spp_profinet

Generated by Doxygen 1.8.10

Sat Jan 30 2016 13:00:17

Contents

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

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

Chapter 3

Data Structure Documentation

3.1 BoxStruct_struct Struct Reference

Use brief, otherwise the index won't have a brief explanation.

`#include <doxygen_c.h>`

Data Fields

- int a
- int b
- double c

3.1.1 Detailed Description

Use brief, otherwise the index won't have a brief explanation.

Detailed explanation.

3.1.2 Field Documentation

3.1.2.1 int BoxStruct_struct::a

Some documentation for the member BoxStruct::a.

3.1.2.2 int BoxStruct_struct::b

Some documentation for the member BoxStruct::b.

3.1.2.3 double BoxStruct_struct::c

Etc.

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

- src/doxygen_c.h

3.2 Dissector Struct Reference

Data Fields

- bool initialized
- const struct Dissector_ops ∗ ops
- Dissector_t ∗ calling

3.2.1 Field Documentation

3.2.1.1 Dissector_t∗ Dissector::calling

the dissector this dissector has been called from

3.2.1.2 bool Dissector::initialized

whether this dissector was initialized

3.2.1.3 const struct Dissector_ops∗ Dissector::ops

the dissectors operations

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

- src/Profinet/Dissector-int.h

3.3 Dissector_ops Struct Reference

Data Fields

- size_t **Dissector_size**
- unit64_t **lower**
- uint64_t **upper**
- void(∗ **Dissector_free**)(Dissector_t ∗dissector)
- Dissector_t ∗(∗ **Dissector_registerSub**)(Dissector_t ∗this, Dissector_t ∗subDissector, Interval interval)
- Dissector_t ∗(∗ **Dissector_getSub_64**)(Dissector_t ∗this, uint64_t data)
- Dissector_t ∗(∗ **Dissector_getSub_32**)(Dissector_t ∗this, uint32_t data)
- Dissector_t ∗(∗ **Dissector_getSub_16**)(Dissector_t ∗this, uint16_t data)
- Dissector_t ∗(∗ **Dissector_getSub_8**)(Dissector_t ∗this, uint8_t data)
- Dissector_t ∗(∗ **Dissector_getSub**)(Dissector_t ∗this, void ∗data, int len)
- int(∗ **Dissector_dissect**)(Dissector_t ∗this, Buffer_t ∗buf, ProtocolTree_t ∗tree)

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

- src/Profinet/Dissector-int.h

3.4 PNRTDissector Struct Reference

Data Fields

- struct Dissector **dissector**

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

- src/Profinet/PNRTDissector.c

Chapter 4

File Documentation

4.1 src/doxygen_c.h File Reference

File containing example of doxygen usage for quick reference.

```
#include <systemheader1.h>
#include <systemheader2.h>
#include <box/header1.h>
#include <box/header2.h>
#include "local_header1.h"
#include "local_header2.h"
```

Data Structures

- struct BoxStruct_struct

  Use brief, otherwise the index won't have a brief explanation.

Typedefs

- typedef enum BoxEnum_enum BoxEnum
- typedef struct BoxStruct_struct BoxStruct

  Use brief, otherwise the index won't have a brief explanation.

Enumerations

- enum BoxEnum_enum { BOXENUM_FIRST, BOXENUM_SECOND, BOXENUM_ETC }

Functions

- BOXEXPORT BoxStruct * Box_The_Function_Name (BoxParamType1 param1, BoxParamType2 param2)

  Example showing how to document a function with Doxygen.
- BOXEXPORT void * Box_The_Second_Function (void)

  A simple stub function to show how links do work.
- BOXEXPORT void Box_The_Last_One (void)
- void DissectorRegister_free (DissectorRegister_t *this)

4.1.1 Detailed Description

File containing example of doxygen usage for quick reference.

Author

My Self

Date

9 Sep 2012 Here typically goes a more extensive explanation of what the header defines. Doxygens tags are words preceeded by either a backslash \ or by an at symbol @.

See also

http://www.stack.nl/~dimitri/doxygen/docblocks.html
http://www.stack.nl/~dimitri/doxygen/commands.html

4.1.2 Typedef Documentation

4.1.2.1 typedef enum BoxEnum_enum BoxEnum

brief Use brief, otherwise the index won't have a brief explanation.

Detailed explanation.

4.1.2.2 typedef struct BoxStruct_struct BoxStruct

Use brief, otherwise the index won't have a brief explanation.

Detailed explanation.

4.1.3 Enumeration Type Documentation

4.1.3.1 enum BoxEnum_enum

brief Use brief, otherwise the index won't have a brief explanation.

Detailed explanation.

Enumerator

BOXENUM_FIRST Some documentation for first.

BOXENUM_SECOND Some documentation for second.

BOXENUM_ETC Etc.

4.1.4 Function Documentation

4.1.4.1 BOXEXPORT BoxStruct* Box_The_Function_Name (BoxParamType1 *param1,* BoxParamType2 *param2*)

Example showing how to document a function with Doxygen.

Description of what the function does. This part may refer to the parameters of the function, like `param1` or `param2`. A word of code can also be inserted like `this` which is equivalent to `this` and can be useful to say that the function returns a `void` or an `int`. If you want to have more than one word in typewriter font, then just use <tt>. We can also include text verbatim,

```
like this
```

Sometimes it is also convenient to include an example of usage:

```
1 BoxStruct *out = Box_The_Function_Name(param1, param2);
2 printf("something...\n");
```

Or,

```
1 pyval = python_func(arg1, arg2)
2 print pyval
```

when the language is not the one used in the current source file (but **be careful** as this may be supported only by recent versions of Doxygen). By the way, **this is how you write bold text** or, if it is just one word, then you can just do **this**.

Parameters

param1	Description of the first parameter of the function.
param2	The second one, which follows `param1`.

Returns

Describe what the function returns.

See also

Box_The_Second_Function
Box_The_Last_One
http://website/

Note

Something to note.

Warning

Warning.

4.1.4.2 BOXEXPORT void Box_The_Last_One (void)

Brief can be omitted. If you configure Doxygen with `JAVADOC_AUTOBRIEF=YES`, then the first Line of the comment is used instead. In this function this would be as if

```
@brief Brief can be omitted.
```

was used instead.

4.1.4.3 BOXEXPORT void* Box_The_Second_Function (void)

A simple stub function to show how links do work.

Links are generated automatically for webpages (like http://www.google.co.uk) and for structures, like BoxStruct_struct. For typedef-ed types use BoxStruct. For functions, automatic links are generated when the parenthesis () follow the name of the function, like Box_The_Function_Name(). Alternatively, you can use Box_↩ The_Function_Name.

Returns

`NULL` is always returned.

4.1.4.4 void DissectorRegister_free (DissectorRegister_t * *this*)

Frees the given DissectorRegister.

4.2 src/Profinet/Buffy.h File Reference

Functions

- uint8_t Buffy_get_bits8 (Buffy_t *buffy, unsigned int bit_offset, const int no_of_bits)
- uint16_t Buffy_get_bits16 (Buffy_t *buffy, unsigned int bit_offset, const int no_of_bits, const unsigned int encoding)
- uint32_t Buffy_get_bits32 (Buffy_t *buffy, unsigned int bit_offset, const int no_of_bits, const unsigned int encoding)
- uint64_t Buffy_get_bits64 (Buffy_t *buffy, unsigned int bit_offset, const int no_of_bits, const unsigned int encoding)

4.2.1 Detailed Description

Snort Preprocessor Plugin Header

This file gets included in plugbase.h when it is integrated into the rest of the program.

4.2.2 Function Documentation

4.2.2.1 uint16_t Buffy_get_bits16 (Buffy_t * *buffy,* unsigned int *bit_offset,* const int *no_of_bits,* const unsigned int *encoding*)

Get 1 - 16 bits returned in a uint16.

Parameters

buffy	the calling buffer
bit_offset	the offset for from the currenty buffer position
the	number of bits to be read

Returns

unsigned 16 bit value representing the specified bit range

4.2.2.2 uint32_t Buffy_get_bits32 (Buffy_t * *buffy,* unsigned int *bit_offset,* const int *no_of_bits,* const unsigned int *encoding*)

Get 1 - 32 bits returned in a uint32.

Parameters

buffy	the calling buffer
bit_offset	the offset for from the currenty buffer position
the	number of bits to be read

Returns

unsigned 32 bit value representing the specified bit range

4.2.2.3 uint64_t Buffy_get_bits64 (Buffy_t * *buffy,* unsigned int *bit_offset,* const int *no_of_bits,* const unsigned int *encoding*)

Get 1 - 64 bits returned in a uint64.

Parameters

buffy	the calling buffer
bit_offset	the offset for from the currenty buffer position
the	number of bits to be read

Returns

unsigned 64 bit value representing the specified bit range

4.2.2.4 uint8_t Buffy_get_bits8 (Buffy_t * *buffy*, unsigned int *bit_offset*, const int *no_of_bits*)

Get 1 - 8 bits returned in a uint8.

Parameters

buffy	the calling buffer
bit_offset	the offset for from the currenty buffer position
the	number of bits to be read

Returns

unsigned 8 bit value representing the specified bit range

4.3 src/Profinet/Dissector-int.h File Reference

Data Structures

- struct Dissector_ops
- struct Dissector

Functions

- struct Dissector **Dissector_new** (const struct Dissector_ops *ops)

Variables

- bool initialized
- const struct Dissector_ops * ops
- Dissector_t * calling

4.3.1 Detailed Description

This Header discribes the internal structure of the Dissector type. It defines the basic interface for operations.

4.3.2 Variable Documentation

4.3.2.1 Dissector_t* calling

the dissector this dissector has been called from

4.3.2.2 bool initialized

whether this dissector was initialized

4.3.2.3 const struct Dissector_ops* ops

the dissectors operations

4.4 src/Profinet/Dissector.h File Reference

Typedefs

- typedef struct Dissector **Dissector_t**

Functions

- Dissector_t * Dissector_new (const struct dissector_ops *ops)
- void Dissector_free (Dissector_t *dissector)
- Dissector_t * Dissector_registerSub (Dissector_t *this, Dissector_t *subDissector)
- Dissector_t * Dissector_getSub (Dissector_t *this, uint64_t data)
- int Dissector_dissect (Dissector_t *this, Buffer_t *buf, ProtocolTree_t *tree)

4.4.1 Detailed Description

The Base Dissector abstraction. Every implementation of a Dissector will use and implement the operations described in this interface. Dissector are used to dissect certain ranges of data in a network package, while having the possibility to link to further dissectors when the dissection of the desired range is complete.

-> It is possible to link several Dissectors together building a tree of dissectors and subdissectors that call each other when their dissection part is completed.

4.4.2 Function Documentation

4.4.2.1 int Dissector_dissect (Dissector_t * *this*, Buffer_t * *buf*, ProtocolTree_t * *tree*)

Dissects the package the given buffer is pointing to.

Parameters

this	the calling Dissector
buf	the buffer pointing to the package data currently being processed
tree	the tree strcture to save the package data in

Returns

0 if the dissection was successful wihtout any failures, -1 if it was a faulty package. The fault flag will be set in the ProtocolTree accordingly

4.4.2.2 void Dissector_free (Dissector_t * *dissector*)

Frees the given dissector.

4.4.2.3 Dissector_t* Dissector_getSub (Dissector_t * *this*, uint64_t *data*)

Returns the sub dissector that is register for the given unsigned long.

Parameters

this	the dissector calling Dissector_getSub
data	the value for looking up in the dissector register

Returns

the registered sub dissector if any, NULL otherwise

4.4.2.4 Dissector_t* Dissector_new (const struct dissector_ops * *ops*)

Creates a new Dissector with the given operations. This Function is the interface constructor for every Dissector implementation.

Parameters

ops	the pointer to the operations used for this dissector

Returns

a pointer to the created dissector

4.4.2.5 Dissector_t* Dissector_registerSub (Dissector_t * *this,* Dissector_t * *subDissector*)

Registers a given sub dissector on this dissector.

Parameters

this	the dissector to register the subDissector on
subDissector	the dissector to be registered as sub

Returns

NULL if there was no other dissector registered for the given interval otherwise the Dissector will be overwritten and returned.

4.5 src/Profinet/DissectorRegister.h File Reference

Typedefs

- typedef struct DissectorRegister **DissectorRegister_t**

Functions

- DissectorRegister_t * DissectorRegister_new (const struct DissectorRegister_ops *ops)
- void DissectorRegister_free (DissectorRegister_t *this)
- Dissector_t * DissectorRegister_insert (DissectorRegister_t *this, DissectorRegister_t *dissector)
- Dissector_t * DissectorRegister_get (DissectorRegister_t *this, uint64_t data)

4.5.1 Detailed Description

The dissector register is used to register dissectors to intervals. Thereby making it possible to dissect a package while using certain data ranges for calling a next dissector that is mapped to the given data.

4.5.2 Function Documentation

4.5.2.1 void DissectorRegister_free (DissectorRegister_t * *this*)

Frees the given DissectorRegister.

4.5.2.2 Dissector_t* DissectorRegister_get (DissectorRegister_t * *this*, uint64_t *data*)

Returns the sub DissectorRegister that is register for the given unsigned long.

Parameters

this	the DissectorRegister calling
data	the value for looking up in the DissectorRegister

Returns

the registered Dissector if any, NULL otherwise

4.5.2.3 Dissector_t* DissectorRegister_insert (DissectorRegister_t * *this*, DissectorRegister_t * *dissector*)

Inserts a new Dissector for the given interval. If a dissector is allready mapped

Parameters

this	the calling register
the	interval the given dissector will be mapped to
dissector	the dissector to be inserted

Returns

NULL if there is no previous dissector registered within the given interval, otherwise overwrites the old dissector and returns it

4.5.2.4 DissectorRegister_t* DissectorRegister_new (const struct DissectorRegister_ops * *ops*)

Creates a new DissectorRegister with the given operations. This Function is the interface constructor for every DissectorRegister implementation.

Parameters

ops	the pointer to the operations used for this DissectorRegister

Returns

a pointer to the created DissectorRegister

4.6 src/Profinet/Sender.h File Reference

Typedefs

- typedef struct Sender **Sender_t**

Functions

- Sender_t * Sender_new (const struct sender_ops *ops)
- void Sender_free (Sender_t *dissector)
- int Sender_send (Sender_t *this, Truffle_t *truffle)

4.6.1 Detailed Description

The basic Sender abstraction. Every implementation of a Sender will use and implement the operations described in this interface. A Sender is used to send truffles to a certain port, socket, or messagequeue, depending on the implemented send operation.

-> It is possible to link several Dissectors together building a tree of dissectors and subdissectors that call each other when their dissection part is completed.

4.6.2 Function Documentation

4.6.2.1 void Sender_free (Sender_t * *dissector*)

Frees the given dissector.

4.6.2.2 Sender_t* Sender_new (const struct sender_ops * *ops*)

Creates a new Dissector with the given operations. This Function is the interface constructor for every Dissector implementation.

Parameters

ops	the pointer to the operations used for this dissector

Returns

a pointer to the created dissector

4.6.2.3 int Sender_send (Sender_t * *this*, Truffle_t * *truffle*)

Sends the given truffle to the specified ipc

Parameters

this	the calling sender
truffle	the truffle to be send

Returns

0 if the sending was successful, -1 if no client is detected for receiving, or on other errors.

4.7 src/Profinet/Truffle.h File Reference

Typedefs

- typedef struct Truffle **Truffle_t**

Functions

- Truffle_t * Truffle_new ()
- Entry_t * Truffle_get (Truffle_t *this, char *key)
- Entry_t * Truffle_put (Truffle_t *this, char *key, Entry_t entry)

4.7.1 Detailed Description

Truffle header file

4.7.2 Function Documentation

4.7.2.1 Entry_t* Truffle_get (Truffle_t * *this*, char * *key*)

Returns the entry that is mapped to the given key in this truffle.

Parameters

this	the calling truffle
key	the key to be looking for

4.7.2.2 Truffle_t* Truffle_new ( )

Creates a new truffle that can be filled with data from the package.

4.7.2.3 Entry_t* Truffle_put (Truffle_t * *this*, char * *key*, Entry_t *entry*)

Inserts the given entry into the truffle mapped to the key.

Parameters

this	the calling truffle
key	the value to map the entry to
entry	the entry to be put into the truffle

4.8 src/spp_profinet.c File Reference

Functions

- void SetupProfiNet ()
- void DissectorInit ()

Variables

- DissectorRegister_t * tlRegister
- Sender_t * sender

4.8.1 Detailed Description

Id Snort Preprocessor Plugin Source File ProfiNet Purpose:

Preprocessors perform some function *once* for *each* packet. This is different from detection plugins, which are accessed depending on the standard rules. When adding a plugin to the system, be sure to add the "Setup" function to the InitPreprocessors() function call in plugbase.c!

Arguments:

This is the list of arguements that the plugin can take at the "preprocessor" line in the rules file

Effect:

What the preprocessor does. Check out some of the default ones (e.g. spp_frag2) for a good example of this description.

Comments:

Any comments?

4.8.2 Function Documentation

4.8.2.1 void DissectorInit ( )

Initializes the dissectors for the profinet protocols.

4.8.2.2 void SetupProfiNet ( )

Registers the preprocessor keyword and initialization function into the preprocessor list. This is the function that gets called from InitPreprocessors() in plugbase.c.

4.8.3 Variable Documentation

4.8.3.1 Sender_t* sender

The ipc sender.

4.8.3.2 DissectorRegister_t* tlRegister

The top level dissector register.

4.9 src/spp_profinet.h File Reference

Functions

- void SetupProfiNet ()

4.9.1 Detailed Description

Snort Preprocessor Plugin Header

This file gets included in plugbase.h when it is integrated into the rest of the program.

4.9.2 Function Documentation

4.9.2.1 void SetupProfiNet ( )

list of function prototypes to export for this preprocessor

Registers the preprocessor keyword and initialization function into the preprocessor list. This is the function that gets called from InitPreprocessors() in plugbase.c.

Index