StartTime

5.8.1 Detailed Description

Important class that manages the flows.

< It creates a static array of FLOWBUFFERSIZE (65536) Flow-objects. By inspection of packets it creates or updates flows. For fast search a hash table is used with FLOWHASHSIZE (1024) entries. Each entry is the start of a linked list of Flows. Instead of Pointers the array-indices are used in the linked list.New flows are placed in a randomly picked empty flow of the static array.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 TFlowAggregator::TFlowAggregator (uint32_t *IIP*)

Constructor. During creation a fixed array of Flows with hashtable is created for storage.

Parameters

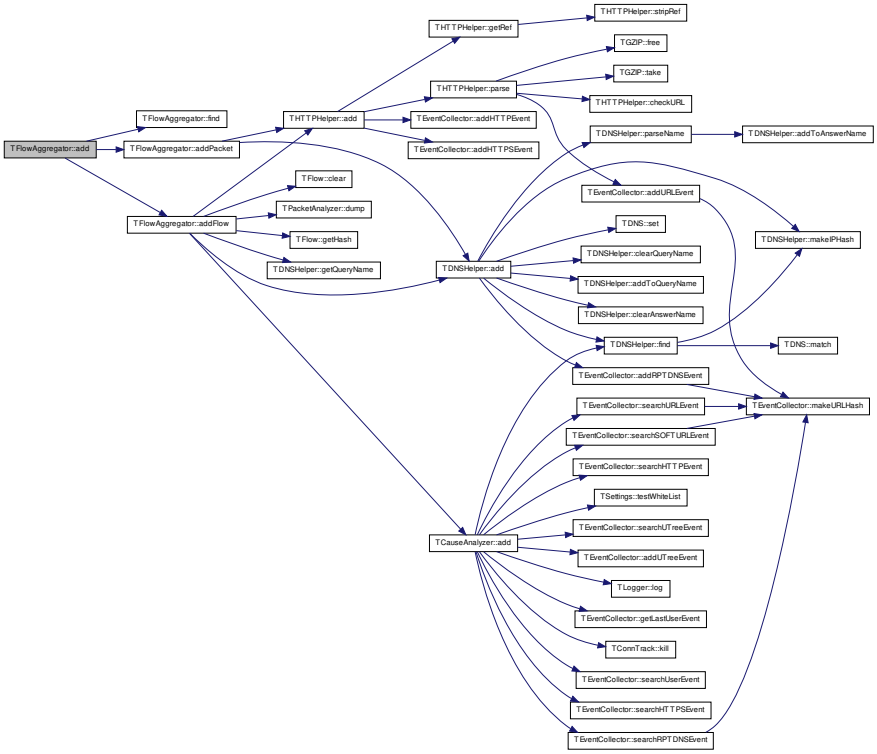
IIP	The IP-address of the observed device. This will be the localIP-adddress.

5.8.3 Member Function Documentation

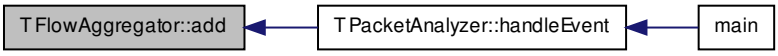
5.8.3.1 uint8_t TFlowAggregator::add (void) [virtual]

Reimplemented from TAggregator.

Here is the call graph for this function:



Here is the caller graph for this function:



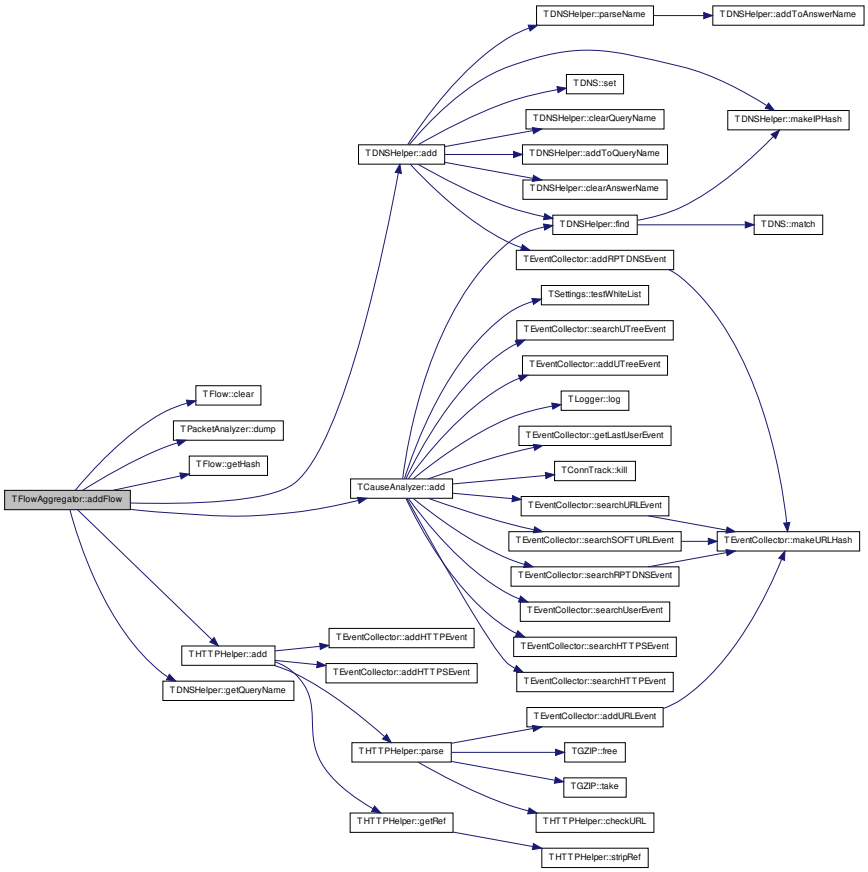
5.8.3.2 uint8_t TFlowAggregator::addFlow (void) [private]

Makes a new flow of the current Packet in PacketAnalyzer

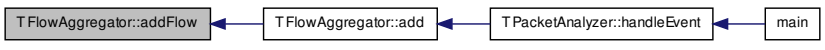
Returns

1=ok, 0=Buffer full or no matching LocalIP or No SYN in HTTP(S)

Here is the call graph for this function:



Here is the caller graph for this function:



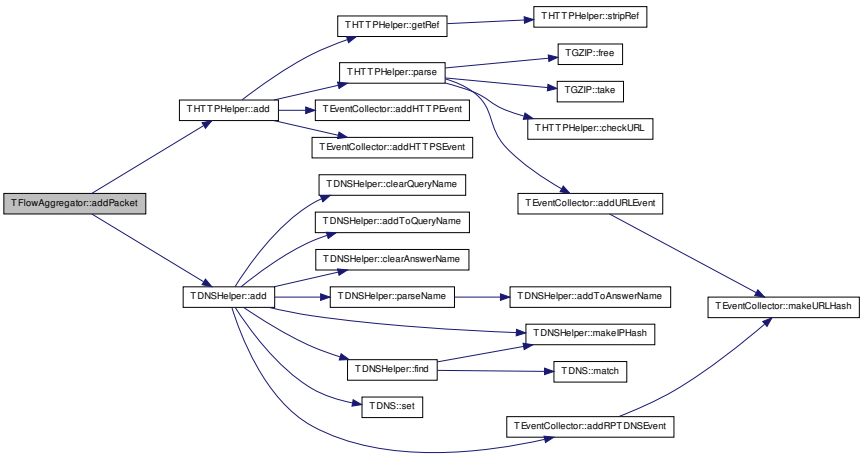
5.8.3.3 uint8_t TFlowAggregator::addPacket (void) [private]

adds packet to existing flow or creates a new flow. Important to call first "find", because it uses Index to link with an existing flow.

Returns

1=success

Here is the call graph for this function:



Here is the caller graph for this function:

5.8.3.4 uint8_t TFlowAggregator::deleteFlow (int32_t *i*) [private]

Frees a flow. Untested at this moment.

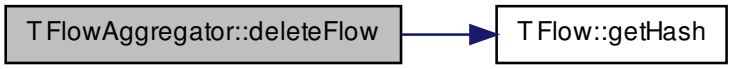
Returns

1=ok

Parameters

i	Index of the flow in the static array of flows.

Here is the call graph for this function:



5.8.3.5 void TFlowAggregator::dump (char * *content*, int *dest*)

updates existing flow or creates a new flow with the current packet in PacketAnalyzer. return 1 = success Dumps the content of all flows to stdout or logfile

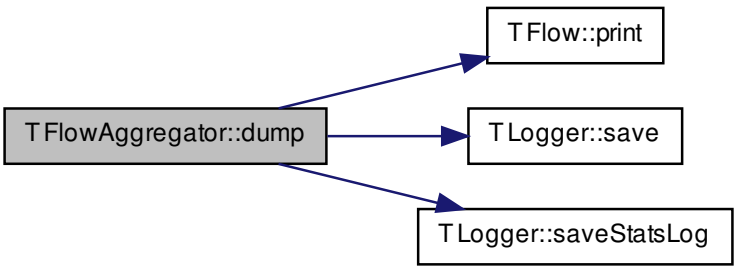
Returns

void

Parameters

**content*	pointer to content that must be printed. NULL is print to stdout
dest	destination of the log: 0= to stdout, 1=to logfile

Here is the call graph for this function:



Here is the caller graph for this function:

5.8.3.6 uint8_t TFlowAggregator::find (void) [private]

Finds the flow of current packet in PacketAnalyzer if it exists.

Returns

result of the find action

- 1 = Flow found, the index can be found in Index
- 2 = Flow not found, Index=-1
- 0 = No local IP present in the Packet

Here is the caller graph for this function:

5.8.3.7 uint8_t TFlowAggregator::find (uint8_t *p*, uint32_t *slP*, uint32_t *dlP*, uint16_t *sP*, uint16_t *dP*, uint16_t *id*) [private]

Finds the flow specified IP/Port parameters. return 1=success

Returns

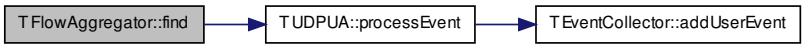
result of the find action

- 1 = Flow found, the index can be found in Index
- 2 = Flow not found, Index=-1
- 0 = No local IP present in the Packet

Parameters

p	Protocol number (e.g. 6=TCP)
sIP	Source IP address
dIP	Destination IP address
sP	Source Port
dP	Destination Port
id	Identification field, as used in DNS, 0 for other protocols

Here is the call graph for this function:



5.8.4 Member Data Documentation

5.8.4.1 int32_t TFlowAggregator::AlreadyOpenCounter [private]

5.8.4.2 uint8_t TFlowAggregator::Direction

Flow direction of last analyzed packet: INGRESS, EGRESS or UNDEFINED.

5.8.4.3 int32_t TFlowAggregator::EgressDNSFlowCounter [private]

5.8.4.4 int32_t TFlowAggregator::EgressFlowCounter [private]

5.8.4.5 int32_t TFlowAggregator::EgressHTTPFlowCounter [private]

5.8.4.6 int32_t TFlowAggregator::EgressHTTPSFlowCounter [private]

5.8.4.7 int32_t TFlowAggregator::EgressICMPFlowCounter [private]

5.8.4.8 int32_t TFlowAggregator::EgressOtherFlowCounter [private]

5.8.4.9 int32_t TFlowAggregator::EgressTCPFlowCounter [private]

5.8.4.10 int32_t TFlowAggregator::EgressUDPFlowCounter [private]

5.8.4.11 int32_t TFlowAggregator::IngressFlowCounter [private]

5.8.4.12 int32_t TFlowAggregator::IngressICMPFlowCounter [private]

5.8.4.13 int32_t TFlowAggregator::IngressOtherFlowCounter [private]

5.8.4.14 int32_t TFlowAggregator::IngressTCPFlowCounter [private]

5.8.4.15 int32_t TFlowAggregator::IngressUDPFlowCounter [private]

5.8.4.16 int32_t TFlowAggregator::InPacketCounter [private]

5.8.4.17 uint32_t TFlowAggregator::LocalIP [private]

IP-address of local as a reference for the flow direction.

5.8.4.18 int32_t TFlowAggregator::NoLocalPacketCounter [private]

5.8.4.19 int32_t TFlowAggregator::NonEmptyRRUDPCounter [private]

5.8.4.20 int32_t TFlowAggregator::OutPacketCounter [private]

5.8.4.21 int64_t TFlowAggregator::StartTime [private]

5.8.4.22 int32_t TFlowAggregator::TotalPacketCounter [private]

The documentation for this class was generated from the following files:

- src/FlowAggregator.h
- src/FlowAggregator.cpp

5.9 TGZIP Class Reference

Class for deflating gzip data. Wraps around zlib.

`#include <GZIP.h>`

Public Member Functions

- TGZIP (void)
- int isInUse (void)
- void take (void)
- void free (void)
- int uncompress (void)

Public Attributes

- int Processing
- z_stream strm
- unsigned char * InBuffer
- int InLength

- unsigned char OutBuffer [GZIP_BUFFERSIZE]
- int OutLength

Private Attributes

- int InUse

5.9.1 Detailed Description

Class for deflating gzip data. Wraps around zlib.

<

5.9.2 Constructor & Destructor Documentation

5.9.2.1 TGZIP::TGZIP (void)

The constructor. Resets all the data-members, including NextFlow

Here is the call graph for this function:

5.9.3 Member Function Documentation

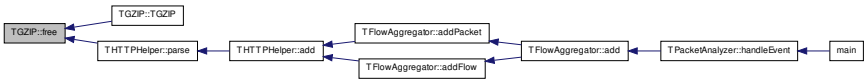
5.9.3.1 void TGZIP::free (void)

Frees the gzip. Memory is flushed cleared etc.

Returns

void

Here is the caller graph for this function:



5.9.3.2 int TGZIP::isInUse (void)

Tests if the object is free to take.

Returns

0=free 1=in use

5.9.3.3 void TGZIP::take (void)

Takes the gzip-object. Memory is assumed to be flushed, cleared

Returns

void

Here is the caller graph for this function:

5.9.3.4 int TGZIP::uncompress (void)

Uncompresses a blok of data. Inlength and ∗Inbuffer must be prepared before this call

Returns

int 0=ok, -1 wrong magic number, -2=no data left

5.9.4 Member Data Documentation

5.9.4.1 unsigned char* TGZIP::InBuffer

Pointer to the input buffer

5.9.4.2 int TGZIP::InLength

Lenth of the input buffer

5.9.4.3 int TGZIP::InUse [private]

Boolean 0=object is free, 1=object is already taken

5.9.4.4 unsigned char TGZIP::OutBuffer[GZIP_BUFFERSIZE]

Pointer to the output buffer

5.9.4.5 int TGZIP::OutLength

Lenth of the output buffer

5.9.4.6 int TGZIP::Processing

5.9.4.7 z_stream TGZIP::strm

The documentation for this class was generated from the following files:

- src/GZIP.h
- src/GZIP.cpp

5.10 THTTP Class Reference

Storage class for HTTP-events. Every HTTP-flow or HTTPS-flow that can potentially trigger new traffi by embedded URL's has an object of this class.

`#include <HTTP.h>`

Public Member Functions

- THTTP (void)
- void clear (void)
- void print (char *content)

Public Attributes

- int64_t Time

  Timestamp of the last HTTP-event.
- uint8_t Status

  State of the HTTP-dialog.
- uint32_t InByteCounter

  Number of ingress bytes during a recieve phase.
- uint32_t OutByteCounter

  Number of egress bytes during a send phase.
- uint32_t TotalInByteCounter

  Total number of egress payload bytes.
- uint32_t TotalOutByteCounter

  Total number of ingress payload bytes.
- int32_t FlowIndex

  Index to Flow that created the cause.
- uint8_t ParseState

  Main State of the FSM for parsing URL's.
- uint8_t ParseSubState

  Sub State inside a Main State for parsing URL's.
- uint16_t ParseMicroState

  Micro State inside a SubState for parsing URL's.
- int32_t PayloadSize

  Size of the HTTP(S) payload in the last received packet.
- uint8_t ContentType

  0=unknown, 1=text
- uint8_t Encoding

  0=unknown, 1=gzip
- uint8_t Chunked

  0=unknown, 1=chunked
- int32_t GZIPIndex

  pointer to GZIP-object if Encoding=1
- int PostDotLength
- uint8_t RefStat

  Status of Referrer field (0=waitig , 1= 1=first parse failed, 2=second parse failed.
- char URLBuffer [256]

  Buffer to hold URL if it passes over multiple packets.
- char RefBuffer [80]

  Buffer to hold Referer.
- int64_t LastHeaderTime
- int64_t LastTailTime

5.10.1 Detailed Description

Storage class for HTTP-events. Every HTTP-flow or HTTPS-flow that can potentially trigger new traffi by embedded URL's has an object of this class.

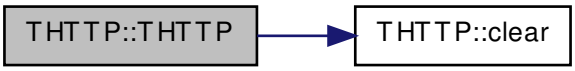
<

5.10.2 Constructor & Destructor Documentation

5.10.2.1 THTTP::THTTP (void)

The constructor. Resets all the data-members, including NextFlow

Here is the call graph for this function:



5.10.3 Member Function Documentation

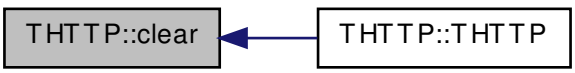
5.10.3.1 void THTTP::clear (void)

Clears all the data-members.

Returns

void

Here is the caller graph for this function:



5.10.3.2 void THTTP::print (char * *content*)

Prints a cause in readable format to stdout

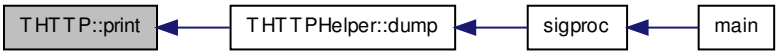
Returns

void

Parameters

**content*	pointer to content that must be printed. NULL is print to stdout

Here is the caller graph for this function:



5.10.4 Member Data Documentation

5.10.4.1 uint8_t THTTP::Chunked

0=unknown, 1=chunked

5.10.4.2 uint8_t THTTP::ContentType

0=unknown, 1=text

5.10.4.3 uint8_t THTTP::Encoding

0=unknown, 1=gzip

5.10.4.4 int32_t THTTP::FlowIndex

Index to Flow that created the cause.

5.10.4.5 int32_t THTTP::GZIPIndex

pointer to GZIP-object if Encoding=1

5.10.4.6 uint32_t THTTP::InByteCounter

Number of ingress bytes during a recieve phase.

5.10.4.7 int64_t THTTP::LastHeaderTime

5.10.4.8 int64_t THTTP::LastTailTime

5.10.4.9 uint32_t THTTP::OutByteCounter

Number of egress bytes during a send phase.

5.10.4.10 uint16_t THTTP::ParseMicroState

Micro State inside a SubState for parsing URL's.

5.10.4.11 uint8_t THTTP::ParseState

Main State of the FSM for parsing URL's.

5.10.4.12 uint8_t THTTP::ParseSubState

Sub State inside a Main State for parsing URL's.

5.10.4.13 int32_t THTTP::PayloadSize

Size of the HTTP(S) payload in the last received packet.

5.10.4.14 int THTTP::PostDotLength

-1=no dot detected, other is number of characters after the dot

5.10.4.15 char THTTP::RefBuffer[80]

Buffer to hold Referer.

5.10.4.16 uint8_t THTTP::RefStat

Status of Referrer field (0=waitig , 1= 1=first parse failed, 2=second parse failed.

5.10.4.17 uint8_t THTTP::Status

State of the HTTP-dialog.

5.10.4.18 int64_t THTTP::Time

Timestamp of the last HTTP-event.

5.10.4.19 uint32_t THTTP::TotalInByteCounter

Total number of egress payload bytes.

5.10.4.20 uint32_t THTTP::TotalOutByteCounter

Total number of ingress payload bytes.

5.10.4.21 char THTTP::URLBuffer[256]

Buffer to hold URL if it passes over multiple packets.

The documentation for this class was generated from the following files:

- src/HTTP.h
- src/HTTP.cpp

5.11 THTTPEvent Class Reference

`#include <EventCollector.h>`

Public Member Functions

- THTTPEvent (void)

  The constructor. Clears all the data-members.
- void clear (void)

  Clears all data-members.
- void print (char *content)

  Prints a cause in readable format to stdout.

Public Attributes

- int64_t TimeStamp

  TimeStamp: Time since 1970 in us.
- uint8_t EventCode

  EventNumber: 0=UNDEFINED, 1=HTTPDATARECEIVED, 2=HTTPDATAPUSH, 3=-CAUSE_HTTPWITHDRAWN.
- int32_t FlowIndex

  Index to flow that contains the event.

- uint8_t CauseCount

  number of times the event caused new flows
- int64_t DeltaT

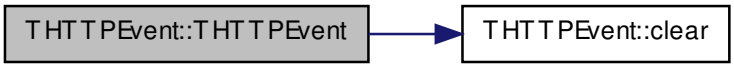
  Time contributed to openwindow.

5.11.1 Constructor & Destructor Documentation

5.11.1.1 THTTPEvent::THTTPEvent (void)

The constructor. Clears all the data-members.

Here is the call graph for this function:



5.11.2 Member Function Documentation

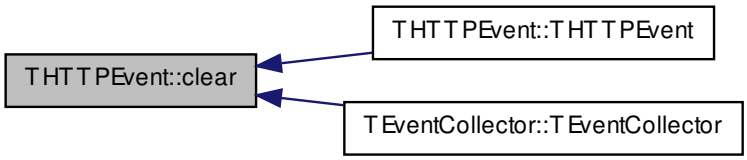
5.11.2.1 void THTTPEvent::clear (void)

Clears all data-members.

Returns

void

Here is the caller graph for this function:



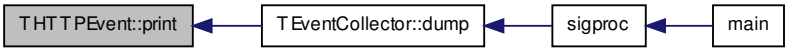
5.11.2.2 void THTTPEvent::print (char * *content*)

Prints a cause in readable format to stdout.

Returns

void

Here is the caller graph for this function:



5.11.3 Member Data Documentation

5.11.3.1 uint8_t THTTPEvent::CauseCount

number of times the event caused new flows

5.11.3.2 int64_t THTTPEvent::DeltaT

Time contributed to openwindow.

5.11.3.3 uint8_t THTTPEvent::EventCode

EventNumber: 0=UNDEFINED, 1=HTTPDATARECEIVED, 2=HTTPDATAPUSH, 3=CAUSE_HTTPWITHDRAWN.

5.11.3.4 int32_t THTTPEvent::FlowIndex

Index to flow that contains the event.

5.11.3.5 int64_t THTTPEvent::TimeStamp

TimeStamp: Time since 1970 in us.

The documentation for this class was generated from the following files:

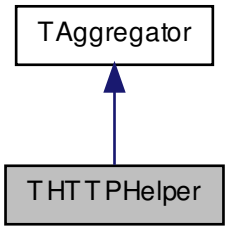
- src/EventCollector.h
- src/EventCollector.cpp

5.12 THTTPHelper Class Reference

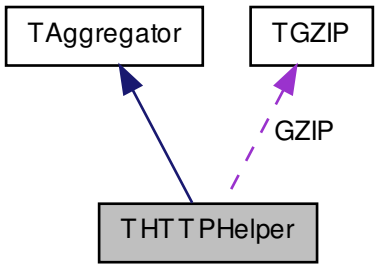
Class that aggregates HTTP traffic. It aggregates information that predicts the cause of new traffic flows THTTPHelper creates a static array of HTTP_BUFFER_SIZE (65536) HTTP-objects. The ClusterAggregator uses this class to check for potential HTTP-parent flows. TFlowAggregator updates the information of this class.

`#include <HTTPHelper.h>`

Inheritance diagram for THTTPHelper:



Collaboration diagram for THTTPHelper:



Public Member Functions

- THTTPHelper (void)
- uint8_t add (void)

- void dump (char *dm, int dest)

Private Member Functions

- int parse (void)
- int getRef (void)
- int stripRef (void)
- int checkURL (void)

Private Attributes

- TGZIP GZIP [TOTAL_GZIP]

  GZIP-objects.
- char DeChunkedContent [32768]

  intermediate buffer for chunk header removal
- uint32_t SuccessRefCounter
- uint32_t RefCounter
- uint32_t GetRequestCounter
- int32_t GZIPIndex

  index to most recent GZIP-buffer or -1 if empty

5.12.1 Detailed Description

Class that aggregates HTTP traffic. It aggregates information that predicts the cause of new traffic flows THTTPHelper creates a static array of HTTP_BUFFER_SIZE (65536) HTTP-objects. The ClusterAggregator uses this class to check for potential HTTP-parent flows. TFlowAggregator updates the information of this class.

<

5.12.2 Constructor & Destructor Documentation

5.12.2.1 THTTPHelper::THTTPHelper (void)

Constructor. During creation a fixed array of Causes is created for storage.

5.12.3 Member Function Documentation

5.12.3.1 uint8_t THTTPHelper::add (void) [virtual]

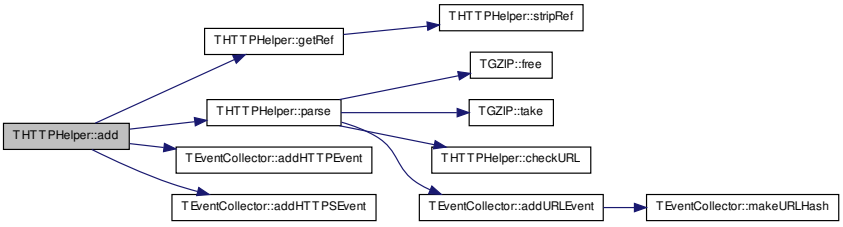
Adds HTTP-packets of new or existing flows to the aggregator

Returns

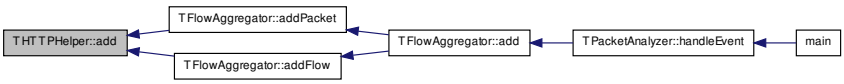
1=success: data updated in existing records, 2=succcess: new HTTP-record added, 0 = parse error, 10=Cannot create because HTTPBuffer Full

Reimplemented from TAggregator.

Here is the call graph for this function:

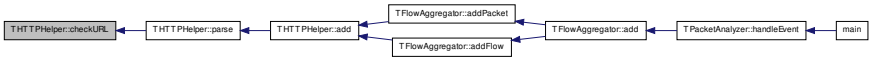


Here is the caller graph for this function:



5.12.3.2 int THTTPHelper::checkURL (void) [private]

Here is the caller graph for this function:



5.12.3.3 void THTTPHelper::dump (char * *dm*, int *dest*)

Dumps the content of all recent causes to stdout

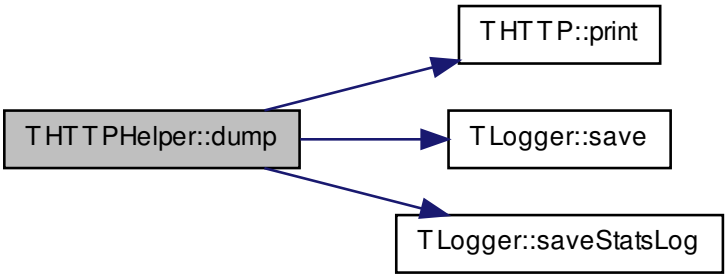
Returns

void

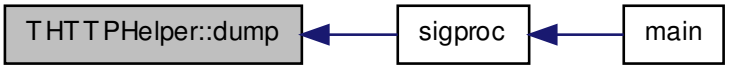
Parameters

∗dm	pointer to content that must be printed. NULL is print to stdout
dest	destination of the log: 0= to stdout, 1=to logfile

Here is the call graph for this function:



Here is the caller graph for this function:



5.12.3.4 int THTTPHelper::getRef (void) [private]

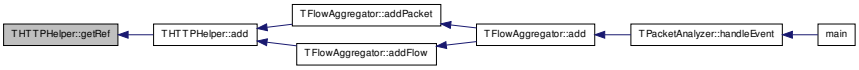
Parses HTTP get requests for referred domains.

Returns

1 = referred domain found, 2 = referred domain parsed but not found, 0 = no referred domain parsed, -1 = error

Here is the call graph for this function:

Here is the caller graph for this function:



5.12.3.5 int THTTPHelper::parse (void) [private]

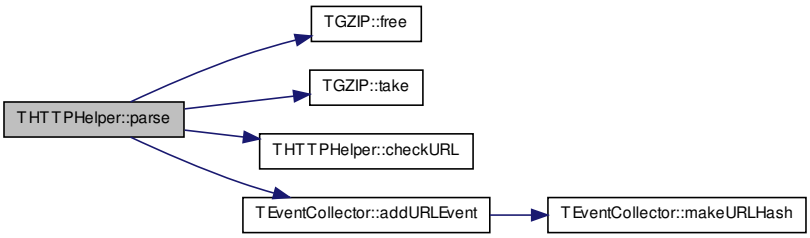
Parses HTTP. First the header and subsequently the body if the content-type=text. - Dechunking and decompression is included if necessary. The domain names of all url's, found in the body, are placed in the URL_EventList.

Returns

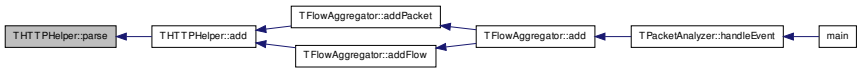
0 or positive=number of parsed URL's, negative=failure

Start first gzip

Here is the call graph for this function:



Here is the caller graph for this function:



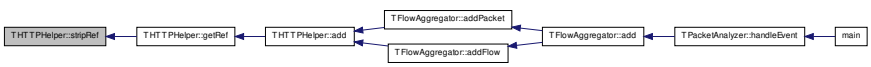
5.12.3.6 int THTTPHelper::stripRef (void) [private]

Strips the domain part of the referer from other stuff.

Returns

1 = referred domain successfully stripped

Here is the caller graph for this function:



5.12.4 Member Data Documentation

5.12.4.1 char THTTPHelper::DeChunkedContent[32768] [private]

intermediate buffer for chunk header removal

5.12.4.2 uint32_t THTTPHelper::GetRequestCounter [private]

5.12.4.3 TGZIP THTTPHelper::GZIP[TOTAL_GZIP] [private]

GZIP-objects.

5.12.4.4 int32_t THTTPHelper::GZIPIndex [private]

index to most recent GZIP-buffer or -1 if empty

5.12.4.5 uint32_t THTTPHelper::RefCounter [private]

5.12.4.6 uint32_t THTTPHelper::SuccessRefCounter [private]

The documentation for this class was generated from the following files:

- src/HTTPHelper.h
- src/HTTPHelper.cpp

5.13 THTTPSEvent Class Reference

`#include <EventCollector.h>`

Public Member Functions

- THTTPSEvent (void)

  The constructor. Clears all the data-members.
- void clear (void)

  Clears all data-members.
- void print (char *content)

  Prints a cause in readable format to stdout.

Public Attributes

- int64_t TimeStamp

  TimeStamp: Time since 1970 in us.
- uint8_t EventCode

EventNumber: 0=UNDEFINED, 1=HTTPDATARECEIVED, 2=HTTPDATAPUSH, 3=-CAUSE_HTTPWITHDRAWN.

- int32_t FlowIndex

  Index to flow that contains the event.

- uint8_t CauseCount

  number of times the event caused new flows

- int64_t DeltaT

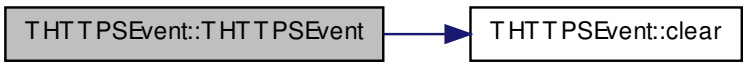
  Time contributed to openwindow.

5.13.1 Constructor & Destructor Documentation

5.13.1.1 THTTPSEvent::THTTPSEvent (void)

The constructor. Clears all the data-members.

Here is the call graph for this function:



5.13.2 Member Function Documentation

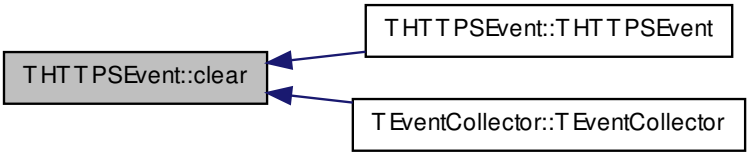
5.13.2.1 void THTTPSEvent::clear (void)

Clears all data-members.

Returns

void

Here is the caller graph for this function:



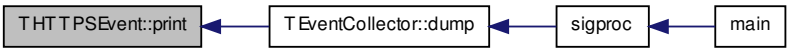
5.13.2.2 void THTTPSEvent::print (char * *content*)

Prints a cause in readable format to stdout.

Returns

void

Here is the caller graph for this function:



5.13.3 Member Data Documentation

5.13.3.1 uint8_t THTTPSEvent::CauseCount

number of times the event caused new flows

5.13.3.2 int64_t THTTPSEvent::DeltaT

Time contributed to openwindow.

5.13.3.3 uint8_t THTTPSEvent::EventCode

EventNumber: 0=UNDEFINED, 1=HTTPDATARECEIVED, 2=HTTPDATAPUSH, 3=CAUSE_HTTPWITHDRAWN.

5.13.3.4 int32_t THTTPSEvent::FlowIndex

Index to flow that contains the event.

5.13.3.5 int64_t THTTPSEvent::TimeStamp

TimeStamp: Time since 1970 in us.

The documentation for this class was generated from the following files:

- src/EventCollector.h
- src/EventCollector.cpp

5.14 TLogger Class Reference

Class that aggregates log messages in flat file format in RAM.

`#include <Logger.h>`

Public Member Functions

- TLogger (const char *name)
- int log (int sev, int fac, char *message)
- int log (int sev, int fac, int32_t flowindex, char *id, int64_t delay)
- int saveLog (void)
- int save (const char *ext, char *dm)
- int initStatsLog (void)
- int saveStatsLog (char *dm)

Private Attributes

- int LineCounter
- char Log [LOGSIZE]
- char * Buffer
- const char * FileName

5.14.1 Detailed Description

Class that aggregates log messages in flat file format in RAM.

< Finally it can save the log in a file. Every field is separated with a comma. Every record is separated with a newline. The format is:

Type1: Timestamp, Severity, Facility, message

Type2: Timestamp, Severity, Facility, message, ID, delay

Type3: Timestamp, Severity, Facility, message, index

5.14.2 Constructor & Destructor Documentation

5.14.2.1 TLogger::TLogger (const char * *name*)

Constructor. During creation a fixed array of Causes is created for storage.

Returns

void

Parameters

**name*	filename

5.14.3 Member Function Documentation

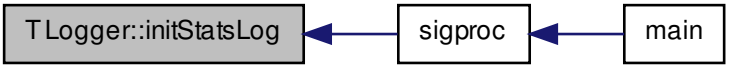
5.14.3.1 int TLogger::initStatsLog (void)

Initializes a new file for statistics

Returns

0=success, -1=error

Here is the caller graph for this function:



5.14.3.2 int TLogger::log (int *sev,* int *fac,* char * *message*)

Writes a generic log record

Returns

0=success, -1=error

Parameters

sev	Severity (0 t/m 7)
fac	Facility (number identifying which functional unit is the source
**message*	string with the message to write

Here is the caller graph for this function:

5.14.3.3 int TLogger::log (int *sev,* int *fac,* int32_t *flowindex,* char * *id,* int64_t *delay*)

Writes a new flow record

Returns

0=success, -1=error

Parameters

sev	Severity (0 t/m 7)
fac	Facility (number identifying which functional unit is the source
flowindex	Index of the new flow
**id*	IP or name of the new flow
delay	Time in us between new flow and cause

5.14.3.4 int TLogger::save (const char * *ext,* char * *dm*)

Saves a string in a file with specified extension

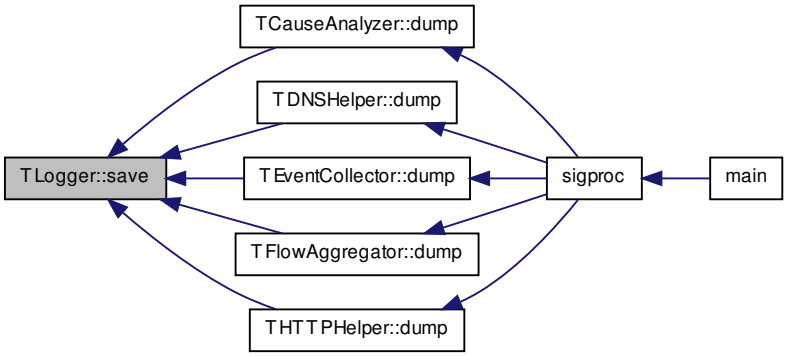
Returns

0=success, -1=error

Parameters

∗ext	pointer to extension of the filename, starting with a dot e.g. .flow, .dat
∗dm	pointer to complete string that must be saved

Here is the caller graph for this function:



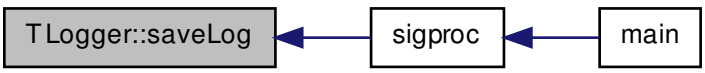
5.14.3.5 int TLogger::saveLog (void)

Saves the complete log in a file

Returns

0=success, -1=error

Here is the caller graph for this function:



5.14.3.6 int TLogger::saveStatsLog (char ∗ *dm*)

Adds a string with statistics to the statistics file

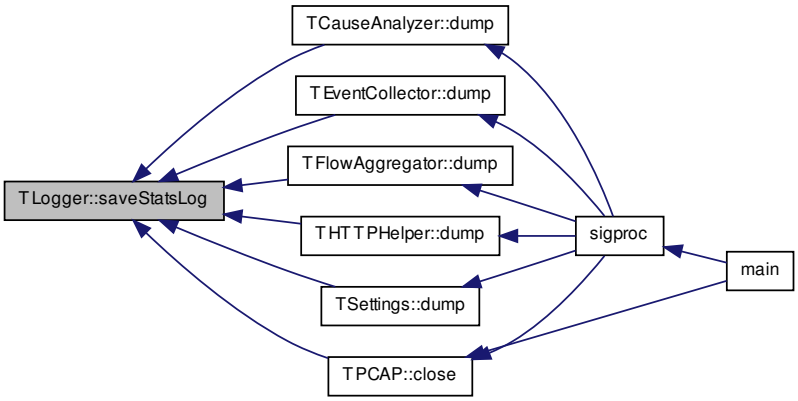
Returns

0=success, -1=error

Parameters

∗dm	pointer to complete string that must be saved

Here is the caller graph for this function:



5.14.4 Member Data Documentation

5.14.4.1 char∗ TLogger::Buffer [private]

Pointer in the storage

5.14.4.2 const char∗ TLogger::FileName [private]

File name after save to non-volatile memory

5.14.4.3 int TLogger::LineCounter [private]

Number of lines in the log

5.14.4.4 char TLogger::Log[LOGSIZE] [private]

RAM Storage for logging

The documentation for this class was generated from the following files:

- src/Logger.h
- src/Logger.cpp

5.15 TPacketAnalyzer Class Reference

Class for analyzing an arrived packet.

`#include <PacketAnalyzer.h>`

Public Member Functions

- TPacketAnalyzer ()
- void handleEvent (void)
- void dump (void)

Public Attributes

- int64_t Time

  Time of Packet-arrival in seconds sinze 1970.
- int32_t Length

  Number of available packet-bytes.
- uint8_t Protocol

  IP-Protocol-Field:1=ICMP, 6=TCP, 17=UDP, 255=IP-other.
- uint32_t SourceIP

  IP source sddress.
- uint32_t DestIP

  IP destination address.
- uint16_t SourcePort

  PORT if UDP/TCP else IDENTIFIER if ICMP or 0 if other L4-protocol.
- uint16_t DestPort

  PORT if UDP/TCP else SEQUENCE NUMBER if ICMP or 0 if other L4-protocol.
- uint8_t TCPFlag

  TCP FlagRegister bit0=FIN, bit1=SYN, bit2=RST, bit3=PSH bit4=ACK, bit 5=URG.
- uint32_t TCPSEQ

  TCP SEQ NR.
- uint32_t TCPACK

  TCP ACK NR.
- uint16_t Identification

  Identification (DNS)
- int32_t PayloadIndex

  Start of the Payload after L4 in Packet.
- int8_t TCPValid

  1=TCP_Packet expected, 0=Undefined, -1=TCP_Packet unexpected (written from - Flow)

Private Member Functions

- int8_t makePacket (void)

5.15.1 Detailed Description

Class for analyzing an arrived packet.

< If a packet arrives, this singleton will handle the event.

After analysis of L3 and L4, the FlowAnalyzer class is called for further processing.

5.15.2 Constructor & Destructor Documentation

5.15.2.1 TPacketAnalyzer::TPacketAnalyzer ( )

The constructor. Resets all the data-members

5.15.3 Member Function Documentation

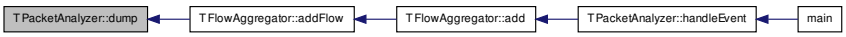
5.15.3.1 void TPacketAnalyzer::dump (void)

Dumps packet shortlist to stdout for debug purpose.

Returns

void

Here is the caller graph for this function:



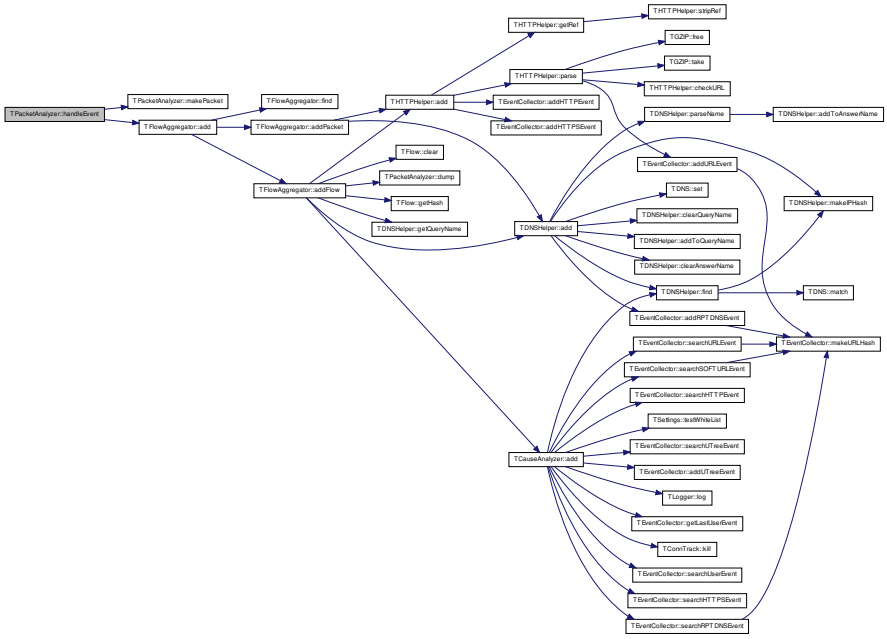
5.15.3.2 void TPacketAnalyzer::handleEvent (void)

Analyzes a packet and fills the public members with actual values of the arrived packet.

Returns

void

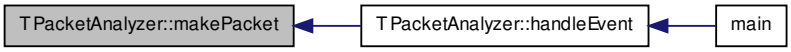
Here is the call graph for this function:



Here is the caller graph for this function:

5.15.3.3 int8_t TPacketAnalyzer::makePacket (void) [private]

Here is the caller graph for this function:



5.15.4 Member Data Documentation

5.15.4.1 uint32_t TPacketAnalyzer::DestIP

IP destination address.

5.15.4.2 uint16_t TPacketAnalyzer::DestPort

PORT if UDP/TCP else SEQUENCE NUMBER if ICMP or 0 if other L4-protocol.

5.15.4.3 uint16_t TPacketAnalyzer::Identification

Identification (DNS)

5.15.4.4 int32_t TPacketAnalyzer::Length

Number of available packet-bytes.

5.15.4.5 int32_t TPacketAnalyzer::PayloadIndex

Start of the Payload after L4 in Packet.

5.15.4.6 uint8_t TPacketAnalyzer::Protocol

IP-Protocol-Field:1=ICMP, 6=TCP, 17=UDP, 255=IP-other.

5.15.4.7 uint32_t TPacketAnalyzer::SourceIP

IP source sddress.

5.15.4.8 uint16_t TPacketAnalyzer::SourcePort

PORT if UDP/TCP else IDENTIFIER if ICMP or 0 if other L4-protocol.

5.15.4.9 uint32_t TPacketAnalyzer::TCPACK

TCP ACK NR.

5.15.4.10 uint8_t TPacketAnalyzer::TCPFlag

TCP FlagRegister bit0=FIN, bit1=SYN, bit2=RST, bit3=PSH bit4=ACK, bit 5=URG.

5.15.4.11 uint32_t TPacketAnalyzer::TCPSEQ

TCP SEQ NR.

5.15.4.12 int8_t TPacketAnalyzer::TCPValid

1=TCP_Packet expected, 0=Undefined, -1=TCP_Packet unexpected (written from Flow)

5.15.4.13 int64_t TPacketAnalyzer::Time

Time of Packet-arrival in seconds sinze 1970.

The documentation for this class was generated from the following files:

- src/PacketAnalyzer.h
- src/PacketAnalyzer.cpp

5.16 TPCAP Class Reference

The "singleton" object of TPCAP is a wrapper around the libpcap library. It can read from a device or a file and return a pointer to a received packet.

```
#include <PCAP.h>
```

Public Member Functions

- TPCAP (void)
- void openNIC (char *dev)
- void openFile (char *name)
- void openDevice (char *device, int live)
- int8_t applyFilter (char *filter)
- int8_t getPacketEvent (void)
- void close (char *dm, int dest)

Public Attributes

- uint64_t TimeStamp

  Time of Packet-arrival in useconds sinze 1970.
- uint32_t Length

  Number of available packet-bytes.
- const uint8_t * Packet

  Pointer the datastructure of the received Packet.

Private Attributes

- pcap_t * Handle

  Handle to openened NIC or file.
- char * DeviceName

  Name of the Device or File, like "eth0".
- char * Filter

  Packet filter e.g. "not port 22" , default is "ip".

5.16.1 Detailed Description

The "singleton" object of TPCAP is a wrapper around the libpcap library. It can read from a device or a file and return a pointer to a received packet.

<

5.16.2 Constructor & Destructor Documentation

5.16.2.1 TPCAP::TPCAP (void)

The constructor.

5.16.3 Member Function Documentation

5.16.3.1 int8_t TPCAP::applyFilter (char * *filter*)

Applies a libpcap preprocessor filter.

Returns

0=ok

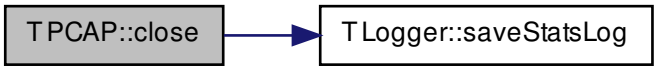
Parameters

filter	The pointer to a string with the filter equation e.g. "IP".

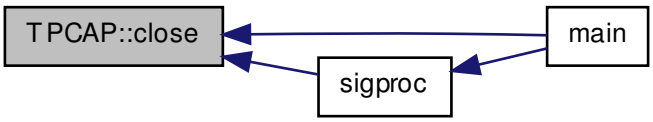
Here is the caller graph for this function:

5.16.3.2 void TPCAP::close (char * *dm*, int *dest*)

Here is the call graph for this function:



Here is the caller graph for this function:



5.16.3.3 int8_t TPCAP::getPacketEvent (void)

Event check with immediate return, for use in a event loop. ;

Returns

0=no event, 1=packet received, -1=error, -2=EOF

Here is the caller graph for this function:

5.16.3.4 void TPCAP::openDevice (char * *device*, int *live*)

Opens a libcap-file or network device with recorded traffic.

If the name ends with a number an interface is assumed.

In other cases a file is assumed.

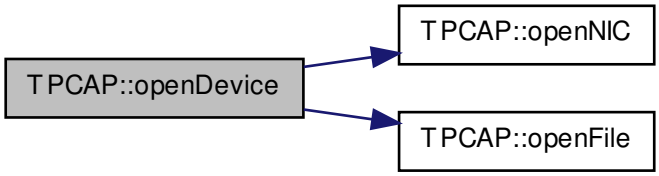
Returns

void

Parameters

live	0=from file, 1=from NIC
device	The pointer to a string with the name of the interface or file.

Here is the call graph for this function:



Here is the caller graph for this function:

5.16.3.5 void TPCAP::openFile (char * *name*)

Opens a libpcap file with recorded traffic e.g. "trace.pcap".

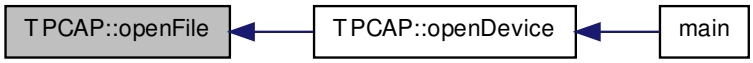
Returns

void

Parameters

name	The pointer to a string with the name of the file.

Here is the caller graph for this function:



5.16.3.6 void TPCAP::openNIC (char * *dev*)

Opens a network device (interface) e.g. "eth0".

Returns

void

Parameters

dev	The pointer to a string with the name of the device.

Here is the caller graph for this function:

5.16.4 Member Data Documentation

5.16.4.1 char* TPCAP::DeviceName [private]

Name of the Device or File, like "eth0".

5.16.4.2 char* TPCAP::Filter [private]

Packet filter e.g. "not port 22" , default is "ip".

5.16.4.3 pcap_t* TPCAP::Handle [private]

Handle to openened NIC or file.

5.16.4.4 uint32_t TPCAP::Length

Number of available packet-bytes.

5.16.4.5 const uint8_t* TPCAP::Packet

Pointer the datastructure of the received Packet.

5.16.4.6 uint64_t TPCAP::TimeStamp

Time of Packet-arrival in useconds sinze 1970.

The documentation for this class was generated from the following files:

- src/PCAP.h
- src/PCAP.cpp

5.17 TRPTDNSEvent Class Reference

`#include <EventCollector.h>`

Public Member Functions

- TRPTDNSEvent (void)

  The constructor. Clears all the data-members.
- void clear (void)

  Clears all data-members.
- void print (char *content)

  Prints a cause in readable format to stdout.

Public Attributes

- int64_t TimeStamp

  TimeStamp: Time since 1970 in us.
- char Name [64]

  domain name in lowercase ending with '/0'(<64)
- int32_t FlowIndex

  Index to flow that contains the event.
- uint8_t CauseCount

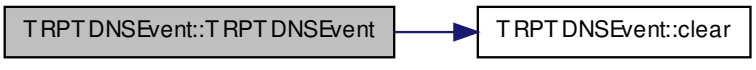
  number of times the event caused new flows

5.17.1 Constructor & Destructor Documentation

5.17.1.1 TRPTDNSEvent::TRPTDNSEvent (void)

The constructor. Clears all the data-members.

Here is the call graph for this function:



5.17.2 Member Function Documentation

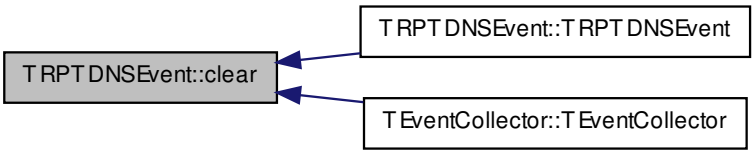
5.17.2.1 void TRPTDNSEvent::clear (void)

Clears all data-members.

Returns

void

Here is the caller graph for this function:



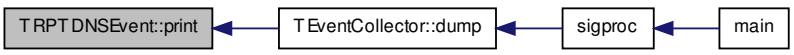
5.17.2.2 void TRPTDNSEvent::print (char * *content*)

Prints a cause in readable format to stdout.

Returns

void

Here is the caller graph for this function:



5.17.3 Member Data Documentation

5.17.3.1 uint8_t TRPTDNSEvent::CauseCount

number of times the event caused new flows

5.17.3.2 int32_t TRPTDNSEvent::FlowIndex

Index to flow that contains the event.

5.17.3.3 char TRPTDNSEvent::Name[64]

domain name in lowercase ending with '/0'(<64)

5.17.3.4 int64_t TRPTDNSEvent::TimeStamp

TimeStamp: Time since 1970 in us.

The documentation for this class was generated from the following files:

- src/EventCollector.h
- src/EventCollector.cpp

5.18 TSettinggs Class Reference

Class for parsing and distributing the settings of the application.

5.18.1 Detailed Description

Class for parsing and distributing the settings of the application.

<

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

- src/Settings.h

5.19 TSettings Class Reference

```
#include <Settings.h>
```

Public Member Functions

- TSettings (const char *name)
- int parseFile (void)
- int testWhiteList (char *id)
- int testWhiteList (uint32_t ip)
- void dump (char *content, int dest)

Public Attributes

- int64_t DefinedValues [SETTINGSSIZE]
- const char * DefinedSettings [SETTINGSSIZE]
- uint32_t WhiteIPLow [256]
- uint32_t WhiteIPHigh [256]
- char WhiteName [256][256]

Private Attributes

- char Name [100]
- int WhiteIPTotal
- int WhiteNameTotal

5.19.1 Constructor & Destructor Documentation

5.19.1.1 TSettings::TSettings (const char * *name*)

The constructor. Opens file

Parameters

name	filename with the settings

Here is the call graph for this function:

5.19.2 Member Function Documentation

5.19.2.1 void TSettings::dump (char * *content*, int *dest*)

Dumps the settings to stats file

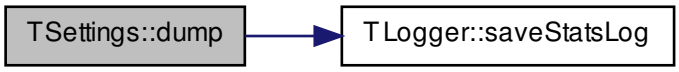
Returns

void

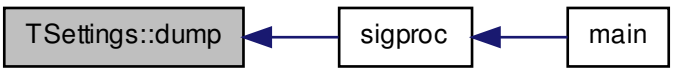
Parameters

∗content	pointer to content that must be printed. NULL is print to stdout
dest	destination of the log: 0= to stdout, 1=to logfile

Here is the call graph for this function:



Here is the caller graph for this function:



5.19.2.2 int TSettings::parseFile (void)

Parses the settings file.

Returns

number of recognized lines or -1 if error

Here is the caller graph for this function:

5.19.2.3 int TSettings::testWhiteList (char * *id*)

Tests if ID is in the whitelist

Returns

0=not in whitelist 1=in whitelist

Here is the caller graph for this function:

5.19.2.4 int TSettings::testWhiteList (uint32_t *ip*)

Tests if ip is in the whitelist

Returns

0=not in whitelist 1=in whitelist

5.19.3 Member Data Documentation

5.19.3.1 const char* TSettings::DefinedSettings[SETTINGSSIZE]

5.19.3.2 int64_t TSettings::DefinedValues[SETTINGSSIZE]

5.19.3.3 char TSettings::Name[100] [private]

Stores the Filename

5.19.3.4 uint32_t TSettings::WhiteIPHigh[256]

5.19.3.5 uint32_t TSettings::WhiteIPLow[256]

5.19.3.6 int TSettings::WhiteIPTotal [private]

Total Number of Whitelist IP-ranges

5.19.3.7 char TSettings::WhiteName[256][256]

5.19.3.8 int TSettings::WhiteNameTotal [private]

Total Number of Whitelist Names

The documentation for this class was generated from the following files:

- src/Settings.h
- src/Settings.cpp

5.20 TTree Class Reference

Storage class for a tree of network flows with a causal relationship.

`#include <Tree.h>`

Public Member Functions

- TTree (void)
- void clear (void)
- void print (char *content)

Public Attributes

- int64_t StartTime

  Timestamp of the root flow.
- int64_t StopTime

  Timestamp of any last activity of a flow in the tree.
- int32_t RootFlow

  ID of the root flow.
- int8_t RootCause

  Cause of the root flow.
- uint16_t NumberOfFlows

  Total number of flows (root + all offspring), 0 = tree empty.
- uint16_t MaxDepth

  Longest line of descendants.
- uint8_t DNSOther

  Default zero. Will be 1 if there if at least one other flow than a resolver flow.
- char ID [256]

  ID of the root Flow.

5.20.1 Detailed Description

Storage class for a tree of network flows with a causal relationship.

<

5.20.2 Constructor & Destructor Documentation

5.20.2.1 TTree::TTree (void)

The constructor. Resets all the data-members, including NextFlow

Here is the call graph for this function:

5.20.3 Member Function Documentation

5.20.3.1 void TTree::clear (void)

Clears all the data-members.

Returns

void

Here is the caller graph for this function:

5.20.3.2 void TTree::print (char * *content*)

Prints a tree in readable format to stdout

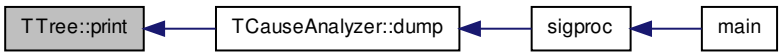
Returns

void

Parameters

∗content	pointer to content that must be printed. NULL is print to stdout

Here is the caller graph for this function:



5.20.4 Member Data Documentation

5.20.4.1 uint8_t TTree::DNSOther

Default zero. Will be 1 if there if at least one other flow than a resolver flow.

5.20.4.2 char TTree::ID[256]

ID of the root Flow.

5.20.4.3 uint16_t TTree::MaxDepth

Longest line of descendants.

5.20.4.4 uint16_t TTree::NumberOfFlows

Total number of flows (root + all offspring), 0 = tree empty.

5.20.4.5 int8_t TTree::RootCause

Cause of the root flow.

5.20.4.6 int32_t TTree::RootFlow

ID of the root flow.

5.20.4.7 int64_t TTree::StartTime

Timestamp of the root flow.

5.20.4.8 int64_t TTree::StopTime

Timestamp of any last activity of a flow in the tree.

The documentation for this class was generated from the following files:

- src/Tree.h
- src/Tree.cpp

5.21 TUDPUA Class Reference

One object of this class (class should be used as singleton) recieves and filters user events, received from UDP with a special agent. Received results can be polled in a Event-loop.

`#include <UDPUA.h>`

Public Member Functions

- TUDPUA (void)
- uint8_t open (void)
- void close (void)
- uint8_t getEvent (void)
- int processEvent (void)
- void dump (void)

Public Attributes

- char Message [256]

  Received raw message, only used in an opened socket after succesful getEvent.
- char Event [256]

  String, describing the event (LMB=Left Mouse Button, Enter, F5).
- char Process [256]

  String, describing the generating process that generates the user event.
- int64_t TimeStamp

  TimeStamp: Time since 1970 in us.
- uint8_t EventCode

  EventNumber: 0=none, 1=F5, 2=LMB, 3=Enter.

Private Attributes

- int SocketDescriptor

  Socket descriptor for UDP-communication, listening on port UDP_Port.

5.21.1 Detailed Description

One object of this class (class should be used as singleton) recieves and filters user events, received from UDP with a special agent. Received results can be polled in a Event-loop.

5.21.2 Constructor & Destructor Documentation

5.21.2.1 TUDPUA::TUDPUA (void)

The constructor that defines the UDP connection.

Clears the strings and EventCode;

5.21.3 Member Function Documentation

5.21.3.1 void TUDPUA::close (void)

Closes the UDP-socket.

Returns

void

5.21.3.2 void TUDPUA::dump (void)

Prints User Events with Timestamp, Event and Process.

Returns

void

5.21.3.3 uint8_t TUDPUA::getEvent (void)

Reads UDP received events from an opened socket

Returns

0 if no event and something or 1 if something is received.

5.21.3.4 uint8_t TUDPUA::open (void)

Binds the UDP-socket.

Returns

0 = ok 1+ = problem

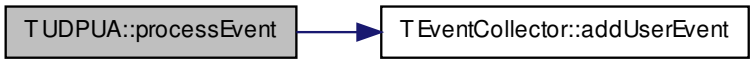
5.21.3.5 int TUDPUA::processEvent (void)

Reads potential UDP received events from PCAP-packet. Remember that it will not use an open socket. Thi can result in ICMP-responses in the pcap-trace.

Returns

0=format ok but no event, 1=format ok and event, -1 = format not ok

Here is the call graph for this function:



Here is the caller graph for this function:

5.21.4 Member Data Documentation

5.21.4.1 char TUDPUA::Event[256]

String, describing the event (LMB=Left Mouse Button, Enter, F5).

5.21.4.2 uint8_t TUDPUA::EventCode

EventNumber: 0=none, 1=F5, 2=LMB, 3=Enter.

5.21.4.3 char TUDPUA::Message[256]

Received raw message, only used in an opened socket after succesful getEvent.

5.21.4.4 char TUDPUA::Process[256]

String, describing the generating process that generates the user event.

5.21.4.5 int TUDPUA::SocketDescriptor [private]

Socket descriptor for UDP-communication, listening on port UDP_Port.

5.21.4.6 int64_t TUDPUA::TimeStamp

TimeStamp: Time since 1970 in us.

The documentation for this class was generated from the following files:

- src/UDPUA.h
- src/UDPUA.cpp

5.22 TURLEvent Class Reference

`#include <EventCollector.h>`

Public Member Functions

- TURLEvent (void)

  The constructor. Clears all the data-members.
- void clear (void)

  Clears all data-members.
- void print (char *content)

  Prints a cause in readable format to stdout.

Public Attributes

- int64_t TimeStamp

  TimeStamp: Time since 1970 in us.
- char URL [64]

  domain name in lowercase ending with '/0'(<64)
- char SUBURL [64]

  2nd level domain name substring for softcompare
- int32_t FlowIndex

  Index to flow that contains the event.
- uint8_t CauseCount

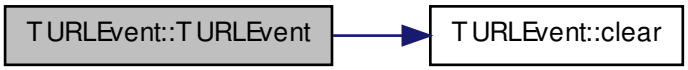
  number of times the event caused new flows

5.22.1 Constructor & Destructor Documentation

5.22.1.1 TURLEvent::TURLEvent (void)

The constructor. Clears all the data-members.

Here is the call graph for this function:



5.22.2 Member Function Documentation

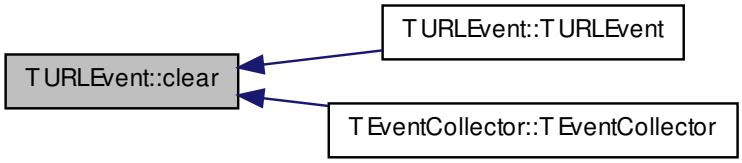
5.22.2.1 void TURLEvent::clear (void)

Clears all data-members.

Returns

void

Here is the caller graph for this function:



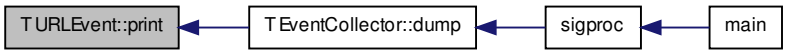
5.22.2.2 void TURLEvent::print (char * *content*)

Prints a cause in readable format to stdout.

Returns

void

Here is the caller graph for this function:



5.22.3 Member Data Documentation

5.22.3.1 uint8_t TURLEvent::CauseCount

number of times the event caused new flows

5.22.3.2 int32_t TURLEvent::FlowIndex

Index to flow that contains the event.

5.22.3.3 char TURLEvent::SUBURL[64]

2nd level domain name substring for softcompare

5.22.3.4 int64_t TURLEvent::TimeStamp

TimeStamp: Time since 1970 in us.

5.22.3.5 char TURLEvent::URL[64]

domain name in lowercase ending with '/0'(<64)

The documentation for this class was generated from the following files:

- src/EventCollector.h
- src/EventCollector.cpp

5.23 TUserEvent Class Reference

`#include <EventCollector.h>`

Public Member Functions

- TUserEvent (void)

  The constructor. Clears all the data-members.
- void clear (void)

  Clears all data-members.
- void print (char ∗content)

  Prints a cause in readable format to stdout.

Public Attributes

- int64_t TimeStamp

  TimeStamp: Time since 1970 in us.
- uint8_t EventCode

  EventNumber: 0=none, 1=F5, 2=LMB, 3=Enter.
- char Process [64]

  String, describing the generating process that generates the user event in lowercase ending with '/0'(<64)
- uint8_t CauseCount

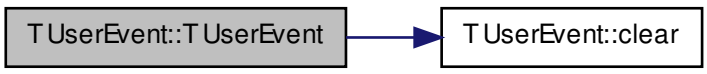
  number of times the event caused new flows

5.23.1 Constructor & Destructor Documentation

5.23.1.1 TUserEvent::TUserEvent (void)

The constructor. Clears all the data-members.

Here is the call graph for this function:



5.23.2 Member Function Documentation

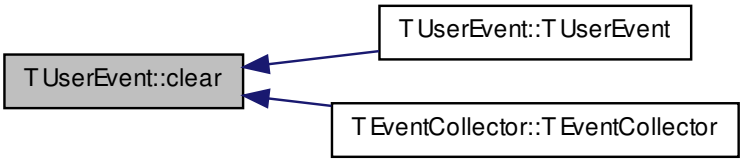
5.23.2.1 void TUserEvent::clear (void)

Clears all data-members.

Returns

void

Here is the caller graph for this function:



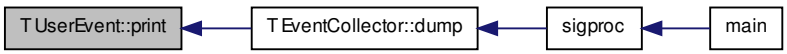
5.23.2.2 void TUserEvent::print (char * *content*)

Prints a cause in readable format to stdout.

Returns

void

Here is the caller graph for this function:



5.23.3 Member Data Documentation

5.23.3.1 uint8_t TUserEvent::CauseCount

number of times the event caused new flows

5.23.3.2 uint8_t TUserEvent::EventCode

EventNumber: 0=none, 1=F5, 2=LMB, 3=Enter.

5.23.3.3 char TUserEvent::Process[64]

String, describing the generating process that generates the user event in lowercase ending with '/0'(<64)

5.23.3.4 int64_t TUserEvent::TimeStamp

TimeStamp: Time since 1970 in us.

The documentation for this class was generated from the following files:

- src/EventCollector.h
- src/EventCollector.cpp

5.24 TUTreeEvent Class Reference

`#include <EventCollector.h>`

Public Member Functions

- TUTreeEvent (void)

  The constructor. Clears all the data-members.
- void clear (void)

  Clears all data-members.
- void print (char *content)

  Prints a cause in readable format to stdout.

Public Attributes

- int64_t TimeStamp

  TimeStamp: Time since 1970 in us.
- int32_t TreeIndex

  Index to tree that contains the event.
- uint8_t CauseCount

  number of times the event caused new flows
- int64_t DeltaT

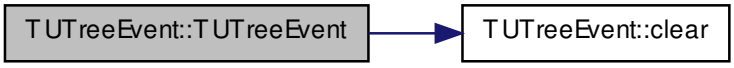
  Time contributed to openwindow.

5.24.1 Constructor & Destructor Documentation

5.24.1.1 TUTreeEvent::TUTreeEvent (void)

The constructor. Clears all the data-members.

Here is the call graph for this function:



5.24.2 Member Function Documentation

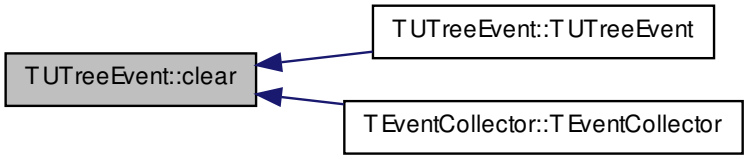
5.24.2.1 void TUTreeEvent::clear (void)

Clears all data-members.

Returns

void

Here is the caller graph for this function:

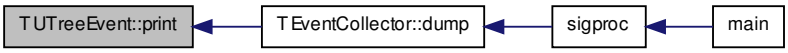


5.24.2.2 void TUTreeEvent::print (char * *content*)

Prints a cause in readable format to stdout.

Returns

Here is the caller graph for this function:



5.24.3 Member Data Documentation

5.24.3.1 uint8_t TUTreeEvent::CauseCount

number of times the event caused new flows

5.24.3.2 int64_t TUTreeEvent::DeltaT

Time contributed to openwindow.

5.24.3.3 int64_t TUTreeEvent::TimeStamp

TimeStamp: Time since 1970 in us.

5.24.3.4 int32_t TUTreeEvent::TreeIndex

Index to tree that contains the event.

The documentation for this class was generated from the following files:

- src/EventCollector.h
- src/EventCollector.cpp

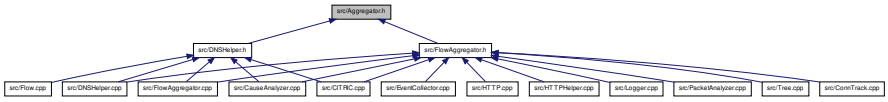
Chapter 6

File Documentation

6.1 src/Aggregator.h File Reference

This file contains the base class TAggregator.

This graph shows which files directly or indirectly include this file:



Classes

- class TAggregator

  Baseclass for aggregators. An aggragator is an object dat aggregates multiple data in a combined datastructure or element of a small size. Examples of aggregating elements are Flows, Tree, and DNS-records.

6.1.1 Detailed Description

This file contains the base class TAggregator.

Copyright

Copyright 2012-2013 Delft University of Technology and The Hague University of Applied Sciences. License: LGPL 3+

Author

Pieter Burghouwt

Version

Revision 2.0

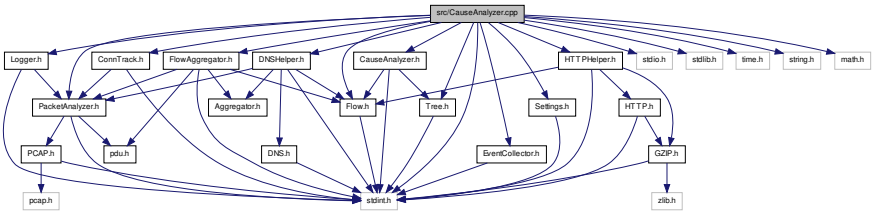
Date

Saturday, October 20, 2012

6.2 src/CauseAnalyzer.cpp File Reference

This file contains the operators of the TCauseAnalyzer class.

`#include <stdint.h> #include <stdio.h> #include <stdlib.h> #include <time.h> #include <string.h> #include <math.h> #include "FlowAggregator.h" #include "Flow.h" #include "DNSHelper.h" #include "Tree.h" #include "CauseAnalyzer.h" #include "HTTPHelper.h" #include "PacketAnalyzer.h" × #include "EventCollector.h" #include "Logger.h" #include "Settings.h" #include "ConnTrack.h"` Include dependency graph for CauseAnalyzer.cpp:



Variables

- TFlowAggregator ∗ FlowAggregator
- TDNSHelper ∗ DNSHelper
- TPCAP ∗ PCAP
- TPacketAnalyzer ∗ PacketAnalyzer
- TFlow Flow [FLOWBUFFERSIZE]
- TDNS DNS [DNSBUFFERSIZE]
- THTTP HTTP [HTTP_BUFFER_SIZE]
- THTTPHelper ∗ HTTPHelper
- TEventCollector ∗ EventCollector
- TLogger ∗ Logger

- TSettings * Settings
- TConnTrack * ConnTrack
- int ENDPROG
- TTree Tree [TREE_BUFFER_SIZE]

6.2.1 Detailed Description

This file contains the operators of the TCauseAnalyzer class.

Author

Pieter Burghouwt

Version

Revision 2.0

Date

Tuesday, March, 12, 2013

6.2.2 Variable Documentation

6.2.2.1 TConnTrack* ConnTrack

6.2.2.2 TDNS DNS[DNSBUFFERSIZE]

6.2.2.3 TDNSHelper* DNSHelper

6.2.2.4 int ENDPROG

6.2.2.5 TEventCollector* EventCollector

6.2.2.6 TFlow Flow[FLOWBUFFERSIZE]

6.2.2.7 TFlowAggregator* FlowAggregator

6.2.2.8 THTTP HTTP[HTTP_BUFFER_SIZE]

6.2.2.9 THTTPHelper* HTTPHelper

6.2.2.10 TLogger* Logger

6.2.2.11 TPacketAnalyzer* PacketAnalyzer

6.2.2.12 TPCAP* PCAP

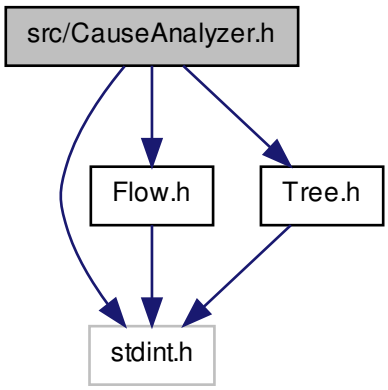
6.2.2.13 TSettings* Settings

6.2.2.14 TTree Tree[TREE_BUFFER_SIZE]

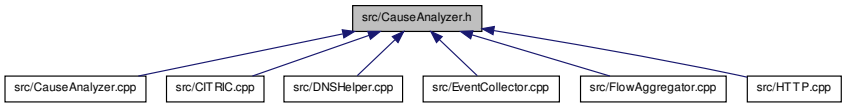
6.3 src/CauseAnalyzer.h File Reference

This file contains the prototypes of the TCauseAnalyzer class.

`#include <stdint.h> #include "Flow.h" #include "Tree.h"` ×
Include dependency graph for CauseAnalyzer.h:



This graph shows which files directly or indirectly include this file:



Classes

- class TCauseAnalyzer

  Class that manages causal trees and sorts new flows in it.

Defines

- #define TREE_BUFFER_SIZE 65536

6.3.1 Detailed Description

This file contains the prototypes of the TCauseAnalyzer class.

Copyright

Copyright 2012-2013 Delft University of Technology and The Hague University of Applied Sciences. License: LGPL 3+

Author

Pieter Burghouwt

Version

Revision 2.0

Date

Tuesday, March, 12, 2013

6.3.2 Define Documentation

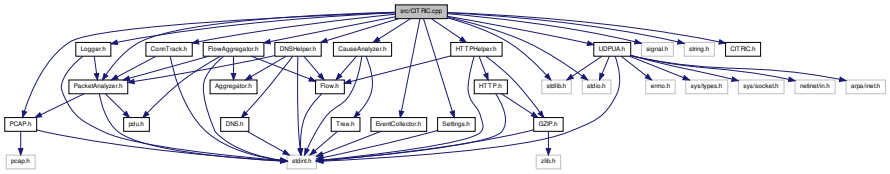
6.3.2.1 #define TREE_BUFFER_SIZE 65536

6.4 src/CITRIC.cpp File Reference

This file contains the main file of CITRIC. CITRIC is an experimental causal detector that analyzes PCAP-data from a device.It aggregates packets in bidirectional flows and it organizes the flows in causal trees. A causal tree is a group of flows with a causal relationship. Causal relationship is determined by time-intervals and in some cases content. This is an experimental version for research.

```
#include <stdlib.h> #include <stdio.h> #include <signal.-
h> #include <string.h> #include "PCAP.h" #include "Packet-
Analyzer.h"   #include "FlowAggregator.h"   #include "DN-
SHelper.h"   #include "CauseAnalyzer.h"   #include "HTTP-
Helper.h"  #include "EventCollector.h"  #include "UDPUA.-
h"  #include "CITRIC.h"  #include "Logger.h"  #include "-
```

`Settings.h" #include "ConnTrack.h"` Include dependency graph for - CITRIC.cpp:



Functions

- int main (int argc, char ∗argv[])
- void sigproc (int signum)

Variables

- TPCAP ∗ PCAP
- TPacketAnalyzer ∗ PacketAnalyzer
- TFlowAggregator ∗ FlowAggregator
- TDNSHelper ∗ DNSHelper
- TCauseAnalyzer ∗ CauseAnalyzer
- THTTPHelper ∗ HTTPHelper
- TUDPUA ∗ UDPUA
- TEventCollector ∗ EventCollector
- TLogger ∗ Logger
- TSettings ∗ Settings
- TConnTrack ∗ ConnTrack
- char DumpMessage [100000000]
- int ENDPROG

6.4.1 Detailed Description

This file contains the main file of CITRIC. CITRIC is an experimental causal detector that analyzes PCAP-data from a device.It aggregates packets in bidirectional flows and it organizes the flows in causal trees. A causal tree is a group of flows with a causal relationship. Causal relationship is determined by time-intervals and in some cases content. This is an experimental version for research.

Copyright

Author

Pieter Burghouwt

Version

Revision 2.0

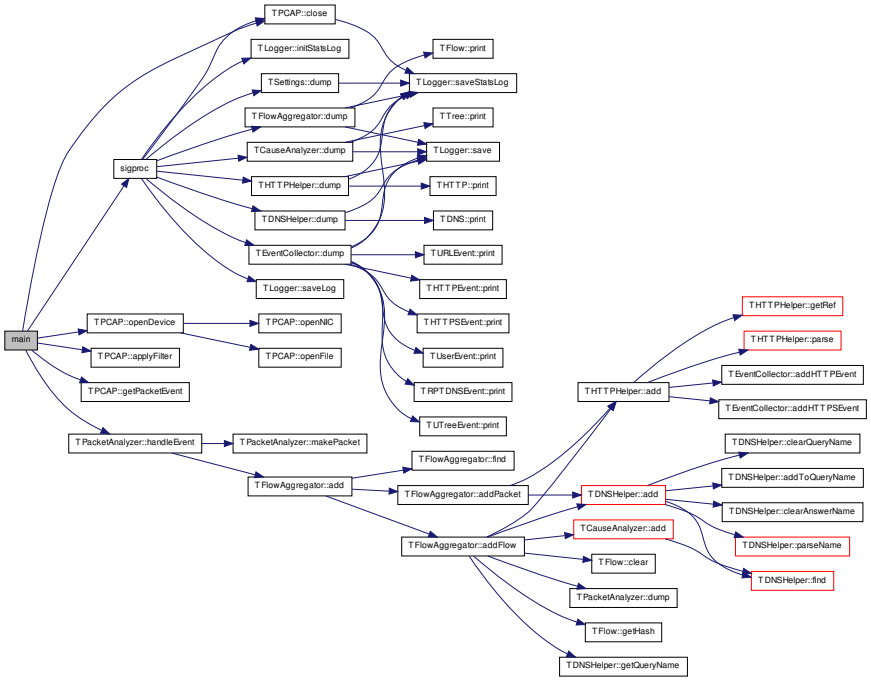
Date

Friday, November 16, 2012

6.4.2 Function Documentation

6.4.2.1 int main (int *argc*, char * *argv[]*)

Here is the call graph for this function:



6.4.2.2 void sigproc (int *signum*)

A handler for OS-signals, to catch the CTRL+C

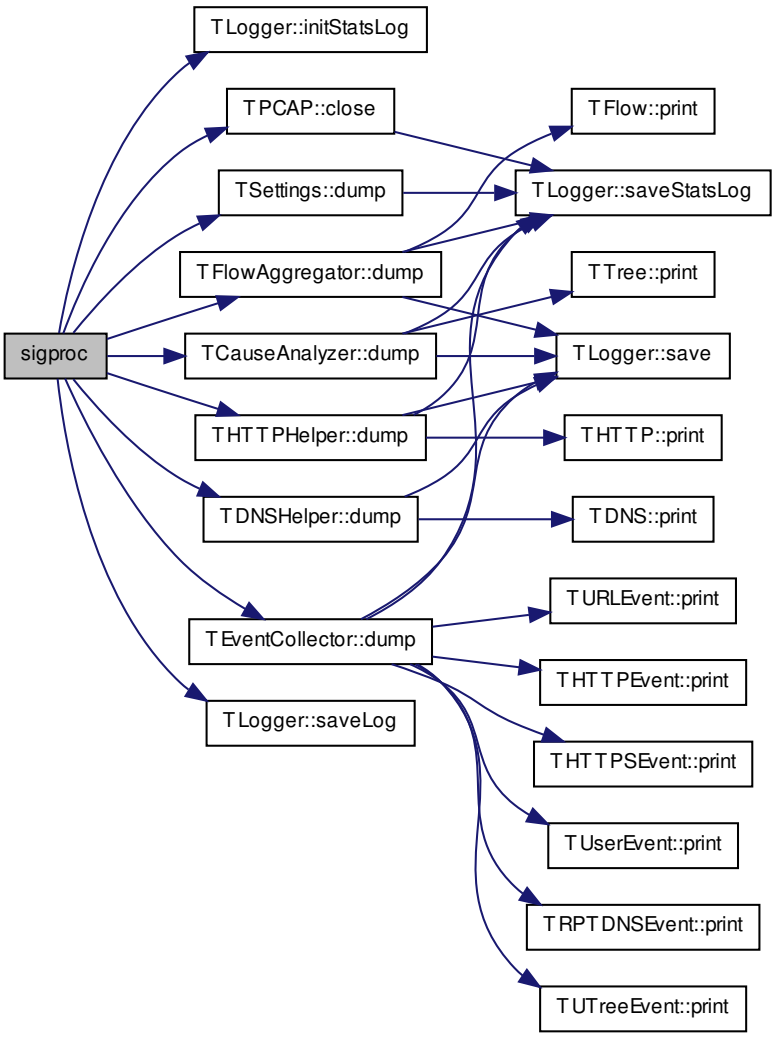
Returns

void

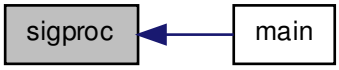
Parameters

signum	=2 (SIGINT) that identifies to CTRL+C

Here is the call graph for this function:



Here is the caller graph for this function:



6.4.3 Variable Documentation

6.4.3.1 TCauseAnalyzer* CauseAnalyzer

6.4.3.2 TConnTrack* ConnTrack

6.4.3.3 TDNSHelper* DNSHelper

6.4.3.4 char DumpMessage[100000000]

6.4.3.5 int ENDPROG

6.4.3.6 TEventCollector* EventCollector

6.4.3.7 TFlowAggregator* FlowAggregator

6.4.3.8 THTTPHelper* HTTPHelper

6.4.3.9 TLogger* Logger

6.4.3.10 TPacketAnalyzer* PacketAnalyzer

6.4.3.11 TPCAP* PCAP

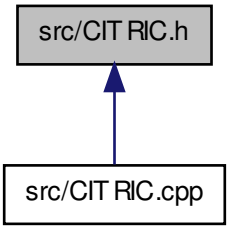
6.4.3.12 TSettings* Settings

6.4.3.13 TUDPUA* UDPUA

6.5 src/CITRIC.h File Reference

This file contains the prototypes of the CITRIC class.

This graph shows which files directly or indirectly include this file:



Functions

- void sigproc (int signum)

6.5.1 Detailed Description

This file contains the prototypes of the CITRIC class.

Copyright

Copyright 2012-2013 Delft University of Technology and The Hague University of Applied Sciences. License: LGPL 3+

Author

Pieter Burghouwt

Version

Revision 2.0

Date

Friday, November 16, 2012

6.5.2 Function Documentation

6.5.2.1 void sigproc (int *signum*)

A handler for OS-signals, to catch the CTRL+C

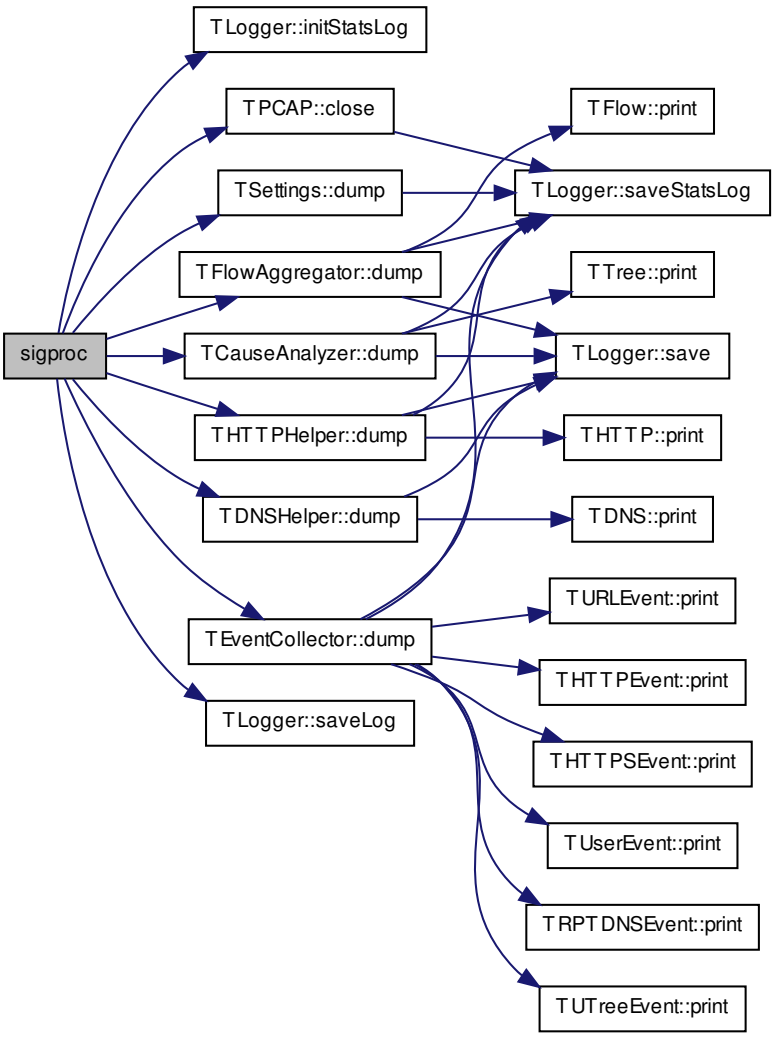
Returns

void

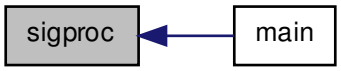
Parameters

signum	=2 (SIGINT) that identifies to CTRL+C

Here is the call graph for this function:



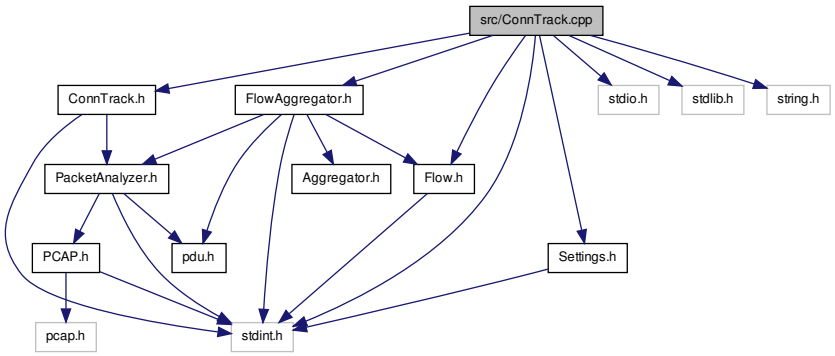
Here is the caller graph for this function:



6.6 src/ConnTrack.cpp File Reference

This file contains the operators of the TConnTrack class.

`#include <stdint.h> #include <stdio.h> #include <stdlib.h> #include <string.h> #include "Flow.h" #include "FlowAggregator.h" #include "ConnTrack.h" #include "Settings.h"` Include dependency graph for ConnTrack.cpp:



Variables

- TSettings * Settings
- TFlow Flow [FLOWBUFFERSIZE]

6.6.1 Detailed Description

This file contains the operators of the TConnTrack class.

Copyright

Copyright 2012-2013 Delft University of Technology and The Hague University of Applied Sciences. License: LGPL 3+

Author

Pieter Burghouwt

Version

Revision 2.0

Date

Wednesday, January 2, 2013

6.6.2 Variable Documentation

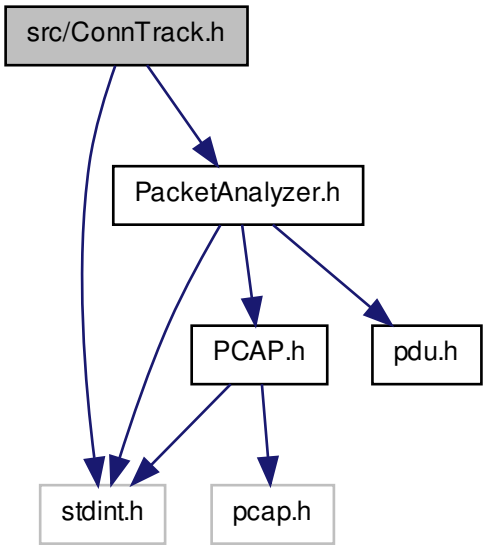
6.6.2.1 TFlow Flow[FLOWBUFFERSIZE]

6.6.2.2 TSettings* Settings

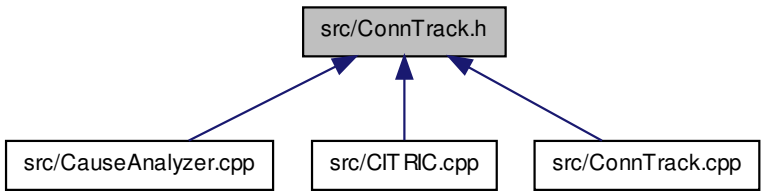
6.7 src/ConnTrack.h File Reference

This file contains the prototypes of the TConnTrack class.

`#include <stdint.h>` `#include "PacketAnalyzer.h"` Include dependency graph for ConnTrack.h:



This graph shows which files directly or indirectly include this file:



Classes

- class TConnTrack

  Class that can kill an establshed connection.

6.7.1 Detailed Description

This file contains the prototypes of the TConnTrack class.

Copyright

Copyright 2012-2013 Delft University of Technology and The Hague University of Applied Sciences. License: LGPL 3+

Author

Pieter Burghouwt

Version

Revision 2.0

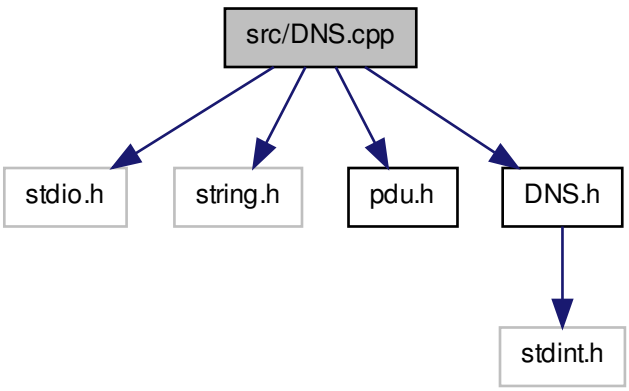
Date

Wednesday, January 1, 2013

6.8 src/DNS.cpp File Reference

This file contains the operators of the TDNS class.

`#include <stdio.h> #include <string.h> #include "pdu.h"` × `#include "DNS.h"` Include dependency graph for DNS.cpp:



6.8.1 Detailed Description

This file contains the operators of the TDNS class.

Copyright

Copyright 2012-2013 Delft University of Technology and The Hague University of Applied Sciences. License: LGPL 3+

Author

Pieter Burghouwt

Version

Revision 2.0

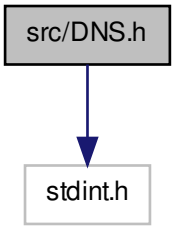
Date

Thursday, February 28, 2013

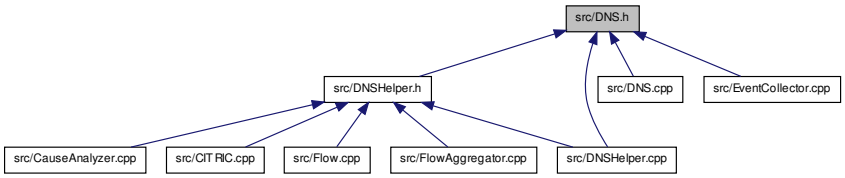
6.9 src/DNS.h File Reference

This file contains the prototypes of the TDNS class.

`#include <stdint.h>` Include dependency graph for DNS.h:



This graph shows which files directly or indirectly include this file:



Classes

- class TDNS

  Storage class for DNS-data. Objects of this class are managed by DNSHelper.

Defines

- #define MAXDOMAINNAMELENGTH 64

6.9.1 Detailed Description

This file contains the prototypes of the TDNS class.

Copyright

Copyright 2012-2013 Delft University of Technology and The Hague University of Applied Sciences. License: LGPL 3+

Author

Pieter Burghouwt

Version

Revision 2.0

Date

Thursday, February 28, 2013

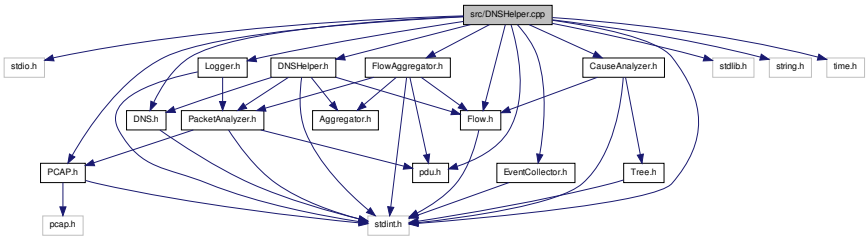
6.9.2 Define Documentation

6.9.2.1 #define MAXDOMAINNAMELENGTH 64

6.10 src/DNSHelper.cpp File Reference

This file contains the operators of the TDNSHelper class.

`#include <stdio.h> #include <stdint.h> #include <stdlib.h> #include <string.h> #include <time.h> #include "pdu.h" #include "PCAP.h" #include "DNSHelper.h" #include "DNS.h" #include "FlowAggregator.h" #include "EventCollector.h" #include "CauseAnalyzer.h" #include "Flow.h" #include "Logger.h"` Include dependency graph for DNSHelper.cpp:



Variables

- TPCAP * PCAP
- TPacketAnalyzer * PacketAnalyzer
- TFlowAggregator * FlowAggregator
- TEventCollector * EventCollector
- TCauseAnalyzer * CauseAnalyzer
- TLogger * Logger
- TFlow Flow [FLOWBUFFERSIZE]
- TDNS DNS [DNSBUFFERSIZE]

6.10.1 Detailed Description

This file contains the operators of the TDNSHelper class.

Copyright

Copyright 2012-2013 Delft University of Technology and The Hague University of Applied Sciences. License: LGPL 3+

Author

Pieter Burghouwt

Version

Revision 2.0

Date

Thursday, February 28, 2013

6.10.2 Variable Documentation

6.10.2.1 TCauseAnalyzer* CauseAnalyzer

6.10.2.2 TDNS DNS[DNSBUFFERSIZE]

6.10.2.3 TEventCollector* EventCollector

6.10.2.4 TFlow Flow[FLOWBUFFERSIZE]

6.10.2.5 TFlowAggregator* FlowAggregator

6.10.2.6 TLogger* Logger

6.10.2.7 TPacketAnalyzer* PacketAnalyzer

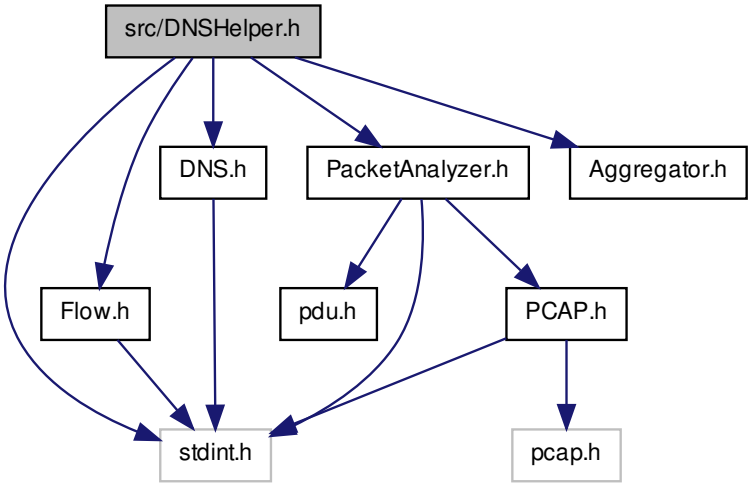
6.10.2.8 TPCAP* PCAP

6.11 src/DNSHelper.h File Reference

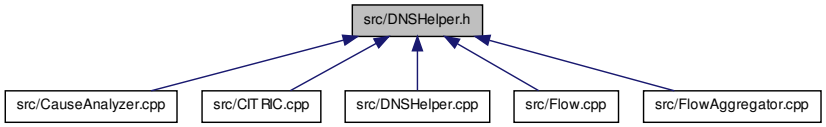
This file contains the prototypes of the TDNSHelper class.

`#include <stdint.h> #include "PacketAnalyzer.h" #include "Flow.h" #include "DNS.h" #include "Aggregator.h"` Include de-

pendency graph for DNSHelper.h:



This graph shows which files directly or indirectly include this file:



Classes

- class TDNSHelper

  Class specialized in aggregating DNS events. It only processes replies from egress UDP DNS-flows. DNS objects are stored as a chained hashtable of unidirectional linked lists. The hashtable is based on a modulo 256 sum of the IP-address. This results in a fast looukup by IP-adress. Newer records with the same name and IP-adress are replaced. TODO: cleanup-routine that checks one record at a time on expiry.

Defines

- #define DNSBUFFERSIZE 65536

total buffer size of DNS-objects

- #define DNSHASHSIZE 1024

  must be a power of 2. Max is 2^16

6.11.1 Detailed Description

This file contains the prototypes of the TDNSHelper class.

Copyright

Copyright 2012-2013 Delft University of Technology and The Hague University of Applied Sciences. License: LGPL 3+

Author

Pieter Burghouwt

Version

Revision 2.0

Date

Thursday, February 28, 2013

6.11.2 Define Documentation

6.11.2.1 #define DNSBUFFERSIZE 65536

total buffer size of DNS-objects

6.11.2.2 #define DNSHASHSIZE 1024

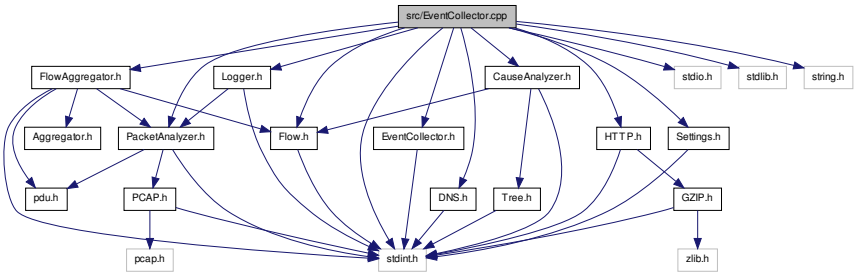
must be a power of 2. Max is 2^16

6.12 src/EventCollector.cpp File Reference

This file contains the operators of the EventCollector class.

```
#include <stdint.h> #include <stdio.h> #include <stdlib.-
h> #include <string.h> #include "FlowAggregator.h" #include
```

"Flow.h" #include "HTTP.h" #include "EventCollector.h" × #include "DNS.h" #include "PacketAnalyzer.h" #include "-Logger.h" #include "CauseAnalyzer.h" #include "Settings.-h" Include dependency graph for EventCollector.cpp:



Variables

- TPacketAnalyzer * PacketAnalyzer
- TFlowAggregator * FlowAggregator
- TLogger * Logger
- TFlow Flow []
- THTTP HTTP []
- TDNS DNS []
- TSettings * Settings

6.12.1 Detailed Description

This file contains the operators of the EventCollector class.

Copyright

Copyright 2012-2013 Delft University of Technology and The Hague University of Applied Sciences. License: LGPL 3+

Author

Pieter Burghouwt

Version

Revision 2.0

Date

Monda, March 11, 2013

6.12.2 Variable Documentation

6.12.2.1 TDNS DNS[]

6.12.2.2 TFlow Flow[]

6.12.2.3 TFlowAggregator* FlowAggregator

6.12.2.4 THTTP HTTP[]

6.12.2.5 TLogger* Logger

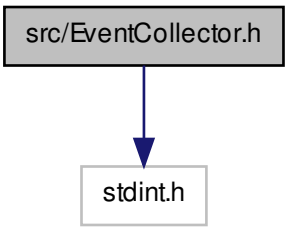
6.12.2.6 TPacketAnalyzer* PacketAnalyzer

6.12.2.7 TSettings* Settings

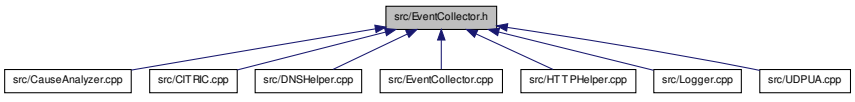
6.13 src/EventCollector.h File Reference

This file contains the prototypes of the TEventCollector class.

`#include <stdint.h>` Include dependency graph for EventCollector.h:



This graph shows which files directly or indirectly include this file:



Classes

- class TUserEvent
- class THTTPEvent
- class THTTPSEvent
- class TURLEvent
- class TRPTDNSEvent
- class TUTreeEvent
- class TEventCollector

  Class specialized in aggregating events that can cause new traffic. Every potential event is buffered here. Hashing is used for fast lookup. FIFO's are used to prevent overflow.

Defines

- #define EVENTBUFFERSIZE 10000
- #define RPTDNSEVENTBUFFERSIZE 100
- #define URLEVENTBUFFERSIZE 1000
- #define URLEVENTHASHSIZE 256
- #define CAUSE_HTTPDATARECEIVED 1
- #define CAUSE_HTTPDATAPUSH 2
- #define CAUSE_HTTPSIZEDECREASE 3
- #define CAUSE_HTTPWITHDRAWN 4

6.13.1 Detailed Description

This file contains the prototypes of the TEventCollector class.

Copyright

Copyright 2012-2013 Delft University of Technology and The Hague University of Applied Sciences. License: LGPL 3+

Author

Pieter Burghouwt

Version

Revision 2.0

Date

Monday, March 11, 2013

6.13.2 Define Documentation

6.13.2.1 #define CAUSE_HTTPDATAPUSH 2

6.13.2.2 #define CAUSE_HTTPDATARECEIVED 1

6.13.2.3 #define CAUSE_HTTPSIZEDECREASE 3

6.13.2.4 #define CAUSE_HTTPWITHDRAWN 4

6.13.2.5 #define EVENTBUFFERSIZE 10000

6.13.2.6 #define RPTDNSEVENTBUFFERSIZE 100

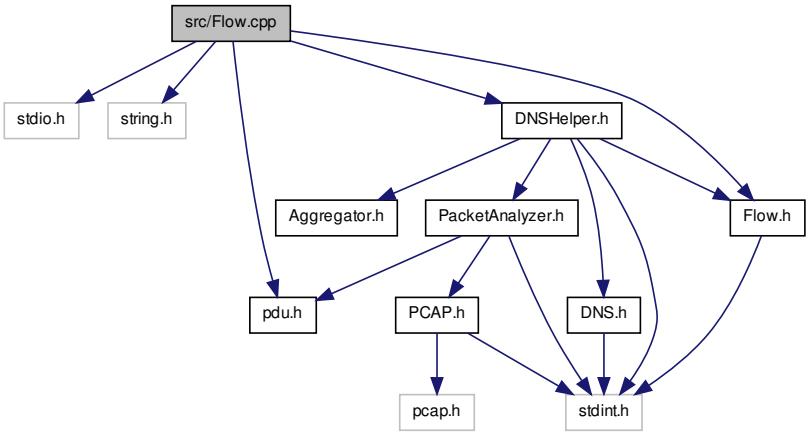
6.13.2.7 #define URLEVENTBUFFERSIZE 1000

6.13.2.8 #define URLEVENTHASHSIZE 256

6.14 src/Flow.cpp File Reference

This file contains the operators of the TFlow class.

`#include <stdio.h> #include <string.h> #include "pdu.h" × #include "Flow.h" #include "DNSHelper.h"` Include dependency graph for Flow.cpp:



Variables

- TDNS DNS [DNSBUFFERSIZE]

6.14.1 Detailed Description

This file contains the operators of the TFlow class.

Copyright

Copyright 2012-2013 Delft University of Technology and The Hague University of Applied Sciences. License: LGPL 3+

Author

Pieter Burghouwt

Version

Revision 0.9

Date

Tuesday, March 12, 2013

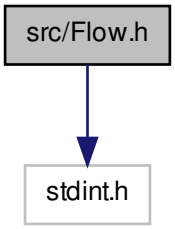
6.14.2 Variable Documentation

6.14.2.1 TDNS DNS[DNSBUFFERSIZE]

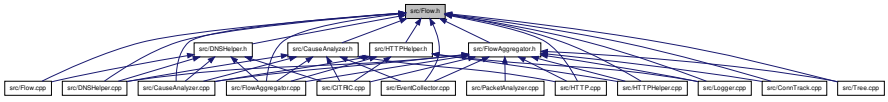
6.15 src/Flow.h File Reference

This file contains the prototypes of the TFlow class.

`#include <stdint.h>` Include dependency graph for Flow.h:



This graph shows which files directly or indirectly include this file:



Classes

- class TFlow

  Storage class for a bidirectional flow.

Defines

- #define FLOWHASHSIZE 1024
- #define UNDEFINED_DIR 0
- #define INGRESS 1
- #define EGRESS 2
- #define RESOLVER_UNDEFINED 0
- #define RESOLVER_PROTO_STATIC 1
- #define RESOLVER_PROTO_DNS 2
- #define RESOLVER_PROTO_OTHER 3
- #define RESOLVER_NONAME 128
- #define RESOLVER_STATICNAME 129
- #define RESOLVER_DNSNAME 130
- #define RESOLVER_OTHERNAME 131
- #define CAUSE_UNKNOWN 0
- #define CAUSE_SERVER 1
- #define CAUSE_DNS 2
- #define CAUSE_HTTP_URL 3
- #define CAUSE_HTTP_SOFTURL 4
- #define CAUSE_HTTP_REFERER 5
- #define CAUSE_HTTP_GEN 6
- #define CAUSE_HTTPS_GEN 7
- #define CAUSE_USER 8
- #define CAUSE_PROTODNS 9
- #define CAUSE_WHITELIST 10
- #define CAUSE_ALREADYOPEN 11
- #define CAUSE_DNS_REPEAT 12
- #define CAUSE_UTREE 13
- #define CAUSEQ_NOID_NOTIME 0
- #define CAUSEQ_NOID_SOFTTIME 1
- #define CAUSEQ_NOID_TIME 2
- #define CAUSEQ_TIME 3

- #define CAUSEQ_SOFTID_NOTIME 4
- #define CAUSEQ_SOFTID_SOFTTIME 5
- #define CAUSEQ_SOFTID_TIME 6
- #define CAUSEQ_ID_NOTIME 7
- #define CAUSEQ_ID_SOFTTIME 8
- #define CAUSEQ_ID_TIME 9

6.15.1 Detailed Description

This file contains the prototypes of the TFlow class.

Copyright

Copyright 2012-2013 Delft University of Technology and The Hague University of Applied Sciences. License: LGPL 3+

Author

Pieter Burghouwt

Version

Revision 2.0

Date

Tuesday, March 12, 2013

6.15.2 Define Documentation

6.15.2.1 #define CAUSE_ALREADYOPEN 11

6.15.2.2 #define CAUSE_DNS 2

6.15.2.3 #define CAUSE_DNS_REPEAT 12

6.15.2.4 #define CAUSE_HTTP_GEN 6

6.15.2.5 #define CAUSE_HTTP_REFERER 5

6.15.2.6 #define CAUSE_HTTP_SOFTURL 4

6.15.2.7 #define CAUSE_HTTP_URL 3

6.15.2.8 #define CAUSE_HTTPS_GEN 7

6.15.2.9 #define CAUSE_PROTODNS 9

6.15.2.10 #define CAUSE_SERVER 1

6.15.2.11 #define CAUSE_UNKNOWN 0

6.15.2.12 #define CAUSE_USER 8

6.15.2.13 #define CAUSE_UTREE 13

6.15.2.14 #define CAUSE_WHITELIST 10

6.15.2.15 #define CAUSEQ_ID_NOTIME 7

6.15.2.16 #define CAUSEQ_ID_SOFTTIME 8

6.15.2.17 #define CAUSEQ_ID_TIME 9

6.15.2.18 #define CAUSEQ_NOID_NOTIME 0

6.15.2.19 #define CAUSEQ_NOID_SOFTTIME 1

6.15.2.20 #define CAUSEQ_NOID_TIME 2

6.15.2.21 #define CAUSEQ_SOFTID_NOTIME 4

6.15.2.22 #define CAUSEQ_SOFTID_SOFTTIME 5

6.15.2.23 #define CAUSEQ_SOFTID_TIME 6

6.15.2.24 #define CAUSEQ_TIME 3

6.15.2.25 #define EGRESS 2

6.15.2.26 #define FLOWHASHSIZE 1024

6.15.2.27 #define INGRESS 1

6.15.2.28 #define RESOLVER_DNSNAME 130

6.15.2.29 #define RESOLVER_NONAME 128

6.15.2.30 #define RESOLVER_OTHERNAME 131

6.15.2.31 #define RESOLVER_PROTO_DNS 2

6.15.2.32 #define RESOLVER_PROTO_OTHER 3

6.15.2.33 #define RESOLVER_PROTO_STATIC 1

6.15.2.34 #define RESOLVER_STATICNAME 129

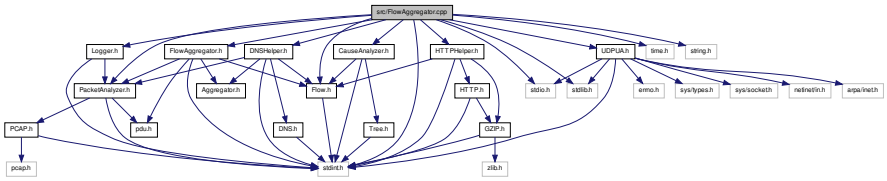
6.15.2.35 #define RESOLVER_UNDEFINED 0

6.15.2.36 #define UNDEFINED_DIR 0

6.16 src/FlowAggregator.cpp File Reference

This file contains the operators of the TFlowAggregator class.

`#include <stdint.h> #include <stdio.h> #include <stdlib.h> #include <time.h> #include <string.h> #include "PacketAnalyzer.h" #include "FlowAggregator.h" #include "Flow.h" #include "DNSHelper.h" #include "CauseAnalyzer.h" #include "HTTPHelper.h" #include "UDPUA.h" #include "Logger.h"` Include dependency graph for FlowAggregator.cpp:



Variables

- TPCAP * PCAP
- TPacketAnalyzer * PacketAnalyzer
- TDNSHelper * DNSHelper
- TCauseAnalyzer * CauseAnalyzer
- THTTPHelper * HTTPHelper
- TUDPUA * UDPUA
- TLogger * Logger
- int32_t FlowTable [FLOWHASHSIZE]
- TFlow Flow [FLOWBUFFERSIZE]

6.16.1 Detailed Description

This file contains the operators of the TFlowAggregator class.

Copyright

Copyright 2012-2013 Delft University of Technology and The Hague University of Applied Sciences. License: LGPL 3+

Author

Pieter Burghouwt

Version

Revision 2.0

Date

Tuesday, March 12, 2013

6.16.2 Variable Documentation

6.16.2.1 TCauseAnalyzer* CauseAnalyzer

6.16.2.2 TDNSHelper* DNSHelper

6.16.2.3 TFlow Flow[FLOWBUFFERSIZE]

6.16.2.4 int32_t FlowTable[FLOWHASHSIZE]

6.16.2.5 THTTPHelper* HTTPHelper

6.16.2.6 TLogger* Logger

6.16.2.7 TPacketAnalyzer* PacketAnalyzer

6.16.2.8 TPCAP* PCAP

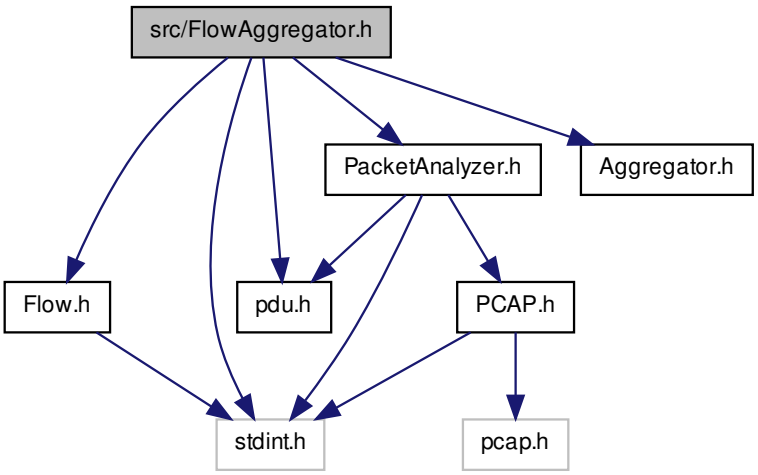
6.16.2.9 TUDPUA* UDPUA

6.17 src/FlowAggregator.h File Reference

This file contains the prototypes of the TFlowAggregator class.

`#include <stdint.h> #include "PacketAnalyzer.h" #include "Flow.h" #include "Aggregator.h" #include "pdu.h"` Include de-

pendency graph for FlowAggregator.h:



This graph shows which files directly or indirectly include this file:

Classes

- class TFlowAggregator

  Important class that manages the flows.

Defines

- #define FLOWBUFFERSIZE 65536

6.17.1 Detailed Description

This file contains the prototypes of the TFlowAggregator class.

Copyright

Copyright 2012-2013 Delft University of Technology and The Hague University of Applied Sciences. License: LGPL 3+

Author

Pieter Burghouwt

Version

Revision 2.0

Date

Tuesday, December 4, 2012

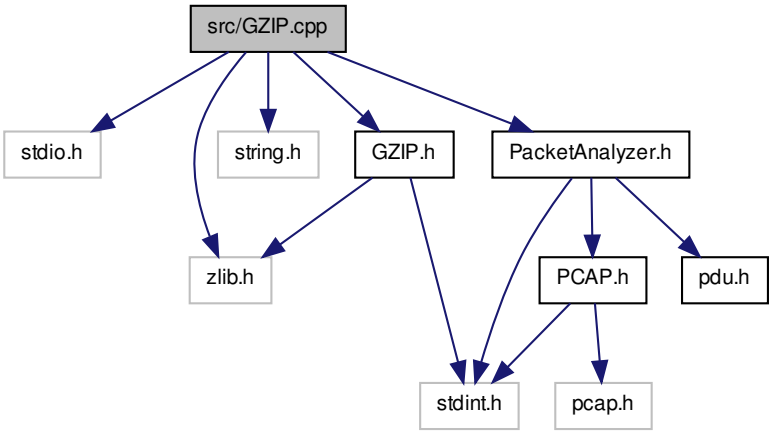
6.17.2 Define Documentation

6.17.2.1 #define FLOWBUFFERSIZE 65536

6.18 src/GZIP.cpp File Reference

This file contains the operators of the TGZIP class.

`#include <stdio.h> #include <zlib.h> #include <string.h> #include "GZIP.h" #include "PacketAnalyzer.h"` Include dependency graph for GZIP.cpp:



Defines

- #define windowBits 15
- #define ENABLE_ZLIB_GZIP 32

Variables

- TPacketAnalyzer * PacketAnalyzer

6.18.1 Detailed Description

This file contains the operators of the TGZIP class.

Copyright

Copyright 2012-2013 Delft University of Technology and The Hague University of Applied Sciences. License: LGPL 3+

Author

Pieter Burghouwt

Version

Revision 2.0

Date

Tuesday, March 12, 2013

6.18.2 Define Documentation

6.18.2.1 #define ENABLE_ZLIB_GZIP 32

6.18.2.2 #define windowBits 15

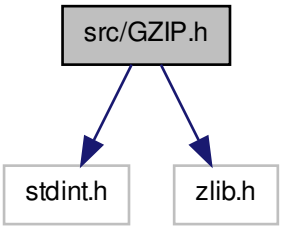
6.18.3 Variable Documentation

6.18.3.1 TPacketAnalyzer* PacketAnalyzer

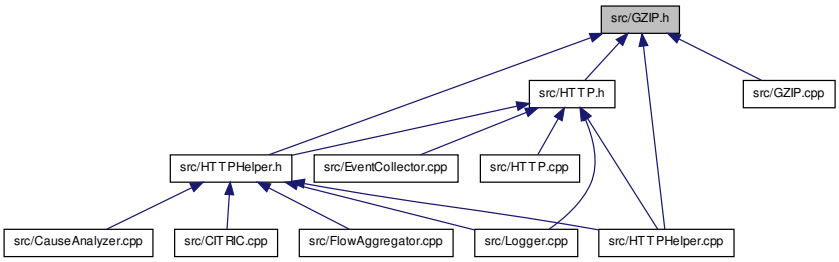
6.19 src/GZIP.h File Reference

This file contains the prototypes of the TGZIP class.

`#include <stdint.h>` `#include <zlib.h>` Include dependency graph for GZIP.h:



This graph shows which files directly or indirectly include this file:



Classes

- class TGZIP

  Class for deflating gzip data. Wraps around zlib.

Defines

- #define GZIP_BUFFERSIZE 32768

6.19.1 Detailed Description

This file contains the prototypes of the TGZIP class.

Copyright

Copyright 2012-2013 Delft University of Technology and The Hague University of Applied Sciences. License: LGPL 3+

Author

Pieter Burghouwt

Version

Revision 2.0

Date

Thursday, February 28, 2013

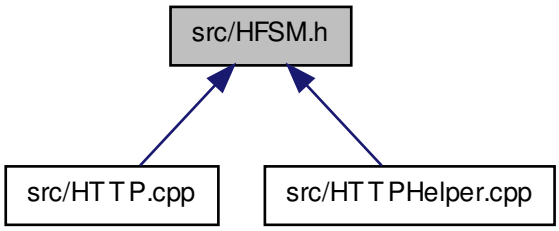
6.19.2 Define Documentation

6.19.2.1 #define GZIP_BUFFERSIZE 32768

6.20 src/HFSM.h File Reference

This file contains HFSM-state definitions.

This graph shows which files directly or indirectly include this file:



Defines

- #define HFSM1_IDLE 0
- #define HFSM1_NO_HTTP 1

- #define HFSM1_HEADER 2
- #define HFSM1_BODY 3
- #define HFSM1_PARSED 4
- #define HFSM1_CHUNK 5
- #define HFSM2_IDLE 0
- #define HFSM2_NEWLINERECEIVED 1
- #define HFSM2_CONTENT 2
- #define HFSM2_CONTENTTYPE 3
- #define HFSM2_ENCODING 4
- #define HFSM2_CHUNKED 5
- #define HFSM2_CONTENTLENGTH 6
- #define HFSM2_WHITELINE 7
- #define HFSM2_CHAR_RECEIVED 8
- #define HFSM2_DOT_RECEIVED 9
- #define HFSM2_OVERFLOW 10
- #define HFSM2_MOVED 11
- #define HFSM2_

6.20.1 Detailed Description

This file contains HFSM-state definitions.

Copyright

Copyright 2012-2013 Delft University of Technology and The Hague University of Applied Sciences. License: LGPL 3+

Author

Pieter Burghouwt

Version

Revision 2.0

Date

Wednesday, November 28, 2012

6.20.2 Define Documentation

6.20.2.1 #define HFSM1_BODY 3

6.20.2.2 #define HFSM1_CHUNK 5

6.20.2.3 #define HFSM1_HEADER 2

6.20.2.4 #define HFSM1_IDLE 0

6.20.2.5 #define HFSM1_NO_HTTP 1

6.20.2.6 #define HFSM1_PARSED 4

6.20.2.7 #define HFSM2_

6.20.2.8 #define HFSM2_CHAR_RECEIVED 8

6.20.2.9 #define HFSM2_CHUNKED 5

6.20.2.10 #define HFSM2_CONTENT 2

6.20.2.11 #define HFSM2_CONTENTLENGTH 6

6.20.2.12 #define HFSM2_CONTENTTYPE 3

6.20.2.13 #define HFSM2_DOT_RECEIVED 9

6.20.2.14 #define HFSM2_ENCODING 4

6.20.2.15 #define HFSM2_IDLE 0

6.20.2.16 #define HFSM2_MOVED 11

6.20.2.17 #define HFSM2_NEWLINERECEIVED 1

6.20.2.18 #define HFSM2_OVERFLOW 10

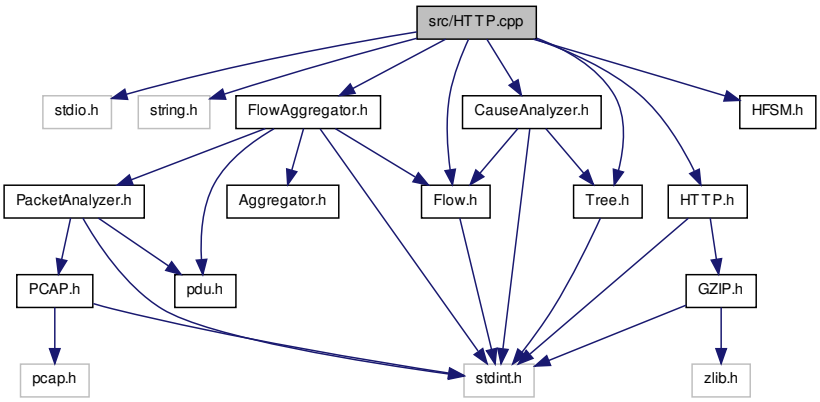
6.20.2.19 #define HFSM2_WHITELINE 7

6.21 src/HTTP.cpp File Reference

This file contains the operators of the THTTTP class.

```
#include <stdio.h>  #include <string.h>  #include "HTT-
P.h" #include "HFSM.h" #include "Flow.h" #include "Flow-
Aggregator.h" #include "CauseAnalyzer.h" #include "Tree.-
```

h" Include dependency graph for HTTP.cpp:



Variables

- TFlow Flow [FLOWBUFFERSIZE]
- TTree Tree [TREE_BUFFER_SIZE]

6.21.1 Detailed Description

This file contains the operators of the THTTTP class.

Copyright

Copyright 2012-2013 Delft University of Technology and The Hague University of Applied Sciences. License: LGPL 3+

Author

Pieter Burghouwt

Version

Revision 2.0

Date

Tuesday, March 5, 2013

6.21.2 Variable Documentation

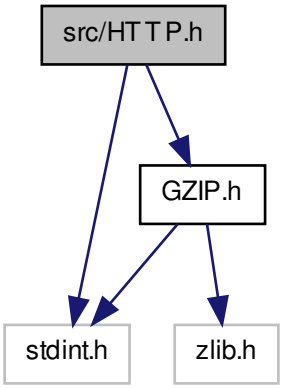
6.21.2.1 TFlow Flow[FLOWBUFFERSIZE]

6.21.2.2 TTree Tree[TREE_BUFFER_SIZE]

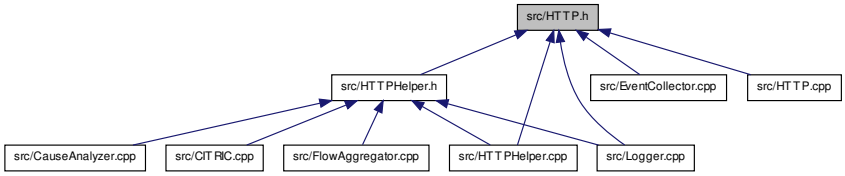
6.22 src/HTTP.h File Reference

This file contains the prototypes of the THTTP class.

`#include <stdint.h>` `#include "GZIP.h"` Include dependency graph for HTTP.h:



This graph shows which files directly or indirectly include this file:



Classes

- class THTTP

  Storage class for HTTP-events. Every HTTP-flow or HTTPS-flow that can potentially trigger new traffi by embedded URL's has an object of this class.

Defines

- #define HTTPSTATUS_UNDEFINED 0
- #define HTTPSTATUS_WAITINGFORSEND 1
- #define HTTPSTATUS_SENT 2
- #define HTTPSTATUS_RECEIVED 3
- #define REFSTAT_UNDEFINED 0
- #define REFSTAT_REF 1
- #define REFSTAT_NOREF 2
- #define REFSTAT_GETSEEN 3

6.22.1 Detailed Description

This file contains the prototypes of the THTTP class.

Copyright

Copyright 2012-2013 Delft University of Technology and The Hague University of Applied Sciences. License: LGPL 3+

Author

Pieter Burghouwt

Version

Revision 2.0

Date

Tuesday, March 5, 2013

6.22.2 Define Documentation

6.22.2.1 #define HTTPSTATUS_RECEIVED 3

6.22.2.2 #define HTTPSTATUS_SENT 2

6.22.2.3 #define HTTPSTATUS_UNDEFINED 0

6.22.2.4 #define HTTPSTATUS_WAITINGFORSEND 1

6.22.2.5 #define REFSTAT_GETSEEN 3

6.22.2.6 #define REFSTAT_NOREF 2

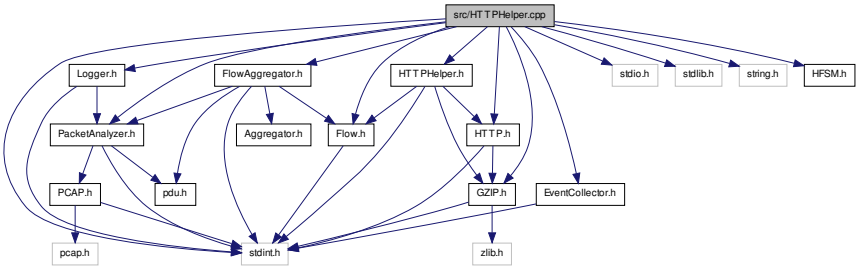
6.22.2.7 #define REFSTAT_REF 1

6.22.2.8 #define REFSTAT_UNDEFINED 0

6.23 src/HTTPHelper.cpp File Reference

This file contains the operators of the THTTPAgggregator class.

`#include <stdint.h> #include <stdio.h> #include <stdlib.h> #include <string.h> #include "FlowAggregator.h" #include "Flow.h" #include "HTTPHelper.h" #include "PacketAnalyzer.h" #include "EventCollector.h" #include "HTTP.h" #include "HFSM.h" #include "GZIP.h" #include "Logger.h"` Include dependency graph for HTTPHelper.cpp:



Variables

- TFlowAggregator * FlowAggregator
- TPCAP * PCAP
- TPacketAnalyzer * PacketAnalyzer
- TFlow Flow [FLOWBUFFERSIZE]
- TEventCollector * EventCollector
- TLogger * Logger
- THTTP HTTP [HTTP_BUFFER_SIZE]
- const char * PatternHTTP = "HTTP"
- const char * PatternContentTypeText = "Content-Type:text "
- const char * PatternContentTypeJava = "Content-Type:application/x-java "
- const char * PatternContentTypeJS = "Content-Type:application/javascript "
- const char * PatternContentTypeFlash = "Content-Type:application/x-shoc "

- const char * PatternContentTypeJSON = "Content-Type:application/json "
- const char * PatternEncodingGZIP = "Encoding:gzip"
- const char * PatternTransfer = "Transfer-Encoding:chunked"
- const char * PatternLocation = "Location:"
- const char * PatternWhiteLine = "\r\n"
- const char * Patternhttp = "http"
- const char * TLD [26]

6.23.1 Detailed Description

This file contains the operators of the THTTPAgggregator class.

Copyright

Copyright 2012-2013 Delft University of Technology and The Hague University of Applied Sciences. License: LGPL 3+

Author

Pieter Burghouwt

Version

Revision 2.0

Date

Tuesday, March 12, 2013

6.23.2 Variable Documentation

6.23.2.1 TEventCollector* EventCollector

6.23.2.2 TFlow Flow[FLOWBUFFERSIZE]

6.23.2.3 TFlowAggregator* FlowAggregator

6.23.2.4 THTTP HTTP[HTTP_BUFFER_SIZE]

6.23.2.5 TLogger* Logger

6.23.2.6 TPacketAnalyzer* PacketAnalyzer

6.23.2.7 const char* PatternContentTypeFlash = "Content-Type:application/x-shoc "

6.23.2.8 const char* PatternContentTypeJava = "Content-Type:application/x-java "

6.23.2.9 const char* PatternContentTypeJS = "Content-Type:application/javascript "

6.23.2.10 const char* PatternContentTypeJSON = "Content-Type:application/json "

6.23.2.11 const char* PatternContentTypeText = "Content-Type:text "

6.23.2.12 const char* PatternEncodingGZIP = "Encoding:gzip"

6.23.2.13 const char* PatternHTTP = "HTTP"

6.23.2.14 const char* Patternhttp = "http"

6.23.2.15 const char* PatternLocation = "Location:"

6.23.2.16 const char* PatternTransfer = "Transfer-Encoding:chunked"

6.23.2.17 const char* PatternWhiteLine = "\r\n"

6.23.2.18 TPCAP* PCAP

6.23.2.19 const char* TLD[26]

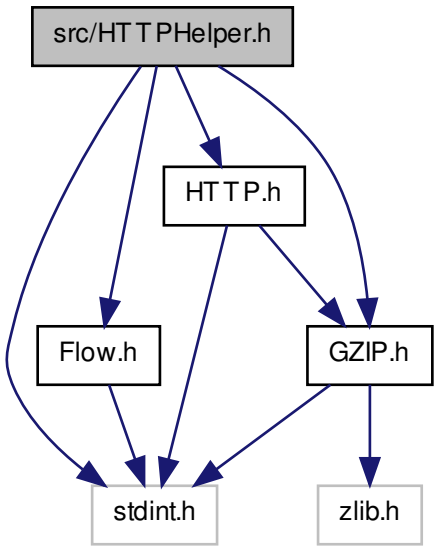
Initial value:

```
{
".ac.ad.ae.aero.af.ag.ai.al.am.an.ao.aq.ar.arpa.as.asia.at.au.aw.ax.az",
".ba.bb.bd.be.bf.bg.bh.bi.biz.bj.bl.bm.bn.bo.bq.br.bs.bt.bv.bw.by.bz",
".ca.cat.cc.cd.cf.cg.ch.ci.ck.cl.cm.cn.co.com.coop.cr.cu.cv.cw.cx.cy.cz",
".de.dj.dk.dm.do.dz",
".ec.edu.ee.eg.eh.er.es.et.eu",
".fi.fj.fk.fm.fo.fr",
".ga.gb.gd.ge.gf.gg.gh.gi.gl.gm.gn.gov.gp.gq.gr.gs.gt.gu.gw.gy",
".hk.hm.hn.hr.ht.hu",
".id.ie.il.im.in.info.int.io.iq.ir.is.it",
".je.jm.jo.jobs.jp",
".ke.kg.kh.ki.km.kn.kp.kr.kw.ky.kz",
".la.lb.lc.li.lk.lr.ls.lt.lu.lv.ly",
"
      .ma.mc.md.me.mf.mg.mh.mil.mk.ml.mm.mn.mo.mobi.mp.mq.mr.ms.mt.mu.museum.mv.mw.mx.my.mz",
".na.name.nc.ne.net.nf.ng.ni.nl.no.np.nr.nu.nz",
".om.org",
".pa.pe.pf.pg.ph.pk.pl.pm.pn.post.pr.pro.ps.pt.pw.py",
".qa",
".re.ro.rs.ru.rw",
".sa.sb.sc.sd.se.sg.sh.si.sj.sk.sl.sm.sn.so.sr.ss.st.su.sv.sx.sy.sz",
".tc.td.tel.tf.tg.th.tj.tk.tl.tm.tn.to.tp.tr.travel.tt.tv.tw.tz",
".ua.ug.uk.um.us.uy.uz",
".va.vc.ve.vg.vi.vn.vu",
".wf.ws",
".xxx.xn--",
".ye.yt",
".za.zm.zw"
}
```

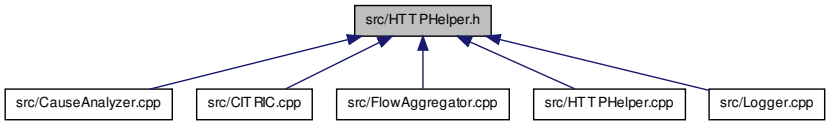
6.24 src/HTTPHelper.h File Reference

This file contains the prototypes of the THTTPHelper class.

`#include <stdint.h> #include "Flow.h" #include "HTTP.h"` × `#include "GZIP.h"` Include dependency graph for HTTPHelper.h:



This graph shows which files directly or indirectly include this file:



Classes

- class THTTPHelper

  Class that aggregates HTTP traffic. It aggregates information that predicts the cause of new traffic flows THTTPHelper creates a static array of HTTP_BUFFER_SIZE (65536)

HTTP-objects. The ClusterAggregator uses this class to check for potential HTTP-parent flows. TFlowAggregator updates the information of this class.

Defines

- #define HTTP_BUFFER_SIZE 65536
- #define TOTAL_GZIP 2000
- #define BINARYCONTENT 0

6.24.1 Detailed Description

This file contains the prototypes of the THTTPHelper class.

Copyright

Copyright 2012-2013 Delft University of Technology and The Hague University of Applied Sciences. License: LGPL 3+

Author

Pieter Burghouwt

Version

Revision 2.0

Date

Tuesday, March 5, 2013

6.24.2 Define Documentation

6.24.2.1 #define BINARYCONTENT 0

6.24.2.2 #define HTTP_BUFFER_SIZE 65536

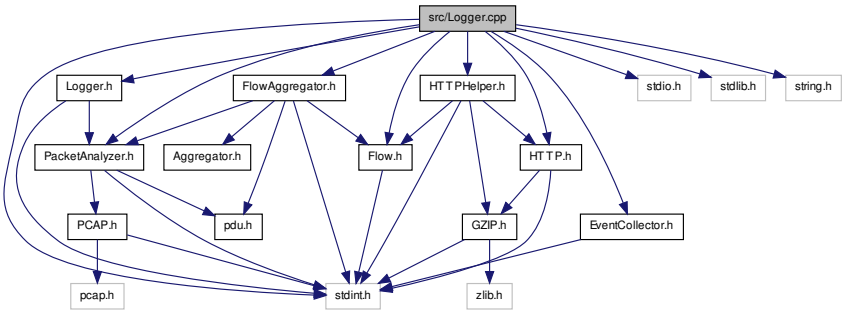
6.24.2.3 #define TOTAL_GZIP 2000

6.25 src/Logger.cpp File Reference

This file contains the operators of the TLogger class.

```
#include <stdint.h> #include <stdio.h> #include <stdlib.-
h> #include <string.h> #include "FlowAggregator.h" #include
```

"Flow.h" #include "HTTPHelper.h" #include "PacketAnalyzer.h" #include "EventCollector.h" #include "HTTP.h" #include "Logger.h" Include dependency graph for Logger.cpp:



Variables

- TFlowAggregator * FlowAggregator
- TPCAP * PCAP
- TPacketAnalyzer * PacketAnalyzer
- TFlow Flow [FLOWBUFFERSIZE]
- TEventCollector * EventCollector
- THTTP * HTTP
- const char * Severity [8] = {"EMERGENCY", "ALERT", "CRITICAL", "ERROR", "WARNING", "NOTICE", "INFORMATIONAL", "DEBUG"}
- const char * Facility [7] = {"PACKET_ANALYZER", "FLOW_AGGREGATOR", "DNS_HELPER", "HTTP_HELPER", "EVENT_COLLECTOR", "CAUSE_ANALYZER", "MAIN"}
- const char * Protocol [7] = {"ICMP", "TCP", "UDP", "HTTP", "HTTPS", "DNS", "UNKNOWN"}
- const char * CauseDesc [13] = {"CAUSE_UNKNOWN", "CAUSE_SERVER", "CAUSE_DNS", "CAUSE_HTTP_URL", "CAUSE_HTTP_SOFTURL", "CAUSE_HTTP_REFERER", "CAUSE_HTTP_GEN", "CAUSE_HTTPS_GEN", "CAUSE_USER", "CAUSE_PROTODNS", "CAUSE_WHITELIST", "CAUSE_ALREADYOPEN", "CAUSE_DNS_REPEAT"}

6.25.1 Detailed Description

This file contains the operators of the TLogger class.

Copyright

Copyright 2012-2013 Delft University of Technology and The Hague University of Applied Sciences. License: LGPL 3+

Author

Pieter Burghouwt

Version

Revision 2.0

Date

Wednesday, December 5, 2012

6.25.2 Variable Documentation

6.25.2.1 const char* CauseDesc[13] = {"CAUSE_UNKNOWN", "CAUSE_SERVER", "CAUSE_DNS", "CAUSE_HTTP_URL", "CAUSE_HTTP_SOFTURL", "CAUSE_HTTP_REFERER", "CAUSE_HTTP_GEN", "CAUSE_HTTPS_GEN", "CAUSE_USER", "CAUSE_PROTODNS", "CAUSE_WHITELIST", "CAUSE_ALREADYOPEN", "CAUSE_DNS_REPEAT"}

6.25.2.2 TEventCollector* EventCollector

6.25.2.3 const char* Facility[7] = {"PACKET_ANALYZER", "FLOW_AGGREGATOR", "DNS_HELPER", "HTTP_HELPER", "EVENT_COLLECTOR", "CAUSE_ANALYZER", "MAIN"}

6.25.2.4 TFlow Flow[FLOWBUFFERSIZE]

6.25.2.5 TFlowAggregator* FlowAggregator

6.25.2.6 THTTP* HTTP

6.25.2.7 TPacketAnalyzer* PacketAnalyzer

6.25.2.8 TPCAP* PCAP

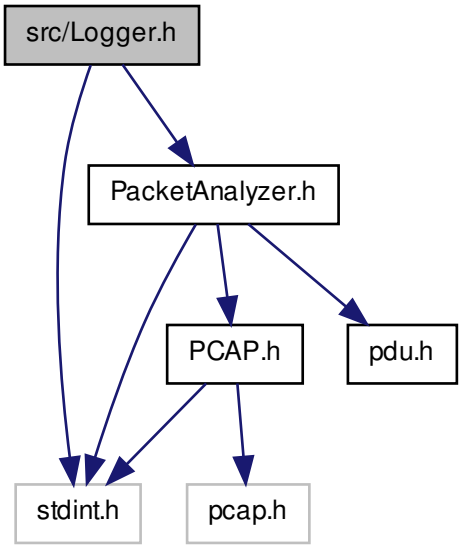
6.25.2.9 const char* Protocol[7] = {"ICMP", "TCP", "UDP", "HTTP", "HTTPS", "DNS", "UNKNOWN"}

6.25.2.10 const char* Severity[8] = {"EMERGENCY", "ALERT", "CRITICAL", "ERROR", "WARNING", "NOTICE", "INFORMATIONAL", "DEBUG"}

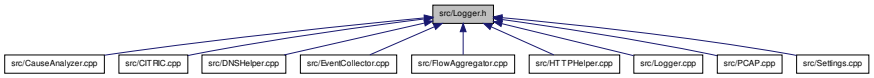
6.26 src/Logger.h File Reference

This file contains the prototypes of the TLogger class.

`#include <stdint.h>` `#include "PacketAnalyzer.h"` Include dependency graph for Logger.h:



This graph shows which files directly or indirectly include this file:



Classes

- class TLogger

  Class that aggregates log messages in flat file format in RAM.

Defines

- #define LOGSIZE 10000000
- #define DEBUG 7
- #define INFORMATIONAL 6

- #define NOTICE 5
- #define WARNING 4
- #define ERROR 3
- #define CRITICAL 2
- #define ALERT 1
- #define EMERGENCY 0
- #define FAC_PACKET 0
- #define FAC_FLOW 1
- #define FAC_DNS 2
- #define FAC_HTTP 3
- #define FAC_EVENT 4
- #define FAC_CAUSE 5
- #define FAC_MAIN 6

6.26.1 Detailed Description

This file contains the prototypes of the TLogger class.

Copyright

Copyright 2012-2013 Delft University of Technology and The Hague University of Applied Sciences. License: LGPL 3+

Author

Pieter Burghouwt

Version

Revision 2.0

Date

Wednesday, December 5, 2012

6.26.2 Define Documentation

6.26.2.1 #define ALERT 1

6.26.2.2 #define CRITICAL 2

6.26.2.3 #define DEBUG 7

6.26.2.4 #define EMERGENCY 0

6.26.2.5 #define ERROR 3

6.26.2.6 #define FAC_CAUSE 5

6.26.2.7 #define FAC_DNS 2

6.26.2.8 #define FAC_EVENT 4

6.26.2.9 #define FAC_FLOW 1

6.26.2.10 #define FAC_HTTP 3

6.26.2.11 #define FAC_MAIN 6

6.26.2.12 #define FAC_PACKET 0

6.26.2.13 #define INFORMATIONAL 6

6.26.2.14 #define LOGSIZE 10000000

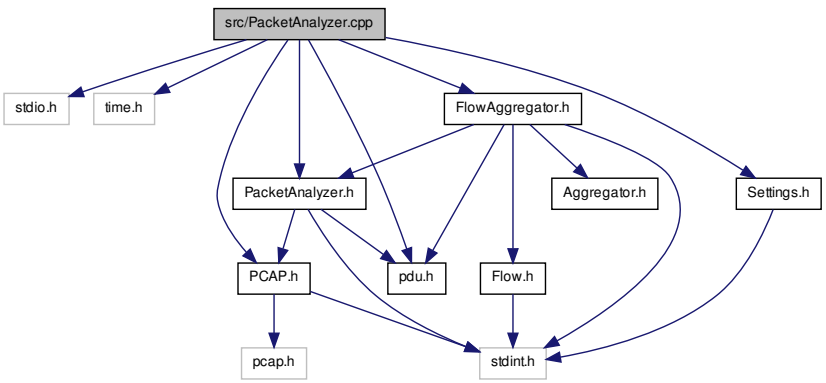
6.26.2.15 #define NOTICE 5

6.26.2.16 #define WARNING 4

6.27 src/PacketAnalyzer.cpp File Reference

This file contains the operators of the TPacketAnalyzer class.

`#include <stdio.h> #include <time.h> #include "pdu.h"` × `#include "PacketAnalyzer.h" #include "PCAP.h" #include "-FlowAggregator.h" #include "Settings.h"` Include dependency graph for PacketAnalyzer.cpp:



Variables

- TPCAP * PCAP
- TFlowAggregator * FlowAggregator
- TSettings * Settings

6.27.1 Detailed Description

This file contains the operators of the TPacketAnalyzer class.

Copyright

Copyright 2012-2013 Delft University of Technology and The Hague University of Applied Sciences. License: LGPL 3+

Author

Pieter Burghouwt

Version

Revision 2.0

Date

Tuesday, March 12, 2013

6.27.2 Variable Documentation

6.27.2.1 TFlowAggregator* FlowAggregator

6.27.2.2 TPCAP* PCAP

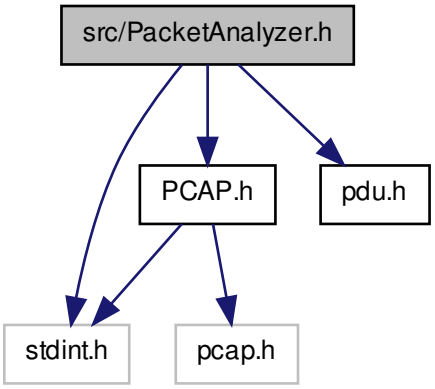
6.27.2.3 TSettings* Settings

6.28 src/PacketAnalyzer.h File Reference

This file contains the prototypes of the TPacketAnalyzer class.

```
#include <stdint.h>  #include "PCAP.h"  #include "pdu.h"
```

Include dependency graph for PacketAnalyzer.h:



This graph shows which files directly or indirectly include this file:

Classes

- class TPacketAnalyzer

  Class for analyzing an arrived packet.

6.28.1 Detailed Description

This file contains the prototypes of the TPacketAnalyzer class.

Copyright

Copyright 2012-2013 Delft University of Technology and The Hague University of Applied Sciences. License: LGPL 3+

Author

Pieter Burghouwt

Version

Revision 2.0

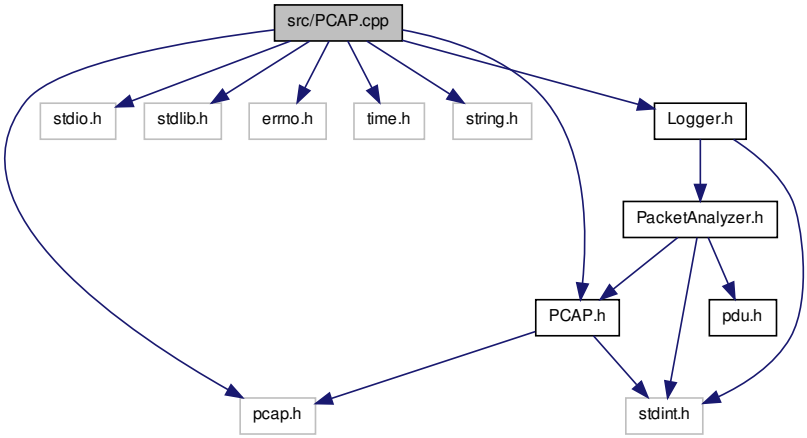
Date

Tuesday, March 12, 2013

6.29 src/PCAP.cpp File Reference

This file contains the operators of the TPCAP class.

`#include <pcap.h> #include <stdio.h> #include <stdlib.h> #include <errno.h> #include <time.h> #include <string.h> #include "PCAP.h" #include "Logger.h"` Include dependency graph for PCAP.cpp:



Variables

- TLogger * Logger

6.29.1 Detailed Description

This file contains the operators of the TPCAP class.

Copyright

Copyright 2012-2013 Delft University of Technology and The Hague University of Applied Sciences. License: LGPL 3+

Author

Pieter Burghouwt

Version

Revision 2.0

Date

Tuesday, November 22, 2011

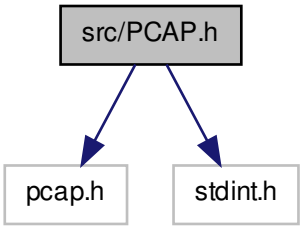
6.29.2 Variable Documentation

6.29.2.1 TLogger* Logger

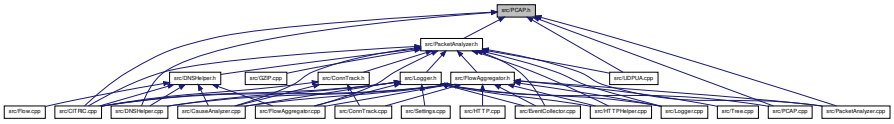
6.30 src/PCAP.h File Reference

This file contains the prototypes of the TPCAP class.

`#include <pcap.h>` `#include <stdint.h>` Include dependency graph for PCAP.h:



This graph shows which files directly or indirectly include this file:



Classes

- class TPCAP

  The "singleton" object of TPCAP is a wrapper around the libpcap library. It can read from a device or a file and return a pointer to a received packet.

6.30.1 Detailed Description

This file contains the prototypes of the TPCAP class.

Copyright

Copyright 2012-2013 Delft University of Technology and The Hague University of Applied Sciences. License: LGPL 3+

Author

Pieter Burghouwt

Version

Revision 2.0

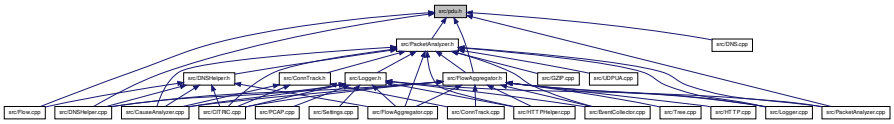
Date

Tuesday, November 22, 2011

6.31 src/pdu.h File Reference

This file contains PDU-fields, places and values.

This graph shows which files directly or indirectly include this file:



Defines

- #define ETHERNET_LENGTH 14
- #define IP_MIN_LENGTH 20
- #define UDP_LENGTH 8
- #define ICMP_LENGTH 4
- #define ETH_PROT_OFFSET 12
- #define IP_VERSION_OFFSET 0
- #define IP_LENGTH_OFFSET 0
- #define IP_PROT_OFFSET 9
- #define IP_SOURCE_OFFSET 12
- #define IP_DEST_OFFSET 16
- #define UDP_LENGTH 8
- #define TCP_LENGTH_OFFSET 12
- #define TCP_SEQ_OFFSET 4
- #define TCP_ACK_OFFSET 8
- #define TCP_FLAG_OFFSET 13
- #define SOURCE_PORT_OFFSET 0
- #define DEST_PORT_OFFSET 2
- #define ICMP_TYPE_OFFSET 0
- #define ICMP_CODE_OFFSET 1
- #define ICMP_ID_OFFSET 4
- #define ICMP_SEQ_OFFSET 6
- #define TCP 6
- #define UDP 17
- #define ICMP 1
- #define OTHER 255
- #define DNS_IDENTIFICATION 0
- #define DNS_OPCODE_OFFSET 2
- #define DNS_RCODE_OFFSET 3
- #define DNS_NUMBER_OF_QUESTIONS_OFFSET 4
- #define DNS_NUMBER_OF_ANSWERS_OFFSET 6
- #define DNS_QUESTIONS_OFFSET 12

6.31.1 Detailed Description

This file contains PDU-fields, places and values.

Copyright

Copyright 2012-2013 Delft University of Technology and The Hague University of Applied Sciences. License: LGPL 3+

Author

Pieter Burghouwt

Version

Revision 2.0

Date

Saturday, March 9, 2013

6.31.2 Define Documentation

6.31.2.1 #define DEST_PORT_OFFSET 2

6.31.2.2 #define DNS_IDENTIFICATION 0

6.31.2.3 #define DNS_NUMBER_OF_ANSWERS_OFFSET 6

6.31.2.4 #define DNS_NUMBER_OF_QUESTIONS_OFFSET 4

6.31.2.5 #define DNS_OPCODE_OFFSET 2

6.31.2.6 #define DNS_QUESTIONS_OFFSET 12

6.31.2.7 #define DNS_RCODE_OFFSET 3

6.31.2.8 #define ETH_PROT_OFFSET 12

6.31.2.9 #define ETHERNET_LENGTH 14

6.31.2.10 #define ICMP 1

6.31.2.11 #define ICMP_CODE_OFFSET 1

6.31.2.12 #define ICMP_ID_OFFSET 4

6.31.2.13 #define ICMP_LENGTH 4

6.31.2.14 #define ICMP_SEQ_OFFSET 6

6.31.2.15 #define ICMP_TYPE_OFFSET 0

6.31.2.16 #define IP_DEST_OFFSET 16

6.31.2.17 #define IP_LENGTH_OFFSET 0

6.31.2.18 #define IP_MIN_LENGTH 20

6.31.2.19 #define IP_PROT_OFFSET 9

6.31.2.20 #define IP_SOURCE_OFFSET 12

6.31.2.21 #define IP_VERSION_OFFSET 0

6.31.2.22 #define OTHER 255

6.31.2.23 #define SOURCE_PORT_OFFSET 0

6.31.2.24 #define TCP 6

6.31.2.25 #define TCP_ACK_OFFSET 8

6.31.2.26 #define TCP_FLAG_OFFSET 13

6.31.2.27 #define TCP_LENGTH_OFFSET 12

6.31.2.28 #define TCP_SEQ_OFFSET 4

6.31.2.29 #define UDP 17

6.31.2.30 #define UDP_LENGTH 8

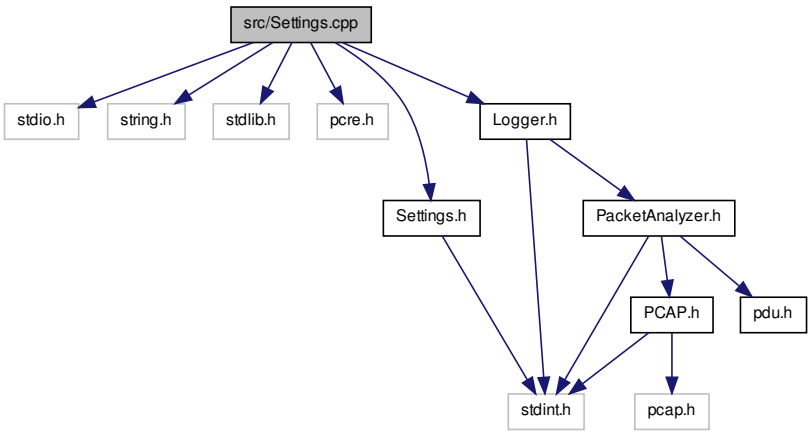
6.31.2.31 #define UDP_LENGTH 8

6.32 src/Settings.cpp File Reference

This file contains the operators of the TSettings class.

```
#include <stdio.h> #include <string.h> #include <stdlib.-
h>  #include <pcre.h>  #include "Settings.h"  #include "-
```

Logger.h" Include dependency graph for Settings.cpp:



Variables

- TLogger * Logger

6.32.1 Detailed Description

This file contains the operators of the TSettings class.

Copyright

Copyright 2012-2013 Delft University of Technology and The Hague University of Applied Sciences. License: LGPL 3+

Author

Pieter Burghouwt

Version

Revision 2.0

Date

Saturday, December 22, 2012

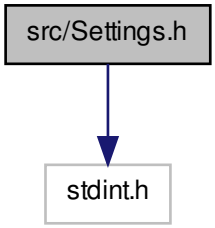
6.32.2 Variable Documentation

6.32.2.1 TLogger* Logger

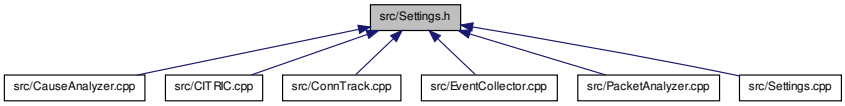
6.33 src/Settings.h File Reference

This file contains the prototypes of the TSettings class.

`#include <stdint.h>` Include dependency graph for Settings.h:



This graph shows which files directly or indirectly include this file:



Classes

- class TSettings

Defines

- #define SETTINGSSIZE 13
- #define DELTA_T_DNS Settings->DefinedValues[0]
- #define DELTA_T_DNS_DEL Settings->DefinedValues[1]
- #define DELTA_T_DNS_RPT Settings->DefinedValues[2]
- #define DELTA_T_URL Settings->DefinedValues[3]

- #define DELTA_T_URL_DEL Settings->DefinedValues[4]
- #define DELTA_T_HTTP Settings->DefinedValues[5]
- #define DELTA_T_HTTPS Settings->DefinedValues[6]
- #define DELTA_T_USER Settings->DefinedValues[7]
- #define DELTA_T_UTREE Settings->DefinedValues[8]
- #define IPS_ENABLE Settings->DefinedValues[9]
- #define DNS_PORTPATCH Settings->DefinedValues[10]
- #define IDL_MAX_TOKENS Settings->DefinedValues[11]
- #define IDL_MAX_LENGTH Settings->DefinedValues[12]

6.33.1 Detailed Description

This file contains the prototypes of the TSettings class.

Copyright

Copyright 2012-2013 Delft University of Technology and The Hague University of Applied Sciences. License: LGPL 3+

Author

Pieter Burghouwt

Version

Revision 2.0

Date

Saturday, December 22, 2012

6.33.2 Define Documentation

6.33.2.1 #define DELTA_T_DNS Settings->DefinedValues[0]

6.33.2.2 #define DELTA_T_DNS_DEL Settings->DefinedValues[1]

6.33.2.3 #define DELTA_T_DNS_RPT Settings->DefinedValues[2]

6.33.2.4 #define DELTA_T_HTTP Settings->DefinedValues[5]

6.33.2.5 #define DELTA_T_HTTPS Settings->DefinedValues[6]

6.33.2.6 #define DELTA_T_URL Settings->DefinedValues[3]

6.33.2.7 #define DELTA_T_URL_DEL Settings->DefinedValues[4]

6.33.2.8 #define DELTA_T_USER Settings->DefinedValues[7]

6.33.2.9 #define DELTA_T_UTREE Settings->DefinedValues[8]

6.33.2.10 #define DNS_PORTPATCH Settings->DefinedValues[10]

6.33.2.11 #define IDL_MAX_LENGTH Settings->DefinedValues[12]

6.33.2.12 #define IDL_MAX_TOKENS Settings->DefinedValues[11]

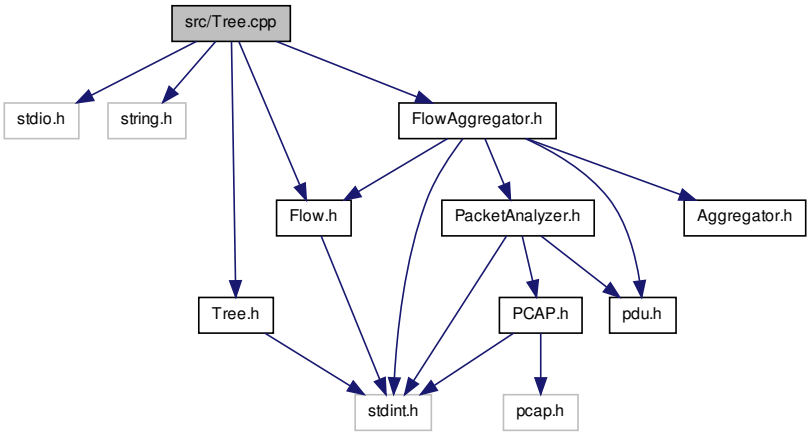
6.33.2.13 #define IPS_ENABLE Settings->DefinedValues[9]

6.33.2.14 #define SETTINGSSIZE 13

6.34 src/Tree.cpp File Reference

This file contains the operators of the TTree class.

`#include <stdio.h> #include <string.h> #include "Tree.h" #include "Flow.h" #include "FlowAggregator.h"` Include dependency graph for Tree.cpp:



Variables

- TFlow Flow [FLOWBUFFERSIZE]

6.34.1 Detailed Description

This file contains the operators of the TTree class.

Copyright

Copyright 2012-2013 Delft University of Technology and The Hague University of Applied Sciences. License: LGPL 3+

Author

Pieter Burghouwt

Version

Revision 2.0

Date

Friday, March 8, 2013

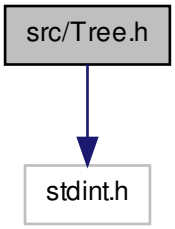
6.34.2 Variable Documentation

6.34.2.1 TFlow Flow[FLOWBUFFERSIZE]

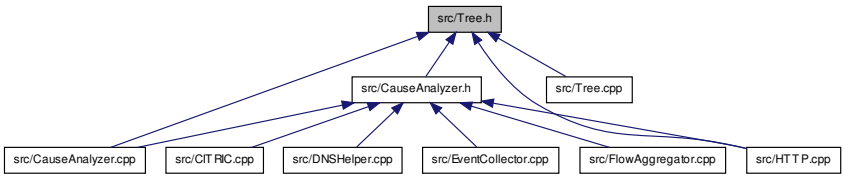
6.35 src/Tree.h File Reference

This file contains the prototypes of the TTree class.

`#include <stdint.h>` Include dependency graph for Tree.h:



This graph shows which files directly or indirectly include this file:



Classes

- class TTree

  Storage class for a tree of network flows with a causal relationship.

6.35.1 Detailed Description

This file contains the prototypes of the TTree class.

Copyright

Copyright 2012-2013 Delft University of Technology and The Hague University of Applied Sciences. License: LGPL 3+

Author

Pieter Burghouwt

Version

Revision 2.0

Date

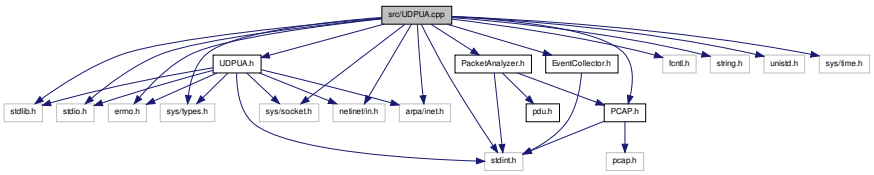
Friday, March 8, 2013

6.36 src/UDPUA.cpp File Reference

This file contains the operators of the TUDPUA class.

```
#include <stdlib.h> #include <stdio.h> #include <fcntl.-
h>  #include <errno.h>  #include <sys/types.h>  #include
```

<sys/socket.h> #include <netinet/in.h> #include <arpa/inet.-h> #include <string.h> #include <unistd.h> #include <stdint.-h> #include "UDPUA.h" #include <sys/time.h> #include "-PacketAnalyzer.h" #include "PCAP.h" #include "EventCollector.-h" Include dependency graph for UDPUA.cpp:



Variables

- TPacketAnalyzer * PacketAnalyzer
- TPCAP * PCAP
- TEventCollector * EventCollector

6.36.1 Detailed Description

This file contains the operators of the TUDPUA class.

Copyright

Copyright 2012-2013 Delft University of Technology and The Hague University of Applied Sciences. License: LGPL 3+

Author

Pieter Burghouwt

Version

Revision 2.0

Date

Wednesday, April 04, 2012

6.36.2 Variable Documentation

6.36.2.1 TEventCollector* EventCollector

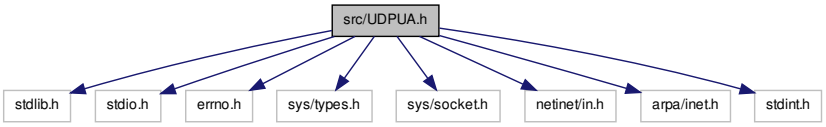
6.36.2.2 TPacketAnalyzer* PacketAnalyzer

6.36.2.3 TPCAP* PCAP

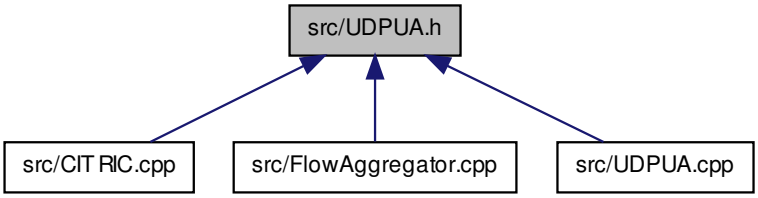
6.37 src/UDPUA.h File Reference

This file contains the prototypes of the UDPUA class.

`#include <stdlib.h> #include <stdio.h> #include <errno.h> #include <sys/types.h> #include <sys/socket.h> #include <netinet/in.h> #include <arpa/inet.h> #include <stdint.h>` Include dependency graph for UDPUA.h:



This graph shows which files directly or indirectly include this file:



Classes

- class TUDPUA

  One object of this class (class should be used as singleton) recieves and filters user events, received from UDP with a special agent. Received results can be polled in a Event-loop.

Defines

- #define UDP_Port 1234

6.37.1 Detailed Description

This file contains the prototypes of the UDPUA class.

Copyright

Copyright 2012-2013 Delft University of Technology and The Hague University of Applied Sciences. License: LGPL 3+

Author

Burghouwt:Pieter

Version

Revision 2.0

Date

Wednesday, April 4, 2012

6.37.2 Define Documentation

6.37.2.1 #define UDP_Port 1234