Project Plan 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)

Version	Date	Author	Change
1.0	10/02	AH	Created 1st version of project plan
2.0	11/06	AH	Created 2nd version of project plan

Table of Contents

1 Introduction 3

1.1 Problem Statement 3

1.2 Market / Literature Survey 3

1.3 Definitions, Acronyms, Abbreviations 3

1.4 References 3

2 Project Management 4

2.1 Work Breakdown Structure 4

2.2 Project Schedule 4

2.3 Risks and Mitigation Strategies 5

3 Requirements 7

3.1 System Diagrams 7

3.2 System Description 8

3.3 Operating Environment 8

3.4 User Interface Description 8

3.5 Functional Requirements 9

3.6 Non-Functional Requirements 9

3.7 Resources 10

3.8 Deliverables 10

1 Introduction

1.1 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.

1.2 MARKET / LITERATURE SURVEY

The client had previously looked at Xplico as a possible GUI for 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, and therefore, they do not work for our purposes.

Part of the interface between Xplico and DeepSweep will require creating pcap files. For creating pcap files, the most common library is libpcap, which is written in C. While Xplico is written in C, DeepSweep is written in Java. Therefore, we will eventually need a Java library for handling pcaps. There are several alternatives, but we plan to use jnetpcap because it is well-documented (e.g. tutorials, example code, Javadoc).

1.3 DEFINITIONS, ACRONYMS, ABBREVIATIONS

(None at this time – Will add definitions as needed)

1.4 REFERENCES

(None at this time – Will add references as needed)

2 Project Management

2.1 WORK BREAKDOWN STRUCTURE

Manager	Component
Altay Ozen	Bug/Feature Tracking
Abraham Devine	Web Site
Andy Heintz	Documents (Project Plan / Design Doc)
Everyone	Research
Everyone	Development
Everyone	Testing

Table 1: Work Breakdown Structure

The leader of each part delegates tasks as necessary to other team members.

Altay, as team lead, serves primary liaison for both Curt Schwaderer (our client) and Dr. Joseph Zambreno (our advisor).

2.2 PROJECT SCHEDULE

Task	Days	Start Date	End Date
Project Research	15	09/12/14	10/02/14
Investigate alternatives to Xplico	10	09/12/14	09/25/14
Study Xplico code and docs	5	09/12/14	09/18/14
Study DeepSweep code and docs	5	09/19/14	09/25/14
Study Go Collector code and docs	5	09/26/14	10/02/14
Documentation and Requirements	59	09/19/14	12/10/14
Project Plan	55	09/19/14	12/04/14
Create 1st version of plan	10	09/19/14	10/02/14
Create 2nd version of plan	10	10/24/14	11/06/14
Create final version of plan	10	11/21/14	12/04/14
Design Document	40	10/10/14	12/04/14
Create initial design	10	10/10/14	10/23/14
Create final design	15	11/14/14	12/04/14
Web Site	11	09/26/14	10/10/14
Build initial/basic web site	6	09/26/14	10/03/14
Build final web site	6	10/03/14	10/10/14
Final Presentation	3	12/08/14	12/10/14

Table 2: Research and Documentation Schedule

Task	Days	Start Date	End Date
Development	161	09/05/14	04/17/15
Phase 1	66	09/05/14	12/05/14
Install Xplico on Linux VM	5	09/05/14	09/11/14
Compile and run Go Collector	25	10/03/14	11/06/14
Write Perl Pusher	5	10/17/14	10/23/14
Connect all parts of the system	10	11/07/14	11/20/14
Release initial product to client	1	12/05/14	12/05/14
Phase 2	66	01/16/15	04/17/15
Write Java Collector	10	01/16/15	01/29/15
Write Java Convertor	10	01/23/15	02/05/15
Write Java Pusher	10	01/30/15	02/12/15
Modify Xplico by adding report generation	10	03/27/15	04/09/15
Replace Go Collector with Java modules	10	03/06/15	03/19/15
Release final product to client	1	04/17/15	04/17/15
Testing	115	11/07/14	04/16/15
Phase 1	20	11/07/14	12/04/14
Test Pusher with Collector	5	11/07/14	11/13/14
Test Pusher with Xplico	5	11/14/14	11/20/14
Test integrated system	10	11/21/14	12/04/14
Phase 2	45	02/13/15	04/16/15
Test new Java modules	15	02/13/15	03/05/15
Test integrated system again	10	03/20/15	04/02/15
Test Xplico modifications	5	04/10/15	04/16/15

Table 3: Development and Testing Schedule

2.3 RISKS AND MITIGATION STRATEGIES

Risk	Probability	Criticality	Risk Factor	Mitigation
Obtaining a test environment for testing the entire system may be difficult to obtain.	20%	90	0.2*80 = 18	We will work with the client to figure out the best solution for obtaining a test environment. If one can't be obtained, we will obtain test data for those parts which we cannot set up
Collector may not be able to handle the rate at which collected traffic comes from DeepSweep.	20%	80	0.15*80 = 16	Our team left plenty of time in our schedule for running soak / stress tests on the entire system and making corrections based on the test results.

Team does not have sufficient experience and/or knowledge to complete the requirements.	30%	50	0.3 * 40 = 15	A number of team members have experience with the project's technologies. When planning the project's schedule, we included time for researching the technologies.
Pusher may not be able to process pcaps at or faster than they are created.	20%	50	0.2 * 50 = 12	Our team left plenty of time in our schedule for running soak / stress tests on the entire system and making corrections based on the test results. If necessary, we will run several threads for the Pusher module.
Team members do not finish their assigned tasks on schedule.	40%	25	0.4 * 25 = 10	Track each team member's progress in weekly reports. If tasks are not finished on time, talk with them about these concerns. Adjust their tasks and schedule if necessary.
There may be challenges finding a library for handling pcap files.	10%	90	0.10*90 = 10	One of the team members has looked into this, and has found a Java library that might work. If it does not work, it may be possible to access a C library via JNI (Java Native Interface).
Extending Xplico goes beyond what the team is able to finish in two semesters.	20%	20	0.2*20 = 4	Create a set list of deliverables and functionalities by the end of this semester. If scope creep happens, separate the list into essential tasks that need to be finished and less essential tasks that aren't necessary.
Application will not run on a major distribution of Linux	15%	20	0.15*20 = 3	Our application will be tested on all major Linux distributions if necessary (Ubuntu, Red Hat, Kali, Fedora, openSUSE).

Table 4: Risks and Mitigation Strategies

3 Requirements

3.1 SYSTEM DIAGRAMS

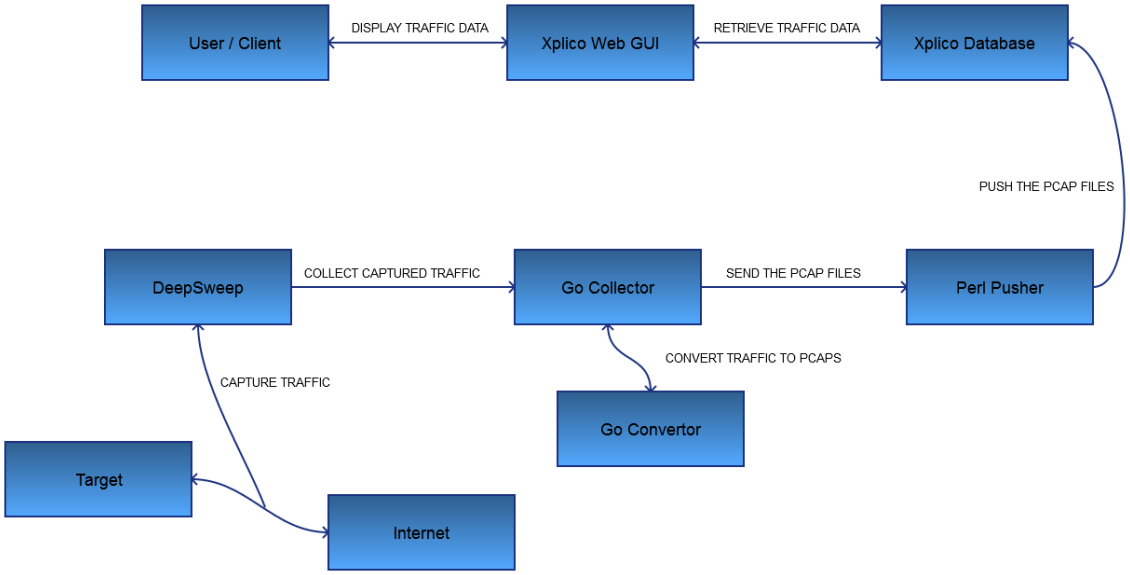

Figure 1: System Diagram for Phase One

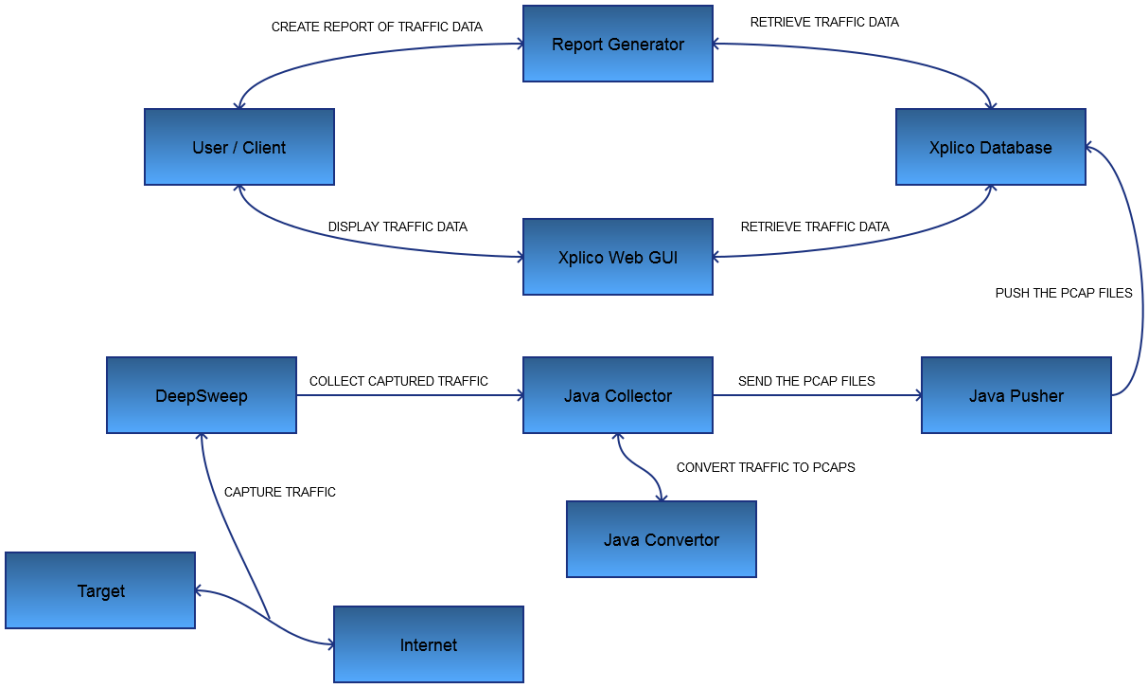

Figure 2: System Diagram for Phase Two

3.2 SYSTEM DESCRIPTION

In Phase One (Figure 1), a user configures the DeepSweep device to filter only the traffic from their desired target. Deepsweep captures the traffic between the target and the internet, and sends the raw traffic to the Go Collector. The Go Collector collects the traffic and converts it into pcap files. The Perl Pusher scans a directory for new pcap files and uploads them into Xplico's server, at which point the user can view them in Xplico's web GUI.

Phase Two (Figure 2) functions essentially the same way, except the Collector, Converter, and Pusher 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.

3.3 OPERATING ENVIRONMENT

Our client's DeepSweep API and Go Collector are built to run on Linux. Xplico can run on any version of Linux, but is easiest to install on Ubuntu 11.04 or later. Installation instructions for Xplico can be found here: http://wiki.xplico.org/doku.php?id=ubuntu.

3.4 USER INTERFACE DESCRIPTION

We will be using Xplico, which already contains a user interface. This user interface is a web based UI developed in PHP, accessed at http://localhost:9876 or http://localhost:9880. Before logging in, the user must start Xplico on the command line.

Once logged in, a user can create cases for uploading pcap files. Inside a case, the user can create a new session to upload pcap files for a specific time frame. Once a case is created with a new session, the user can upload pcap files. After Xplico finishes decoding the file, it will update the page listing how much of each data types (e.g. pictures, emails, http, ftp, etc.) was found. The data itself can be viewed by selecting the appropriate category from a menu on the left.

Xplico will be modified, so users can generate reports of the uploaded traffic. When viewing a session, the user can select which the data types to include and Xplico will generate a PDF report of only those data types. When viewing the list of cases, the user can select a case and Xplico will generate a PDF report for that case. Once generated, the user can view it on a page listing all generated reports. Clicking a report will open it up in the web browser.

3.5 FUNCTIONAL REQUIREMENTS

Phase 1

1. Collector (existing): Collect traffic from DeepSweep
2. Convertor (existing): Create pcap files from traffic collected by Collector
3. Pusher: Search directory for new pcap files and upload them to Xplico
4. Xplico (existing): Display uploaded traffic in its web GUI

Phase 2

1. Xplico: Start Java app containing Collector, Convertor, and Pusher
2. Collector: Collect raw traffic from DeepSweep and pass it to the Convertor
3. Convertor: Take traffic from the Collector and put it into pcap files
4. Pusher: Take pcap files and upload them to Xplico
5. Xplico (existing): Display uploaded traffic in its web GUI
6. Xplico: Provide options in its web GUI for generating reports of uploaded traffic

3.6 NON-FUNCTIONAL REQUIREMENTS

1. Reliability: Application should return information consistently and recover from errors.
2. Maintainability: Documentation should be up-to-date and accurate; application should be easy to upgrade to the latest version.
3. Security: Application should protect users' passwords and data from unauthorized access.
4. Extensibility: Future developers should be able to easily add additional features using the existing architecture.
5. Response Time: Application should respond promptly to users' queries and requests.

3.7 RESOURCES

- Hardware
    - DeepSweep device (Provided by client)
    - Linux Server or VMs (Provided by us)
- System Software
    - DeepSweep APIs (Provided by client)
    - Go Collector (Provided by client)
    - Xplico (Open source – Available online)
    - Pcap libraries (Open source – Available online)
- Development Software
    - Eclipse (Open source – Available online)
    - Java 7 (Open source – Available online)
    - Go (Open source – Available online)
    - Perl (Open source – Available online)

3.8 DELIVERABLES

- Project Plan: Documentation of project's schedule and requirements
- Design Document: Documentation of project's design
- Xplico: Program modified so users can create reports
- Pusher: Program for uploading pcap files to Xplico via command line
- Collector: Java replacement for existing Go Collector
- Convertor: Java replacement for existing Go Convertor