Design Document for IP Fabrics

Author: May06-15 (Network Forensic UI)

Andy Heintz (Communication Leader)

Abraham Devine (Webmaster)

Altay Ozen (Team Leader and Team Key Concept Holder)

Dr. Joseph Zambreno (Adviser)

Curt Schwaderer (Client)

Version	Date	Author	Change
1.0	10/26	AH	Created initial version of design document
2.0	11/23	AH	Created final version of design document

Table of Contents

1 Problem Statement

Currently, DeepSweep, a device/program made by the client, emits traffic as a raw stream of data and does not provide a graphical user interface for viewing the output. While there are tools for processing traffic, many of them require input as pcap files or some format other than the raw stream of data. Therefore, our project's goal is to develop an interface between an open source application for processing internet traffic and DeepSweep. Once the interface is developed, we will extend the other program to add additional functionality for our client.

2 System Design

2.1 SYSTEM REQUIREMENTS

Xplico can run on any version of Linux, but is easiest to install on Ubuntu 11.04 or later. Instructions for installing Xplico can be found here: http://wiki.xplico.org/doku.php?id=ubuntu. The installation instructions include all third-party packages or programs used by Xplico.

Xplico does not list on its website how much memory or CPU is required. Currently, Xplico is being run on virtual machines with small amount of RAM and CPU. As of yet, performance impacts have not been seen with this limited amount of RAM and CPU. Eventually, tests will be run to determine how much memory and RAM Xplico requires.

2.2 FUNCTIONAL REQUIREMENTS

Phase 1

1. Xplico (modified): Start Pusher and the appropriate Collector
2. C Collector (new): Collect raw traffic from DeepSweep and create pcaps
3. Go Collector (existing): Collect ASN.1 traffic from DeepSweep and create pcaps
4. Xplico Pusher (existing): Take new pcap files and upload them to Xplico
5. Xplico (existing): Display uploaded traffic in its web GUI

(The primary purpose of Phase 1 is to connect all components of the system.)

Phase 2

1. Xplico (modified): Start Pusher and the appropriate Collector
2. C Collector (new): Collect raw traffic from DeepSweep and create pcaps
3. Java Collector (new): Collect ASN.1 traffic from DeepSweep and create pcaps
4. Xplico Pusher (existing): Take new pcap files and upload them to Xplico
5. Xplico (existing): Display uploaded traffic in its web GUI
6. Xplico (modified): Provide options for generating reports of uploaded traffic

2.3 FUNCTIONAL DECOMPOSITION

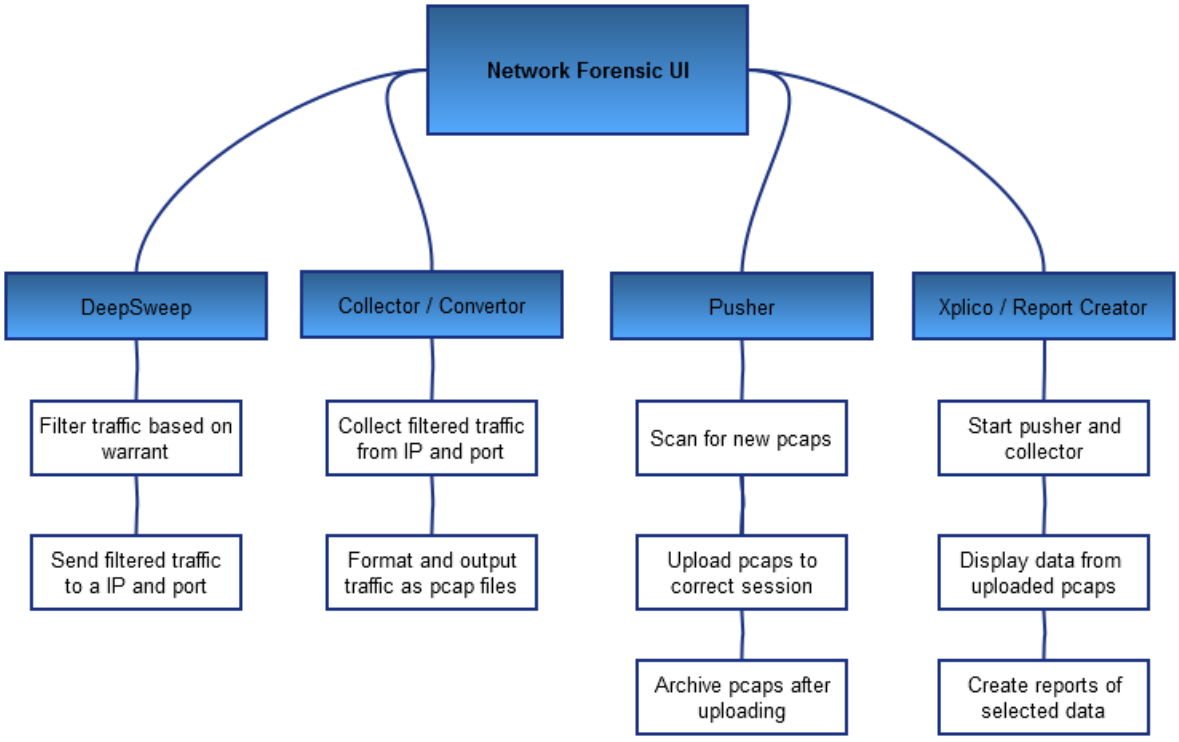

Figure 1: Functional Decomposition Diagram

2.4 SYSTEM ANALYSIS

Network Forensic Tool: Xplico

The client had previously looked at Xplico as a possible tool, which would provide a GUI for viewing output from DeepSweep. We compared Xplico to other programs for decoding pcaps; however, we did not find any other programs that would work better. The vast majority of decoding programs are command line only. Since the client wants a tool with a GUI, these alternatives don't work for our purposes.

Java PCap Library: jnetpcap

In the second phase of our project, the client wants us to replace the existing collector with a new Java collector. As Java doesn't have native libraries for creating pcaps, we will need to find a third party library. We plan to use jnetpcap, since it is well-documented. Their web site contains tutorials, example code, and Javadoc, making it easier to figure out how the library works.

PHP PDF Library: TCPDF

In the second phase of our project, the client wants us to add a report generation feature to Xplico. We plan to create the reports as PDFs, since this is a fairly universal file format that will display properly regardless of operating system. PHP does not contain libraries for creating PDFs, so we will need to use a third party library. We intend to use TCPDF, since it is well-documented and regularly updated.

3 Detailed Design

3.1 INPUT / OUTPUT SPECIFICATION

1. Our system's input will be internet traffic, which will be filtered by DeepSweep. The traffic will either be raw or formatted in the ASN.1 standard's BER encoding.

2. The Collector will take this traffic and output it as pcap files. The C Collector handles raw traffic; the Go and Java Collectors handles ASN.1 traffic.

3. The Pusher will take these pcaps and push them into Xplico.

4. Xplico will decode these pcaps and load the data into Xplico's database.

5. Users can either view the data in a web GUI or export the data as a PDF report.

Module	Input	Output
DeepSweep	Unfiltered traffic	Filtered traffic, in ber format
Collector	Filtered traffic, in ber format	Pcap files
Pusher Script	Pcap files	Data in Xplico's database
Xplico	Data in Xplico's database	Pcap contents and PDF reports

Table: List of all components and their inputs/outputs

3.2 USER INTERFACE SPECIFICATION

Our project will modify Xplico's existing GUI and add additional functionality requested by the client. The below images show what the screens will look like after modification.

Figure 2: Case page

This page lists all cases created by the user, which contain sessions for uploading pcaps. This page will be modified to add two new options. First, there will be an option for automatically uploading pcaps output by DeepSweep to a case's most recent session. Second, there will be an option to generate a report for a case.

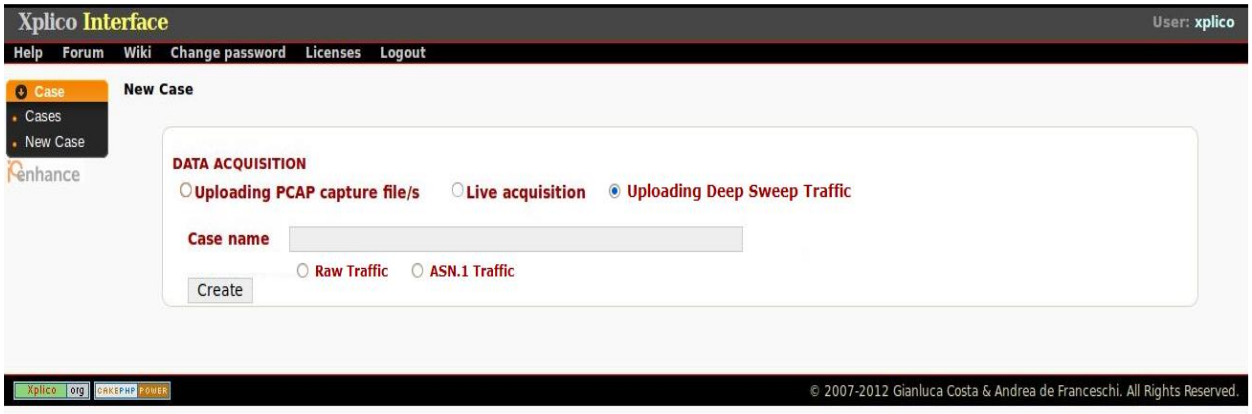

Figure 3: New Case page

This page provides options for creating new cases. This page will be modified to add an option for creating a case that automatically uploading traffic from DeepSweep. The user will select whether the traffic is raw or ASN.1 and press the "Create" button. If this option is selected, Xplico will first create a new session. Once the session is created, Xplico will start the correct Collector and Pusher.

Figure 4: Session page

This page lists a summary of the data uploaded for one session. On the left, a user can choose to view specific types of data, e.g. web, mail, chat.

This page will be modified to add an option for generating a report with the session's content. The user will select the desired types of data and press the "Generate Report" button. Once the report is generated, the user can click "List of All Reports" to find the report.

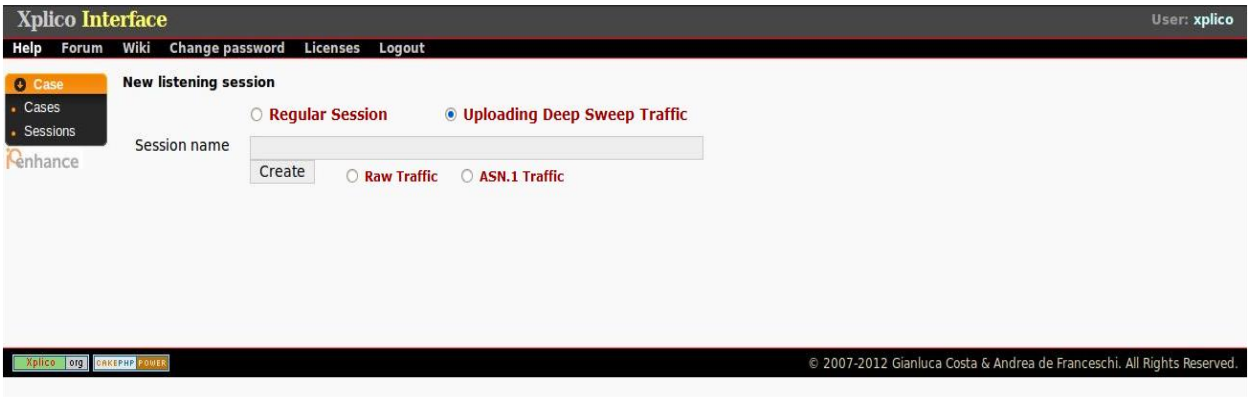

Figure 5: New Session page

This page provides options for creating new sessions. This page will be modified to add an option for automatically uploading traffic output by DeepSweep to the new session. The user will select the type of traffic and press the "Create" button. Once the button is pressed, Xplico will start the correct Collector and Pusher.

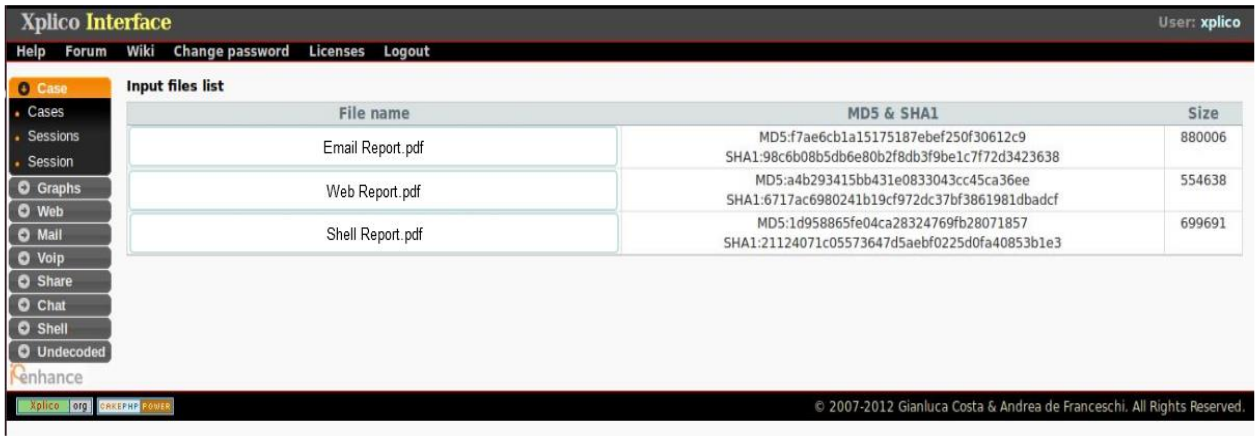

Figure 6: Report page

This is an entirely new page, although it will use the style of existing pages. On this page, all generated reports will be listed along with their size and possibly their hash info. Clicking on the name of report will open the report.

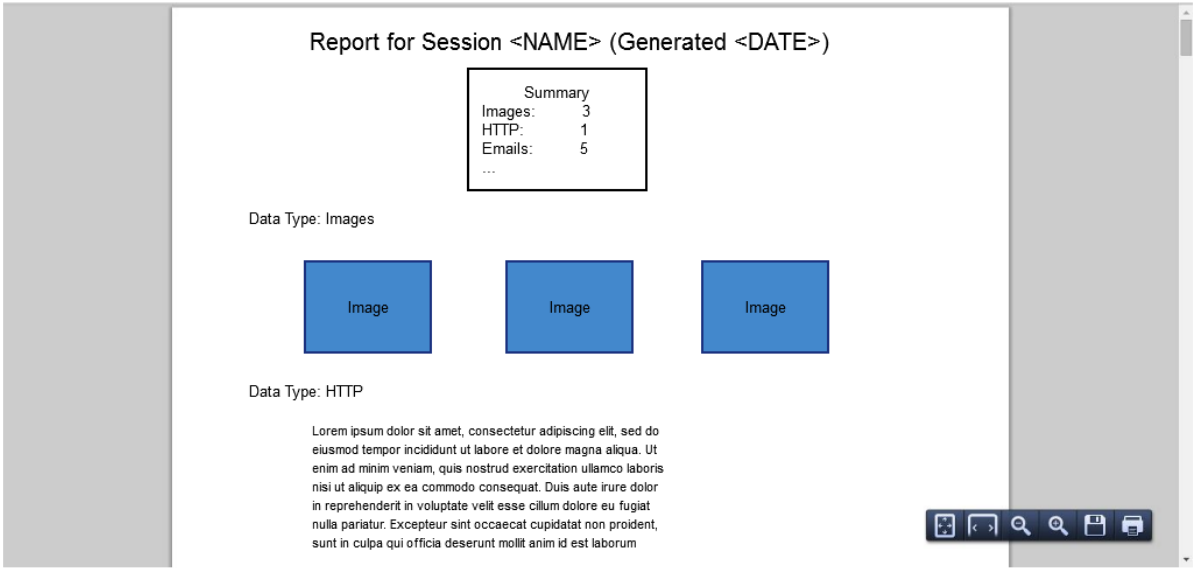

Figure 7: PDF Report

Once a report is clicked, it will be opened in the web browser. This is what the report itself will look like. The top of the report will list the session or case name along with the date and time when the report was generated. Below this will be a quick summary of the data types found in the report. After the summary, the actual data is listed, sorted by data type.

3.3 HARDWARE / SOFTWARE SPECIFICATION

DeepSweep: Hardware supplied by client

Go Collector, Phase 1: Software supplied by client in Go
Java Collector, Phase 2: Software written by our team in Java
C Collector, Phase 1 + 2: Software written by our team in C

Xplico Pusher, Phase 1 + 2: Open source software in C/Python
Xplico, Phase 1: Open source software in C and PHP
Xplico, Phase 2: Modified version in C and PHP

(For more details, see below under Software Design.)

3.4 PROTOTYPES

Phase 1A Prototype: The initial prototype will use the C Collector, which we will write. Both the C Collector and Xplico Pusher will be started from the command line.

Phase 1B Prototype: This prototype will build on Phase 1A Prototype by using the Go Collector supplied by the client. We will also modify Xplico to start the Pusher and Collectors. Users will have options for selecting which case and session will store the data.

A diagram of the initial prototype can be found below in Software Design.

Phase 2A Prototype: For this prototype, we will replace the existing Go Collector with a new Java-based Collector. Xplico will be modified to call this new Collector instead.

Phase 2B Prototype: This prototype will build on Phase 2A Prototype by extending Xplico. The primary new functionality will be a report generation. Users will be able to generate a report consisting of all or part of the data for each session and/or case.

A diagram of the final prototype can be found below in Software Design.

3.5 SOFTWARE DESIGN

As our project is entirely software, there is no mechanical CAD, electronic CAD, or PCB.

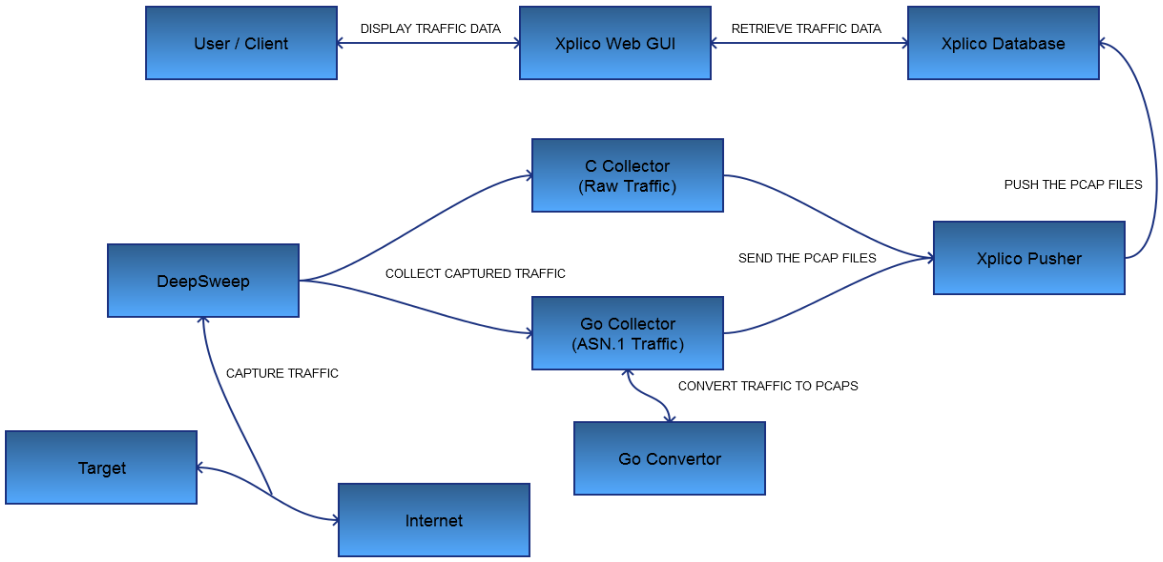

Figure 8: System Diagram for Phase One

In Phase One (Figure 1), a user configures the system to filter traffic from their desired target. Deepsweep captures the traffic between the target and the internet, and sends the traffic to the appropriate Collector. Raw traffic is handled by the C Collector, while traffic in ASN.1 format is handled by the Go Collector. Both Collectors create pcap files from the traffic. Xplico's Pusher scans a directory for new pcap files and uploads them into Xplico's database, at which point the user can view them in Xplico's web GUI. Once uploaded, the pcaps are moved to another directory, so the user can retrieve them if necessary.

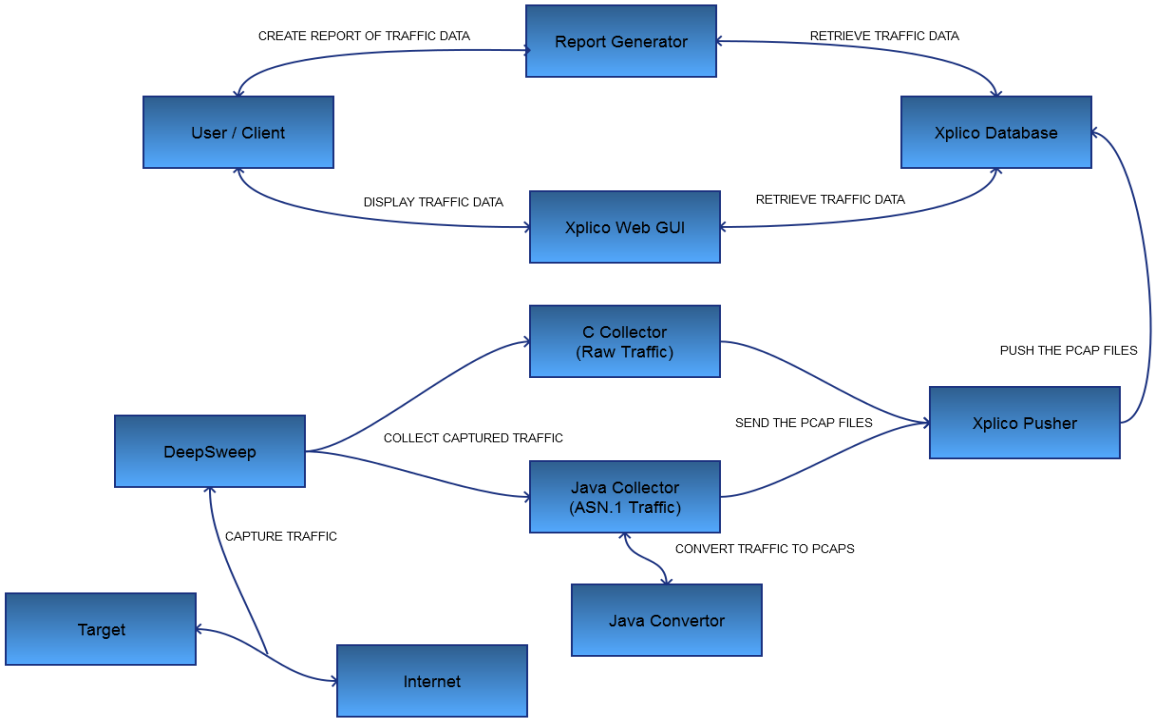

Figure 9: System Diagram for Phase Two

Phase Two (Figure 2) functions essentially the same way, except the Collector and Converter will be replaced with versions written in Java. Users also will be able to either view the traffic in Xplico's web GUI or create PDF reports of the traffic once it is uploaded.

3.6 TEST SPECIFICATION

Several different kinds of tests will be run for our project:

White/Black Box Unit Tests: When we write code for the Collectors, we will write tests for each class and/or package. Whenever changes are made to the code, the person who changed it will be responsible for updating the tests. Tests will be a mix of white box and black box tests.

Black Box Component Tests: Once we complete a component of our project, we will perform black box tests of the component. These tests will verify that the component generates the correct output when given valid input and that it handles error conditions properly.

System/Integration Tests: Once all components are complete, we will perform a black box test of the entire system whenever it is feasible. These tests will ensure that all components of the system work properly with each other.

Soak and Stress Tests: Soak tests will be performed if possible, so we can ensure the system works reliably over time. Stress tests will also be performed if possible, so we can verify the system can handle heavy loads of traffic.

Benchmark Tests: Benchmark tests will be performed if possible, so we can determine what limitations the system has. In particular, we plan to test how fast Xplico can upload pcaps and how quickly our Collectors can create pcaps.

Phase 1

1. Verify that the Pusher and Collector start correctly when started via the GUI
2. Test that the C Collector collects raw traffic and properly creates pcaps from it
3. Verify that data from pcaps created by both collectors uploads to Xplico correctly
4. Run integration and stress tests on the entire system once connected

Phase 2

1. Test that the Java Collector collects ASN.1 traffic and properly creates pcaps from it
2. Verify creating a report includes all the data selected by the user
3. Run regression tests consisting of tests from Phase 1 on the entire system
4. Run integration and stress tests on the entire system once connected

4 Conclusion

Our project will create an interface between Xplico and DeepSweep. Since DeepSweep can emit traffic as raw or in ASN.1 format, we will write a program for each format. These collectors will convert the traffic into pcaps, since Xplico only imports traffic stored in pcaps.

Once a prototype of the interface is developed, we will modify the program's existing graphical user interface. This existing GUI displays traffic sorted by its type (e.g. pictures, emails, http, ftp, etc.) and provides a summary of the total traffic uploaded. Our modifications will allow a user to select a case and session for uploading traffic and then start the collector. Users can also generate reports consisting of some or all of the uploaded traffic in a session or case.