

Pre-engineer Internship Report

akoudanilo@gmail.com

OSID: 15P080

2019-11-04

Contents

1	Preamble	5
1.1	Acknowledgement	5
1.2	Sigles et abbreviations	5
1.3	Abstract	5
1.4	Résumé	6
1.5	General Introduction	6
1.6	Presentation of Google Summer of Code	6
1.6.1	Overview	6
1.6.2	How it works	7
1.6.3	History	7
1.6.3.1	GSoC in numbers	8
2	State of the art of capture files diff tools	11
2.1	Introduction	11
2.1.1	Problematic	11
2.2	Classical approach: view each file individually	11
2.3	Middle approach: generate diffs using the text output of wireshark	11
2.4	Our approach: Use the PDML output of Wireshark to make diffs	12
2.4.1	Text	12
2.4.2	Json	12
2.4.3	XML (PDML)	14
3	Design of the solution	15
3.1	Analysis	15
3.1.1	Requirements engineering	15
3.1.1.1	Functionnal requirements	15
3.1.1.2	Non-functional requirements	15
3.1.2	Use case diagram (GUI)	15
3.1.3	Class diagram	17
3.2	Technologies used	19
3.2.1	For the command line interface	19
3.2.2	Graphical User Interface	21
4	Implementation	22
4.1	Algorithm 1: recursive diffs between 2 nodes of the XML graph	22
4.1.1	Principle	22
4.1.2	Pseudo-code	22
4.1.3	Complexity analysis	22

4.1.3.1	Time complexity	23
4.1.3.2	Space complexity	23
4.2	Algorithm 2: Circular search inside a non-circular list	23
4.2.1	Principle	23
4.2.2	Pseudo code	23
4.2.3	Complexity analysis	24
4.2.3.1	Time complexity	24
4.2.3.2	Space complexity	24
5	Evaluation of results	25
5.1	Diffs	25
5.1.1	Different command fields	25
5.1.2	Same command fields	27
5.2	GUI	29
5.2.1	Windows	29
5.2.2	Linux	30
6	Prospect	32
6.1	Add documentation for config file and usage	32
6.2	Install documentation and desktop file in packaging	32
6.3	Pass down arguments passed to windows launcher to the Python scripts	32
6.4	Implement ignored fields in smbcmp-gui	32
7	Conclusion	34
8	Appendix	35
8.1	Appendix A - Proposal	35
8.1.1	Introduction	35
8.1.2	Project Goals	35
8.1.2.1	Use or combine current tshark output with the XML output	35
8.1.2.1.1	Add an html output	35
8.1.2.1.2	Deliverable	35
8.1.2.2	Make smbcmp highlight diffs from the packet summary listing	36
8.1.2.2.1	Deliverable	36
8.1.2.3	Automate the creation of .smbcmp file	36
8.1.2.3.1	Deliverable	36
8.1.2.4	Make a proper delivery way	36
8.1.2.4.1	Deliverable	36
8.1.2.5	Correct soome flaws of the tool	36
8.1.2.5.1	smbcmp throws an error if there is no samba packet in the pcap file . .	36
8.1.2.5.2	smbcmp close unexpectedly after resizing to critical values	36
8.1.3	Timeline	36
8.1.3.1	Community Bonding Period[May 6 - May 26]	37
8.1.3.2	Coding officially begins [May 27]	37
8.1.3.2.1	June 7 - June 28 (First second and third Weeks)	37
8.1.3.2.2	June 28 - July 26 (Weeks 4 - 7)	37
8.1.3.2.3	July 26 - August 19 (Week 7 - end)	37
8.1.4	About Me	37
8.1.4.1	Name	37
8.1.4.2	Email	37

<i>CONTENTS</i>	3
8.1.4.3 Github	37
8.1.4.4 Phone Number	37
8.1.4.5 Summary	38
8.1.4.6 Contributions to OSS :	38
8.1.4.6.1 Merged PR	38
8.1.4.6.2 Unmerged PR	38
8.2 Appendix B - Recursive PDML diffs algorithm	38
Bibliography	42

List of Figures

1.1	Google Summer of Code Logo	6
2.1	Textual output Wireshark	12
2.2	Json output Wireshark	13
2.3	XML output Wireshark	14
3.1	Use case diagram smbcmp-gui	16
3.2	Class diagram	18
5.1	Plain text output for different command fields	26
5.2	PDML output for different command fields	27
5.3	Plain text output for same command fields	28
5.4	PDML output for same command fields	29
5.5	Windows view smbcmp	30
5.6	Linux view smbcmp	31

Chapter 1

Preamble

1.1 Acknowledgement

I would like to express my deepest appreciation to all those who provided me the possibility to complete this report. First thing first my parents, without who I wouldn't even exist in the first place and logically God Who makes all things possible. A special gratitude goes to my mentor, Mr Aurelien Aptel for his time taken to explain things to me, for cleaning up and correcting errors that I might have missed otherwise. Furthermore I would also like to acknowledge with much appreciation the crucial role of my teachers, especially Mr KOUAMOU Georges, who gave me the basics in algorithms needed to achieve my work, specifically the algorithms I had to develop during this internship. A special thanks goes to my family, friends and classmates who helped me in their own ways by providing support, empathy and courage to go through all that.

1.2 Sigles et abbreviations

- **SMB**: Server Message Block
- **PDML**: Packet Description Markup Language
- **GSoC**: Google Summer of Code
- **cET**: cElementTree
- **CLI**: Command Line Interface
- **GUI**: Graphical User Interface
- **CIFS**: Common Internet File System

1.3 Abstract

Smbcmp is a CLI tool for making diffs between two pcap files containing SMB packets and rendering them using ncurses. The first part of the project consisted in making better diffs by using the PDML output of Tshark and the second part consisted in adding a GUI and port it to windows. More details in the Appendix A

1.4 Résumé

Smbcmp est un outil en ligne de commandes permettant de visualiser les différences entre deux fichiers pcap contenant des paquets SMB et de les afficher à l'aide de la bibliothèque ncurses . La première partie du projet consistait à améliorer les différences en utilisant la sortie PDML de Tshark et la seconde à ajouter une interface graphique et à créer une version pour Windows. Plus de détails sur la proposition dans l'appendice A

1.5 General Introduction

SMB is a protocol providing shared access to files, printers, and serial ports between nodes on a network, one version of SMB was known as CIFS. It also provides an authenticated inter-process communication mechanism. Most usage of SMB involves computers running Microsoft Windows, where it was known as “Microsoft Windows Network” before the introduction of Active Directory. Knowing how to troubleshoot connectivity problems can be cumbersome, hence network sniffers like Wireshark/Tshark and many others. While this may be considered as good enough by many, the problem of comparison arise very quickly, i.e. the ability to know between two situations what exactly went wrong, an use case that Wireshark/Tshark can't handle properly that's why smbcmp has been created.

1.6 Presentation of Google Summer of Code



Google Summer of Code

Figure 1.1: Google Summer of Code Logo

1.6.1 Overview

Google Summer of Code is a global initiative focused on bringing more students (undergrads, grads and post grads), mainly developers into open source software development. Students work with an open source organization on a 3 months programming project during their break from school, typically during the summer.

1.6.2 How it works

As a part of Google Summer of Code, student participants are paired with a mentor from the participating organizations, gaining exposure to real-world software development and techniques. Students have the opportunity to spend the break between their school semesters earning a stipend while working in areas related to their interests. Practically, we have the following relationship.

Students contact the mentor organizations they want to work with and write up a project proposal for the summer. If accepted, students spend a month integrating with their organizations prior to the start of coding. Students then have three months to code, meeting the deadlines agreed upon with their mentors.

Open source projects apply to be mentor organizations. Once accepted, organizations discuss possible ideas with students and then decide on the proposals they wish to mentor for the summer. They provide mentors to help guide each student through the program.

Existing contributors with the organizations can choose to mentor a student project. Mentors and students work together to determine appropriate milestones and requirements for the summer. Mentor interaction is a vital part of the program.

In turn, the participating organizations are able to identify and bring in new developers who implement new features and hopefully continue to contribute to open source even after the program is over. Most importantly, more code is created and released for the use and benefit of all.

1.6.3 History

The idea for the Summer of Code came directly from Google's founders, Sergey Brin and Larry Page. ("Google Summer of Code") From 2007 until 2009 Leslie Hawthorn, who has been involved in the project since 2006, was the program manager. From 2010 until 2015, Carol Smith was the program manager. In 2016, Stephanie Taylor took over management of the program. In 2005, more than 8,740 project proposals were submitted for the 200 available student positions. Due to the overwhelming response, Google expanded the program to 419 positions.

The mentoring organizations were responsible for reviewing and selecting proposals, and then providing guidance to those students to help them complete their proposal. Students that successfully completed their proposal to the satisfaction of their mentoring organization were awarded a stipend and a Google Summer of Code T-shirt, while \$500 per project was sent to the mentoring organization. Approximately 80% of the projects were successfully completed in 2005, although completion rates varied by organization: Ubuntu, for example, reported a completion rate of only 64%, and KDE reported a 67% completion rate. Many projects were continued past summer, even though the SOC period was over, and some changed direction as they developed.

For the first Summer of Code, Google was criticized for not giving sufficient time to open source organizations so they could plan projects for the Summer of Code. Despite these criticisms there were 41 organizations involved, including FreeBSD, Apache, KDE, Ubuntu, Blender, Mozdev, and Google it

According to a blog post by Chris DiBona, Google's open source program manager, "something like 30 percent of the students stuck with their groups past SoC [Summer of Code]." Mozilla developer Gervase Markham also commented that none of the 10 Google-sponsored Mozilla projects survived after the event. However, the Gaim (now Pidgin) project was able to enlist enough coding support through the event to include the changes into Gaim (now Pidgin) 2.0; the Jabber Software Foundation (now XMPP Standards Foundation) and KDE project also counted a few surviving projects of their own from the event (KDE only counted 1 continuing project from out of the 24 projects which it sponsored). For the following years, the program kept growing with more and more applicants among organizations and students, we have the following metrics ("Statistics Google Summer of Code")

1.6.3.1 GSoC in numbers

- 2019

Coding Dates: May 27th - August 19th

1,276 students accepted from 63 countries

2,000+ mentors with active projects from 72 countries

201 open source organizations

89.05% overall success rate

- 2018

Coding Dates: May 14th - August 14th

1,264 students accepted from 62 countries

2,117 mentors with active projects from 73 countries

206 open source organizations

86.24% overall success rate

- 2017

Coding Dates: May 30th - August 29th

1,318 students accepted from 72 countries

1,647 mentors with active projects from 69 countries

3,439 registered mentors

198 open source organizations

86.2% overall success rate

- 2016

Coding Dates: May 23rd - August 23rd

1,206 students accepted from 67 countries

2,524 mentors from 66 countries

178 open source organizations

85.6% overall success rate

- 2015

Coding Dates: May 25th - August 21st

1,051 students accepted from 73 countries

1,903 mentors from 68 countries

137 open source organizations

88.2% overall success rate

- 2014

Coding Dates: May 19th - August 18th

1,307 students accepted from 72 countries

2,491 mentors from 78 countries

190 open source organizations

89.7% overall success rate

- 2013

Coding Dates: June 17th - September 27th

1,192 students accepted from 70 countries

2,212 mentors from 70 countries

177 open source organizations

88.9% overall success rate

- 2012

Coding Dates: May 21st - August 20th

1,212 students accepted from 69 countries

2,220 mentors from 66 countries

180 open source organizations

88.5% overall success rate

- 2011

Coding Dates: May 23rd - August 22nd

1,115 students accepted from 68 countries

2,096 mentors from 55 countries

175 open source organizations

88% overall success rate

- 2010

Coding Dates: May 24th - August 16th

1,026 students accepted from 69 countries

2,045 mentors from 52 countries

150 open source organizations

89% overall student success rate

- 2009

Coding Dates: May 23rd - August 25th

1,000 students accepted from 70 countries

1,991 mentors from 65 countries

150 open source organizations

85% overall student success rate

- 2008

Coding Dates: May 26th - August 18th

1,126 students accepted from 64 countries

2,164 mentors from 66 countries

175 open source organizations

83% overall student success rate

- 2007

Coding Dates: May 28th - August 31st

905 students accepted from 62 countries

1,819 mentors from 75 countries

135 open source organizations

81% overall student success rate

- 2006

Coding Dates: May 1st - September 8th

630 students accepted from 55 countries

1,252 mentors from 57 countries

102 open source organizations

82% overall student success rate

In Cameroon, the university of Buea is the most represented institution with 37 students coming from it since the beginning of the program in 2015.

Chapter 2

State of the art of capture files diff tools

2.1 Introduction

Let's say we have a network of computers with active directory configured, for one reason or another, when we try to enable one feature of active directory the connectivity seems to have performance issues. The issues can be that the network is either failing or too slow and we have two capture files obtained using any capture tool (wireshark, tcpdump, ...) of the two states of the network. One question we can ask is how can we analyze what changed in the network between the activation of the feature and when everything worked perfectly ?

2.1.1 Problematic

We want to visualize the diffs between 2 capture files and we are only interested in the SMB packets.

2.2 Classical approach: view each file individually

Thanks to the open source community effort we have a very powerful network analyzer: wireshark. There are other proprietary solutions, but generally they lack many features that wireshark have. The idea is to find any misconfigured thing by browsing manually the first capture file then the second one, and since what interest us here is the SMB protocol, we will use the smb or smb2 filters depending on which version of the protocol we are using.

2.3 Middle approach: generate diffs using the text output of wireshark

There are many programs like that which typically rely on the textual output of wireshark, and use a text differ to highlight changes, on Unix like systems a good diffs utility is `diff` present in the GNU coreutils. One example of a program like that is pcap-diff (isginf).

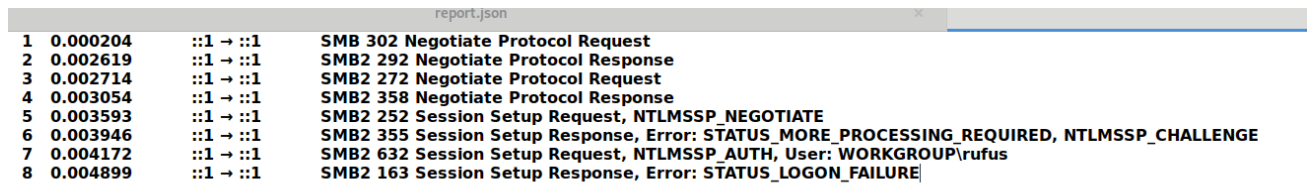
2.4 Our approach: Use the PDML output of Wireshark to make diffs

Knowing the advantages and the disadvantages of the two previous approaches, we tried to address all this using the XML output of Wireshark/Tshark in fact, there are typically 3 interesting, human-readable formats:

2.4.1 Text

Each line is a packet and sub-packets are indented by one tab. We can obtain it with the command:

```
tshark -2 -R "smb or smb2" -r /path/of/capture/file
```



No.	Time	Source	Destination	Protocol	Description
1	0.000204	::1	::1	SMB	302 Negotiate Protocol Request
2	0.002619	::1	::1	SMB2	292 Negotiate Protocol Response
3	0.002714	::1	::1	SMB2	272 Negotiate Protocol Request
4	0.003054	::1	::1	SMB2	358 Negotiate Protocol Response
5	0.003593	::1	::1	SMB2	252 Session Setup Request, NTLMSSP_NEGOTIATE
6	0.003946	::1	::1	SMB2	355 Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHALLENGE
7	0.004172	::1	::1	SMB2	632 Session Setup Request, NTLMSSP_AUTH, User: WORKGROUP\rufus
8	0.004899	::1	::1	SMB2	163 Session Setup Response, Error: STATUS_LOGON_FAILURE

Figure 2.1: Textual output Wireshark

2.4.2 Json

The json file is an array of objects and each object is a packet, each packet contains many key/value pairs corresponding to a field or a subfield of the protocol used (header, flags, length, ...). We can obtain it with the command:

```
tshark -2 -R "smb or smb2" -r /path/of/capture/file -T json
```

```

report.json
[
  {
    "_index": "packets-2019-06-12",
    "_type": "pcap_file",
    "_score": null,
    "_source": {
      "layers": {
        "frame": {
          "frame.encap_type": "1",
          "frame.time": "Jun 12, 2019 16:32:06.239260000 WAT",
          "frame.offset_shift": "0.000000000",
          "frame.time_epoch": "1560353526.239260000",
          "frame.time_delta": "0.000000000",
          "frame.time_delta_displayed": "0.000000000",
          "frame.time_relative": "0.000204000",
          "frame.number": "1",
          "frame.len": "302",
          "frame.cap_len": "302",
          "frame.marked": "0",
          "frame.ignored": "0",
          "frame.protocols": "eth:ethertype:ipv6:tcp:nbss:smb"
        },
        "eth": {
          "eth.dst": "00:00:00:00:00:00",
          "eth.dst_tree": {
            "eth.dst_resolved": "00:00:00_00:00:00",
            "eth.addr": "00:00:00:00:00:00",
            "eth.addr_resolved": "00:00:00_00:00:00",
            "eth.lg": "0",
            "eth.lg": "0"
          },
          "eth.src": "00:00:00:00:00:00",
          "eth.src_tree": {
            "eth.src_resolved": "00:00:00_00:00:00",
            "eth.addr": "00:00:00:00:00:00",
            "eth.addr_resolved": "00:00:00_00:00:00",
            "eth.lg": "0",
            "eth.lg": "0"
          },
          "eth.type": "0x000086dd"
        },
        "ipv6": {
          "ipv6.version": "6",
          "ip.version": "6",
          "ipv6.tclass": "0x00000000",
          "ipv6.tclass_tree": {
            "ipv6.tclass.dscp": "0",
            "ipv6.tclass.ecn": "0"
          },
          "ipv6.flow": "0x000678a6"
        }
      }
    }
  }
]

```

Figure 2.2: Json output Wireshark

2.4.3 XML (PDML)

A PDML file contains multiple packets, denoted by the `<packet>` tag. A packet will contain multiple protocols, denoted by the `<proto>` tag. A protocol might contain one or more fields, denoted by the `<field>` tag. We can obtain it with the command:

```
tshark -2 -R "smb or smb2" -r /path/of/capture/file -T PDML
```

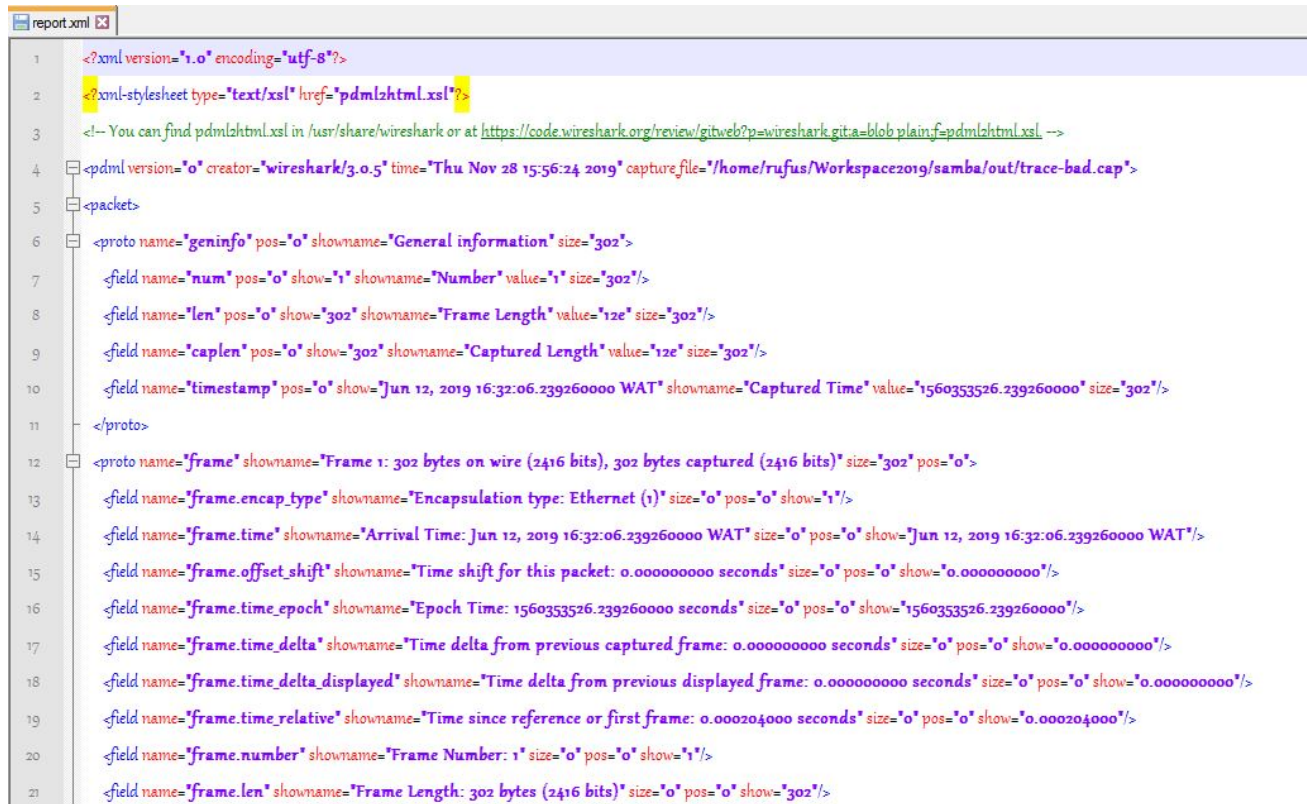


Figure 2.3: XML output Wireshark

Chapter 3

Design of the solution

3.1 Analysis

In this section, we discuss about the stakeholders of our system, the different actors, the functional and non-functional requirements and finally establish the use cases and class diagram.

3.1.1 Requirements engineering

3.1.1.1 Functionnal requirements

- visualize diffs side by side
- permute packets: Usually diffs are from packet A to packet B, if the user wants from B to A, he must not reload the packets
- Change different settings: colors, diff method
- search inside the packets

3.1.1.2 Non-functional requirements

- Use the programming language python
- Must use external dependencies only if necessary, moreover, the program must work standalone
- No databases, store everything in files

3.1.2 Use case diagram (GUI)

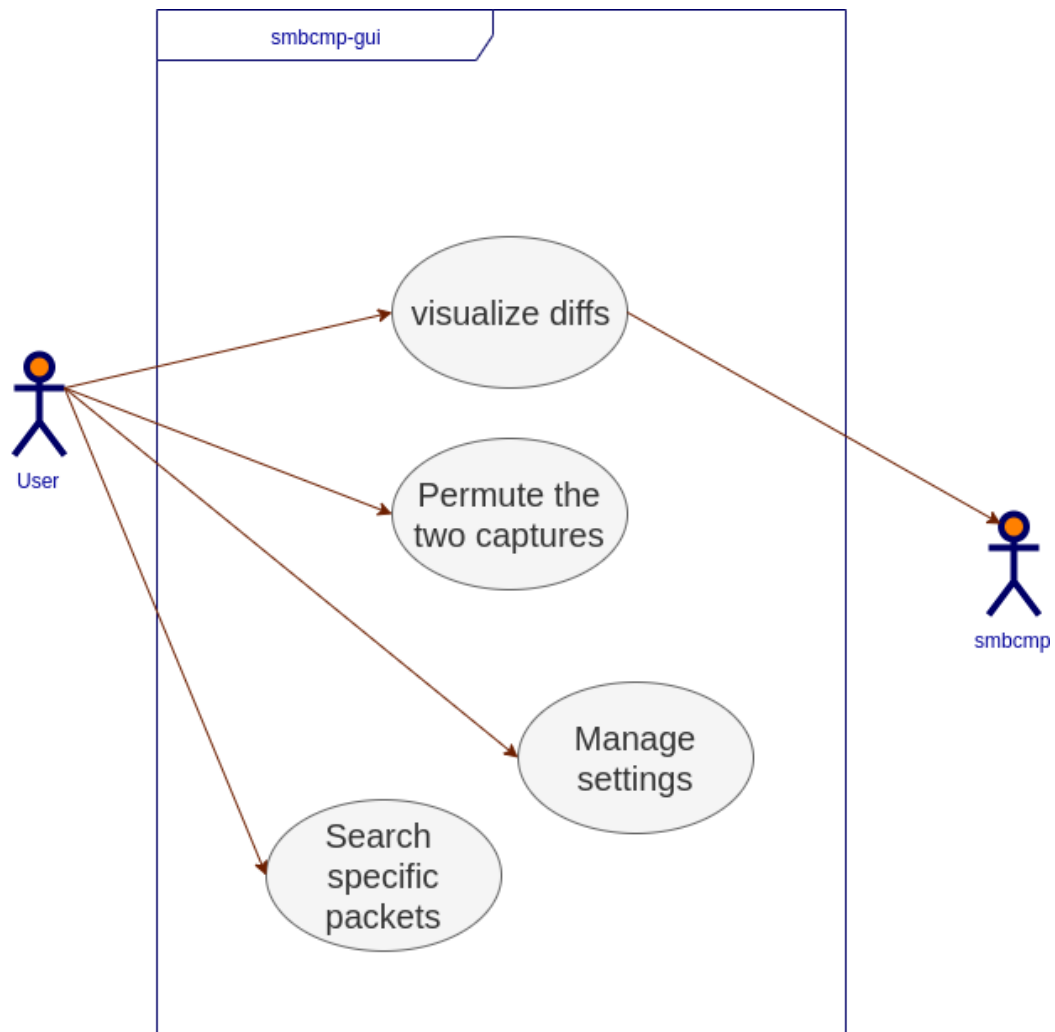


Figure 3.1: Use case diagram smbcmp-gui

- **Stakeholders** The main stakeholder here are the people inside the company because it's an internal tool open sourced.
- **Actors**
 - Main actor
 - User
 - Secondary actor
 - smbcmp
- **Use cases**
 - Open files : load the files inside the buffers
 - Permute buffers : exchange files
 - Change settings : like colors, diff engine
 - Search packets : search packet in each buffer

3.1.3 Class diagram

We have the following class diagram:

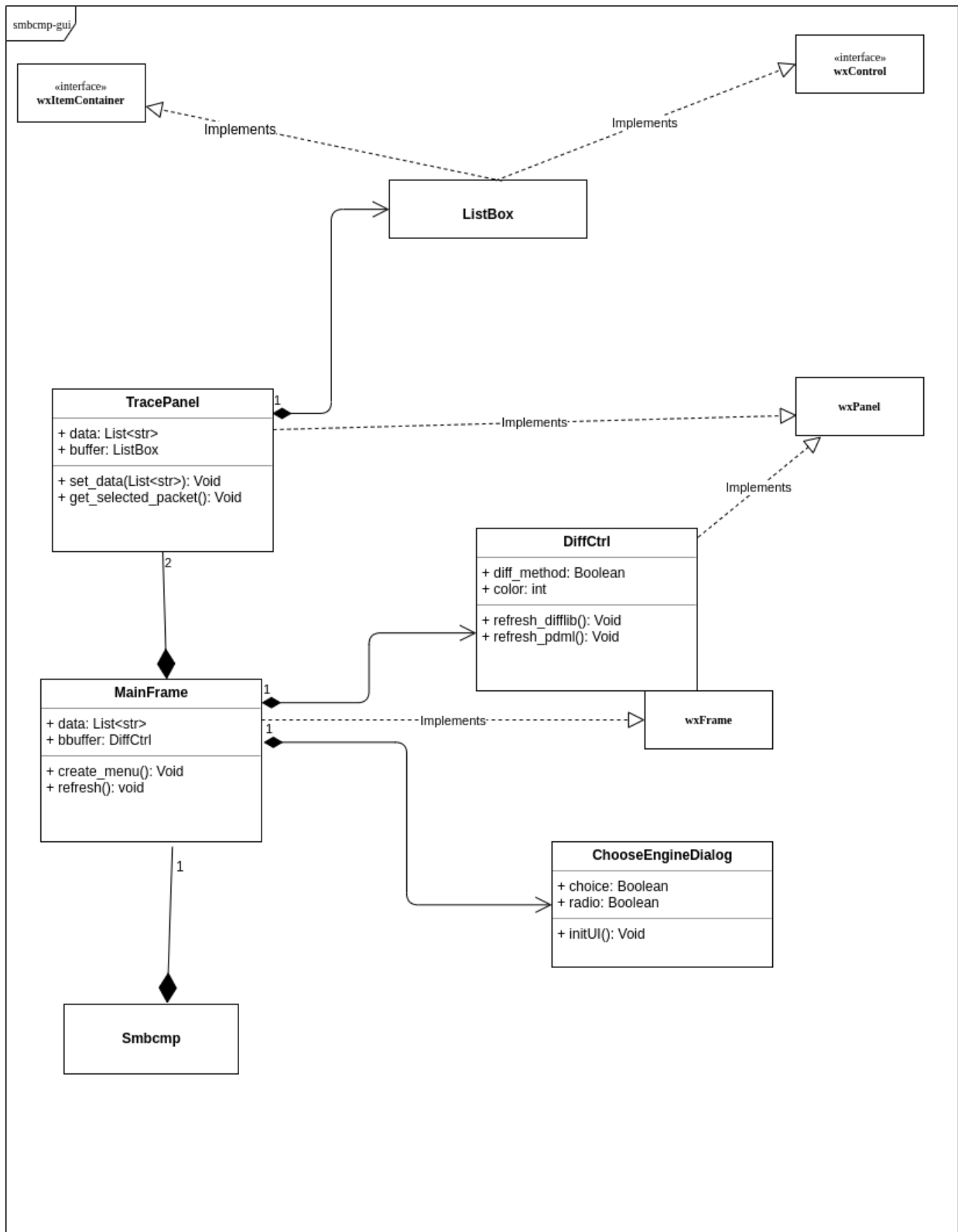


Figure 3.2: Class diagram

3.2 Technologies used

3.2.1 For the command line interface

The objective was to have as little dependencies as possible so in this part, there is only an *optionnal* dependency: `lxml` one may ask why, if it's optionnal it can also be removed without many harm but in our research, `lxml` has been proven to be quicker than `cElementTree` (which is the standard library to parse xml files) for the operation we had to do on the XML tree (mainly serialising and parsing) as a side note, the two libraries propose the same API thus reducing the overhead of using them concurrently.

The following subsections are about the benchmark: (“Benchmarks and Speed”)

• Serialising

For serialising, `lxml` is more than 10 times as fast as the much improved `ElementTree` 1.3 in recent Python versions:

<code>lxe: tostring_utf16</code>	(S-TR T1)	7.9958 msec/pass
<code>cET: tostring_utf16</code>	(S-TR T1)	83.1358 msec/pass
<code>lxe: tostring_utf16</code>	(UATR T1)	8.3222 msec/pass
<code>cET: tostring_utf16</code>	(UATR T1)	84.4688 msec/pass
<code>lxe: tostring_utf16</code>	(S-TR T2)	8.2297 msec/pass
<code>cET: tostring_utf16</code>	(S-TR T2)	87.3415 msec/pass
<code>lxe: tostring_utf8</code>	(S-TR T2)	6.5677 msec/pass
<code>cET: tostring_utf8</code>	(S-TR T2)	76.2064 msec/pass
<code>lxe: tostring_utf8</code>	(U-TR T3)	1.1952 msec/pass
<code>cET: tostring_utf8</code>	(U-TR T3)	22.0058 msec/pass

• Parsing

For parsing, `lxml.etree` and `cElementTree` compete for the medal. Depending on the input, either of the two can be faster. The (c)ET libraries use a very thin layer on top of the `expat` parser, which is known to be very fast. Here are some timings from the benchmarking suite:

<code>lxe: parse_bytesIO</code>	(SAXR T1)	13.0246 msec/pass
<code>cET: parse_bytesIO</code>	(SAXR T1)	8.2929 msec/pass
<code>lxe: parse_bytesIO</code>	(S-XR T3)	1.3542 msec/pass
<code>cET: parse_bytesIO</code>	(S-XR T3)	2.4023 msec/pass
<code>lxe: parse_bytesIO</code>	(UAXR T3)	7.5610 msec/pass
<code>cET: parse_bytesIO</code>	(UAXR T3)	11.2455 msec/pass

• Child Access

The same tree overhead makes operations like collecting children as in `list(element)` more costly in `lxml`. Where `cET` can quickly create a shallow copy of their list of children, `lxml` has to create a Python object for each child and collect them in a list:

<code>lxe: root_list_children</code>	(--TR T1)	0.0038 msec/pass
--------------------------------------	-----------	------------------

cET: root_list_children	(--TR T1)	0.0010 msec/pass
lxe: root_list_children	(--TR T2)	0.0455 msec/pass
cET: root_list_children	(--TR T2)	0.0050 msec/pass

• Element creation

As opposed to ET, libxml2 has a notion of documents that each element must be in. This results in a major performance difference for creating independent Elements that end up in independently created documents:

lxe: create_elements	(--TC T2)	1.0045 msec/pass
cET: create_elements	(--TC T2)	0.0753 msec/pass

• Merging different sources

A critical action for lxml is moving elements between document contexts. It requires lxml to do recursive adaptations throughout the moved tree structure.

The following benchmark appends all root children of the second tree to the root of the first tree:

lxe: append_from_document	(--TR T1,T2)	1.0812 msec/pass
cET: append_from_document	(--TR T1,T2)	0.1104 msec/pass
lxe: append_from_document	(--TR T3,T4)	0.0155 msec/pass
cET: append_from_document	(--TR T3,T4)	0.0060 msec/pass

• Deepcopy

Deep copying a tree is fast in lxml:

lxe: deepcopy_all	(--TR T1)	3.1650 msec/pass
cET: deepcopy_all	(--TR T1)	53.9973 msec/pass
lxe: deepcopy_all	(-ATR T2)	3.7365 msec/pass
cET: deepcopy_all	(-ATR T2)	61.6267 msec/pass
lxe: deepcopy_all	(S-TR T3)	0.7913 msec/pass
cET: deepcopy_all	(S-TR T3)	13.6220 msec/pass

• Tree traversal

Another important area in XML processing is iteration for tree traversal which is especially useful if the algorithms can benefit from step-by-step traversal of the XML tree bonus points if few elements are of interest or the target element tag name is known.

lxe: iter_all	(--TR T1)	1.0529 msec/pass
cET: iter_all	(--TR T1)	0.2635 msec/pass
lxe: iter_islice	(--TR T2)	0.0110 msec/pass
cET: iter_islice	(--TR T2)	0.0050 msec/pass
lxe: iter_tag	(--TR T2)	0.0079 msec/pass
cET: iter_tag	(--TR T2)	0.0112 msec/pass
lxe: iter_tag_all	(--TR T2)	0.1822 msec/pass
cET: iter_tag_all	(--TR T2)	0.5343 msec/pass

From this benchmark, it's clear that lxml outperforms blatantly cET for our use cases and it even propose methods not implemented yet in cET, like a xpath engine but for practical reasons (keep the dependency optionnal) we only used common features.

3.2.2 Graphical User Interface

In the Python GUI multiplatform programming ecosystem, there are a few legit choices (mature enough to be considered):

- **WXpython**: Python bindings of the WXWidgets framework
- **Tkinter**: Standard library for GUI programming
- **Pyside 2 (Qt for python)**: Python bindings for the QT framework
- **PyQT**
- **Kivy**
- **PyGTK**
- **PySimpleGUI**

There is a comparative table for all of them based on the following criteria:

Framework	License	Documentation	Modern UI	Wysiwyg	Target	Native
Wxpython	GPL v3	Good	Yes	Yes	Desktop	Yes
Tkinter	GPL v3	Good	No	No	Desktop	No
Pyside 2	GPL v3	Poor	Yes	Yes	Desktop	No
PyQT	Commercial	Medium	Yes	Yes	Desktop	No
Kivy	BSD	Good	Yes	No	Mobile	No
PyGTK	GPL v3	Medium	No	Yes	Desktop	Only on Gnome
PySimpleGUI	GPL v3	Medium	Yes	No	Desktop	Yes

Because fo the requirements and the documentation, we ended up choosing *WXpython* for the project.

Chapter 4

Implementation

For this part, we had to implement a few algorithms, here we highlight 2 of those which are recursive:

4.1 Algorithm 1: recursive diffs between 2 nodes of the XML graph

4.1.1 Principle

We have two top-level packets A and B, we face the following cases:

- **Cases where A and B have different numbers of children**

In this case we identify each child, match those which are the same and mark the remaining as either removed or added depending on which the parent, if it's A we mark added and if it's B we mark removed.

- **1 Leaf node vs 1 Folder node**

If not child_a.children or not child_b.children: In this case we just keep going deeper inside the folder node, and we print all that as diffs.

- **Terminal cases: 2 leaf nodes**

If the nodes are roughly the same (just the values changed) we store this change else we mark as two different lines.

- **Recursive case: 2 Tree nodes**

We call the function again on the two nodes.

4.1.2 Pseudo-code

The pseudo-code is quite long, you can find it in Appendix B

4.1.3 Complexity analysis

We make a few assumptions, the packets A and B which are represented as trees have respectively n and m elements

4.1.3.1 Time complexity

In the worst case, the packets aren't the same but have the same structure with the same number of nodes, in this case, we have to go to each leaf just to notice that the leafs aren't the same and marked the first one as added and the second one as removed. We have obviously $O(n^2)$

4.1.3.2 Space complexity

We store the both trees and use a constant space to make diffs, so $O(n^2)$.

4.2 Algorithm 2: Circular search inside a non-circular list

4.2.1 Principle

We have a list of strings (packets) `data` inside a `buffer` and we are looking for a certain string `text` and if we find it, (even a partial match) we update the `buffer` selected item and stop the search.

In order to achieve that, we make a fuzzy search (finding approximate substring matches inside a given list of strings) and stop at the next occurrence, repeat this process while the user hit the search button and if we reach the bottom of the list, the search restart automatically at the top of the list.

4.2.2 Pseudo code

```
function search(text, rec=True):
    """ Cyclic search inside summaries

    Keyword arguments:

    text -- expression searched
    rec -- keep track of recursive calls
    """
    sel = buffer.GetSelection()

    if rec:
        search_index = sel
    else:
        search_index = -1

    for i in range(search_index + 1, len(data) + 1):
        # edge case : The last packet is selected
        if i >= len(data):
            found = False
            break

        found = text.lower() in data[i].lower()
        if found:
```



```
        search_index = i
        break

    if not found:
        if rec:
            search(text, rec=False)
    else:
        buffer.SetSelection(search_index)
```

4.2.3 Complexity analysis

4.2.3.1 Time complexity

If we assume that the list `data` has n elements and the max size of elements inside `data` is p ($p = \max(\text{data}[i], 0 < i < n+1)$), in the worst case there are no matches in the whole list, the complexity is then $O(n)$.

4.2.3.2 Space complexity

The space corresponding is $O(n)$

Chapter 5

Evaluation of results

Typically, SMB 2 Protocol messages begin with a fixed-length SMB2 header that is described in the SMB specifications (openspecs-office). The SMB2 header contains a Command field indicating the operation code that is requested by the client or responded to by the server. An SMB 2 Protocol message is of variable length, depending on the **Command field** in the SMB2 header and on whether the SMB 2 Protocol message is a client request or a server response.

So basically, when you are trying to make diffs on SMB packets, there are two possibilities:

- the command fields of the packets are different thus the packets aren't of the same type
- the command fields are the same thus the packets are of the same type

5.1 Diffs

5.1.1 Different command fields

```

Negotiate Protocol Request
Negotiate Protocol Response
Negotiate Protocol Request
Negotiate Protocol Response
Session Setup Request, NTLMSSP_NEGOTIATE
Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHALLENGE
Session Setup Request, NTLMSSP_AUTH, User: WORKGROUP\rufus
Session Setup Response
Tree Connect Request Tree: \\localhost\IPC$
Tree Connect Response
Ioctl Request FSCTL_DFS_GET_REFERRALS, File: \\localhost\mon_partage
Ioctl Response, Error: STATUS_NOT_FOUND
Tree Disconnect Request
Tree Disconnect Response
Tree Connect Request Tree: \\localhost\mon_partage
Tree Connect Response
Create Request File:
Find Request File: SMB2_FIND_ID_BOTH_DIRECTORY_INFO Pattern: *
Find Response SMB2_FIND_ID_BOTH_DIRECTORY_INFO Pattern: *
Find Request File: SMB2_FIND_ID_BOTH_DIRECTORY_INFO Pattern: *
Find Response, Error: STATUS_NO_MORE_FILES SMB2_FIND_ID_BOTH_DIRECTORY_INFO Pattern: *

--- /home/rufus/Workspace2019/samba/out/trace.cap
+++ /home/rufus/Workspace2019/samba/out/trace-bad.cap
@@ -2,48 +2,46 @@
SMB2 Header
Server Component: SMB2
Header Length: 64
Credit Charge: 0
NT Status: STATUS_SUCCESS (0x00000000)
Command: Negotiate Protocol (0)
Credits granted: 1
Flags: 0x00000001, Response
.....0 = Response: This is a RESPONSE
+
Credit Charge: 1
Channel Sequence: 0
Reserved: 0000
Command: Session Setup (1)
Credits requested: 8192
Flags: 0x00000010, Priority
.....0 = Response: This is a REQUEST
.....0.. = Async command: This is a SYNC command
.....0.. = Chained: This pdu is NOT a chained command
.....0.. = Signing: This pdu is NOT signed
.....000 .... = Priority: This pdu does NOT contain a PRIORITY

```

Figure 5.1: Plain text output for different command fields

- **Interpretation** As you can notice from the previous plain text output, diffs were done blocks by blocks instead of in this case line by line which lead to something like the **Credit Charge** field has only one minor change but in the plain text output it's considered as part of the 6 lines change while in this case the **Credit Charge: 0 -> 1** is clearly highlighted. Moreover, in the PDML output, the following changes are highlighted one at the time so that the user know exactly what really change without putting too much effort of making the visual correspondance by him

```

Negotiate Protocol Request
Negotiate Protocol Response
Negotiate Protocol Request
Negotiate Protocol Response
Session Setup Request, NTLMSSP_NEGOTIATE
Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHALLENGE
Session Setup Request, NTLMSSP_AUTH, User: WORKGROUP\rufus
Session Setup Response
Tree Connect Request Tree: \\localhost\IPC$
Tree Connect Response
Ioctl Request FSCTL_DFS_GET_REFERRALS, File: \\localhost\mon_partage
Ioctl Response, Error: STATUS_NOT_FOUND
Tree Disconnect Request
Tree Disconnect Response
Tree Connect Request Tree: \\localhost\mon_partage
Tree Connect Response
Create Request File:
Create Response File:
Find Request File: SMB2_FIND_ID_BOTH_DIRECTORY_INFO Pattern: *
Find Response SMB2_FIND_ID_BOTH_DIRECTORY_INFO Pattern: *
Find Request File: SMB2_FIND_ID_BOTH_DIRECTORY_INFO Pattern: *
Find Response, Error: STATUS_NO_MORE_FILES SMB2_FIND_ID_BOTH_DIRECTORY_INFO Pattern: *

: Root
: SMB2 (Server Message Block Protocol version 2)
: SMB2 Header
  Server Component: SMB2
  Header Length: 64
  Credit Charge: 0 → 1
  -NT Status: STATUS_SUCCESS (0x00000000)
  +Channel Sequence: 0
  -Command: Negotiate Protocol (0)
  +Reserved: 0000
  -Credits granted: 1
  +Command: Session Setup (1)
  -Flags: 0x00000001, Response
    -! ..... = Response: This is a RESPONSE
    -! ..... = Async command: This is a SYNC command
    -! ..... = Chained: This pdu is NOT a chained command
    -! ..... = Signing: This pdu is NOT signed
    -! ..... = Priority: This pdu does NOT contain a PRIORITY
    -! ..0 ..... = DFS operation: This is a normal operation
    -! ..0 ..... = Replay operation: This is NOT a replay operation
  +Credits requested: 8192
  +Flags: 0x00000010, Priority
    +! ..... = Response: This is a REQUEST

```

Figure 5.2: PDML output for different command fields

5.1.2 Same command fields

```

Negotiate Protocol Request
Negotiate Protocol Response
Negotiate Protocol Request
Negotiate Protocol Response
Session Setup Request, NTLMSSP_NEGOTIATE
Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHALLENGE
Session Setup Request, NTLMSSP_AUTH, User: WORKGROUP\rufus
Session Setup Response
Tree Connect Request Tree: \\localhost\IPC$
Tree Connect Response
Ioctl Request FSCTL_DFS_GET_REFERRALS, File: \\localhost\mon_portage
Ioctl Response, Error: STATUS_NOT_FOUND
Tree Disconnect Request
Tree Disconnect Response
Tree Connect Request Tree: \\localhost\mon_portage
Tree Connect Response
Create Request File:
Create Response File:
Find Request File: SMB2_FIND_ID_BOTH_DIRECTORY_INFO Pattern: *
Find Response SMB2_FIND_ID_BOTH_DIRECTORY_INFO Pattern: *
Find Request File: SMB2_FIND_ID_BOTH_DIRECTORY_INFO Pattern: *
Find Response, Error: STATUS_NO_MORE_FILES SMB2_FIND_ID_BOTH_DIRECTORY_INFO Pattern: *

-- /home/rufus/Workspace2019/samba/out/trace.cap
++ /home/rufus/Workspace2019/samba/out/trace-bad.cap
@@ -21,9 @@
    Session Id: 0x0000000000000000
    Signature: 00000000000000000000000000000000
    [Response to: 8]
    [Time from request: 0.000300000 seconds]
    [Time from request: 0.000340000 seconds]
    Negotiate Protocol Response (0x00)
    [Preauth Hash: 0bda319050ee237507361ab710098c06a8cb5762f30c5339_]
    [Preauth Hash: 42584fe2ef6108eda64acfee437f53c51049130f63524bec_]
    StructureSize: 0x0041
    0000 0000 0100 000. = Fixed Part Length: 32
    .... .... .... ...1 = Dynamic Part: True
@@ -44,7 @@
    Max Transaction Size: 8388608
    Max Read Size: 8388608
    Max Write Size: 8388608
    Current Time: Jun 12, 2019 16:29:41.116240000 WAT
    Current Time: Jun 12, 2019 16:32:06.241787000 WAT
    Boot Time: No time specified (0)
    Blob Offset: 0x00000080
    Blob Length: 74

```

Figure 5.3: Plain text output for same command fields

- **Interpretation** Here it's even more visible, using the plain text output, it's very difficult to separate a field from his value, that's what you can notice from the first output; the fields **Time for request**, **Preaut Hash** and **Current Time** are only changing values but with the first version they are highlighted as one line change while the second are highlighted more precisely as values change.

```

Negotiate Protocol Request
Negotiate Protocol Response
Negotiate Protocol Request
Negotiate Protocol Response
Session Setup Request, NTLMSSP_NEGOTIATE
Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHALLENGE
Session Setup Request, NTLMSSP_AUTH, User: WORKGROUP\rufus
Session Setup Response
Tree Connect Request Tree: \\localhost\IPC$
Tree Connect Response
Ioctl Request FSCTL_DFS_GET_REFERRALS, File: \\localhost\mon_partage
Ioctl Response, Error: STATUS_NOT_FOUND
Tree Disconnect Request
Tree Disconnect Response
Tree Connect Request Tree: \\localhost\mon_partage
Tree Connect Response
Create Request File:
Create Response File:
Find Request File: SMB2_FIND_ID_BOTH_DIRECTORY_INFO Pattern: *
Find Response SMB2_FIND_ID_BOTH_DIRECTORY_INFO Pattern: *
Find Request File: SMB2_FIND_ID_BOTH_DIRECTORY_INFO Pattern: *
Find Response, Error: STATUS_NO_MORE_FILES SMB2_FIND_ID_BOTH_DIRECTORY_INFO Pattern: *

Session Id: 0x0000000000000000
Signature: 00000000000000000000000000000000
Response to: 8
Time from request: 0.000300000 seconds → 0.000340000 seconds
: Negotiate Protocol Response (0x00)
Preauth Hash: 0bda31905e0e237507381ab710098c06a8cb5762f30c5339\xe2\x80\xa6 → 42584fe2ef6108eda64acf6e437f53c51049130f63524bec\xe2\x80\xa6
StructureSize: 0x0041
0000 0000 0100 000. = Fixed Part Length: 32
? .... ..1 = Dynamic Part: True
Security mode: 0x01, Signing enabled
? .... ..1 = Signing enabled: True
? .... ..0. = Signing required: False
Dialect: 0x0311
NegotiateContextCount: 2
Server Guid: 61636f6c-686c-736f-7400-000000000000
Capabilities: 0x00000007, DFS, LEASING, LARGE_MTU
? .... ..1 = DFS: This host supports DFS
? .... ..1. = LEASING: This host supports LEASING
? .... ..1.. = LARGE_MTU: This host supports LARGE_MTU
? .... ..0... = MULTI_CHANNEL: This host does NOT support MULTI_CHANNEL
? .... ..0 .... = PERSISTENT_HANDLES: This host does NOT support PERSISTENT_HANDLES
? .... ..0. .... = DIRECTORY_LEASING: This host does NOT support DIRECTORY_LEASING
? .... ..0.. .... = ENCRYPTION: This host does NOT support ENCRYPTION

```

Figure 5.4: PDML output for same command fields

5.2 GUI

One objective was to port to windows, the tests have been realized on Windows 10 Professional and Fedora Workstation 31 the views are following:

5.2.1 Windows

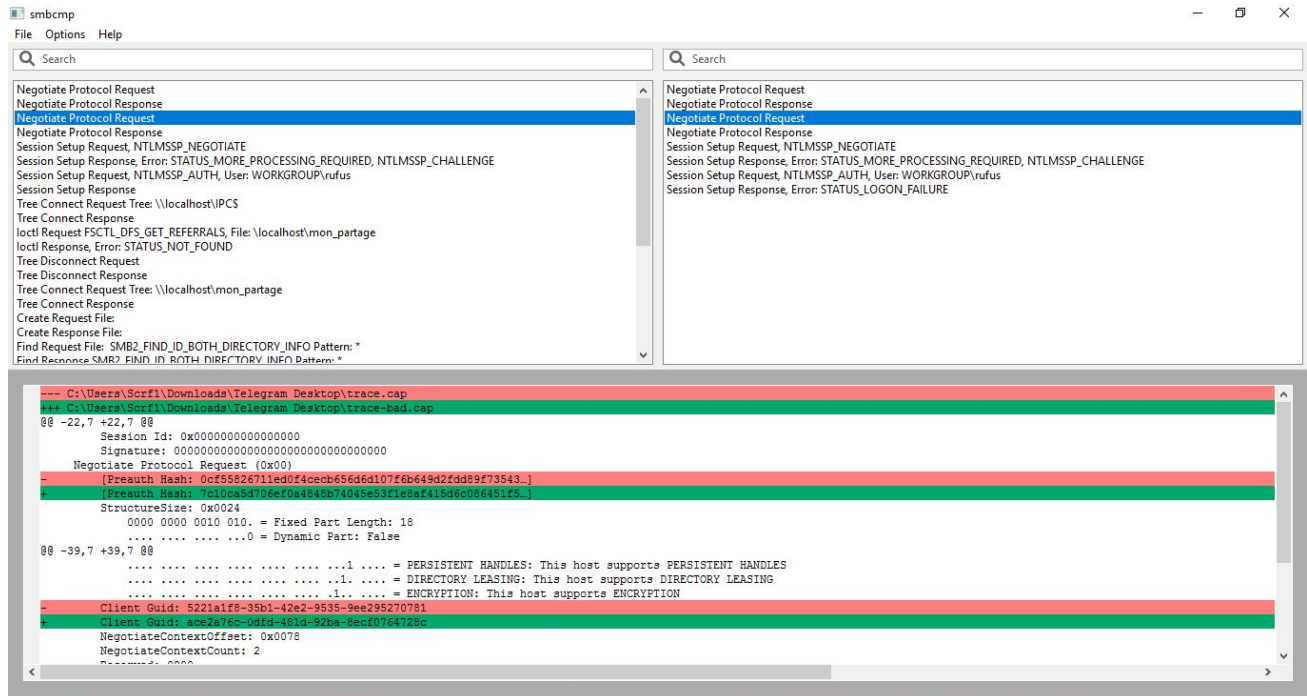


Figure 5.5: Windows view smbcmp

5.2.2 Linux

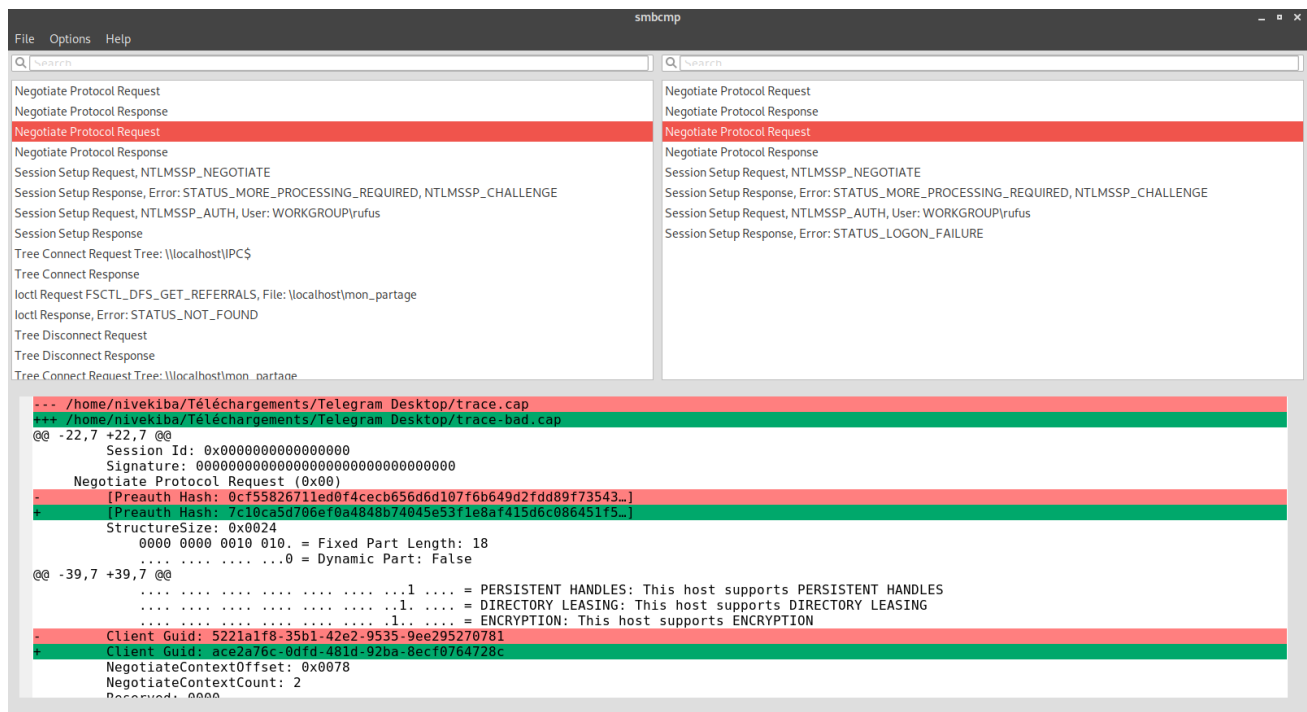


Figure 5.6: Linux view smbcmp

Chapter 6

Prospect

To improve the usefulness of our system, we can think of many things that can be added or done differently:

6.1 Add documentation for config file and usage

About : - file format of the config file (possible options, possible values,...) - the location it will be loaded from - how to use -k, how to generate keys from kernel or samba

6.2 Install documentation and desktop file in packaging

Modify packaging scripts for all OS to install the man pages and desktop file to make it available on all major Linux distributions, actually it's only on Opensuse repository.

6.3 Pass down arguments passed to windows launcher to the Python scripts

The windows port is not perfect, there are some features present on the Linux version which aren't yet on windows. Add arguments to smbcmp.exe (-h) and Drag and drop support to load capture files more easily and intuitively.

6.4 Implement ignored fields in smbcmp-gui

Add or remove ignored field from the diff view (right click) Add a main menu item to list and manage ignored fields. Implement ignored fields filters :

- value greater than, equal to, less than
- value is one of x, y, z
- value contains x

We can think of creating a little programming language with its interpreter which will be able to parse and validate rules.

Chapter 7

Conclusion

Our target during this internship at Samba conducted by Google was to improve an existing tool by adding features (better and deeper diffs, GUI, Windows port). For that, we had to make some research about the state of the art of Capture diff tooling, after what we started developing our solution according to the software reuse principle called : encapsulating an existing system, we ended up with 3 components : smbcmp (the CLI), smbcmp-gui (the GUI) and smbcmp-common (a set of common utilities for the previous components). Despite the fact that we were new in the working internals of a protocol namely SMB which has been around for quite a while, with multiple revisions through the years, we developed a working solution. At the end, we developed some skills in Python programming (debugging, testing), remote work, steady communication, Git and Github workflow.

Chapter 8

Appendix

8.1 Appendix A - Proposal

8.1.1 Introduction

At present, the smbcmp tool can view single capture or diff 2 captures side by side with a diff on the bottom pane but smbcmp has a very little user base and is nothing more finally than a script, at first my reflex was to verify if it's available on the main repository of my distro but I think this is such a great program which deserves more than `git clone && chmod +x`, instead, regular updates and improvements.

8.1.2 Project Goals

The aim of this project is (as stated in the title) to improve the tool by adding some features, the first are the ideas taken on the idea list, then some of my ideas to actually improve the tooling

8.1.2.1 Use or combine current tshark output with the XML output

This is for doing better and deeper diffs by ignoring indentation differences, adding ways to let users add/ignore rules, etc. The XML output is known in the Wireshark world as PDML (Product Data Markup Language) after some researches, I found a resource explaining the specifications <http://xml.coverpages.org/PDML.html> I think that in order to do that, I'll use the `##-TPDML##` option of tshark and use the output to apply filters (add/ignore rules).

8.1.2.1.1 Add an html output

According to <https://wiki.wireshark.org/PDML>, making an html output from the xml one is pretty easy, with `##xsltproc##` from the xslt library with a simple call, then style the webpage to render a pretty thus more readable output.

8.1.2.1.2 Deliverable

A pull request containing all the changes ready for review.

8.1.2.2 Make smbcmp highlight diffs from the packet summary listing

The current implementation of this is satisfying (at least for me) but one thing I think I can add is a descriptive text stating that white packets are the same, it may be confusing at first for newbies who doesn't really know the structure of a samba packet.

8.1.2.2.1 Deliverable

A pull request

8.1.2.3 Automate the creation of .smbcmp file

For now, we need to manually copy the settings from the sample in the readme then paste it and tweak to our desires, first i want to automate the process of a config file, and maybe add a frontend for configuration.

8.1.2.3.1 Deliverable

A pull request

8.1.2.4 Make a proper delivery way

By packaging as flatpak, appimage or snap. At least creating an "install/configure" script

8.1.2.4.1 Deliverable

It depends on what will be chosen, but at this point, the Readme of the repos should be updated with the new delivery method.

8.1.2.5 Correct soome flaws of the tool

8.1.2.5.1 smbcmp throws an error if there is no samba packet in the pcap file

this is already listed on this issue <https://github.com/aaptel/smbcmp/issues/3>, I plan to tackle it in my work.

8.1.2.5.2 smbcmp close unexpectedly after resizing to critical values

8.1.3 Timeline

The breakdown structure here is just as the program announce it and as an approximative indication on what I will be working on.

8.1.3.1 Community Bonding Period[May 6 - May 26]

Since I have exams starting in this period, I'll need a little break of maximum 3 weeks but during that time I won't be idle as I can start prototyping the html page output and get insight and advices about my weak areas + a better understanding on how the core samba protocol works, I will talk regularly to my mentor(s) about any specifications on various fonctionnalités.

8.1.3.2 Coding officially begins [May 27]

I will be able to work full time on the project after the end of my exams (around the 7th of June). The classes restart by the 9th of September.

8.1.3.2.1 June 7 - June 28 (First second and third Weeks)

I will be working on the use and combination of xml output for better overview of results + the implementation of the html view and diff highlighting, on the official timeline it's stated that from 24 to 28 of June there are evaluations it's also included in my planning

8.1.3.2.2 June 28 - July 26 (Weeks 4 - 7)

Automate the creation of .smbcmp file and correct some flaws of the tools, including those that I would have eventually added. This phase is also marked by an evaluation.

8.1.3.2.3 July 26 - August 19 (Week 7 - end)

Work on the delivery process of the application, here I will make the final decision on how I should distribute it.

8.1.4 About Me

8.1.4.1 Name

Mairo Paul Rufus

8.1.4.2 Email

akoudanilo@gmail.com

8.1.4.3 Github

<https://github.com/RMPR>

8.1.4.4 Phone Number

+237 690823108

8.1.4.5 Summary

Currently I'm pursuing a Bachelor degree in computer engineering (2015-2020) in National Advanced School of Engineering of Yaounde, Cameroon. I'm a self-taught considering python since we are most working with JAVA. As for my work experience, last year we implemented a recommender system hosted at <http://52.23.196.12> on an AWS machine and the years before that I used to do "IT-hopping" (test many IT fields in order to see what suits me the most, hence many contributions in web development).

8.1.4.6 Contributions to OSS :

8.1.4.6.1 Merged PR

- Powerline
- Telegram Desktop
- Awesome design tools
- Awesome design tools
- Smbcmp
- Source code pro

8.1.4.6.2 Unmerged PR

- Daily Coding Problem
- yii2-user-extended
- Typescript react starter
- Polybar

8.2 Appendix B - Recursive PDML diffs algorithm

```
function smb_diff():
    """Compute PDML diff and return DiffOutput instance"""

    output = DiffOutput()

    function rec_diff(a, b, n=0):
        final_eq = SAME

        #
        # Diff a folder attributes
        #

        eq, ign = diff_attr_with_rules(a, b)
        if eq == SAME:
            # doesnt matter which
            output.print_field(OUT_SAME, a, indent=n)
        elif eq == MOD:
            output.print_mod_field(a, b, indent=n)
        if not ign:
```

```

        final_eq = max(final_eq, eq)
    elif eq == DIFF:
        output.dump(OUT_REM, a, ignored=ign, indent=n)
        output.dump(OUT_ADD, b, ignored=ign, indent=n)
        if not ign:
            final_eq = max(final_eq, eq)
    return final_eq

#
# Diff the children of the folder
#

sm = difflib.SequenceMatcher(None, a.children, b.children)
for tag, i1, i2, j1, j2 in sm.get_opcodes():
    if tag == 'delete':
        for child_a in a.children[i1:i2]:
            ign = ignored_with_rules(child_a)
            output.dump(OUT_REM, child_a, ignored=ign, indent=n+1)
            if not ign:
                final_eq = max(final_eq, eq)
        continue
    elif tag == 'insert':
        for child_b in b.children[j1:j2]:
            ign = ignored_with_rules(child_b)
            output.dump(OUT_ADD, child_b, ignored=ign, indent=n+1)
            if not ign:
                final_eq = max(final_eq, eq)
        continue
    elif tag == 'equal':
        for child_a, child_b in zip_longest(a.children[i1:i2],
                                             b.children[j1:j2]):
            eq, ign = diff_field_with_rules(child_a, child_b)
            if eq == SAME:
                # doesnt matter which
                output.dump(OUT_SAME, child_a, indent=n+1)
            elif eq == MOD:
                output.print_mod_field(child_a, child_b,
                                       ignored=ign, indent=n+1)
                if not ign:
                    final_eq = max(final_eq, eq)
            else:
                raise Exception("nodes should not be diff here")
        continue
    elif tag == 'replace':
        for child_a, child_b in zip_longest(a.children[i1:i2],
                                             b.children[j1:j2]):

#
# Cases where A and B have different numbers

```



```

# of children

if child_a is None:
    # B has more children
    ign = ignored_with_rules(child_b)
    output.dump(OUT_ADD, child_b, ignored=ign, indent=n+1)
    if not ign:
        final_eq = max(final_eq, eq)
    continue

if child_b is None:
    # A has more children
    ign = ignored_with_rules(child_a)
    output.dump(OUT_REM, child_a, ignored=ign, indent=n+1)
    if not ign:
        final_eq = max(final_eq, eq)
    continue

#
# Terminal cases: 2 leaf nodes
#

if not child_a.children and not child_b.children:
    eq, ign = diff_field_with_rules(child_a,
                                    child_b)

    if eq == SAME:
        # doesnt matter which
        output.dump(OUT_SAME, child_a, indent=n+1)
    elif eq == MOD:
        output.print_mod_field(child_a, child_b,
                               ignored=ign, indent=n+1)

        if not ign:
            final_eq = max(final_eq, eq)
    else:
        output.dump(OUT_REM, child_a, ignored=ign,
                    indent=n+1)
        output.dump(OUT_ADD, child_b, ignored=ign,
                    indent=n+1)

        if not ign:
            final_eq = max(final_eq, eq)
    continue

#
# 1 Leaf node vs 1 Folder node
#

if not child_a.children or not child_b.children:
    # doesn't make sense to diff deeper,

```

```
# consider one has been removed
# and the other added

ign = ignored_with_rules(child_a) and \
    ignored_with_rules(child_b)
output.dump(OUT_REM, child_a, ignored=ign,
            indent=n+1)
output.dump(OUT_ADD, child_b, ignored=ign,
            indent=n+1)
if not ign:
    final_eq = max(final_eq, DIFF)
continue

#
# Recursive case: 2 Tree nodes
#

eq = rec_diff(child_a, child_b, n=n+1)
final_eq = max(final_eq, eq)

return final_eq

rec_diff(pkt_a, pkt_b)
return output
```

Bibliography

“Benchmarks and Speed.” Web. 26 Nov. 2019.

“Google Summer of Code.” *Wikipedia* Nov. 2019. Web. 22 Nov. 2019.

isginf. “Isginf/Pcap-Diff.” Oct. 2019. Web. 23 Nov. 2019.

openspecs-office. “[MS-SMB2]: Server Message Block (SMB) Protocol Versions 2 and 3.” Web. 27 Nov. 2019.

“Statistics Google Summer of Code.” *Google Developers*. Web. 22 Nov. 2019.