

# **SECURITY ORIENTED TCP PARSER FOR PCAP FILES**

## **Team Members**

**Surendra Sharma (459205)**

**Rajesh Mehta (459194)**

**T Ramu (459204)**

# **Agenda**

- 1. Aim of the project**
- 2. Assumptions & Boundary conditions**
- 3. Design Architecture/Function description**
- 4. Secure coding principles**
- 5. Tests & Results**
- 6. Discussions**

# Aim of Project

To design and implement a TCP packet parser for extracting values in header fields / application data of valid TCP packets and output on stdout and file

## Development Environment / Tools

1. Ubuntu Linux (14.xx/16.xx) / QT Creator 3.5.1
2. Language C, C-std Library
3. GNU CC, GDB
4. Wireshark (Data Capture and analysis)
5. Stored PCAP files (Restricted types)
6. Testing – TCP/File Fuzzers

# Assumptions & Boundary Conditions

1. No live Packet processing / only stored PCAP files used
2. Input file as command line arg, output default - parserdata.txt
3. File validity – Magic numbers of selected formats of PCAP  
( 0xD4C3B2A1 & 0x34CDB2A1)
4. Input limited to sizeof (int) number of input frames, Max Frame size = 1600 Bytes
5. Processing of Packets of IPv4 std
6. Tested with Fuzzed input on TCP headers /data

# Design Architecture / Functional Description

1. Three categories of Functionalities :-
  - File Input and File level checks/Validation
  - Framing/packetization of file stream data in Ethernet/TCP/IP header structures
  - Validation of Ethernet/TCP/IP headers and Extraction of header field elements
2. Modular approach with small functions for specific/small jobs
3. Only main Function and functions pertaining to validation and header data extraction in “Main.c” file
4. All auxiliary functions requiring no access to global variables/data of main shifted to “functions.c”

# Secure Coding Principles

1. Simple Design and Implementation
2. Modular code in smaller specific job oriented functions
3. No trust in user file input :-
  - Sanitisation/validation checks of FileName length
  - Detection/Escaping special input characters
  - Forced String termination - appending '\0' in filename
4. Compartmentalization – Modular, Auxiliary functions separated from main and integral function
5. Least privilege – No system level function called / output File writing & location not accepted from the user
6. Default Deny/Fail on detection of Invalidity first before processing in function

# Tests And Results

## Execution of TCPparser

```
File Edit View Search Terminal Help
|-- Sequence Number : 3973
|-- Acknowledgement Number : 62014
|-- TCP Data Offset : 20
|-- Flags : 0x0018
|-- TCP Window Size : 8190
|-- Checksum : 0xe368
|-- Urgent Field value : 0
App Payload Size 204
Application Payload (204 bytes):
00000 d4 f3 8a 9e 53 58 49 46 a9 eb 4c fb 3d 67 cf f2 ...SXIF..L.=g..
00016 e5 c9 38 77 d5 3f 30 18 19 0c ec 96 3f 36 53 40 ...8w.70.....?6S@
00032 31 2f f5 ea ff 48 29 43 32 f0 72 64 c4 b2 36 1b 17/...H)C2.rd..6.
00048 97 ee 1b 01 7b 22 9a bf bf bc 5e c5 80 20 22 09 {"...^..".
00064 2c 6f 65 6b af e5 bf a6 67 92 0b 70 e3 7c 37 f8 ,oek...g..p.|7.
00080 2c be 00 b1 b9 b5 5a ef 48 5b 06 ec f4 75 57 23 In function 'ProcessFrame':
00096 73 a3 0a 15 c5 ea a3 9d 6b b6 32 83 c1 13 0e 5b comparison between signed and unsigned integer expressions [-Wsign-compare]
00112 06 1e 5e bb b7 30 80 4f ef e8 1b cc 24 3d d3 e8 format '%x' expects argument of type 'unsigned int', but argument 2 has type 'unsigned int' [-Wformat]
00128 c5 f8 30 7e c9 21 3e 2c dd e8 40 42 18 a3 c7 21 comparison between signed and unsigned integer expressions [-Wsign-compare]
00144 f4 51 14 22 62 a4 61 ce f7 fc 96 31 14 86 52 93 comparison between signed and unsigned integer expressions [-Wsign-compare]
00160 c6 ee 2c ce 83 38 b7 5c 2b 8b ea 9c ea 9e 6d 91 comparison between signed and unsigned integer expressions [-Wsign-compare]
00176 42 da 1b cd a6 5b b8 d9 d1 95 72 ed 16 b0 72 55 In function 'main':
00192 ab 45 c3 61 95 37 d4 99 23 6b 1a d2 ordered comparison of pointer with integer zero [-Wextra]
.E.a.7...#k..
control reaches end of non-void function [-Wreturn-type]
The previous frame was processed...Going for next Frame
FrameLength1: 54
FrameLength2: 54
Processing Frame
Frame_Header 15a2960
```

# Tests And Results

## Output of Memory Leak tool - Valgrind

```
surendra@LXBOX-SS: /media/surendra/DATA/MasarykCourse/6_PA193-SecureCoding_8/Parser_Project/Parser_5/parser_5
File Edit View Search Terminal Help
Pro...  main.c  # Line: 550, Col: 14
*** SUMMARY OF PARSER DATA ***
TotalFrame=1 FrameEscape= 0
Total Frames Analysed = 1
Total BadTCPSegment = 0
Total valid IP Packets Processed = 1
Total valid TCP Segments Processed= 0
Total NonTCP Segments= 0
Total ErrorFrameCounter = 0
!!!EXITING...GOOD-BYE!!!!
==13329==
==13329== FILE DESCRIPTORS: 3 open at exit.
==13329== Open file descriptor 2: /dev/pts/16
==13329== <inherited from parent>
==13329== Open file descriptor 1: /dev/pts/16
==13329== <inherited from parent>
==13329== Open file descriptor 0: /dev/pts/16
==13329== <inherited from parent>
==13329== HEAP SUMMARY:
==13329==   in use at exit: 0 bytes in 0 blocks
==13329== total heap usage: 8 allocs, 8 frees, 12,120 bytes allocated
==13329== All heap blocks were freed -- no leaks are possible
==13329== For counts of detected and suppressed errors, rerun with: -v
==13329== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
surendra@LXBOX-SS:/media/surendra/DATA/MasarykCourse/6_PA193-SecureCoding_8/Parser_Project/Parser_5/parser_5$ ^C
surendra@LXBOX-SS:/media/surendra/DATA/MasarykCourse/6_PA193-SecureCoding_8/Parser_Project/Parser_5/parser_5$ ^C
surendra@LXBOX-SS:/media/surendra/DATA/MasarykCourse/6_PA193-SecureCoding_8/Parser_Project/Parser_5/parser_5$
```

# Tests And Results

## Summary of results for Different file types and sizes

\*\*\*\*\* SUMMARY OF PARSER DATA testFile-2.pcap \*\*\*\*\*

TotalFrame=45605      FrameEscape= 8      Total ErrorFrameCounter = 19  
Total Frames Analysed = 45578  
Total valid IP Packets Processed = 45578      Total Invalid IP datagrams = 19  
Valid TCP Segments Processed= 44013      Total NonTCP Segments= 1565      Total BadTCPSegment  
= 0

\*\*\*\*\* SUMMARY OF PARSER DATA testFile-1.pcap \*\*\*\*\*

TotalFrame=6306      FrameEscape= 0      Total ErrorFrameCounter = 7  
Total Frames Analysed = 6299  
Total valid IP Packets Processed = 6299      Total Invalid IP datagrams = 7  
Valid TCP Segments Processed= 6012      Total NonTCP Segments= 287      Total BadTCPSegment = 0

\*\*\*\*\* SUMMARY OF PARSER DATA slammer.pcap\*\*\*\*\*

TotalFrame=1      FrameEscape= 0      Total ErrorFrameCounter = 0  
Total Frames Analysed = 1  
Total valid IP Packets Processed = 1      Total Invalid IP datagrams = 0  
Valid TCP Segments Processed= 0      Total NonTCP Segments= 1      Total BadTCPSegment = 0

\*\*\*\*\* SUMMARY OF PARSER DATA test.cap\*\*\*\*\*

TotalFrame=4      FrameEscape= 0      Total ErrorFrameCounter = 0  
Total Frames Analysed = 4  
Total valid IP Packets Processed = 4      Total Invalid IP datagrams = 0  
Valid TCP Segments Processed= 4      Total NonTCP Segments= 0      Total BadTCPSegment = 0

\*\*\*\*\* SUMMARY OF PARSER DATA (Data FUzzed )FuzOfIwFile-1.pcap\*\*\*\*\*

TotalFrame=6306      FrameEscape= 0      Total ErrorFrameCounter = 7  
Total Frames Analysed = 6299  
Total valid IP Packets Processed = 6299      Total Invalid IP datagrams = 7  
Valid TCP Segments Processed= 6012      Total NonTCP Segments= 287      Total BadTCPSegment = 0

# **DISCUSSIONS**



## Secure Coding Principles – Validation Function

```
CheckFrameLength(unsigned char* FrameHeader);  
CheckLinkLayerProtocol(unsigned char* FileHeader);  
CheckMagicNumber(unsigned char* FileHeader);  
CheckVersionNumber(unsigned char* FileHeader);  
ValidateFileName(int argc, char *argv[]);
```

# Design Architecture / Functional Description

Functional Breakdown of Parser to Read .PCAP type of file created by TcpDUMP for analysis

```
// Library includes including pcap.h from libpcap library
```

```
#include <inet.h>
```

```
//-----
```

```
define MACROS
```

```
//Declaration of structures of Packet headers - ----
```

```
/* e.g. Eth_header/ IP_header/TCP_header */
```

```
struct ethernet {
```

```
    //-----
```

```
};
```

```
//// FUNCTION DEFINITIONS for
```

```
/*
```

1. Open the PCAP file to read. Open another file for writing the parser output (may be in parallel to std out)
2. Function to sanitise / validate the input files.
  - (a) (check the validity of type of file - reqd to study the PCAP header struct)
  - (b) Verify the file header block for correct PCAP file type
  - (c) Check the Frame header before commencement for every packet ethernet frme for total num of bytes in packet
  - (d) Reader each packet and call function to extract indivial headers from packet
3. Function to sanitize the input packet for correct type and lenght
  - (a) receive input packet in pointer and length
  - (b) strip individual header and parse into the header structure individually
  - (c) verify the header length/checksum
  - (d) call function to extract data / validate and print the data from structure pointers
4. Call function to check the headertype and extract relevant info
  - (a) Receive headers into individual structure pointer for Ethernet , IP and TCP headers
  - (b) Validate/chack/Sanitise individual headers for expected information in header fields
  - (c) extract relevant info of field from the individual headers and print the info. Print detailed info for TCP header. Print summary info of packet if UDP packet is received.
  - (d) Write the all packet info to std out and output file including TCP payload .

# Design Architecture / Functional Description

```
void function_readSanitiseInputfile() {  
/* Function to sanitise / validate the input files.  
  (a) (check the validity of type of file - reqd to study the PCAP header struct)  
  (b) Verify the file header block for correct PCAP file type  
  (c) Check the Frame header before commencement for every packet ethernet frme for total num of bytes in packet  
  (d) Reader each packet and call function to extract indivial headers from packet  
}
```

```
void function_extractHeadersFromPackets() {  
/* Function to sanitize the input packet for correct type and lenght  
  (a) receive input packet in pointer and length  
  (b) strip individual header and parse into the header structure individually  
  (c) verify the header length/checksum  
*/ (d) call function to extract data / validate and print the data from structure pointers  
}
```

```
void function_extractCheckHeaderInfo_writetoFile() {  
/* Call function to check the headertype and extract relevant info  
  (a) Receive headers into individual structure pointer for Ethernet , IP and TCP headers  
  (b) Validate/chack/Sanitise individual headers for expected information in header fields  
  (c) extract relevant info of field from the individual headers and print the info. Print detailed info for TCP header. Print  
  summary info of packet if UDP packet is received.  
*/ (d) Write the all packet info to std out and output file including TCP payload .  
}  
int main(int argc, char **argv)  
{
```

```
FILE f_in*; f_in = fopen("input_file", 'r');  
FILE f_out*; f_in = fopen("output_file", 'w');  
function_readSanitiseInputfile()  
function_extractHeadersFromPackets()  
function_extractCheckHeaderInfo_writetoFile();
```