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)

Curt Schwaderer (Client)

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
3.0	11/23	AH	Created final version of project plan

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 Project Management

2.1 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).

2.2 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 our client and adviser.

There are no costs for this project, since the software is either free or supplied by our client.

2.3 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
Presentation	13	11/21/14	12/09/14
Presentation Slides	5	11/21/14	11/27/14
Final Presentation	1	12/09/14	12/09/14

Table 2: Research and Documentation Schedule

Task	Days	Start Date	End Date
Development	150	9/5/2014	4/2/2015
Phase 1A	65	9/5/2014	12/4/2014
Install Xplico on Linux VM	5	9/5/2014	9/11/2014
Investigate Pusher options	25	10/17/2014	11/20/2014
Write C Collector	20	11/7/2014	12/4/2014
Phase 1B	95	9/26/2014	2/5/2015
Compile and run Go Collector	10	9/26/2014	10/9/2014
Start C Collector from Xplico	10	1/16/2015	1/29/2015
Start Go Collector from Xplico	5	1/30/2015	2/5/2015
Phase 2A	45	1/23/2015	3/26/2015
Write Java Collector	25	1/23/2015	2/26/2015
Replace Go Collector with Java Collector	10	3/13/2015	3/26/2015
Phase 2B	30	2/20/2015	4/2/2015
Generate reports for a session in Xplico	10	2/20/2015	3/5/2015
Generate reports for a case in Xplico	10	3/20/2015	4/2/2015
Testing	113	11/18/2014	4/23/2015
Phase 1A	24	11/18/2014	12/19/2014
Test C Collector	13	11/18/2014	12/4/2014
Test integrated system	10	12/5/2014	12/18/2014
Phase 1B	26	1/16/2015	2/20/2015
Test Go Collector	5	1/16/2015	1/22/2015
Test modifications to Xplico	5	2/6/2015	2/12/2015
Test integrated system	5	2/13/2015	2/19/2015
Phase 2A	31	2/27/2015	4/10/2015
Test Java Collector	10	2/27/2015	3/12/2015
Test integrated system	10	3/27/2015	4/9/2015
Phase 2B	35	3/6/2015	4/23/2015
Test session report generation in Xplico	10	3/6/2015	3/19/2015
Test case report generation in Xplico	10	4/3/2015	4/16/2015
Test integrated system	5	4/17/2015	4/23/2015

Table 3: Development and Testing Schedule

2.4 RISKS AND MITIGATION STRATEGIES

Risk	Probability	Criticality	Risk Factor	Mitigation
Determining the start and end of packets of data may be difficult to determine.	70%	30	0.7 * 40 = 24	Some formats of data do contain info about their packet, which our application will use for finding the packet start and end. For formats that don't have packet info, our application will process the data after a large amount has been received.
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
Xplico may not be able to process pcaps at or faster than they are created.	20%	80	0.2 * 70 = 16	Our team included time in our schedule for running benchmark tests on the entire system as well as soak and stress tests. If necessary, we will run several threads for pushing pcap files.
Team does not have sufficient experience and/or knowledge to complete the requirements.	30%	50	0.3 * 40 = 15	Some team members have experience with the project's technologies. When planning the project's schedule, we included time for research.
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.
Collector may not be able to handle the rate at which collected traffic comes from DeepSweep.	10%	80	0.10*80 = 8	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.
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

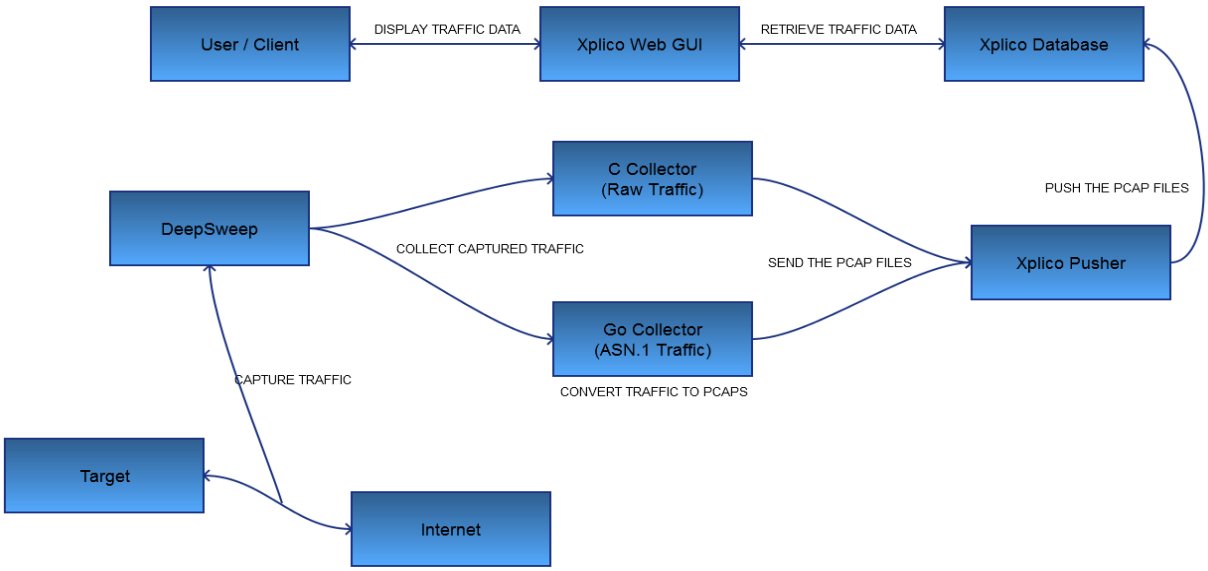

Figure 1: System Diagram for Phase One

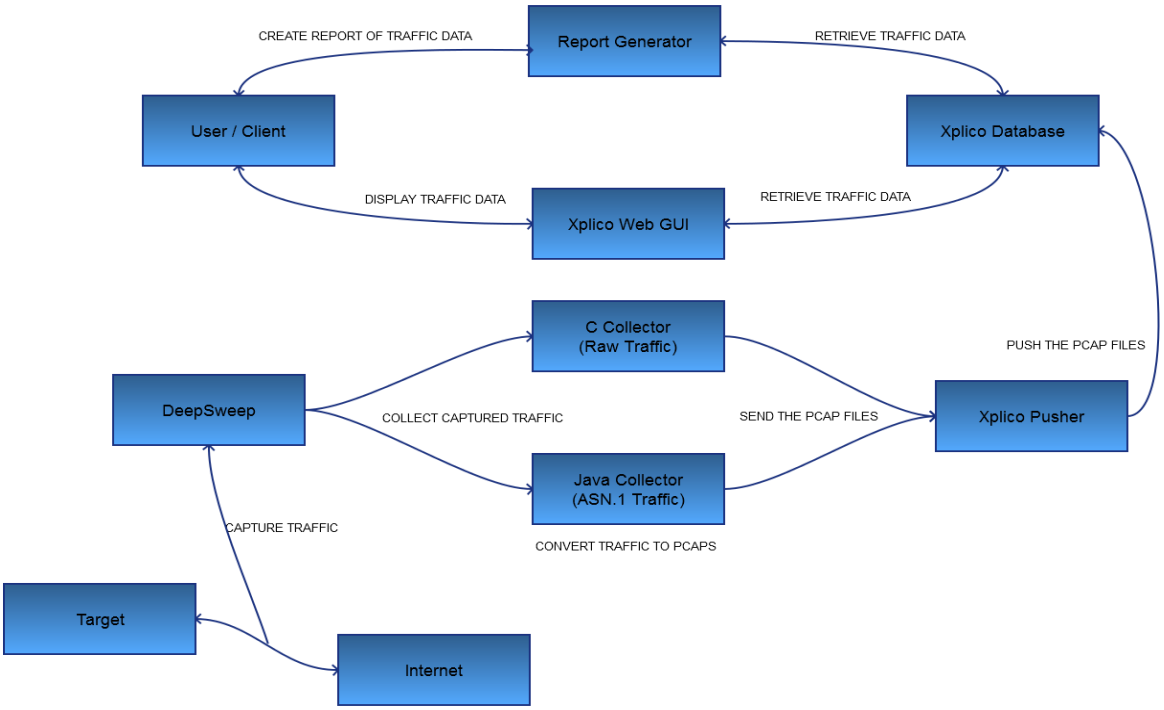

Figure 2: System Diagram for Phase Two

3.2 SYSTEM DESCRIPTION

In Phase One (Figure 1), a user configures the 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.

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.

3.3 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.

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.

Once logged in, a user can create cases and then create sessions inside of a case. Once a case is created with a session, the user can upload pcap files. After Xplico finishes decoding the file, it will update the page with statistics showing 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.

After Phase 1 is finished, a user can start the Collector and Pusher from Xplico. The user can start them from inside a session or from the list of cases. If they are started inside a session, the traffic will be uploaded to that session. If they are started from the list of cases, the user can choose whether the traffic is uploaded to a session in an existing or new case.

After the Phase 2 modifications to Xplico, a user can generate reports of the uploaded traffic. When viewing a session, a user can select which the data types to include in the report. When viewing the list of cases, the user can select a case and Xplico will generate a PDF report for all data in 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. 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

(See Test Plan for details on validation and acceptance tests.)

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 TEST PLAN

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

3.8 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)
- Languages
    - C (Open source – Part of Linux)
    - Java 7 (Open source – Available online)
    - Go (Open source – Available online)
    - PHP (Open source – Available online)

3.9 DELIVERABLES

- Project Plan: Documentation of project's schedule and requirements
- Design Document: Documentation of project's design
- Xplico: Program modified so users can start Collector and Pusher
- Xplico: Program modified so users can create reports
- C Collector: Program for creating pcap files from raw traffic
- Java Collector: Program for creating pcap files from ASN.1 traffic

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.