HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
INFORMATIK

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

HAPviewer v2.0

Bachelorarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Frühjahrssemester 2011

Autoren:	Reto Schneider, Sebastian Hügli
Betreuer:	Prof. Eduard Glatz
Projektpartner:	ETH Zürich
Gegenleser:	Roberto Pajetta

Erklärung über Eigenständigkeit der Arbeit

Wir erklären hiermit,

- dass wir die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt habe, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde,
- dass wir sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben haben.

Rapperswil, June 17, 2011:

Name, Unterschrift

Name, Unterschrift

Abstract

Aufgabenstellung

Der HAPviewer ist eine Software, mit welcher sich NetFlow Daten visualisieren und analysieren lassen.
In dieser Arbeit soll der Funktionsumfang des HAPviewers erweitert werden. Zu den Erweiterungen gehört eine Möglichkeit, Graphen partiell zu desummarisieren, sowie die Unterstützung von IPv6 Daten.
Unterstützung für die neuen Features soll auch in das im Herbstsemester 2010/2011 im Rahmen einer Studienarbeit erstellte HAP4NfSen Plugin eingebaut werden.

Ziel der Arbeit

Die Funktionalität der Standalone Applikation HAPviewer soll um wesentliche Eigenschaften erweitert werden. Im Einzelnen sind dies die Möglichkeit, mit IPv6-Daten umgehen zu können, eine partielle Desummarisierung zu implementieren sowie die Rollendetektion zu verbessern. Auch sollte der bestehende C++-Code wo sinnvoll einem Refactoring unterzogen werden und das bisherige Makefile durch ein flexibleres Build-System ersetzt werden. Bei genügend Zeit kann zudem der HAPviewer an weitere Betriebsysteme angepasst werden, eine Importmöglichkeit für Argus-Daten geschaffen werden sowie ein alternativer Einstiegspunkt für das NfSen-Plugin erstellt werden.

Lösung

Die Software HAPviewer kann neu IPv6-Daten in den Formaten cflow, IPFIX, pcap und nfdump einlesen und als cflow abspeichern. Es wurde die Möglichkeit geschaffen, summarisierte Konten im HAPviewer anzuklicken und damit Rollen an Stelle von ganzen Rollentypen zu desummarisieren. Bei auftretenden Rollenkonflikten versucht der HAPviewer neu, diese intelligenter als zuvor zu eliminieren, indem er eine Bewertung aus Sicht der entfernten IP vornimmt und dabei die gesamten Flowdaten zur Beurteilung heranzieht.

Contents

Part I
Technischer Bericht

1 Ausgangslage

In der letztjährigen Studienarbeit wurde ein Plugin erstellt, mit dem NetFlow Daten aus der Webapplikation NfSen mit Hilfe einer Library Version des HAPviewer visualisiert werden können.
Das Plugin erzeugt klickbare Graphiken, mit denen es möglich ist, dynamisch NetFlow Filter zu setzen und Rollensummarisierungen aufzuheben.[1]

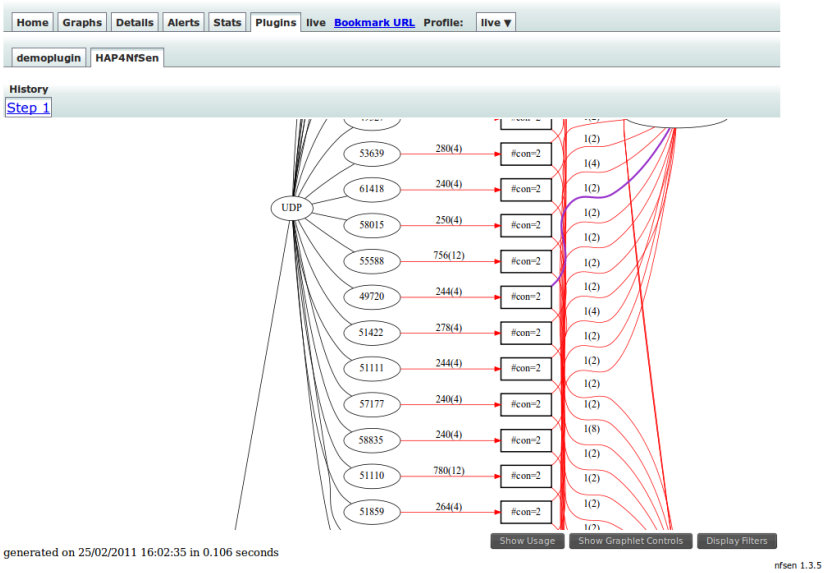


Figure 1: Beispiel eines Graphlets, generiert durch das HAP4NfSen Plugin

2 Ergebnissbericht

2.1 Domain Modell

Beim HAPviewer handelt es sich um eine bereits bestehende Applikation deren Funktion und Einsatz bereits mehrfach dokumentiert wurde. Es sei daher an dieser Stelle auf die Präsentationunterlagen[7] des im Jahr 2010 von Herrn Glatz an der VizSec gehaltenen Vortrags sowie der Projektseite bei SourceForge[2] verwiesen.

2.2 Requirements

2.2.1 Anwendungsspezifikation

Beim HAPviewer handelt es sich um eine Applikation, welche vorrangig wissenschaftlichen Zwecken dient und daher für ein begrenztes und fachlich versiertes Publikum bestimmt ist. Allem voran ist es wichtig, dass die Daten korrekt ausgewertet werden. Eine perfekt durchdachte Handhabung oder optimierte Performance etwa sind zwar wünschenswert, haben für diese Arbeit aber keine Priorität.

2.2.2 Funktionale Anforderungen

Die funktionalen Anforderungen lassen sich der Aufgabenstellung entnehmen:

- Support von IPv6 Daten (Handhabung als auch Import)
- Partielle Desummarisierung
- Flexibles Build-System, verschlankte Abhängigkeiten
- Verbesserung der Rollendetektion bzw. Konfliktlösung
- Optional: Klickbare Graphlets
- Optional: Importmöglichkeiten für Argus Daten
- Optional: Unterstützung weiterer Betriebsysteme

2.2.3 Nichtfunktionale Anforderungen

Technologien

Der HAPviewer setzt auf C++, GTK(mm), Graphviz, Boost sowie Doxygen auf. Weitere Bibliotheken werden für die Unterstützung der Importformate benötigt.

Abhängigkeiten

Weitere Abhängigkeiten sollten vermieden werden um die Installation nicht weiter zu erschweren. Lösungen müssen sich also innerhalb der bereits existenten Technologien finden lassen. Das neue Buildsystem ist hierbei eine Ausnahme.

Leistung

Graphlets von einigen Tausend Knoten sollten innerhalb innert weniger Sekunden dargestellt werden können. Bei zu vielen Daten muss der Benutzer gewarnt werden.

Bedienbarkeit

Die Bedienung soll sich an den bereits bestehenden Tools(HAPviewer und HAP4NfSen Plugin) orientieren, damit für bestehende Benutzer ein geringer Einarbeitungsaufwand entsteht.
Im Plugin soll die eingebaute Hilfefunktion um die im Rahmen dieser Arbeit erstellten Erweiterungen ergänzt werden.

2.2.4 Use Cases

Bei den folgenden Use Cases handelt es sich grössten Teil um Erweiterungen der bereits in der Vorgängerarbeit erstellten Anwendungsfällen[1]. Daher kommt es zu einigen Überschneidungen.

Essential

IPv6 Graphlet anzeigen

Name:	IPv6 Graphlet anzeigen
Description:	Ein Benutzer ruft zu einer bestimmten IP aus einer Liste ein HAP Graphlet auf.
Preconditions:	Der Benutzer hat sich innerhalb des HAPviewer die Liste der local IPs anzeigen lassen. Die Liste enthält mindestens eine IPv6 Adresse.
Postconditions:	Ein Fenster mit einem IPv6 HAP Graphlet wird dem Benutzer angezeigt.

Ablauf:

1. Der Benutzer möchte ein HAP IPv6 Graphlet zu einer bestimmten IP(v6) innwähreerhalb einer Liste von Netflows sehen.
2. Der Benutzer wählt die gewünschte IP(v6) Adresse.
3. Das System zeigt dem Benutzer das erstellte HAP IPv6 Graphlet in einem neuen Fenster an.
4. Ende des Use Cases.

Fully Dressed

Partielle Desummarisierung

Name: Partielle Desummarisierung

Scope:	System under Design
Level:	User Goal

Primary Actor: Benutzer der Applikation

Stakeholders and Interrests:

- Benutzer
  - Möchte mehr Details zu einem bestimmten Teil des HAP Graphlets sehen, den Rest der Graphik aber summarisiert lassen.

Preconditions:

- Benutzer befindet sich im HAPviewer.
- Dem Benutzer wird bereits ein HAP Graphlet angezeigt.
- Das angezeigte Graphlet besitzt mehrere zusammengefasste Knoten.

Success Guarantee: Ein neues HAP Graphlet wird dem Benutzer angezeigt.

Main Success Scenatio:

1. Der Benutzer möchte Details zu einem der zusammengefassten Knoten eines angezeigten HAP Graphlet sehen.
2. Der Benutzer wählt den gewünschten Knoten aus.
3. Das System hebt die gewünschte Aggregation auf und zeigt dem Benutzer ein neues HAP Graphlet an. Die ausgewählte Partition der gewünschten Rolle wird desummarisiert und die zuvor zusammengefassten Werte werden einzeln angezeigt.
4. Ende des Use Cases.

Extentions:

- Wenn bei der Erstellung des neuen Graphlets ein technischer Fehler auftritt, wird der Benutzer mit einer Meldung darüber informiert.

Technology and Data Variations List: Die verwendeten Daten wurden vom HAPviewer aus Dateien in unterstützen Formaten eingelesen.

Frequency of Occurrence: Pro Benutzer: Mehrmals pro Minute

Alternativer Einstiegspunkt

Name: Alternativer Einstiegspunkt

Scope: System under Design

Level: User Goal

Primary Actor: Benutzer der Applikation

Stakeholders and Interrests:

- Benutzer
  - Möchte eine Menge von NetFlows graphisch dargestellt haben, um die Auswahl einer Einstiegs IP für das HAP4NfSen zu verienfachen.

Preconditions:

- Benutzer befindet sich auf der Details Seite in NfSen.
- Es sind Daten für den aktuell ausgewählten Zeitbereich vorhanden.

Success Guarantee: Eine klickbare Graphik mit einer Visualisierung aller im ausgewählten Zeitbereich vorhandenen IP Adressen wird dem Benutzer angezeigt.

Main Success Scenatio:

1. Der Benutzer möchte eine Visualisierung aller IP Adressen im ausgewählten Zeitbereich sehen.
2. Der Benutzer wählt eine der dargestellten IP Adressen aus.
3. Das System leitet den Benutzer auf die HAP4NfSen Plugin Seite weiter, wo ihm das gewünschte HAP Graphlet angezeigt wird.
4. Ende des Use Cases.

Extentions:

- Wenn bei der Erstellung des neuen Graphlets oder der Visualisierung des Einstiegspunkts ein technischer Fehler auftritt, wird der Benutzer mit einer Meldung darüber informiert.

Technology and Data Variations List:	Die verwendeten Rohdaten werden von NfSen zur Verfügung gestellt.
Frequency of Occurrence:	Pro Benuter: Mehrmals pro Minute

Diagram

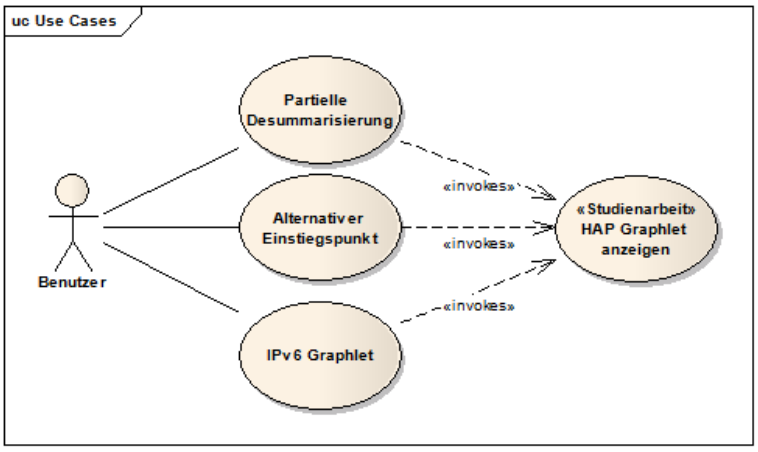


Figure 2: Use Case Übersicht

2.3 Refactoring

In den folgenden Abschnitten werden die wichtigsten Bereiche, welche im Laufe der Arbeit refactored, resp. überarbeitet wurden aufgelistet.
Dabei ist jeder Abschnitt in drei Teile gegliedert, wobei der erste die Ausgangslage, der zweite unsere Überlegungen und Verbesserungsideen und der dritte die erstellte Lösung beschreibt.

2.3.1 Abhängigkeiten zu Programmbibliotheken zur Library Version

Die Library Version des HAPviewers referenziert einige externe Bibliotheken, welche für die Generierung von Graphlets eigentlich nicht benötigt werden.
Dies ist besonders auf reinen Serversystemen ein Problem, wo die Installation der Libraries(z.B. GTK) einen grossen Zusatzaufwand bedeuten würde.

Ausgangslage

Zu Beginn der Arbeit hatte der HAPviewer folgende Abhängigkeiten:

- boost (regex, thread, iostreams, filesystem)
- graphviz
- gtk2mm
- pcap & pcap++
- ipfix

Die Bibliotheksversion benötigte folgende Software:

- boost (regex, thread, iostreams, filesystem)
- graphviz
- giomm
- swig
- pcap & pcap++
- ipfix

Anpassungen

Der HAPviewer als auch die Bibliotheksversion wurden so geändert, dass es möglich ist, einzelne, oftmals nicht benötigte Importfilter (cflow, pcap, ipfix, nfdump) nicht zu kompilieren. Somit müssen dann auch die damit verbundenen Abhängkeiten nicht auf dem Zielrechner vorhanden sein.

Lösung

Bei folgenden Libraries hat sich etwas geändert:

- boost: boost_thread ist nicht mehr erforderlich
- pcap & pcap++: Unterstützung für pcap ist neu optional.

- ipfix: Unterstützung für ipfix inkl. der Abhängigkeit GLib ist nur noch optional.
- cflow: Unterstützung für cflow inkl. den Abhängigkeiten boost_iostreams sowie boost_filesystem ist optional
- argus: Die neu hinzugefügte Unterstützung für Argus-Daten ist ebenfalls nur optional.
- giomm: Wird für die Bibliotheksversion nicht mehr benötigt.

2.3.2 Interne Speicherstruktur der Flows

Intern wird jeder Flow in einem Objekt der Klasse cflow gespeichert.

Ausgangslage

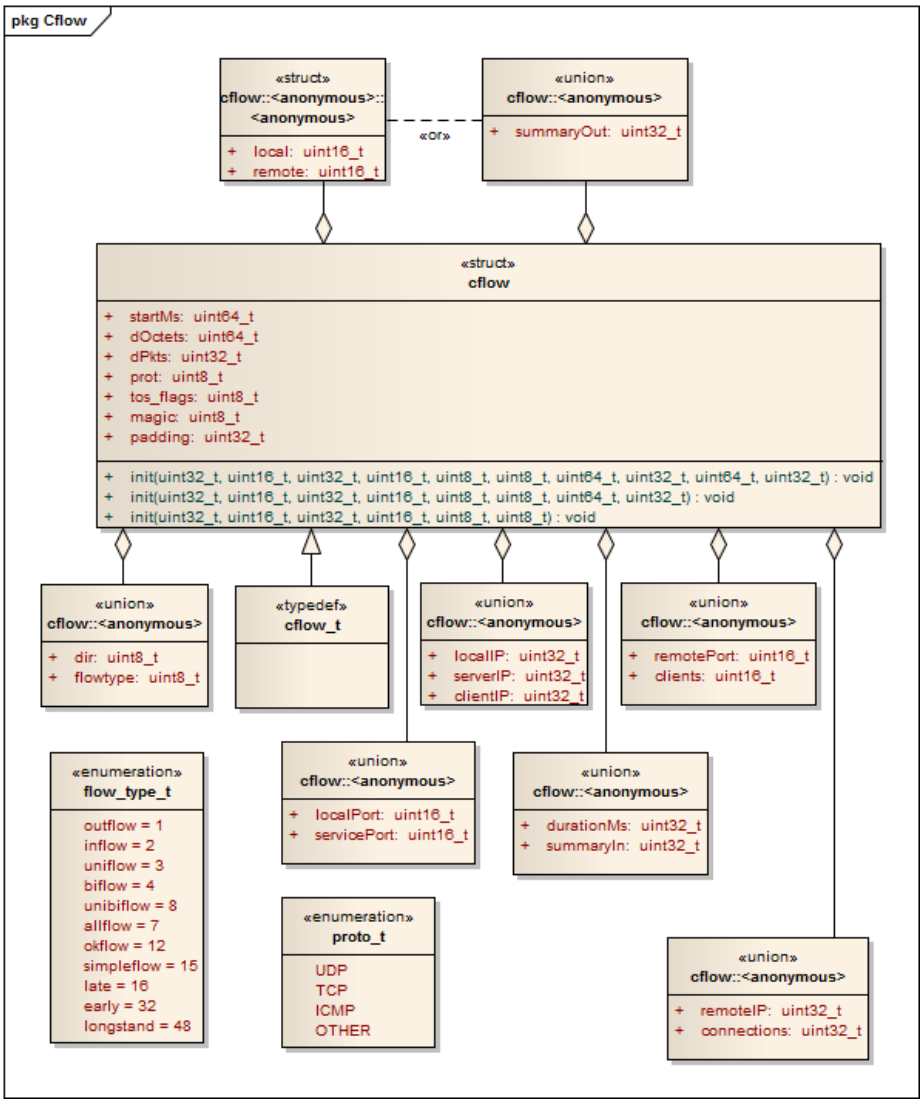


Figure 3: Klassendiagramm der alten cflow Klasse. Die Stereotypen <<union>> und <<or>> stehen für die Union-Funktionalität von C++.

Anpassungen

Zu kleine Datenfelder Die ursprüngliche Variante war nur für IPv4 Daten ausgelegt und hatte entsprechend kleine Felder für die Speicherung der IP. Dasselbe gilt für die 16 Bit Felder welche für die AS-Nummern verwendet wurden - mittlerweile werden bereits 32 Bit Nummern für AS-Netze vergeben[5].

Unions Viele Felder konnten mittels Unions unter mehreren Namen abgerufen werden und es wurden einige Felder je nach Situation zur Speicherung unterschiedlicher Daten verwendet.

Magic Feld Das magic Feld, welches zur Prüfung der Dateiintegrität sowie zur Detektion des cflow-Formats beim Einlesen von der Harddisk eingesetzt wird, befand sich nicht am Anfang der Klasse. Es wird daher durch eine Änderung der Feldgrösse vorangehender Daten verschoben und kann damit nicht mehr zur Versionsdetektierung verwendet werden.

Lösung

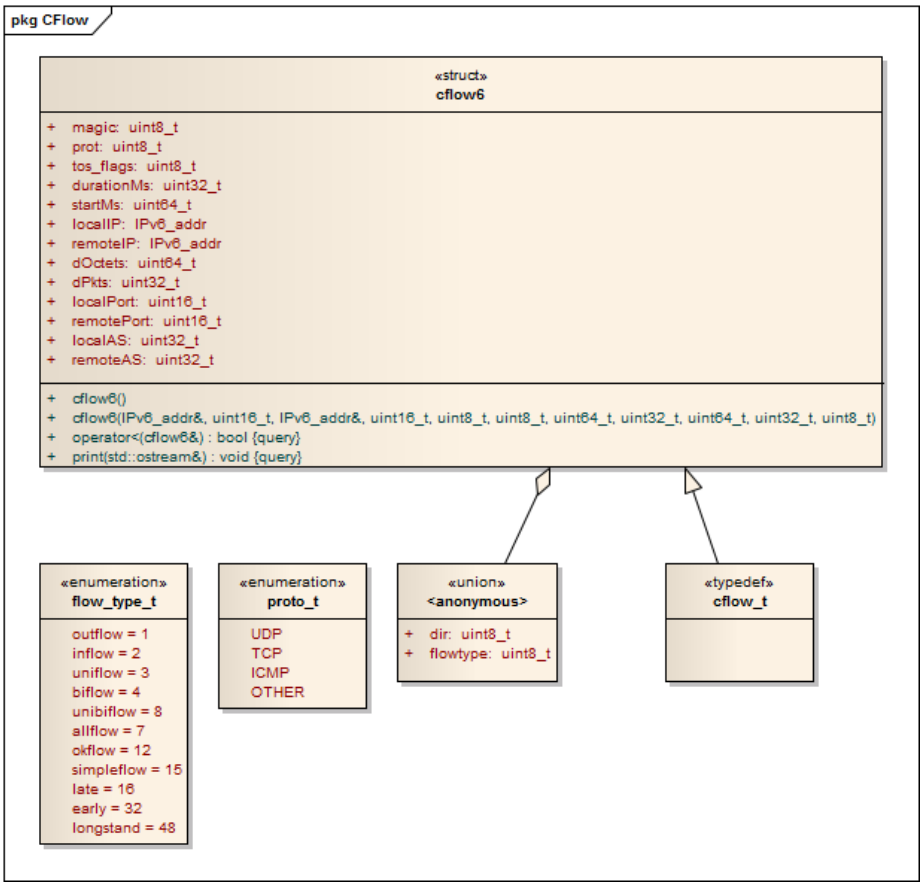


Figure 4: Klassendiagramm der neuen cflow6 Klasse. Die Stereotypen <<union>> und <<or>> stehen für die Union-Funktionalität von C++.

Zu kleine Datenfelder Die 32 Bit Datenfelder für IP Adressen wurden durch 128 bittige ersetzt, die Felder für AS-Nummern wurden von 16 auf 32 Bit vergrössert.

Unions Da IPv6 Adressen nicht mehr in einfachen Dateitypen, welche von C++ zur Verfügung gestellt werden, gespeichert werden können, verwendeten wir dafür eine eigene Klasse, IPv6_addr. Da Unions ein Relikt aus C sind, welche sich nur sehr schlecht mit OOP Prinzipien vereinen lassen und keine, nicht ausschliesslich default-konstruierbaren Typen wie unsere IPv6 Klasse, enthalten können, haben wir weitgehend auf Unions verzichtet.

Magic Feld Das 8 Bit Feld magic liegt neu auf der Adresse 0 der Klasse, sodass in Zukunft bei Erweiterungen dieses Feld auch dann zuverlässig ausgelesen werden kann, wenn sich andere Elemente verschieben. Zwar wäre es auch möglich gewesen, das Feld an seiner bisherigen Position zu belassen und die restlichen Felder entsprechend anzuordnen, im Hinblick auf zukünftige Formate erschien uns dies aber als eine hinderliche Altlast welche wir nicht mitschleppen wollten.

Ausrichtung Dadurch, dass sich neu ein 8 Bit Feld am Anfang der Klasse befindet, mussten, um die 16 Byte IPv6_addr Felder wieder an geeigneten Adressen zu haben, weitere 8 Bit Felder nach vorne verschoben werden. Die Felder konnte damit wieder über Adressen, welche ein Mehrfaches von 8 sind, abgerufen werden, was unnötige Speicherzugriffe verhindert.

Klassennamen Der Klassenname der ursprünglichen Klasse wurde auf cflow4 geändert und ist weiterhin zwecks Import von alten Daten verfügbar. Die von uns modifizierte Variante heisst cflow6. Der typedef cflow_t wurde auf cflow6 gelegt. Innerhalb des Programms wird cflow_t verwendet und ermöglicht damit, in Zukunft an einer einzigen Stelle die intern im Programm verwendete cflow-Klasse zu ändern.

2.3.3 Flow-Datenimport

Der HAPviewer unterstützt den Flowdaten-Import von NfDump-, IPFXI-, pcap- und cflow-Dateien. Zusätzlich können Flows in cflow-Dateien abgespeichert werden.

Ausgangslage

Bisher wurden die unterstützen Importformate (NfDump, IPFIX, pcap) direkt von Methoden der Klasse CImport eingelesen. Gleiches gilt für das Lesen von cflow-Dateien, wobei diese auch geschrieben werden können. Der Code zum Schreiben der cflows befand sich in der GUI-Klasse CflowlistWindow und musste, da es auch möglich sein sollte, bestehende cflow-Dateien zu ergänzen, den gesamten Code um cflow-Datein zu lesen, nochmals enthalten.

Anpassungen

Der Code zum Einlesen von Flow-Daten wurde separiert und in eigene Klassen verschoben.

Lösung

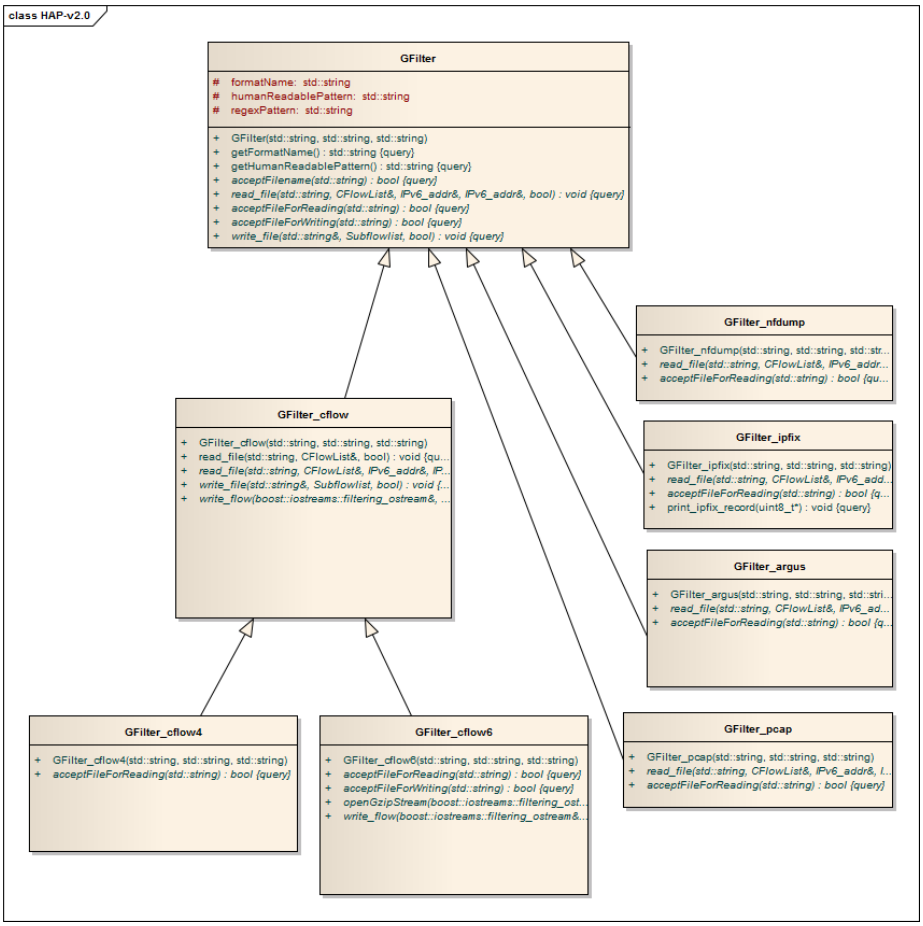


Figure 5: Klassendiagramm der neuen Import-Klassen

Es wurde eine Basisklasse GFilter erstellt, welche gewisse generische Grundfunktionen bereitstellt und von welcher alle Importfilter abzuleiten sind.
Um die Handhabung noch weiter zu vereinfachen wurde die Klasse CImport so erweitert, das unabhängig vom einzulesenden Dateityp immer die gleiche Methode aufgerufen werden kann. Auch ist es mittels statischen Methoden dieser Klasse möglich, aus anderen Teilen der Software zur Laufzeit die unterstützten Dateitypen abzufragen. Beispielsweise die GUI kann so die Filter des Dateidialogs aufgrund der tatsächlich vorhandenen Importfilter setzen.

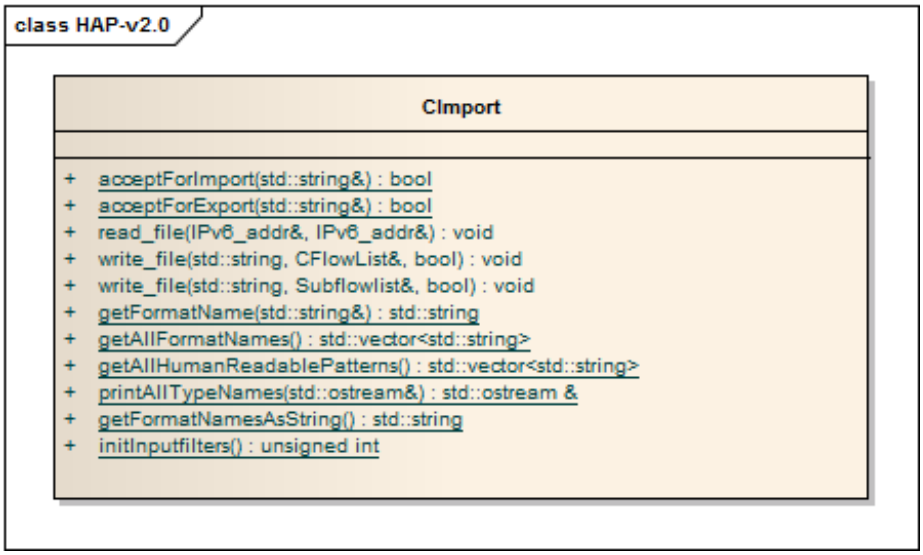


Figure 6: Klassendiagramm der neuen Import-Klassen

2.3.4 Flowliste

Die Flowliste (full_flowlist) wird im HAP Viewer beim Einlesen einer Datei erstellt und enthält die gefundenen Flows. Diese werden intern zur Generierung der Graphlets und zur Darstellung von Flows in Tabellenform verwendet.

Ausgangslage

Die Flowliste wird im Speicher abgelegt, wobei mit einem Pointer auf den ersten Eintrag und einer Variabel mit der gesamten Anzahl der Flows auf einzelne Flows zugegriffen wird. Um zu verhindern, dass das Programm über das letze Element hinausscheisst, wurde jeweils am Ende des Arrays ein zusätzlicher Flow mit dem Wert 0 abgelegt.

Anpassungen

Ein STL Vektor wird verwendet, um den Zugrff und die Handhabung der Flowliste zu vereinfachen.

Lösung

Die neue Flowliste ist folgendermassen definiert:

```
typedef std::vector<cflow_t> CFlowList;
```

Dies erlaubt einige Vereinfachungen im Code. Unter anderem ermöglicht der Vektor die Verwendung von STL Algorithmen, wie etwa das Sortieren. Der bestehende Code musste überall dort angepasst werden, wo als Abbruchbedingung das Erreichen des letzen 0-Elements gesetzt war.

2.3.5 Aktive Flowliste

Der HAPViewer enhält neben der ganzen Flowliste (full_flowlist) auch noch eine zweite Flowliste, welche nur die Elemente enhält, welche aktuell von Interesse sind.

Ausgangslage

Da die ganze Flowliste aus einem Array von Pointern besteht, kann die aktive Flowliste ebenfalls aus dem Startpointer und der Anzahl der Elemente bestehen. Dies ist sehr speichereffizient, da die Daten nicht mehrfach gehalten werden müssen.

Anpassungen

Es wurde eine eigene Klasse "Subflowlist" erstellt, welche Begin- und Enditerator einer CFlowList enhält. Damit kann das bisherige Verhalten imitiert werden.

Lösung

Die neu geschaffene Klasse Subflowliste enhält einen Begin- sowie Enditerator welche sich jeweils auf eine CFlowList beziehen. Es wurde so weit als nötig das Verhalten eines STL-Containers nachgebaut.

2.3.6 HPG Graphenformat

Das HPG Format ist eine kantenbasierte Graphenbeschreibung, wie sie vom HAPviewer gelesen und geschrieben wird.

Ausgangslage

Das ursprüngliche HPG Format erlaubt es aus Platzgründen nicht, IPv6 Adressen abzuspeichern. Details zu diesem Format sind in der Datei "hpg.h" der ursprünglichen Version des HAPviewer zu finden.

Anpassungen

Um Platz für IPv6 Adressen zu schaffen, wurde die Definition einer Kante des Formats von 3x32 Bit auf 3x128 Bit (`hpg_field` anstatt `uint32_t`) erweitert. Daneben musste auch mehr Platz für die Codierung von Rollennummern verwendet werden um deren Eindeutigkeit zu garantieren.

Lösung

Die Grundstruktur des vorgängigen HPG Formats wurde so weit wie möglich beibehalten. Das neue Format ist nun wie folgt aufgebaut:

Part 1	Graphlet Nummer und `rank_t`
Part 2	Erster Wert
Part 3	Zweiter Wert

```
union hpg_field {
    struct
    {
        uint64_t data;
    } eightbytevalue;
    boost::array<unsigned char, 16> data;
    void reset() {
        fill(data.begin(), data.end(), 0);
    }
};
```

Die Codierung der Werte, welche in das zweite und dritte Feld des Arrays geschrieben werden, wurde (bis auf die Erweiterung des Speicherplatzes für Rollennummern und die Anzahl von Hosts, welche von je 8 auf je 24 bit erweitert wurden) nicht angepasst. Um den Code übersichtlicher zu machen und Duplikationen zu reduzieren, wurde die Erstellung von `hpg_field` und die Codierung von Werten in `hpg_field` aus dem Code entfernt und in separate Funktionen ausgelagert.
Eine detailiertere Beschreibung des Formats ist im Code ("hpg.h") zu finden und wird daher hier nicht nochmals wiederholt.

2.3.7 Errorhandling

Ausgangslage

Innerhalb des HAPviewers wurden mehrere Ansätze zum Errorhandling verwendet. So wurde vorrangig mittels Rückgabewert ein unerwartete Fehler signalisiert, in eingen Fällen wurde auch auf den try/catch Mechanismus zurückgegriffen. Als Programmierer musste man den Code jeder verwendete Methode genau ansehen (inkl. den wiederum darin aufgerufenen Methoden) um angemessen auf Fehler zu reagieren.

Anpassungen

Wir versuchten, unerwartete Fehler immer mittels throw zu signalisieren und diese später mit catch zu behandeln.

Lösung

Es wurden die meisten Methoden so abgeändert, dass diese unerwartete Fehler mittels throw signalisieren und selbst mögliche Fehler mittels catch fangen und/oder weiter werfen. Um den zukünftigen Programmierern der Software die Weiterentwicklung der Software zu erleichtern, wurde die Doxygen-Dokumentation entsprechend um exception-Hinweise erweitert.

2.4 Erweiterungen

In den folgenden Abschnitten wird die Implementierung der Erweiterungen für den HAPviewer sowie das HAP4NfSen Plugin beschrieben.

2.4.1 Partielle Desummarisierung

Bei der partiellen Desummarisierung geht es um die Auflösung von einzelnen Rollen innerhalb des Graphlets. Bisher war dies nicht möglich.
Der HAP Viewer bietet die Möglichkeit, Kategorien von Rollen(z.B. Peer-to-Peer Rollen) zu desummarisieren und das HAP4NfSen Plugin bietet eine Drilldown Funktion an.
Diese bestehenden Analysefunktionen führen jedoch in vielen Fällen zu Graphlets, bei denen die Übersicht verloren geht oder nicht alle gewünschten Informationen angezeigt werden können.

Ausgangslage

HAP Viewer und HAP Library bieten Funktionen zur Deaktivierung der Summarisierung von bestimmten Rollentypen an.
Der HAP4NfSen Drilldown ist so implementiert, dass das Inputfile vor dem Aufruf der HAP Library vom HAP4NfSen Back End modifiziert wird. Die HAP Library generiert danach mit der neuen Flow Datei das angepasste Graphlet.

Anpassungen

Die jetztige Lösung kennt vier verschiedene Rollentypen, welche entweder summarisiert oder komplett desummarisiert werden können. Diese Typen sind wie folgt definiert:

```
Summarisierte Paritionen nach Rollentyp:

Server:        ( )-( )-( )-[ ]-[ ]
Client:        ( )-( )-[ ]-( )-( )
Multi Client:  ( )-( )-[ ]-( )-[ ]
Peer to Peer:  ( )-( )-[ ]-[ ]-[ ]

Erklärung:
 - "( )" repräsentiert nicht summarisierte Partitionen, "[ ]" summarisierte
 - Partitionen sind durch "-" getrennt
 - Partitionen sind in der gleichen Reihenfolge wie im Graphlet aufgelistet:
   local IP - protocol - local port - remote port - remote IP
```

Um einzelne Partitionen einer Rolle desummarisieren zu können, ist eine Unterstützung von Sub-Rollen nötig. Für den Rollentyp "Peer to Peer" ergeben sich daher folgende Sub Rollen:

```
Sub-Rollen des Typs "Peer to Peer":
Eine von 3 Partitionen desummarisiert:
( )-( )-( )-[ ]-[ ]
( )-( )-[ ]-( )-[ ]
( )-( )-[ ]-[ ]-( )
Zwei von zwei Partitionen desummarisiert:
```

```
( )-( )-( )-( )-[ ]
( )-( )-[ ]-( )-( )
( )-( )-( )-[ ]-( )
Alle drei Partitionen desummarisiert:
( )-( )-( )-( )-( )
```

Die Umsetzung von Rollen in ein Graphlet wurde bisher durch fünf Methoden erledigt(eine für single Flows, sowie je eine pro Rollentyp). Um den Code trotz den Erweiterungen übersichtlich zu behalten, soll eine neue Methode erstellt werden, welche mit allen Rollen- und Subrollentypen umgehen kann und die vier bisherigen ersetzt.

Lösung

Eine Methode, welche mit beliebigen Rollen und Sub-Rollen umgehen kann (`CGraphlet:: add_generic_role`), wurde erstellt. Um mit generischen Rollen umgehen zu können, mussten die summarisierten Partitionen der Rollen in einem einheitlichen Format vom Typ `role_pattern` abgelegt werden. Die Lösung verwendet dabei folgende Codierung:

```
enum summarization_type {
        summarized   = 0x1,
        desummarized = 0x0
};
enum graphlet_partition {
        local_ip     = 0x1,   // 2^0
        proto        = 0x2,   // 2^1
        local_port   = 0x4,   // 2^2
        remote_port  = 0x8,   // 2^3
        remote_ip    = 0x10   // 2^4
};
enum graphlet_partition_association {
        gpa_1_1, // ..( )-( )..
        gpa_1_n, // ..( )-[ ]..
        gpa_n_1, // ..[ ]-( )..
        gpa_n_n, // ..[ ]-[ ]..
        gpa_unknown// no connection between partitions
};
enum summarization_limits {
        max_pattern = 0 + (summarized * local_ip
                        + summarized * proto
                        + summarized * local_port
                        + summarized * remote_port
                        + summarized * remote_ip),  // (2^5)-1
        min_pattern = 0 + (desummarized * local_ip
                        + desummarized * proto
                        + desummarized * local_port
                        + desummarized * remote_port
                        + desummarized * remote_ip) // 0
};
enum role_type {
```

```
        server = 0 + (summarized * remote_port
                   + summarized * remote_ip),       // ( )-( )-( )-[ ]-[ ]
        client = 0 + (summarized * local_port),     // ( )-( )-[ ]-( )-( )
        multi_client = 0 + (summarized * local_port
                         + summarized * remote_ip), // ( )-( )-[ ]-( )-[ ]
        p2p = 0 + (summarized * local_port
                + summarized * remote_port
                + summarized * remote_ip),          // ( )-( )-[ ]-[ ]-[ ]
        single_flow = min_pattern                   // ( )-( )-( )-( )-( )
};
typedef uint8_t role_pattern;
```

Jede der Sub Rollen ist selbst vom Typ `role_t` und hat daher eine eigene Rollennummer. Diese wird bei der Desummarisierung benötigt, um herauszufinden, welche der Sub Rollen beim Klick auf einen summarisierten Knoten angezeigt werden soll.
Um den Umgang mit Sub Rollen und deren Codierungen zu vereinfachen, wurde die bestehende Klasse `role_t` um folgende Methoden erweitert:

```
role_t* getUsedSubRole(const desummarizedRoles& part_desum_list,
        const desummarizedRoles& mnode_desum_list);
role_t* getUsedSubRole(const role_pattern pattern,
        CRole::role_t* current_sub_role);
uint32_t getSubRoleId(const graphlet_partition&
        partition_to_be_desummarized, const role_t& parent_role);
uint8_t getSummarizationLevel() const;
virtual void create_pseudo_roles();
```

Ein Spezialfall der Desummarisierung sind Multi Summary Nodes, wie sie in der letzten Partition(remote IP) auftreten können. Diese enthalten IPs, welche von mehreren Rollen geteilt werden. Um das gewünschte Desummarisierungsverhalten(Desummarisierung von remote IPs für alle involvierten Rollen) zu erreichen, müssen bei Klicks auf diese Knoten alle betroffenen Rollen ermittelt und zur Liste der desummarisierten Rollen hinzugefügt werden. Darauf werden alle remote IPs aller involvierten Rollen einzeln angezeigt.
Die Erkennung von Multi Summary Nodes ist möglich, da diese negative Rollennummern verwenden.

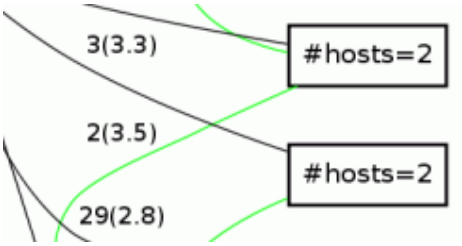


Figure 7: Multi Summary Nodes eines Graphlets

2.4.2 Verbesserte Rollendetektion

Ziel ist es, Rollenkonflikte bei der Graphletgenerierung aufzulösen.

Ausgangslage

Es gibt gewisse Fälle, bei denen ein Flow zu mehreren Rollen passt. Wenn so ein Fall auftritt, so ist dies (falls der Debugoutput aktiv ist) auf der Kommandozeilenausgabe des HAPviewer sichtbar.

```
..
INFO: ambiguous roles (client+server) for flow: 36
INFO: ambiguous roles (client+server) for flow: 37
..
```

Anpassungen

Es musste ein Algorithmus gefunden werden, welcher beim Auftreten eines Rollenkonflikts in der Lage ist, zu entscheiden, zu welcher Rolle ein Flow eher passt. Dazu ist eine Bewertung nötig, mit Hilfe von der entschieden werden kann. Diese Bewertung soll in einem vorgängigen Schritt berechnet werden.

Lösung

Die Lösung der Rollenkonflikte ist in verschiedene Phasen aufgeteilt. Diese werden in den folgenden Abschnitten erklärt:

Generierung der Rollen

Zuerst werden alle Rollen wie bisher generiert. Informationen dieser Rollen werden in den nächsten Schritt miteinbezogen.

Bewertung

Die Bewertung der Rollen ist vom Rollentyp abhängig. Ergebniss einer Bewertung ist eine Gleitkommazahl mit einem Wert zwischen 0 und 1, wobei 0 die tiefste und 1 die höchste Wertung ist.
In allen anderen Fällen werden Rollen aus der Remote-Perspektive angesehen und es wird nach weiteren passenden Flows, welche nicht im Graphlet angezeigt wurden, gesucht. Die gefundene Anzahl an Flows (innerhalb und ausserhalb des Graphlets) werden zur Bewertung verwendet.
Allgemeine Regeln
Jeder Rollentyp hat spezifiziert eine minimale Anzahl von Flows, welche jede gültige Instanz dieser Rolle besitzen muss. Damit aus Rollen mit der minimalen Anzahl an Flows keine weiteren entfernt werden, wird in diesem Fall die Höchstwertung vergeben.
Server Rollen
In Server Rollen sind bereits alle möglichen Kandidaten bekannt, da das Graphlet alle Verbindungen, welche die lokale IP involvieren, enthält.
Client- und Multi-Client Rollen
Für diese Rollentypen wird der Rest der Flowliste durchsucht. Kandidaten für zusätzliche Flows sind solche, welche nicht bereits im Graphlet enthalten sind und remote IP(auch mehrere IPs im Fall von Multi-Client Rollen), remote Port und das Protokoll der bereits in der Rolle enthaltenen Flows teilen.
Die Anzahl der so neu gefundenen Flows wird zu den bereits aus dem Graphlet bekannten Flows addiert. Aus der Summe wird die Rollenbewertung erstellt.
Peer to peer Rollen

Um Peer to Peer Rollen zu bewerten, wird ebenfalls der Rest der Flowliste durchsucht. Zu den bereits aus dem Graphlet bekannten Rollen werden solche hinzugefügt, welche folgende, aus der P2P Rollengenerierung bekannten Kriterien erfüllen:

- Der Flow ist nicht im Graphlet enthalten, kommuniziert mit einer der remote IPs der Rolle und besitzt zwei Portnummern über 1024.
- Der Flow gehört zu einer Client Rolle ausserhalb des Graphlets, wobei mit einer remote IP der P2P Rolle kommuniziert werden muss. Zusätzlich muss der remote Port des Flows über 1024 sein.

Um zu einer Liste von Client Rollen ausserhalb des Graphlets zu gelangen, werden diese temporär generiert. Die dazu verwendeten Regeln entsprechen denen der bereits bekannten Summarisierung von Client Rollen.
Wie bei den Client Rollen werden die neu gefundenen Flows zu den bereits bekannten addiert um die Bewertung zu berechnen.

Erkennung der Konflikte

Die Erkennung von Rollenkonflikten war bereits vor dem Beginn dieses Projekts vorhanden. Diese wurde erweitert und ist nun in der Lage, die gefundenen Konflikte aufzulösen.
Zu jedem Rollentyp gibt es Vektoren, welche die grösse der aktiven Flow Liste haben. Wenn ein Rollentyp einen Flow verwendet, so wird im Vektor an der Position, welche der Flow in der Flow Liste hat, die Nummer der Rolle, welche den Flow verwendet, gesetzt. Alle nicht verwendeten Flows werden mit 0 markiert. Die interne Darstellung sieht wie folgt aus:

```
Flow Liste mit 10 Flows:
[flow0,flow1,flow2,flow3,flow3,flow4,flow5,flow6,flow7,flow8,flow9]

Vektoren der Rollentypen:
Clients:
[  0,  5,  0,  5,  0,  5,  0,  0,  0,  5]
Server:
[ 10, 10,  0,  0,  9,  0,  9, 10,  9,  0]
Peer to Peer:
[  0,  0,  0,  0,  0,  0,  0,  0,  0,  0]
```

In diesem Beispiel wird der Index 1 von mehr als einem Rollentyp verwendet. Somit Kann die Situation als Rollenkonflikt zwischen Client Rolle 5 und Server Rolle 10 erkannt werden. Der Flow, welcher beide Rollen beanspruchen, ist der Eintrag in der Flow Liste mit dem Index 1, was `flow1` entspricht.

Auflösung der gefundenen Konflikten

Im Falle eines Konflikts zwischen Rollentypen wird die Bewertung der beiden Kandidaten betrachtet. Dabei wird der Flow der Rolle, welche die höhere Bewertung hat, zugewiesen und aus der anderen entfernt.

Rollenkonflikte durch übernehmen von Client Rollen

Fehler im bestehenden Code zur Summarisierung von Rollen führten zu Zuständen, welche von der Konflikterkennung als Rollenkonflikte interpretiert wurden. Durch ungültige Rollendaten war es nicht möglich, diese aufzulösen. Nach der Behebung dieser Fehler ist die Anzahl an Rollenkonflikten bei der Generierung von Graphlets stark zurückgegangen.
Das Problem im bestehenden Code entstand durch die Übernahme von bestehenden Client Rollen in Multi Client- oder Peer to Peer Rollen. Übernommene Rollen wurden zwar als ungültig markiert(Rollennummer wird auf 0 gesetzt), womit sie nicht mehr auffindbar sind, die Referenzen in den zuvor beschriebenen Vektoren zur Erkennung und Behebung von Rollenkonflikten blieben aber bestehen.

Aktualisierung der Rollendaten

Nachdem ein Flow aus einer Rolle entfernt wurde, müssen Rollendaten, wie die gesamte Anzahl von Paketen oder Flows aktualisiert werden.
Danach müssen von den Änderungen betroffene Multi Summary Nodes aktualisiert und falls nötig von der betroffenen Rolle gelöst werden.

Nicht auflösbare Konflikte

Jeder Rollentyp muss eine minimale Anzahl von Flows enthalten. Der Konfliktlösung ist es daher nicht möglich, beliebig viele Flows aus einer Rolle zu entfernen. Wenn beide der konkurrierenden Rollen nur die kleinstmögliche Anzahl an Flows besitzen, kann der Konflikt nicht gelöst werden. In diesem Fall wird wie bisher eine Warnmeldung auf der Kommandozeile ausgegeben.

2.4.3 Erweiterter Einstiegspunkt für das HAP4NfSen Plugin

Ausgangslage

Der Zugriff auf das HAP4Nfsen Plugin erfolgte in der ersten Version über Icons(in der folgenden Graphik grau hinterlegt) in einer Liste. Beim Klick auf eines dieser Icons, gelangt der Benutzer auf die Plugin Seite, wo das Graphlet für die gewählte IP angezeigt wird. Anhand der IP ist es für einen Benutzer, welcher an bestimmten Netzwerkaktivitäten interessiert ist, schwierig, einen geeigneten Einstiegspunkt für weitere Analysen zu finden.

```
** nfdump -M /data/nfsen/profiles-data/live/simulator  -T  -r 2011/06/15/nfcapd.201106151115 -n 10 -s ip/flows
nfdump filter:
any
Top 10 IP Addr ordered by flows:
Date first seen          Duration Proto           IP Addr    Flows(%)     Packets(%)       Bytes(%)         pps      bps   bpp
2007-04-28 01:20:54.105   541.090 any        34.188.96.43   15874(25.9)    20231( 6.1)    2.3 M( 1.8)       37    34062   113
2007-04-28 01:20:54.076   498.001 any       34.188.96.198   11667(19.0)    29940( 9.0)    2.9 M( 2.3)       60    46135    95
2007-04-28 01:20:54.119   541.140 any       34.188.96.107    8992(14.6)    28520( 8.6)    3.2 M( 2.5)       52    47278   112
2007-04-28 01:20:54.986   531.912 any        34.188.96.92    6678(10.9)    15611( 4.7)    1.2 M( 1.0)       29    18652    79
2007-04-28 01:20:54.081   519.227 any        34.188.96.88    2585( 4.2)     5238( 1.6)   572334( 0.5)       10     8818   109
2007-04-28 01:20:55.200   513.692 any       34.188.96.108    1874( 3.1)    18635( 5.6)    2.2 M( 1.8)       36    34740   119
2007-04-28 01:20:54.146   500.042 any      109.89.107.247    1689( 2.8)     1713( 0.5)   209387( 0.2)        3     3349   122
2007-04-28 01:20:55.161   518.773 any       34.188.96.197    1584( 2.6)    16566( 5.0)    2.7 M( 2.2)       31    42364   165
2007-04-28 01:20:54.149   455.482 any      212.253.161.249    1230( 2.0)    15100( 4.5)    2.5 M( 2.0)       33    44071   166
2007-04-28 01:20:55.289   477.060 any       34.188.97.203    1190( 1.9)     1412( 0.4)   119858( 0.1)        2     2009    84

Summary: total flows: 61397, total bytes: 125.7 M, total packets: 333395, avg bps: 1.9 M, avg pps: 616, avg bpp: 377
Time window: 2007-04-28 01:20:54 - 2007-04-28 01:29:55
Total flows processed: 61397, Blocks skipped: 0, Bytes read: 3192708
Sys: 0.070s flows/second: 877100.0   Wall: 0.139s flows/second: 441444.6
```

Figure 8: Bestehender Einstiegspunkt für das HAP4NfSen Plugin

Anpassungen

Der existierende Einstiegspunkt bleibt in dieser Form bestehen. Zusätzlich werden nun aber mit Hilfe der Visualisierungssoftware Afterglow die häufigsten Verbindungen zwischen Hosts in einer Graphik dargestellt.
Durch Klick auf einen der dargestellten Knoten, wird dieser als Host IP ausgewählt und der Benutzer wird zum HAP4NfSen Plugin weitergeleitet.

Lösung

Die erstellte Lösung betrifft mehrere Komponenten:
NfSen Details Seite
Um den neuen Einstiegspunkt in NfSen einzubinden, musste die NfSen Details Seite nochmals angepasst werden. Wie bereits zuvor, ist der neue Code so organisiert, das die Applikation auch ohne Plugin ohne Probleme läuft.
Die neuen Front End Erweiterungen sind zudem in einer separaten .php Datei ausgelagert, welche eingebunden wird, falls das Plugin aktiv ist. Dies führt zu einer weiteren Trennung von NfSen und dem HAP4NfSen Plugin.
Die Daten, welche für den Einstiegspunkt verwendet werden, lassen sich mit den bestehenden Filtermöglichkeiten, welche auf der Details Seite angeboten werden, beliebig einschränken.

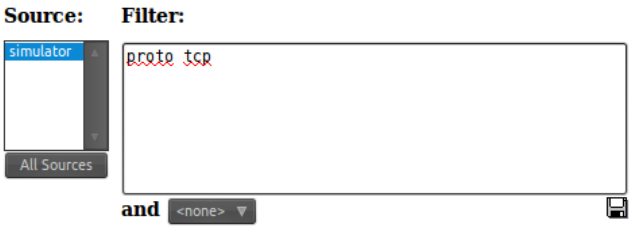


Figure 9: Filter auf der NfSen details Seite

Die Graphik für den Einstiegspunkt wird wie die HAP Graphlets vom HAP4NfSen Back End erstellt. Zum Aufruf dieses, wird die von NfSen bereitgestellte Infrastruktur verwendet.

Um zum neuen Einstiegspunkt zu gelangen, bietet die NfSen Details Seite nun einen zusätzlichen Button("graph"), welcher ein neues Fenster mit dem gewünschten Graphen öffnet.

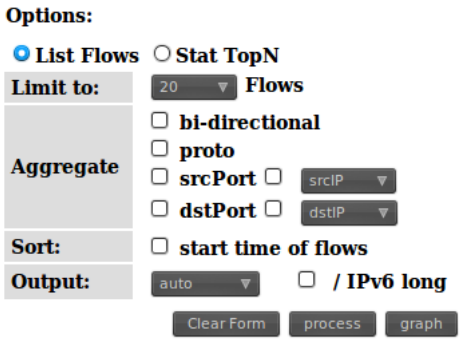


Figure 10: Button zum Aufruf des neuen HAP4NfSen Einstiegspunkts

HAP4NfSen Backend Plugin
Zur Generierung der Einstiegsgraphik wird die Perl Subroutine 'GetIpChooser' verwendet, welche alle Schritte der Generierung des Bilds übernimmt.
Bis zur fertigen Graphik sind folgende Schritte nötig:

```
NfDump => nfdump2csv => Afterglow => GraphViz => sed
          Parser                     neato       Anpassung von Links
```

NfDump	NfDump bietet Flags, mit denen Daten nach unterschiedlichen Kriterien gruppiert und sortiert werden können. Für den alternativen Einstiegspunkt werden die Daten nach Source und Destination IP gruppiert und nach der Auftrittshäufigkeit der so generierten Paare sortiert.
nfdump2csv	Die Daten werden per Pipe an den Parser, welcher separat beschrieben ist, weitergeleitet.
Afterglow	Nachdem die Daten im vom Afterglow gewünschten Format sind, werden sie wiederum per Pipe an Afterglow weitergeleitet, welches einen Graphen im .dot Format erstellt. Die Steuerung von Afterglow erfolgt sowohl über Kommandozeilen Parameter, sowie eine .properties Datei, die Anweisungen zur Formatierung und der Erstellung von Links enthält.

GraphViz	Zur Umwandlung der .dot Graphenbeschreibung in eine SVG Vektorgraphik wird das GraphViz Kommandozeilentool neato verwendet. Die von Afterglow erstellte Graphenbeschreibung wird dabei wieder in einer Pipe übergeben. Wie alle GraphViz Programme, bietet neato eine grosse Anzahl von Parametern, mit welchen sich der generierte Graph beeinflussen lässt. Einige davon werden beim Aufruf gesetzt, um den Graphen lesbarer zu machen und die Generierungszeit in Grenzen zu halten.
sed	Das Unixtool sed wird schliesslich verwendet, um die Links der Knoten zu bearbeiten. Dabei sind jeweils zwei Anpassungen nötig. Die erste ist das 'target' Attribut, welches auf '_parent' gesetzt werden muss, damit der korrekte Teil der HTML Seite mit dem neu generierten HAP Graphlet ersetzt wird. Die zweite Anpassung betrifft die Plugin Id des HAP4NfSen Plugins. Diese muss um sub_tab Teil des Links angegeben werden, ist aber im voraus noch nicht bekannt.

Aus diesen Teilschritten ergibt sich das Kommando, welches aus dem Perl Back End mit Hilfe des 'system' Befehls ausgeführt wird. Folgendes Beispiel illustriert die Zusammenarbeit zwischen den einzelnen Komponenten:

```
/usr/bin/nfdump -M /data/nfsen/profiles-data/live/hsr-nb-vlan-down:hsr-nb-vlan -r
2010/12/23/nfcapd.201012231300 -f /tmp/HAP4NfSen/88088bb8fb6e20e4232a806288a8ca40.filter -o
line6 -A srcip,dstport -s record -n 75|
/data/nfsen/plugins/HAP4NfSen/afterglow/src/perl/parsers/nfdump2csv.pl|
/data/nfsen/plugins/HAP4NfSen/afterglow/src/perl/graph/afterglow.pl -p2 -t -a -c
/data/nfsen/plugins/HAP4NfSen/afterglow/src/perl/parsers/ip_chooser.properties|
neato -Tsvg -v -Gmaxiter=20000 -Nfontname=monospace -Nfontsize=8 -Elen=2 -Goverlap=stretch
-Gsep=0 -Gsplines=true -o /data/nfsen/plugins/c44c1d51b2223624184eac8693a17d4f.svg
```

Mit dem Kommandozeilenprogramm 'grep' wird die erstellte Graphik vor dem zurücksenden an das Front End auf Links durchsucht. Da jeder dargestellte Knoten seinen eigenen Link besitzt, kann daraus bestimmt werden, ob der Graph überhaupt Daten enthält.
Falls dies nicht der Fall ist, wird anstatt eines leeren Graphen dem Benutzer eine Meldung angezeigt.

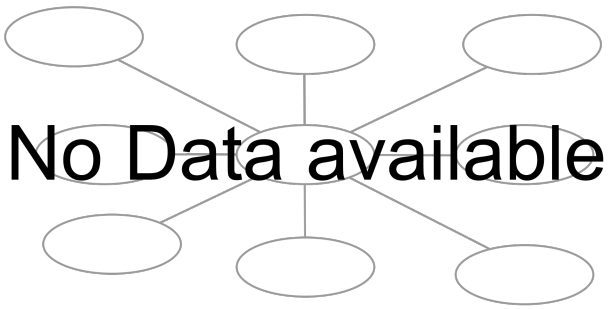


Figure 11: Einstiegspunkt, wenn keine Daten vorhanden sind

Afterglow NfSen Parser
Afterglow erwartet die Inputdaten in einem vorgegebenen CSV-ähnlichen Format.

Da NfDump dieses nicht direkt produziert, wurde ein Perl Skript erstellt, welches die Daten umwandelt.
Die Inputdaten müssen in einem Zwei- oder Dreispaltenformat vorliegen. Der HAP4NfSen Einstiegspunkt verwendet dabei die zweispaltige Variante.
Dabei enthält die erste Spalte jeweils die source IP und die zweite die destination IP.
Beispiel für Afterglow Input:

```
192.168.2.1,192.168.2.2
192.168.2.1,192.168.2.3
192.168.2.2,192.168.2.3
```

Aus diesem einfachen Beispiel ergibt sich folgender Graph:

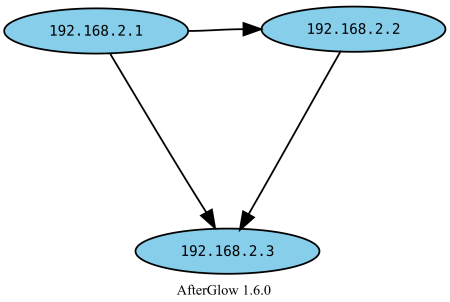


Figure 12: Afterglow Beispielgraph

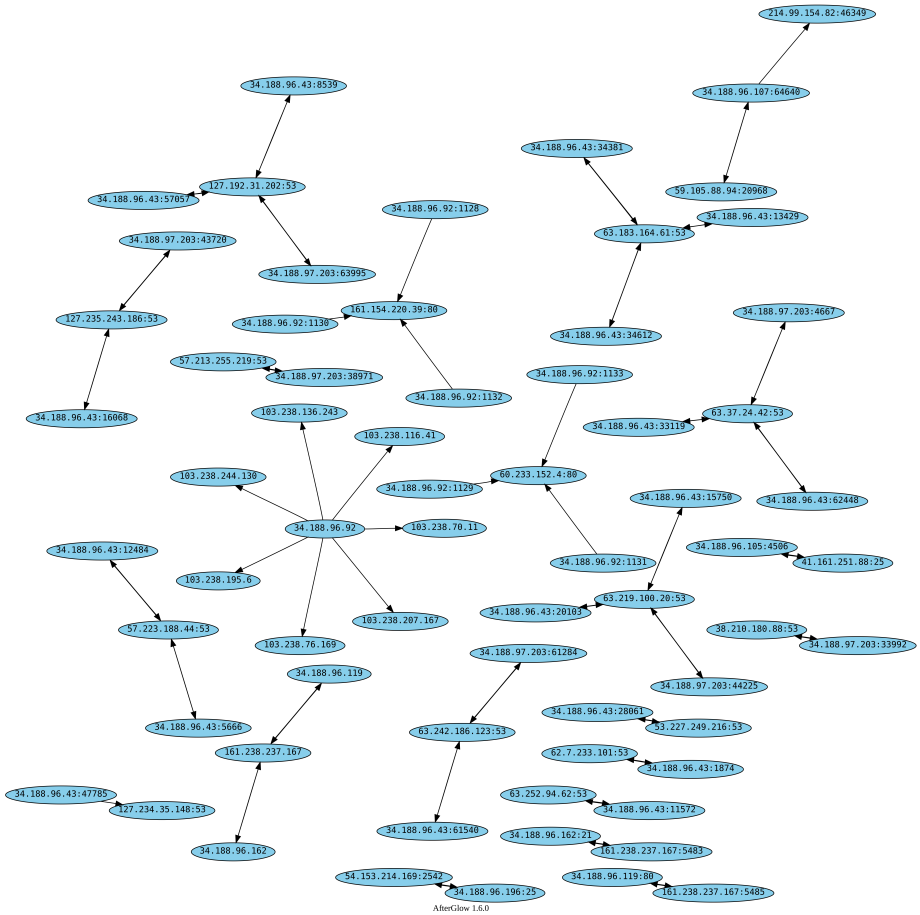


Figure 13: Neuer Einstiegspunkt für das HAP4NfSen Plugin

2.4.4 Klickbare Graphlets im HAPViewer

Ausgangslage

Bisher zeigte der HAP Viewer von GraphViz generierte Graphen im GIF Format an. Die Erstellung von neuen Graphlets, sowie die Anpassung der Summarisierungsoptionen wurde über Menuelemente und Buttons vorgenommen.

Anpassungen

Um die partielle Desummarisierung benutzerfreundlich zu gestalten, sollte es möglich sein, per Mausklick durch Graphlets zu navigieren.
Dies führt zu Anpassungen in der Visualisierung von Graphen, sowie in der Generierung der Graphendefinition, wo festgehalten werden muss, was beim Klick auf einen Knoten geschehen muss.

Lösung

Unsere Lösung wurde nach dem Kriterum, dass keinerlei Änderungen an der zumeist systemweit installierten Bibliotheken Graphviz erforderlich sein dürfen, gewählt. Der erste Ansatz, ein embedded-Browser einzubinden welcher dann ein bereits im HAP4NfSen verwendetes SVG verwendet, musste aufgrund zahlreicher Probleme aufgegeben werden. Auch gibt es keine brauchbaren Lösungen, um DOT-Dateien direkt auf ein GTK-Fenster zeichnen und zusätzlich noch das geklickte Element zu detektieren.

Vorschlag	**Vorteile**	**Nachteile**	**Bewertung**
GtkMozEmbed (SVG)	Bietet sein eigenes GTK Widget an	Extrem veraltet und umfangreich	unbrauchbar
WebKitGTK+ (SVG)	Bietet sein eigenes GTK Widget an	Umfangreich	unbrauchbar
Webkit (SVG)	Bietet sein eigenes GTK Widget an	Umfangreich	unbrauchbar
KGraphViewer	Widget zum Einbinden vorhanden	Basiert auf QT/KDE	unbrauchbar
Graphviz DotEdit	Kann nicht eingebunden, evt. aber Code abgeleitet werden	Verbuggt, unfertig	unbrauchbar
Graphviz dot -Tgtk	Kann nicht eingebunden, evt. aber Code abgeleitet werden	Verbuggt, unfertig, erfordert Main-Loop der Applikation	unbrauchbar
Selbst geschriebene Visualisierung	Sehr flexibel und schlank	Viel Arbeit	alternativlos

Trotz umfangreicher Suche konnte also keine out-of-the-box Lösung gefunden werden. Daher schrieben wir eine eigene, mehrstufige Lösung: Zuerst wird das vom HAPviewer generierte DOT-File mittels Graphviz in das XDOT Format umgewandelt. Dieser nun bereits gelayoutete Graph wird anschliessend mittels der BGL, der Boost Graph Library, wieder in eine eigens dafür geschriebene Klasse eingelesen. Anschliessed kann das GUI auf jedes Element einzeln zugreifen und mittels Cairo auf die GTK DrawingArea zeichnen. Auch ist es möglich, die Klicks eines Users wiederum dem dahinterliegenden Element zuzuordnen und ensprechend darauf zu reagieren.
Die Geschwindigkeit der Lösung ist bisher noch nicht optimiert. Es können aber problemlos mehrere tausend Kanten dargestellt werden, was für den Anwendungszweck des HAPviewers ausreichend ist.

2.4.5 Erweiterte klickbare Graphlets im HAP4NfSen Plugin

Ausgangslage

Durch Klicken auf Knoten des Graphlets mit der linken Maustaste war es bisher möglich, Drill Downs durchzuführen. Dies funktionierte, da das Graphlet(eine SVG Datei) die entsprechenden Links enthielt.
Mit der zusätzlichen Funktionalität der partiellen Desummarisierung muss ein anderer Weg gefunden werden, um mehrere Varianten des Drilldowns zu erlauben.

Anpassungen

Knoten des Graphlets sollen unterscheiden können, mit welcher Maustaste auf einen Knoten geklickt wurde. Bei Klicks mit der linken Taste soll der bereits aus der bisherigen Version bekannte Drill Down ausgeführt werden. Beim Klick auf die rechte Maustaste soll ein Knoten partiell desummarisiert werden.

Lösung

Um die Maustasten unterscheiden zu können, wurde eine JavaScript Library (drilldown.js) erstellt, welche die bestehenden Links mit Klickhandlern ausstattet, welche das Standardverhalten überschreiben.
Die Links innerhalb der SVG Datei werden vom Script gefunden, indem es durch den Baum der SVG Elemente iteriert und nach Elementen mit `tagName == 'a'` sucht.
Die Erkennung des gedrückten Buttons ist browserabhängig und wird daher auf `LEFT`, `MIDDLE`(momentan nicht gebraucht) und `RIGHT` abstrahiert. Anhand des gedrückten Buttons wird danach die Funktion aufgerufen, welche aus dem bestehenden Link die für den Drill Down benötigte URL generiert.

```
// Original Link kann sowohl nodeid als auch desum Parameter enthalten.
// Beispiel:
// nfsen.php?tab=5?sub_tab=1&nodeid=k5_2814748089384990&desum=16777206

/**
 * generates & returns the default drill-down url
 */
function getDefaultDrilldownLink(link) {
        //remove role id information from link
        return (link.getAttribute('xlink:href').replace(/&desum=\d+/, ''));
}

/**
 * generates and returns the drill-down link for part. desummarization
 */
function getPartialDesummarizationDrilldownLink(link) {
        var param = link.getAttribute('xlink:href').match(/&desum=\d+/);
        if (param) {
                // remove node id information from link
                return link.getAttribute('xlink:href').
                        replace(/&nodeid=.[kK_\d]+/, '');
        }
        // no role number available -> use standard drill-down for both clicks
```

```
        return getDefaultDrilldownLink(link);
}
```

Per JavaScript wird der Benutzer danach an die generierte URL weitergeleitet, wo das nächste Graphlet generiert wird.

2.4.6 Argus Daten Importmöglichkeit

Es gilt zu beachten, dass es mehrere Tools namens Argus gibt und die Aufgabenstellung eines[6] davon erwähnt. Abklärungen mit dem Betreuer der Arbeit ergaben, das eigentlich Unterstützung für Netflow Dateien im Argus[4] Format eingebaut werden soll. In allen folgenden Abschnitten ist daher, wenn nicht ausdrücklich erwähnt, Argus[4] gemeint.

Ausgangslage

Bisher wird der Import von Argus Daten vom HAP Viewer nicht unterstützt.

Anpassungen

Durch vorgängige Refactorings hat sich das Hinzufügen weiterer Dateiformate deutlich vereinfacht. Argus selbst ist sehr umfangreich und lässt sich nicht einfach in HAPviewer integrieren. Daher soll eine Möglichkeit gefunden werden, Argus Daten einzulesen, ohne Argus selbst in den Code des HAPviewer aufzunehmen.

Lösung

Die Argus client tools bieten verschiedene Kommandozeilenapplikationen an, mit welchen die gesammelten Argus Daten ausgewertet werden können. Zu diesen gehört `ra`, mit der Argus Flow Daten formatiert auf der Kommandozeile ausgeben werden können.

```
user@host:~$ ra -N 3 -r ./argus_file.log -s startime dur proto saddr sport dir daddr dport - ip
09-21-10 09:41:54.664961   5.284616  tcp   152.96.234.160.47338    ->   217.26.48.124.imap2
09-21-10 09:41:54.747873   5.251401  tcp   152.96.234.160.44260    ->   217.26.49.199.imaps
09-21-10 09:41:54.750597   0.010102  udp   152.96.234.160.35874   <->   152.96.20.10.domain
```

HAPviewer wurde um einen einfachen Parser erweitert, welcher den Output von `ra` in Flows vom Typ `cflow_t` umwandelt.
Nachdem sichergestellt wurde, das `ra` auf dem System installiert ist und die spezifizierte Datei eine gültige Argus Datei ist, wird `ra` in einem eigenen Prozess mit den gewünschten Parametern gestartet und der Output zeilenweise eingelesen. Jede Zeile wird durch einen regulären Ausdruck in die einzelnen Spalten aufgeteilt, welche darauf konvertiert, auf interne Darstellungen gemappt und in einen struct vom Typ `cflow_t` kopiert werden.
Bei diesen Umwandlungen ergeben sich einige Schwierigkeiten:

Protokolle

Argus versucht, Flows mit Protokollen wie UDP Protokollen auf höheren Ebenen, wie etwa RTP oder RTCP zuzuordnen, welche auch in der Ausgabe anstatt des eigentlichen Protokolls erscheinen.
Um zu den gewohnten HAP Graphlets zu gelangen, müssen diese Protokolle (Strings) auf das darunterliegende Protokoll gemappt werden. Falls ein auftretendes Protokoll unbekannt ist, so wird dies ausgegeben und der Import der Datei abgebrochen.

Der Parser kann einfach um weitere Protokolle erweitert werden, indem der Funktion `GFilter_argus:: proto_string_to_proto_num` weitere Mappings hinzugefügt werden

Flow Richtungen und Typen
Argus gibt zu jedem Flow eine Richtung eine Richtungsinformation mit. Diese stimmen nicht genau mit den im HAPViewer definierten `flow_type_t` überein. Bei der Interpretation des `ra` outputs wird daher der am besten passende Wert gewählt. Falls die Wahl nicht ganz eindeutig ist, wird eine Warnung ausgegeben und der Import fortgesetzt. Bei nicht erkannten Flowrichtingen, wird der Import direkt abgebrochen.
Mehr Informationen zu den Bedeutung einzelner Richtungsinformationen ist im Code (`gfilter_argus.cpp`) zu finden.
Der Flowtyp, wie er im HAPviewer für PCAP Dateien berechnet wird, stimmt nicht in allen Fällen mit den von Argus erkannten Flowtypen überein.

IPv6
Der Import von IPv6 Argus Dateien wird in dieser Version des HAPviewers noch nicht unterstützt. Grund dafür ist die Ausgabe von `ra`, welche als Protokoll bei IPv6 Records `ipv6` anzeigt. Es wurde keine Möglichkeit gefunden, auf das verwendete Transport Layer Protokoll, welches für die Generierung von HAP Graphlets benötigt wird, zuzugreifen.

2.4.7 NfSen Interfaces

NfSen bietet Plugins einen Mechanismus zur Kommunikation zwischen Front- und Back End, welche auch vom HAP4NfSen Plugin verwendet wird. Im Laufe dieser Arbeit mussten die bestehenden Interfaces (Dokumentiert im Bericht der Semesterarbeit[1]) erweitert werden. Im folgenden werden die Erweiterungen beschrieben:

generateDotFile

Die Änderungen in der Generierung der .dot Dateien mit Hilfe der HAP Library betreffen nur die input Parameter.

Name	Typ	Erlaubte Werte	Beschreibung
hap4nfsen_ desummarized_ role_ list	String	Ein leerer String oder durch `","` getrennte positive Ganzzahlen.	Dieser Parameter enthält die Information, welche Rollennummern partiell desummarisiert werden müssen. Der Inhalt des Strings wird in unveränderter Form an das Modul "NfDump2Dot", welches für die Anbindung von perl an C++ verantwortlich ist, weitergegeben.

GetConfig

Dieser Aufruf ermöglicht dem Front End, auf Werte der Konfiguration des Plugins zuzugreifen. Der Rückgabewert des Aufrufs wurde um zwei Werte erweitert.

Name	Typ	Erlaubte Werte	Beschreibung
hap4nfsen_ graph_ zoom_ limit	Integer	Eine positive Ganzzahl.	Enthält den Wert, welcher in "nfsen.conf" in der HAP4NfSen Plugin Konfiguration unter dem Wert "graph_ zoom_ limit" gespeichert ist. Default Wert ist 128. Der Wert beschreibt die Maximale Anzahl von Knoten welche ein Graph haben kann, bevor die Zoom Funktionalität deaktiviert wird.
hap4nfsen_ graph_ display_ limit	Integer	Eine positive Ganzzahl.	Enthält den Wert, welcher in "nfsen.conf" in der HAP4NfSen Plugin Konfiguration unter dem Wert "graph_ display_ limit" gespeichert ist. Default Wert ist 1024. Der Wert beschreibt die Maximale Anzahl von Knoten welche ein Graph haben kann, bevor er nur noch auf Bestätigung des Benutzers gerendert wird.

generateIpChooser

Dieser Back End Aufruf ist neu dazugekommen und ermöglicht die Generierung von SVG Graphiken für den alternativen Einstiegspunkt, wie er unter "Erweiterter Einstiegspunkt für das HAP4NfSen Plugin" beschrieben ist.

Input Parameter

Name	Typ	Erlaubte Werte	Beschreibung
hap4nfsen_ type	String	`shadow` oder `real`	Typ des ausgewählten Profils, kopiert aus der NfSen PHP session.
hap4nfsen_ plugin_ id	Integer	Postive Ganzzahl	Id des HAP4NfSen Plugins. Ist abhängig von anderen installierten Plugins und bestimmt den Sub Tab auf der NfSen plugins Seite.

hap4nfsen_ profile	String	Pfad zum ausgewählten Profil(z.B. `"./live(real)"`)	Kopiert aus der NfSen Session.
hap4nfsen_ srcselector	String	Namen von in NfSen vorhandenen Datenquellen, getrennt durch `":"`.	Namen der Profile, aus denen Daten gelesen werden soll. Kopiert aus der NfSen session.
hap4nfsen_ args	String	Gültige Argumente zum Aufruf von NfDump	Argumente, welche auf der NfSen details Seite zum NfDump Aufruf verwendet wurden. Kopiert aus der NfSen session.
hap4nfsen_ filter	String	Gültige NfDump Filter	Ein Filter in der NfDump Syntax, kopiert aus der NfSen session.
hap4nfsen_ and_ filter	String	Leerer String oder Name eines vorhandenen NfSen default Filter	NfSen erlaubt es, häufig verwendete Filter abzuspeichern und zu referenzieren. Hierbei handelt es sich um den Namen eines solchen Filters.
hap4nfsen_ graph_ type	String	`flow` oder `stat`	Art der Auswertung, für die die Graphik generiert werden soll. Bei `stat` werden summarisierte Flows verwendet, während `flow` mit einzelnen Flows arbeitet.
hap4nfsen_ record_ count	Integer	Positive Ganzzahl	Gewünschte grösse des NfDump outputs. Dieser Wert kann von den in der plugin Konfiguration gesetzten Werten eingeschränkt werden.

Output Parameter

Name	Typ	Erlaubte Werte	Beschreibung
hap4nfsen_ ipchooser_ pic	String	Name einer SVG Datei im NfSen Verzeichnis für generierte Bilddateien.	Enthält den Namen der generierten Datei. Mit Hilfe der vom NfSen Framework bereitgestellten pic.php kann die Datei gefunden und dargestellt werden.

2.4.8 Verbessertes Absturzverhalten

Auf vielen System ist das Generieren eines Core-Dumps deaktiviert. Es ist daher zumeist unmöglich, unerwartete, zufällig entdeckte Fehler welche zu einem Absturz des HAPviewers führen, nachträglich zu analysieren. Insbesondere im Falle von Race Conditions ist dies besonders ärgerlich.

Ausgangslage

Bisher stürzte der HAPviewer bei einem Fehler kommentarlos ab. Allenfalls konnte dank vorhergehenden Debug-Ausgaben auf den Fehler geschlossen werden.

Anpassungen

Es soll ein eigener Signalhandler für die gebräuchlichsten Signale (SIGABRT, SIGILL, SIGSEGV) installiert sein. Dieser soll beim Empfang eines solches Signals moglich hilfreiche Informationen ausgeben.

Lösung

Noch bevor der Main-Loop von GTK gestartet wird, wird nun unter Linux ein Signal-Handler für obige Signale installiert. Damit wird bei fehlerhaften Speicherzugriffe, fehlgeschlagenen Assertions und illegalen Instruktionen der aktuelle Stacktrace ausgegeben und anschliessend das Programm beendet. Im Falle des ebenfalls registrierten Signals SIGUSR1 wird lediglich der Stacktrace ausgegeben.
Solange für das Programm die Debugsymbole nicht gestrippt, bzw. überhaupt erst erstellt wurden, kann damit beispielsweise mittels addr2line die Codezeile, an welcher das Programm abgestürzt ist, ermittelt werden.

2.5 Benutzeranleitung

Die im HAP4NfSen Plugin enthaltene Hilfefunktion wurde um Beschreibungen der zusätzlichen Features erweitert. Daneben wird für den erweiterten Einstiegspunkt eine Hilfefunktion, welche vergleichbar mit der vom HAP4Nfsen Plugin ist, erstellt.
Eine eigenständige Benutzeranleitung wurde nicht erstellt.

2.6 Schlussfolgerungen

2.6.1 Zusammenfassung

Alle für diese Arbeit festgelegten Mussziele konnten erreicht werden. Darüber hinaus konnten die meisten der optionalen Ziele erreicht werden und eigene Ideen umgesetzt werden. Der HAPviewer wurde um viele Fearures erweitert, trotzdem gibt es aber noch genügend zu tun, um die eine oder andere Arbeit zu vergeben.

2.6.2 Mögliche Erweiterungen

Bekannte Erweiterungen

Im Laufe der Studienarbeit "Integration des HAPviewers in das NfSen Framework" [1] wurde bereits eine Liste mit möglichen Erweiterungen erstellt. Nicht alle der erwähnten Punkte konnen im Laufe dieser Arbeit implementiert werden. Alle übriggebliebenen Punkte sind daher mögliche Erweiterungen.

Zukünftige Erweiterungen

In den folgenden Abschnitten befinden sich Vorschläge für Erweiterungen, welche während dieses Projekts dazugekommen sind.

Anpassung des HPG Formats

Durch die Unterstützung von IPv6 Adressen sowie anderen Erweiterungen ist das HPG Format um den Faktor vier gewachsen. Ein grosser Teil dieses Platzes wird momentan nicht genutzt.
Mit eine Überarbeitung des Formats könnte dieser Platz eingespart werden. Eine Möglichkeit dazu ist zum Beispiel das Erstellen einer Liste aller verwendeten IP Adressen, welche nur einmal abgespeichert ist und in den Kanten referenziert werden kann.

Zusätzliche Erweiterungen für den Einstiegspunkt

Die aktuelle Version des Einstiegspunkts erlaubt es, Beziehungen zwischen IPs und Ports zu visualisieren. Die Anzahl der darstellbaren Knoten und Kanten ist allerdings stark beschränkt, da der Graph schnell unübersichtlich wird und die Generierung zu viel Rechenzeit beansprucht.
In einer erweiterten Version könnte daher, wie von Peter Haag vorgeschlagen, eine Drilldown Funktion, ähnlich der des HAP4NfSen Plugins, eingebaut werden, bei der zuerst Subnetze visualisert werden. Durch Klicks auf einzelne Knoten müssten dazu im Hintergrund NfDump Filter generiert und die Aggregationsstufe verringert werden.
Am Ende eines Drilldowns würde der Benutzer wie es bereits jetzt möglich ist, auf eine einzene IP Adresse klicken und zum passenden HAP Graphlet gelangen. Um das ganze benutzerfreundlicher zu gestallten, sollten wiederum Zoom & Pan für die SVG Graphik und eine Undo Funktion für Benutzerinteraktionen integriert werden.

Argus IPv6 Unterstützung

Die aktuelle Version des Plugins kann nur mit IPv4 Argus Daten umgehen. Um die ganzen Möglichkeiten des HAPviewer zu nutzen, sollte in einer zukünftigen Version auch der Import von IPv6 Argus Daten unterstützt werden.

Suchen nach Memory Leaks

Im Laufe dieser Arbeit konnten mit Hilfe von Valgrind Memory Leaks und andere Probleme gefunden werden. Aus Zeitgründen konnte nicht allen gefundenen Warnungen nachgegangen werden. Viele der Warnungen stammen aus verwendeten Bibliotheken (Boost, GTK, ..) und sind vermutlich keine

tatsächlichen Leaks. Trotzdem sollte in einer späteren Version der Applikation allen Warnungen nachgegangen werden.

Refactoring

Zwar wurden einige Punkte im Rahmen dieser Arbeit verbessert, es bleibt allerdings immer noch sehr viel Spielraum für Verbesserungen. Es wäre wünschenswert, die Programmlogik weiter von der GUI zu entkoppeln und, soweit es die Performance zulässt, weniger Bit-"Shifterei" zu betreiben.

Fehlerbehandlung konsequent vereinheitlichen

Es wäre sehr hilfreich, wenn im gesamten Code die Fehlerbehandlung durchgehend gleich gestaltet wäre. Zwar wurden im Rahmen dieser Arbeit einige Anstrengungen in diese Richtung unternommen, doch wäre hier noch mehr Arbeit nötig. Da in C++ keinen Mechanismus wie das Java-throws() kennt, ist dabei darauf zu achten, dass die Doxygen-Dokumentation konsequent aktualisiert wird.

Rollenbewertung vereinheitlichen

Derzeit muss für jeden Rollentyp eine Klasse existieren. Es wäre eine Überlegung wert, ob hier eine auf Templates basierende Lösung grosse Vereinfachungen erlauben würden.

Handling der temporären Dateien verbessern

Derzeit legt der HAPviewer zur Laufzeit Dateien im Arbeitsverzeichnis an, welche nicht mehr gelöscht werden. Es wäre wünschenswert, dass diese Dateien an einer geeigneteren Stelle abgelegt werden und nach Gebrauch wieder gelöscht werden.

Part II
Berichtsanhang

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

INFORMATIK

Aufgabenstellung zur Bachelorarbeit FS 2011

„HAPviewer V 2.0“

Gruppe: Sebastian Hügli / Reto Schneider

Ausgangssituation

HAPviewer (host application profile viewer) ist ein Open-Source Tool, das an der ETH Zürich für die verhaltensbasierte Analyse des Netzwerkverkehrs einzelner Rechner entwickelt wurde. Es benutzt eine graphenbasierte Darstellung, die sich an das Berkeley Socketmodell anlehnt, sowie Summarisierungs- und Filtertechniken, die eine kompakte Darstellung von hunderten oder gar tausenden von Netzwerkverbindungen ermöglicht [1].
Im Rahmen einer Studienarbeit wurde zudem ein HAPviewer Plug-In für das NfSen Network Monitoring Framework entwickelt [2, 3], dass den Einsatzbereich erweitert. Auf Grund zunehmender Erfahrungen beim Einsatz wurden verschiedene Bereiche identifiziert, in denen HAPviewer lohnend weiter entwickelt werden kann.

Aufgabe

Im Rahmen dieser Arbeit soll das Stand-Alone Tool HAPviewer und seine Bibliotheks-Variante [4] um wesentliche Eigenschaften erweitert, sowie der Code auf eine professionellere Basis gestellt werden.
Im Einzelnen betrifft dies folgende Muss- und Wunschziele:

1. Verbesserte Code-Basis: Refactoring des C++ Codes wo sinnvoll und Ersatz des Makefile durch ein flexibleres Build-System. Verschlankung der Abhängigkeiten von externen Programmbibliotheken (insbes. der Library Version).
2. Support für IPv6.
3. Partielle De-Summarisierung.
4. Verbesserung der Rollendetektion bzw. Konfliktlösung durch Einführung und Berücksichtigung der Remote-Host-Perspektive.

Bei genügend Zeit können noch folgende optionalen Ziele angegangen werden:

5. Flexibilisierung des Build durch Support weiterer Unix-Systeme, wie z.B. Free BSD, Open BSD, Open Solaris.
6. Importmöglichkeit für Argus-Daten [5].
7. Klickbare Graphlets für Drill-Down bzw. schnellen Wechsel der Host-Perspektive.

Berichtsgestaltung

Der Bericht ist gemäss [6] zu gestalten.

Termine

21. Feb. 2011	Arbeitsbeginn
17. Juni 2011	Abgabe des Berichts an den Betreuer (bis 17:00) sowie „HSR Forum“ mit Vorträgen und Präsentationen der Bachelor- und Diplomarbeiten, 13 bis 18 Uhr
8.-27. Aug. 2011	Mündliche BA-Prüfung (genauer Termin nach Vereinbarung)

Weitere Termine siehe Terminangaben auf dem HSR-Web (intern).

Betreuung

Betreuer: Prof. Eduard Glatz, Email: eglatz@hsr.ch
Während der Durchführung der Arbeit findet nach Möglichkeit regelmässig jede Woche eine Besprechung mit dem Betreuer statt. Dazu werden entsprechende Termine bei Arbeitsbeginn festgelegt.

Referenzen

[1] Eduard Glatz, „Visualizing Host Traffic through Graphs“, 7th International Symposium on Visualization for Cyber Security (VizSec), Ottawa, Ontario, Canada, Sep. 2011.

[2] Sebastian Hügli / Reto Schneider, „Integration des HAPviewers in das NfSen Framework“, Studienarbeit HS 2010, HSR- Hochschule für Technik, Rapperswil

[3] Open Source Projekt Web-Page auf SourceForge: „NfSen - Netflow Sensor“, erreichbar unter http://nfsen.sourceforge.net/

[4] Open Source Projekt Web-Page auf SourceForge: „HAPviewer - Host Application Profile Viewer“, erreichbar unter http://hapviewer.sourceforge.net/

[5] Argus - System and network Monitoring Software, Open Source Projekt, erreichbar unter http://argus.tcp4me.com/

[6] Eduard Glatz, „Vorgaben zur Berichtserstellung“, Ausgabe des 23. September 2010

Rapperswil, den 17. Feb. 2011

Eduard Glatz

3 Projektplan

3.1 Meilensteine

Die hier beschriebene Projektplanung wurde bis und mit der dritten Woche wie hier beschrieben durchgeführt. Danach begannen wir mit der Umsetzung der IPv6 Erweiterungen, was den Projektplan komplett durcheinander brachte.
Die Änderungen nahmen mehrere Wochen in Anspruch und erlaubten es nicht, gleichzeitig an anderen Bereichen des HAPviewer zu arbeiten. Später ergaben sich zudem weitere Änderungen in der Planung. Details zu den Abweichungen und den Ursachen sind im Abschnitt "Auswertung der Arbeitspakete" aufgelistet.

Nummer	**Name**	**Woche**	**Beschreibung**
0	Kickoff	Woche 1	Start der BA
1	Projektplanung abgeschlossen	Woche 2	Der provisorische Projektplan mit Paketen, einer Aufwands- sowie Risikoabschätzung wurde erstellt.
2	Automatisches Buildsystem eingerichtet	Woche 3	Der Code des HAPviewer wurde in einen Zustand gebracht, wo es möglich ist, ihn mit Hilfe von CMake automatisch zu builden. Das Buildsystem wurde auf dem Entwicklungssystem aufgesetzt und automatisiert. Dazu wurden ungewollte Anhängigkeiten auf andere Software entfernt.
3	Prototyp IPv6	Woche 5	Ein Prototyp wurde erstellt, welcher es erlaubt IPv6 Daten in den HAPviewer einzulesen und daraus ein Graphlet zu generieren.
4	Prototyp partielle desummarisierung	Woche 7	Die Schnittstelle der HAP library wurde um einen oder mehrere Parameter erweitert, welche es ermöglichen, einzelne Teile eines Graphlets zu desummarisieren. Der Code des HAPviewer wurde soweit erweitert, das er die Parameter in das gewünschte Graphlet umsetzt.

5	Prototyp alternativer Einstiegspunkt	Woche 8	NfSen wurde um einen neuen Einstiegspunkt für das HAP4NfSen Plugin erweitert. Der zusätzliche Einstiegspunkt erlaubt Benutzern die graphische Auswahl einer IP Adresse, welche sie mit Hilfe des HAP4NfSen Plugins visualisieren möchten.
6	Prototyp verbesserte Rollendetektion und Konfliktlösung	Woche 9	Die Funktionalität des HAPviewer, welche entscheidet, zu welcher Rolle ein Flow gehört wurde überarbeitet und erweitert. Flows werden von beiden Seiten her bewertet und eventuelle Rollenkonflikte wurden behoben.
7	Integration der Prototypen in NfSen	Woche 12	Das HAP4NfSen Plugin wurde erweitert, so dass es möglich ist, die zuvor erstellten HAP Erweiterungen aus der NfSen Umgebung zu nutzen.
8	Argus Daten Import	Woche 15	Der HAPviewer wurde so erweitert, das er in der Lage ist, Daten im Format, welches von Argus verwendet wird, zu lesen.
9	BA Abgabe	Woche 16	Abgabe der Arbeit

3.2 Interationen

Iteration	**Inhalt**	**Start**	**Ende**
Inception	Projekt-/Zeitplanung, Startmeeting	Woche 1	Woche 2
Elaboration	Buildsystem Einrichten, Abklärungen zur IPv6 Unterstützung	Woche 3	Woche 4
Elaboration 2	IPv6 Prototyp, Abklärungen zur partiellen Desummarisierung	Woche 5	Woche 6
Construction	IPv6 Prototyp erweitern, Prototypen: partiellen Desummarisierung, Alternativer Einstiegspunkt & Rollenkonfliktlösung	Woche 7	Woche 9
Construction 2	Erweiterung von Prototypen, Integration der partiellen Desummarisierung in Integration in das NfSen Plugin	Woche 10	Woche 12
Construction 3	Weiterarbeit an Prototypen	Woche 13	Woche 14
Transition	Fertigstellung der Funktionalität der Prototypen, Import von Argus Daten, Fertigstellung aller bisherigen Arbeiten	Woche 15	Woche 16

3.3 Arbeitspakete

In der folgenden Tabelle sind alle Arbeitspakete des Projekts aufgelistet.

Paket	**Beschreibung**	**Geplanter Aufwand Reto Schneider**	**Geplanter Aufwand Sebastian Hügli**
Refactorings	Umstrukturierung des Codes, wo dies für die Ziele dieser Arbeit und spätere Erweiterungen nützlich ist. Entfernung von nicht verwendeten Code und Support von nicht mehr benötigten Versionen des HPG Graphletformats. Pflegen und Vervollständigung der Doxygen Source Code Dokumentation.	120h	120h
Buildsystem	Einrichten und konfigurieren von CMake. Erstellen einer Konfiguration, welche den Buildprozess kontrolliert.	20h	-
IPv6: HAPviewer support	Erweiterungen des HAPviewer, so das es mit Host Adressen im IPv6 Format umgehen kann. Erweiterung des HPG Graphenformats zur Unterstützung von IPv6 Adressen.	50h	60h
IPv6: Datei- import	Erweiterung des bestehenden Imports, um IPv6 Daten in den Formaten IPFIX, pcap, CFlow und NfDump lesen zu können. Export von Daten im neuen CFlow Format mit Unterstützung von IPv6 Addressen. Unterscheidung zwischen unterschiedlichen CFlow Format Versionen beim Lesen aus einer Datei.	20h	20h

Partielle Desummarisierung	Umbau der Graphenberechnung, um mit generischen Rollen umgehen zu können. Desummarisierung einzelner Partitionen. Generierung von Ids zur eindeutigen Identifikation von ausgewählten Rollen.	-	50h
Auflösung von Rollenkonflikten	Bewertung von Rollen unter Einbezug der Remote Perspektive. Auflösung der gefundenen Konflikte.	-	30h
Support für weitere Betriebssysteme	Testen der Funktionalität unter unterschiedlichen Unix Betriebssystemen. Anpassungen wo nötig, um den Support zu gewährleiten.	20h	-
Argus Datei- import	Einbauen einer möglichkeit zum Import von Daten im Argus Format.	20h	-
Erweiterter Einstiegspunkt für HAP4NfSen	Erstellung eines graphischen Einstiegspunkts für das HAP4NfSen Plugin. Verwendung von NfSen Daten zur Generierung eines Graphen mit Afterglow. Darstellung und Funktionalität festgelegt von Peter Haag.	-	30h
Reduktion von Abhängigkeiten	Reduktion der Abhängigkeiten der Library Version, damit nur noch wirklich verwendete Bibliotheken zum Builden benötigt werden.	10h	-

Klickbare HAPviewer Graphen	Bau einer Lösung, welche Klicks auf Knoten erkennen und darauf reagieren kann. Dies setzt die Änderung der bisher verwendeten Darstellung von Graphen voraus.	40h	-
Total	-	**300h**	**310h**

Nicht in den Arbeitspaketen enthalten sind Besprechungen (ca. 20 Stunden pro Person), sowie das Schreiben der Dokumentation (ca. 50 Stunden pro Person). Dazu kommt eine Reserve von 16 Stunden, was sich aus der Risikoabschätzung ergibt.

3.4 Projektrisikoabschätzung

Nachfolgend werden Risiken aufgelistet und bewertet. Nach 4 Projektwochen wurden die Risikopunkte im Team besprochen. Später wurden Risiken, die nicht mehr auftreten, können abgehakt und die verbleibenden Risiken wurden neu beurteilt.

Nr.	**Risiko**	**Vorbeugende Massnahmen und Workaround**	**max. in h**	**p**	**p*h**
R01	Tool zur Integration des neuen Einstiegspunkts unterstützt nicht alle Anforderungen	Frühzeitige und gründliche Abklärung von Anforderungen an, sowie Funktionsumfang des Tools. Eventuell Suche nach Alternativen.	20	0.2	4
R02	Nicht ausreichende Zeit zur Implementation der optionalen Ziele	Gründliche Abklärung des Aufwands bereits in einer frühen Phase des Projekts	-	0.2	-
R03	Performance Probleme oder Memory Leaks, welche im Rahmen der HAPviewer Erweiterungen entstehen	Unterstützende Tools wie etwa Valgrind einsetzen, um die Qualität der Software sicherzustellen. Beim Aufftreten von Problemen muss die Ursache gefunden und das Problem behoben werden.	40	0.1	4

R04	Libraries zum Import der verschieden Formate unterstützen IPv6 nicht	Früh abklären, eventuell Alternativen finden	40	0.2	8
Total					**16**

3.5 Arbeitsaufteilung

In den folgenden Abschnitten sind die Arbeiten aufgelistet, welche zum grössten Teil von einem der Projektteilnehmer ausgeführt wurden. Alles was hier nicht erwähnt ist, wurde gemeinsam erledigt.

3.5.1 Reto Schneider

- Einrichtung und Konfiguration des Buildsystems(CMake)
- Betreiben und Konfiguration der Buildhosts/Wiki/ML
- Refactoring: Reduzierung von Anhängigkeiten
- Refactoring: Struktur des Datei-Imports
- Zeichnen von Graphlets und empfangen von Klick-Eignissen im HAP Viewer
- Installation und Test der Applikation in unterschiedlichen Betriebssystemen

3.5.2 Sebastian Hügli

- Struktur des Berichtes(LaTeX)
- Erweiterter Einstiegspunkt des HAP4NfSen Plugins
- Partielle Desummarisierung
- Refactroing und Erweriterung der .dot Kommentare(benötigt für Filter des HAP4NfSen Plugins)
- Unterstützung von zwei unterschiedlichen Klick-Arten(HAP4NfSen), sowie PHP und perl Modifikationen benötigt zur partiellen Desummarisierung
- Auflösung von Rollenkonflikten

3.6 Auswertung der Arbeitspakete

Die folgende Tabelle bietet eine Zeitauswertung, organisiert nach Arbeitspaketen. Bei Paketen, wo ein grosser Unterchied zwischen dem geplanten und tatsächlichen Zeitaufwand besteht, wird der Grund dafür angegeben.

Paket	**Differenz Reto Schneider**	**Differenz Sebastian Hügli**	**Begründung**
Refactorings	-25h	-50h	Aufgrund von anderen Arbeiten, wie etwa dem Support für IPv6, musste der Umfang der Refactorings angepasst und aufs nötigste reduziert werden.
Buildsystem	+5h	-	
IPv6: HAPviewer support	+50h	+40h	Die Unterstützung des neuen IP Formats war viel aufwändiger als ursprünglich geplant. Ein Grund dafür war, dass in C++ kein einfacher Datentyp für 128 Bit Werte existiert und wir eine eigene Klasse dazu schreiben mussten. Daneben musste auch das HPG Graphenformat angepasst werden. Mit diesen Änderungen musste vieles umgebaut werden, fast jede Klasse war davon betroffen.
IPv6: Datei- import	+5h	+0h	
Partielle Desummarisierung	-	+20h	Die Implementierung war komplizierter als angenommen, da neben der eigentlichen Graphendarstellung auch das HPG Format angepasst werden musste, um Platz für eindeutige Keys (benötigt zum Erkennen von Desummarisierungen) zu schaffen.

Auflösung von Rollenkonflikten	-	+10h	
Support für weitere Betriebssysteme	-5h	-	
Argus Datei- import	-20h	+20h	Die Änderungen wurde aus Zeitgründen von Sebastian Hügli anstatt von Reto Schneider eingebaut.
Erweiterter Einstiegspunkt für HAP4NfSen	-	-5h	
Reduktion von Abhängigkeiten	+0h	-	
Klickbare HAPviewer Graphen	+30h	-	Die Implementation der Erkennung von Klicks war aufwändiger als geplant, da die Ursprünglich vorgesehene Variante (Zeichnen der Graphen mit GraphViz in HAPviewer) nicht möglich war und das Zeichnen und die Klickbarkeit der Graphen selbst implementiert werden musste.
Total	**+40h**	**+35h**	

4 Entscheidungsfindung

Wir führten wie schon bei der SA ein Arbeitswiki. Für Entscheidungen, für welche die Lösung nicht auf der Hand lagen, wurde jeweils eine Tabelle angelegt und die Ansätze gegeneinander abgewogen, bzw. einzelne Ansätze ausgearbeitet.

4.1 Unit-Testing Frameworks

Programm	**Positives**	**Negatives**	**Bewertung**
CTest	• In CMake integriert • Bereits in der SE2-Arbeit damit gearbeitet (Reto) • Lässt sich mit anderen Frameworks kombinieren	eher primitiv	Da in CMake integriert wird auf CTest gesetzt.
CUTE	• Eclipse-Plugin vorhanden • Bereits in der SE2-Arbeit damit gearbeitet (Reto) • Lediglich einige Headers zu includen • Gemäss eigener Aussage einfacher als bspw. CPPUnit		Wird mit CTest kombiniert eingesetzt
CppUnit	Umfangreich	Jede Klasse muss zwingend abgeleitet werden	ungeeignet
Boost Test Library	Boost ist bereits vorhanden	Gemäss Tutorial sehr komplex	brauchbar

4.2 Build Automation Software

Programm	**Positives**	**Negatives**	**Bewertung**
Hudson for C++/ CMake/CppUnit	Hudson ist ein tolles Projekt	“Bastelösung”	nicht geeignet
CDash	gut in CMake integriert	eher altbacken	bestens geeignet

4.3 IPv6

4.3.1 IPv6 Support

Problem	**Lösungsvorschlag**	**Positives**	**Negatives**	**Bewertung**
Datenfelder localIP und remoteIP sind zu klein für IPv6 Adressen	Durch eigene Klasse basierend auf `boost::array<unsigned char, 16>` ersetzen	• Erlaubt Speicherung von IPv6 Adressen • `boost::array` unterstützt STL Algorithmen • OOP Erlaubt die Erweiterung um hilfreiche Methoden und Operatoren, z.B. `operator==`	Erhöhter Speicherverbrauch	angenommen
Datenfelder localAS und remoteAS sind nur 16 bit lang, zukünftige AS werden aber 32 bit Nummern enthalten.	Bisher verwendeten Typ `uint16_t` durch `uint32_t` ersetzen.	• Zukunftssicher • Das bisherige Padding (32 bit) wird damit eingespart und sinnvoller verwendet	32 bit mehr Speicherverbrauch	angenommen

4.3.2 IPv6 only vs. mixed Style

Aktuell unterstützt der HAPviewer nur IPv4. In Zukunft soll auch IPv6 unterstützt werden. Es stellt sich nun die Frage, ob wir unterschiedliche Datenfelder definieren oder aber den gesamten Verkehr intern als IPv6 betrachten.

Kritikpunkt	**IPv6 only**	**mixed style**	**Bewertung**
Konvertierung IPv4<->IPv6	bei gewissen Operation muss eine Fallunterscheidung gemacht werden	IP Adressen müssen nie konvertiert werden	kein ernsthaftest Problem

Speicher	pro IPv4 Adresse werden etwa 96 Bits zusätzlich benötigt	niedrigerer Speicherverbrauch	Irrelevant, da genug RAM

Wie aus der Bewertung hervorgeht, wurde die "IPv6 only" Variante gewählt. Was es neben den erwähnten Punkten ebenfalls zu beachten gilt, ist die zunehmende Verbreitung von IPv6.
Je grösser der Anteil dieses Formates, desto weniger wirken sich die erwähnten Negativpunkte aus.

4.4 Refactoring

Programm	**Lösungsvorschlag**	**Positives**	**Negatives**	**Bewertung**
In der gesamten Software werden jeweils `struct cflow* flowlist, int flow_count` durchgereicht.	Ersetzen durch einen `vector<cflow_t>` der STL	• Flexibel • Sicherheit • Einfacher	• etwas grösserer Overhead	angenommen

4.5 Projektplanung

Programm	Positives	Negatives	Bewertung
GNOME Planner	HTML/PDF Export Möglichkeiten	Keine Übersicht über gearbeitete Stunden/Person	untauglich
KPlato	Umfangreicher als GNOME Planner	• Instabil?! • Iteratives Vorgehen schlecht planbar	untauglich
TaskJuggler	Ausführliche Auswertungen	• Trotz GUI muss zum Ändern des Plans der "Code" des Projektplans angepasst werden • keine Export-Möglichkeiten	untauglich
AgileTrack	Unterstützt iteratives Vorgehen	Kostet 39$ pro User	Wäre ein tolles Tool, möchte aber kein Geld dafür ausgeben (Reto)
acunote	Webbasierend und Multiusertauglich		nicht geeignet
ProjectCard	• Plugin für Eclipse • Kostenlos für Open Source Projekte		nicht geeignet

tinyPM	• Kostenlos für kleine Projekte (bis 5 User) • Scheint unsere Bedürfnisse abzudecken • Daten können relative einfach aus der Datenbank geholt werden	Nicht 100% sicher, ob die Auswertung für uns tauglich ist	• Einen Versuch wert, Testinstallation gemacht • Testlauf erfolgreich verlaufen, werden dieses Tool verwenden

Im Verlaufe des Projekts wurde uns mitgeteilt, dass eine derart detailierte Zeitauswertung wie es tinyPM ermöglicht, nicht erforderlich ist. Wir haben daher während des Projekts auf eine sehr viel einfachere "Zeiterfassung" mittels Papier und Stift umgeschwenkt.

4.6 Klickbares Bild im HAPviewer

Programm	**Positives**	**Negatives**	**Bewertung**
GtkMozEmbed (SVG)	Bietet sein eigenes GTK Widget an	Extrem veraltet und umfangreich	unbrauchbar
WebKitGTK+ (SVG)	Bietet sein eigenes GTK Widget an	extrem umfang- reich	unbrauchbar, zu gross
Webkit (SVG)	Bietet sein eigenes GTK Widget an	extrem umfang- reich	unbrauchbar, zu gross
KGraphViewer	Widget zum Einbinden vorhanden	Basiert auf QT/KDE	unbrauchbar, da falsche Technologie
Graphviz DotEdit	Kann nicht eingebunden, evt. aber Code abgeleitet werden	Verbuggt, unfertig & fehlerhaft	unbrauchbar
Graphviz dot -Tgtk	Kann nicht eingebunden, evt. aber Code abgeleitet werden	Verbuggt, unfertig, erfordert Main-Loop der Applikation	unbrauchbar

Selbst geschriebene Visualisierung	Sehr flexibel und schlank	Viel Arbeit	alternativlos

5 Testprotokolle

In den folgenden Abschnitten werden die durchgeführten Test erklärt und die Ergebnisse aufgelistet.

5.1 Unit Tests

CMake ist in der Lage, die erstellten Unit Tests zu kompilieren, auszuführen, die Ergebnisse zu interpretieren und an einen zentralen Server mit CDash zu schicken, welcher die Ergebnisse von verschiedenen Systemen sammeln und im Fehlerfall die Programmierer per Mail benachrichtigt.
Wir beschränkten unsere UnitTests auf Methoden, bei welchen uns in der Vergangenheit Bugs begegnet sind.
Folgende UnitTests-Dateien wurden erstellt:

Test	**Beschreibung**
cflow	Verschiedene Test zu den Klassen `cflow4` und `cflow6`
gfilter	Allgemeine Tests zum Import von Daten
gfilter_nfdump	Tests zum Import von Daten im NfDump Dateiformat
gfilter_pcap	Tests zum Import von Daten im pcap Format
gutil	Tests zu Funktionen, welche in verschiedenen Teilen des Codes verwendet werden
HashMapE	Tests zu Hash Map Keys
ipv6_addr	Tests zur Klasse, welche IPv6 und IPv4 Adressen repräsentiert

Der HAPviewer wurden auf folgenden Systemen zumindest täglich kompiliert:

- Ubuntu 8.04 x64
- FreeBSD 8.2 x64
- OpenBSD 4.8 x64
- Debian Lenny x86

Alle Tests sowie die Anwendung selbst kompilierten auf allen Testsystemen erfolgreich und ohne Warnungen. Zudem wurden alle erfolgreich ausgeführt.

5.2 System Tests

5.2.1 Graphengenerierung

In diesem Test werden Graphen mit der ursprünglichen Version des HAPviewer(Februar 2011 inklusive Bugfixes von E. Glatz) mit der neuesten Version der Applikation verglichen. Als Testdaten wurde demo.gz verwendet.

Alle Graphlets der Datei wurden für diesen Test verglichen. Es wurde für jede lokale IP von der demo.gz-Datei ein Graplet generiert und abgespeichert. Um der durch die geänderte Node-Namen geschuldete Neuanordnung der Graphlets entgegenzuwirken wurde von jedem Host je vier Graphlets erzeugt: nur mit ICMP/nur TCP/nur UDP/nur "other" Traffic. Das selbe wurde mit unserer Version des HAPviewers gemacht und anschliessend die Bilder verglichen. Bilder welche die gleiche Prüfsumme wie das vom ursprünglichen HAPviewer erzeugte Graphlet hatten, wurden als korrekt markiert. Bei Unterschieden wurden dem Tester drei Bilder angezeigt: Neuer Graph, alter Graph sowie die farblich hervorgehobene Differenz zwischen den Bildern.
Die neuen und alten Testbilder wurden im Source-Repository abgelegt und können in Zukunft als Referenzen dienen.
Folgende Tabelle enthält das Testprotokoll.

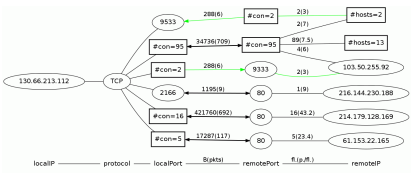
Graphlet lokale IP	**Unterschiede erwartet**	**Bemerkungen**	**Resultat**
113.60.199.251	Ja	Durch einen Bug in der alten Version des HAPviewer, bzw. der Tatsache, dass die Testdaten entgegen den Regeln nicht sortiert sind, ist diese IP zweimal mit jeweils der Hälfte der Flows vorhanden. Dies ist ein Fehler in der Datei. Die neue Version des HAPviewer hat ebenfalls Probleme im Umgang mit den fehlerhaften Daten. Zudem hat die Auflösung der vorhandenen Rollenkonflikte eine Auswirkung auf das Graphlet.	ok
128.253.15.2	Nein	Neu angeordnet	ok
128.253.38.180	Nein		ok
128.253.46.205	Nein		ok
128.253.89.88	Nein		ok
128.253.104.185	Nein	Neu angeordnet	ok
128.253.159.50	Nein	Neu angeordnet	ok

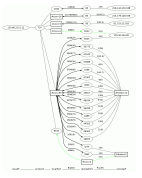
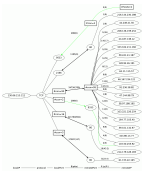
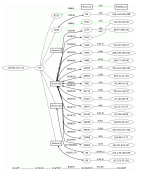
129.37.8.119	Nein	Neu angeordnet	ok
129.37.34.68	Nein	Neu angeordnet	ok
129.37.103.239	Nein	Neu angeordnet	ok
129.68.153.20	Nein	Neu angeordnet	ok
129.68.232.247	Nein		ok
129.68.246.212	Nein		ok
129.68.250.129	Nein	Neu angeordnet	ok
130.50.92.2	Nein	Neu angeordnet	ok
130.50.96.3	Nein	Neu angeordnet	ok
130.50.202.167	Nein	Neu angeordnet	ok
130.52.234.1	Nein	Neu angeordnet	ok
130.66.213.112	Nein	Neu angeordnet	ok
130.122.253.21	Nein	Neu angeordnet	ok
130.176.44.254	Nein	Neu angeordnet	ok
130.176.253.43	Nein	Neu angeordnet	ok
131.231.157.100	Nein		ok
131.231.222.204	Nein	Neu angeordnet	ok
153.110.244.19	Nein	Neu angeordnet	ok
160.2.68.150	Nein	Neu angeordnet	ok
160.66.56.87	Nein	Neu angeordnet	ok
160.66.103.44	Nein	Neu angeordnet	ok
160.66.196.204	Nein		ok

160.98.255.180	Nein		ok
192.31.108.255	Nein	Neu angeordnet	ok
193.113.250.216	Nein	Neu angeordnet	ok
193.229.80.123	Nein	Neu angeordnet	ok
193.229.234.177	Nein		ok
195.128.8.172	Nein	Neu angeordnet	ok
195.128.24.236	Nein	Neu angeordnet	ok
195.128.60.173	Nein		ok
195.128.163.79	Nein		ok
195.128.191.131	Nein		ok
195.128.246.46	Nein	Neu angeordnet	ok
195.128.246.79	Nein	Neu angeordnet	ok
195.128.254.26	Nein	Neu angeordnet	ok
195.128.255.25	Nein	Neu angeordnet	ok
195.128.255.186	Nein	Neu angeordnet	ok

5.2.2 Partielle Desummarisierung

In diesem Test wurde die partielle Desummarisierung inklusive Klickerkennung und Klicks auf remote IPs systematisch geprüft.
Dieser Test verwendet Daten aus der Datei demo.gz. Das Test Graphlet hat die lokale IP `130.66.213.122`, da alle verfügbaren Fälle anhand einer Peer to Peer Rolle dieses Graphlets abgedeckt werden können.
In diesem Fall wird anhand der Rolle mit der Nummer 162, welche 95 Flows enthält, getestet. Aus Übersichtsgründen werden nur TCP Daten, welche Bi-Flows enthalten angezeigt:

Desummarisierte Partitionen	**Graphlet**	**Resultat**
`o-o-[]-[]-[]`		ok

`o-o-o-[]-[]`		ok
`o-o-[]-o-[]`		ok
`o-o-[]-[]-o`		ok
`o-o-[]-o-o`		ok
`o-o-o-[]-o`		ok
`o-o-o-o-[]`		ok
`o-o-o-o-o`		ok

Die angezeigten Graphlets sind aus Platzgründen stark skaliert und sind nur dazu da, die Struktur des Graphlets zu erkennen.
Die Erkennung von Klicks auf Remote IPs wurde anhand des selben Graphlets getestet. Dabei wurde auf alle remote IPs geklickt und das generierte Graphlet kontrolliert.

5.2.3 IPv6 Unterstützung

Der Import und die Anzeige von Dateien mit IPv6 Daten wurde getestet, indem eine pcap Datei geladen und alle Graphlets daraus generiert wurden. Danach wurden die Daten ins Cflow Format exportiert und nochmals geladen. Die generierten Graphlets beider Formate wurden verglichen.

6 Build Anleitung

6.1 Abhängigkeiten

6.1.1 Abhängigkeiten der HAPlib

Zum kompilieren der HAPlib, welche für das HAP4NfSen Plugin benötigt wird, werden zumindest folgende Libraries und Tools vorausgesetzt:

- CMake
- Boost Regex

Eine Anleitung zur Installation kann in der Studienarbeit "Integration des HAPviewers in das NfSen Framework"[1] gefunden werden.

6.1.2 Abhängigkeiten des HAPviewer

Für die GUI Version des HAPviewer ergeben sich die gleichen Abhängigkeiten wie für die Bibliotheksversion sowie folgende zusätzliche:

- Graphviz
- Boost (filesystem, iosystem, graph)
- GTKmm

6.1.3 Zusätzliche Abhängigkeiten

Wenn zusätzliche Dateiformate unterstützt werden müssen, ergeben sich zusätzliche Abhängigkeiten. Die Unterstützung dieser Formate kann in der CMake Konfiguration aktiviert oder deaktiviert werden:

- HAPVIEWER_ENABLE_ARGUS: Argus Tools (zur Laufzeit)
- HAPVIEWER_ENABLE_CFLOW: boost_filesystem, boost_iostreams
- HAPVIEWER_ENABLE_IPFIX: glib
- HAPVIEWER_ENABLE_NFDUMP: keine
- HAPVIEWER_ENABLE_PCAP: pcap++/pcap

6.1.4 Abhängigkeiten des HAP4NfSen Plugins

Folgende Software muss zur Installation und zum Betrieb des Plugins auf dem Zielsystem installiert sein:

- NfSen(Laufzeit)
- NfDump(Laufzeit)
- HAPviewer Bibliothek(Kompilierzeit)
- SWIG(Kompilierzeit)
- sed(Laufzeit)
- Graphviz(Laufzeit)

6.2 Kompilierung

Das Erstellen der HAPlib und des HAPviewer erfolgt durch CMake, welche die zur Kompilierung nötigen Makefile automatisch erstellt.
Dazu muss ein Build Verzeichnis für die generierten Dateien erstellt werden. Zur Illustration des Vorgangs wird folgende Struktur verwendet(Verzeichnisnamen können auf dem Zielsystem abweichen):

```
tree -L 1
.
|-- build      # manuell erstelltes build Verzeichnis, anfangs noch leer
‘-- hapviewer  # Verzeichnis mit source code, z.B. ausgecheckt aus git
```

Danach können die Make Dateien wie folgt generiert werden:

```
cd build
cmake ../hapviewer
```

Danach befinden sich im Build Verzeichnis ein "Makefile", sowie weitere Verzeichnisse und Dateien.
Nun kann der Buildprozess konfiguriert werden. Dies ist mit Hilfe von "ccmake" möglich:

```
# wird immernoch im build Verzeichnis ausgeführt
ccmake .
```

Im folgenden GUI können Flags gesetzt und zahlreiche Werte angepasst werden. Einige Optione erscheinen allerdings erst im “advanced mode”. Im unteren Teil des GUIs wird jeweils beschrieben, welche Auswirkungen die aktuell ausgewählte Einstellung hat.

Figure 14: CCMake GUI

Nachdem die gewünschten Einstellungen gesetzt wurden, können diese durch drücken von `c` gefolgt von `g` übernommen werden.
Nun stehen je nach Konfiguration folgende make Targets zur Verfügung:

Make Target	**Beschreibung**
{default}	Erstellt alle zuvor konfigurierten Executables
HAPviewer	Kompiliert den HAPviewer
hapviz	Kompiliert die HAP-Library
mk_cflows	Kompiliert das mk_cflows-Tool
show_cflows	Kompiliert das show_cflows-Tool
mk_test_cflows	Kompiliert das mk_test_cflows-Tool
doc	Generiert die Doxygen Dokumentation aus den C++ Quelldateien (Target muss in CMake aktiviert sein)

test	Führt Unit-Tests aus (Target muss in CMake aktiviert sein)
install	Installiert HAPlib, HAPviewer sowie die dazugehörige Dokumentation (falls aktiviert)
Nightly	Kompiliert alle konfigurierten Binaries und sendet den Nightly-Report an den Build-Server
Continuous	Kompiliert alle konfigurierten Binaries und sendet den Continuous-Report an den Build-Server
Experimental	Kompiliert alle konfigurierten Binaries und sendet den Experimental-Report an den Build-Server

6.3 Installation

6.3.1 HAPlib

Die Library kann mit Hilfe des Make Targes "install", wie zuvor beschrieben, installiert werden. Dazu sind per default root-Rechte nötig:

```
# im CMake build directory ausführen
make install
```

6.3.2 HAP4NfSen Plugin

Die Installation erfolgt wie in der Studienarbeit "Integration des HAPviewers in das NfSen Framework"[1] beschrieben.

6.4 Plugin Konfiguration

Das NfSen Plugin bietet die Möglichkeit, das Verhalten mit Hilfe von Parametern in der Konfigurationsdatei "nfsen.conf" des Backends anzupassen. Die letzte Version des Plugins bot bereits einige Einstellungen. Diese wurden nun um folgende Parameter ergänzt:

Parameter	**Erlaubte Werte**	**Default**	**Bedeutung**
graph_zoom_limit	Positive Ganzzahlen	128	Beschreibt, ab wann (in Anzahl von Knoten) der JavaScript Drilldown aus Performancegründen deaktiviert werden soll.

graph_display_limit	Positive Ganzzahlen	1024	Beschreibt, ab wann (in Anzahl von Knoten) ein Graphlet nicht mehr automatisch gerendert werden soll, um Perfomance Probleme mit Graphviz zu verhinden.
ip_chooser_top_x	Positive Ganzzahlen	50	Die maximale Anzahl Flows oder Flow Statistiken, welche im alternativen Einstiegspunkt dargestellt werden können.

Alle diese Werte beeinflussen die Performance des Back Ends(ausser graph_zoom_limit, welches sich auf die Browser Performance auswirkt) und können an das vorhandene System angepasst werden.

6.5 Eclipse Projekt

CMake bietet die Möglichkeit, ein Eclipse C++ Projekt zu generieren. Dazu muss im build Verzeichnis(siehe "Kompilierung") folgendes Kommando ausgeführt werden:

```
cmake -G"Eclipse CDT4 - Unix Makefiles" ../hapviewer/
```

Um das Projekt in Eclipse nutzen zu können, muss es per `File=> Import...=> Existing Projects into Workspace` importiert werden.

7 Protokolle der Besprechungen

7.1 Woche 1

7.1.1 Datum und Ort

Mittwoch, 23.2.2011, HSR, Raum 7.2.1, 14:55-15:45

7.1.2 Anwesende

- Eduard Glatz
- Sebastian Hügli
- Reto Schneider

7.1.3 Inhalt

- Antworten auf die Fragen an Herrn Glatz:
  - Import von Argus-Daten: Kein Fehler, ist Wunschziel
  - Projektpartner: neu ETH, nicht mehr Switch
  - Code-Style-Guide von der ETH: Keiner vorhanden ->selbst erstellen
  - Doxygen: weiterführen wie bisher
- Bisherige CFlow-Daten müsen auch zukünftig verwendbar sein
- Nice-to-have: bisheriges HPG-Dateiformat auch in Zukunft les- und schreibbar
- Herr Glatz wird uns sein CFlow-Generierungstool zukommen lassen

7.1.4 Ziele für folgende Woche

- In vorhandener Literatur recherchieren (vizsec.org)
- Projekt- und Zeitplan erstellen
- Build-System erstellen (CMake)
- Test-Umgebung evaluieren und erstellen (CTest + CUTE?)
- Automatische Tests einrichten
- Peter die Aufgabenstellung zukommen lassen

7.2 Woche 2

7.2.1 Datum und Ort

Mittwoch, 2.3.2011, 14:55-16:20, HSR, Raum 7.2.1, Gebäude 7

7.2.2 Anwesende

- Eduard Glatz
- Sebastian Hügli
- Reto Schneider

7.2.3 Inhalt

- Vorstellung des Fortschrittes letzer Woche:
  - Zeitplanungstool
  - CDash
  - VM
  - CMake
  - Tests (CUTE + CTest)

7.2.4 Ziele für folgende Woche

- Build-Name in CDash ansehen
- Projekt- und Zeitplan vervollständigen
- Tests genauer ansehen
- Analyses des bestehenden Codes fertigstellen
- Lösung für klickbare Graphen in GTK suchen
- CMake-fizierten Stand an Herrn Glatz senden

7.3 Woche 3

7.3.1 Datum und Ort

Mittwoch, 9.3.2011, 14:55-16:55, HSR, Raum 7.2.1, Gebäude 7

7.3.2 Anwesende

- Eduard Glatz
- Sebastian Hügli
- Reto Schneider

7.3.3 Inhalt

- Vorstellung des Fortschritts letzter Woche:
  - Afterglow angesehen
  - CMake: erforderliche Version auf 2.6 gesenkt
  - CMake-Eclipse-Projekt (Patch)
  - DOT-Darstellung im FatClient
  - Export Zeiterfassung
  - GIT: öffentliches Repo hier: http://git.hap4nfsen.ch/hapviewer/
  - Anpassungen OpenBSD/FreeBSD/Ubuntu 8.04/Debian Lenny

7.3.4 Ziele für folgende Woche

- Afterglow
  - Beispiele
  - Interessante Infos für Einstiegspunkt?
  - Wie in Rastergrafik rendern?
  - Farben
  - Bei Peter Haag nachfragen
- Abhängigkeiten verkleinern
- Zeitplan für nächste Iteration anpassen
- Argus-Datenimport ansehen (nur oberflächlich)
- Graphviz-Support von Boost ansehen
  - Bei Peter nachfragen
    - BA
    - HAP4Nfsen
  - Codebeispiel DOT in GTK

7.4 Woche 4

7.4.1 Datum und Ort

Mittwoch, 16.3.2011, 14:55-15:15, HSR, Raum 7.2.1, Gebäude 7

7.4.2 Anwesende

- Eduard Glatz
- Sebastian Hügli
- Reto Schneider

7.4.3 Inhalt

- Done:
  - Abhängigkeiten identifizieren
  - Argus-Möglichkeiten abklären
  - Bei Peter nachfragen
    - BA
    - HAP4Nfsen
  - Afterglow-Einstiegspunkt abklären
- Fragen:
  - Wann Sitzung am Do?
- Ideen
  - Cflow-Struct durch etwas Template basierendes ersetzen
- Nächste Sitzung
  - Donnerstag, 24.3.2011: 15:00-17:00 (max), Caffeteria
  - Mittwoch zuvor keine Sitzung

7.4.4 Ziele für folgende Woche

- Codebeispiel DOT in GTK
- IPv6
- HAP4NfSen-Code zurückportieren und Peter zukommen lassen
- Herrn Glatz den Link zu Argus #2 zukommen lassen
- Anschauen, wie wir eine neue cflow-Version mittels GZIP markieren können
- Boost + Graphviz anschauen

7.5 Woche 5

7.5.1 Datum und Ort

15:00-17:45, HSR, Gebäude 1, Cafeteria

7.5.2 Anwesende

- Eduard Glatz
- Sebastian Hügli
- Reto Schneider

7.5.3 Inhalt

- Entscheidungen:
  - Bevorzugter Weg für Argus-Support: C-Code importieren
  - gethgpMetadata(1+2) kann entfernt werden
  - Inhalt von glistview1.cpp/.h ist teilweise entbehrlich
  - Flowliste: Flowrichtung fehlt
  - Es muss nur ein Graphlet jeweils verarbeitet werden können

7.5.4 Ziele für folgende Woche

- Weiterarbeiten an IPv6-Support
- Verschlankte Version von HAP4NfSen an Peter senden
- Herrn Glatz Zugriff auf das Repo geben

7.6 Woche 6

7.6.1 Datum und Ort

1.4.2011 12:00-13:00, HSR, Gebäude 1, Cafeteria

7.6.2 Anwesende

- Eduard Glatz
- Sebastian Hügli

7.6.3 Inhalt

- Besprechung des Fortschritts:
  - Änderungen betreffend IPv6
  - HAP4NfSen Version mit verringerten Abhängigkeiten
- Erweiterung des HPG Formates
  - Temporäre Lösung: Erweiterung von 3x32 Bit Kanten zu 3x128 Bit Kanten
  - Vorschlag 1: Redefinition einer Kante: Jede Kante enthält jedes Attribut einmal
  - Vorschlag 2: Format so umdefinieren, das IP Addressen und andere durch eine Id beschrieben werden. Format müsste zusätzlich eine Mapping Tabelle(Id zu tatsächlichem Wert) enthalten

7.6.4 Ziele für folgende Woche

- Weiterarbeit an den IPv6 Anpassungen. Ziel: Graphlets mit IPv6 Adressen sollen angezeigt werden können
- Fertigstellung der HAP4NfSen Version mit reduzierten Abhängigkeiten

7.7 Woche 7

7.7.1 Datum und Ort

8.4.2011 12:00-12:40, HSR, Gebäude 1, Cafeteria

7.7.2 Anwesende

- Eduard Glatz
- Sebastian Hügli
- Reto Schneider

7.7.3 Inhalt

- HAP4NfSen Version mit reduzierten Abhängigkeiten fertiggestellt und an Peter Haag versendet
- Anzeige von IPv6 Graphlets
- Überarbeitete Version von GImport::set_local_ip()

7.7.4 Ziele für folgende Woche

- Erweiterung des Imports(Support für IPv6)
- Falls genug Zeit vorhanden:
  - Partielle Desummariserung
  - Klickbare Graphlets im HAPviewer

7.8 Woche 8

7.8.1 Datum und Ort

15.4.2011 12:00-12:55, HSR, Gebäude 1, Cafeteria

7.8.2 Anwesende

- Eduard Glatz
- Sebastian Hügli

7.8.3 Inhalt

- Partielle Desummarisierung von Graphlet Knoten
  - Möglichkeiten zur Umsetzung
  - Referenzierung von Rollen
  - Generische Ansätze zur Umsetzung
- Ersetzen der bestehenden Flowliste durch STL Vector

7.8.4 Ziele für folgende Woche

- Erste Implementation der partiellen Desummarisierung
- Fertigstellung und testen der HAP Library version mit reduzierten Abhängigkeiten

7.9 Woche 9

7.9.1 Datum und Ort

20.4.2011 17:00-17:40, HSR, Gebäude 1, Cafeteria

7.9.2 Anwesende

- Eduard Glatz
- Sebastian Hügli
- Reto Schneider

7.9.3 Inhalt

- Gezeigt: Partielle Desummariserung

7.9.4 Ziele für folgende Woche

- Klickbare HAP-Graphen
- Installation bei Peter Haag bug-fixen
- Partielle Desummariserung erweitern
- Importfilter vervollständigen
- stl::sort verwenden

7.10 Woche 10

7.10.1 Datum und Ort

29.4.2011 12:00-13:30, HSR, Gebäude 1, Cafeteria

7.10.2 Anwesende

- Eduard Glatz
- Sebastian Hügli

7.10.3 Inhalt

- Import von bestehenden Formaten(IPv6 Unterstützung)
- Partielle Desummarisierung

7.10.4 Ziele für folgende Woche

- Parititelle Desummarisierung erweitern
- HAP4NfSen Bugs fixen

7.11 Woche 11

7.11.1 Datum und Ort

6.5.2011 12:00-12:25, HSR, Gebäude 1, Cafeteria

7.11.2 Anwesende

- Eduard Glatz
- Sebastian Hügli
- Reto Schneider

7.11.3 Inhalt

- HAP4NfSen Plugin Bugs
- Alternativer Einstiegspunkt für das NfSen Plugin
- Klickbare HAP Graphlets

7.11.4 Ziele für folgende Woche

- Einbau der neuen HAPviewer Funktionalität in das NfSen Plugin
- Prototyp für alternativen Einstiegspunkt
- Klickbare Graphlets im HAP viewer

7.12 Woche 12

Für diese Woche (13.5.2011) findet kein Treffen statt, da Sebastian Hügli sich im Zivilschutz und Reto Schneider in Berlin befindet.

7.13 Woche 13

7.13.1 Datum und Ort

6.5.2011 12:00-12:30, HSR, Gebäude 1, Cafeteria

7.13.2 Anwesende

- Eduard Glatz
- Sebastian Hügli
- Reto Schneider

7.13.3 Inhalt

- HPG-Format wurde erweitert
- GInterface angepasst
- Erste Version eines IP-Graphlets mit Afterglow realisiert

7.13.4 Ziele für folgende Woche

- HAPViewer soll klickbare Graphlets darstellen können
- Front und Backend-Plugins sollen angepasst & aktualisiert werden
- Treffen mit Peter organisieren
- Rollendetektion ansehen

7.14 Woche 14

7.14.1 Datum und Ort

26.5.2011 16:00-17:15, Zürich, Switch

7.14.2 Anwesende

- Eduard Glatz
- Peter Haag
- Sebastian Hügli
- Reto Schneider

7.14.3 Inhalt

- Afterglow-Plugin wurde Peter vorgestellt
- Entscheidungen/Ideen:
  - Afterglow soll grundsätzlich das darstellen, was in NfSen ausgewählt wurde (IP, Netze, etc)
  - Graphlet könnte in einem eigenen Fenster dargestellt werden
  - Da nur noch 3 Wochen Zeit eine einfache Lösung wählen
  - Funktionsfähigkeit: Ubuntu, OpenBSD

7.14.4 Ziele für folgende Woche

- Offene Fragen/Todo:
  - Was tun bei aggregierten Netzen? Einfach immer ein Netzbit entfernen?
  - Was tun bei zu vielen Daten?
  - Wie Drilldown lösen?
  - Bugn (Undefinierter Index) auf OpenBSD fixen

7.15 Woche 15

Da in dieser Woche der Donnerstag ein Feiertag ist, findet keine Sitzung statt.

7.16 Woche 16

7.16.1 Datum und Ort

10.6.2011 13:00-14:30, ETH, Zürich

7.16.2 Anwesende

- Eduard Glatz
- Sebastian Hügli
- Reto Schneider

7.16.3 Inhalt

- Besprechung des Bugs, welcher dazu führt, dass der originale HAPviewer in speziellen Fällen einige Flows "vergisst". Lösung: Zusammenrechnen
- Besprechung der Vorteile eines tempfs für den HAPviewer. Resultat: Irrelevant für HAPviewer
- Vorstellung der aktuellen Version des Afterglow Plugins

7.16.4 Ziele für folgende Woche

- Urheberrechtshinweise anpassen: Nur noch an einer Stelle den Hinweis zum anbringen. Abweichende Dateien weiterhin gesondert deklarieren im Header.
- Import Argus: Hinweisen, warum kein voller Support realisiert
- Falls Zeit vorhanden, Lösung für temporäre Daten finden (Bspw. bei zwei gleichzeitigen Instanzen könnte dies ein Problem werden)
- Schule darauf Hinweisen, dass das Abstract später kommt (E.G. ist bis am 25. Juni in den Ferien)
- Abgabe der Arbeit: Peter Bühler, IFS oder aber im Fächlein von Herrn Glatz. Gesonderte CD an Herrn Stolze.

7.17 Woche 17

In dieser Woche findet keine Sitzung statt, da Prof. Glatz nicht da ist und am Freitag die Abgabe stattindet.

8 Rückblicke

8.1 Reto Schneider

8.1.1 Projektverlauf

Nachdem bereits vor Abschluss der Semesterarbeit es sich abzeichnete, dass wir eine Nachfolgearbeit schreiben können, war es uns möglich, selbst massgeblich an der Aufgabenstellung der BA mitzuarbeiten. Dies hatte zur Folge, dass wir eine Arbeit erstellen durften, welche uns sehr interessierte. Die ersten Wochen verbrachten wir vor allem mit Abklärungen, parallel dazu war ich zudem dabei, ein Buildsystem des HAPviewers auf Basis von CMake zu erstellen und Build-Server einzurichten. Auch musste da der Code an einigen Stellen angepasst werden, damit das Programm gleichzeitig auf FreeBSD, OpenBSD, Ubuntu 8.04 und Debian Lenny funktionierte. Entgegen unserer ursprünglichen Planung starteten wir zwecks Kennenlernen des Codes mit der Implementation der IPv6 Unterstützung. Hier wurden wir das erste Mal etwas überrascht, denn die salopp gesagt simple Erweiterung von zwei mal 32 auf zwei mal 128 Bits für die IPs erforderten umfangreiche Änderungen. Im Endeffekt war der HAPviewer über Wochen hinweg nicht kompilierfähig und wir befanden uns in einer Art Blindflug. Irgendwann dann waren wir so weit, dass der HAPviewer wieder funktionierte - diesmal mit IPv6 Unterstützung. Danach machten wir uns daran, Graphlets klickbar zu machen und partielle Rollendesummarisierung zu implementieren. Ich habe dabei das klickbare Graphlet erstellt - nachdem wir nach langen Recherchen feststellen mussten, dass es keine brauchbare, in eigenen Projekten verwendbare Lösung gibt, um DOT-Graphen zu visualisieren und klickbar zu machen. Herausgekommen ist dabei eine auf der Boost Graph Library, Graphviz und Cairo basierte Lösung, welche mit etwas Anpassungsaufwand auch gut in anderen Projekten verwendet werden könnte. Leider viel später als ursprünglich geplant haben wir uns auch dieses Semester wieder mit Peter getroffen und über den ebenfalls realisierten alternativen Einstiegspunkt gesprochen.

8.1.2 Rückblick

Unsere Vorstellung, welche Arbeit wann zu erledigen sei hat sich überhaupt nicht realisieren lassen. Die anfängliche Planung musste nahezu komplett umgekrempelt und viele Meilensteine nach hinten verschoben werden, sprich hatten einen sich ständig verändernden Zeitplan. Schlussendlich ist es aber dennoch aufgegangen und wir konnten nahezu alle Ziele erreichen.

8.1.3 Gelerntes

Es gibt da ein ganz grosser Punkt: Nichts ist so einfach wie es scheint. Ich persönlich hatte die Änderungen zu Anbeginn des Projekts total unterschätzt. Eine Erfahrung welche sich hoffentlich später im Berufsleben nicht allzu oft wiederholt.
Ebenfalls sehr lehrreich war der Umgang mit C++ in einem grösseren Projekt, der Boost-Library, Open-/FreeBSD, Cairo, CMake und die Dokumentation mit Doxygen (gehört nun zu meiner “Werkzeugkiste”).

8.1.4 Fazit

Die für das Projekt geleistete Arbeit war sehr spannend und abwechslungsreich. Wir konnten den HAPviewer um einige Features erweitern, welche in Zukunft den Benutzern die Arbeit erleichtern. Was mich persönlich am meisten freut, ist der Umstand, dass die Applikation unter einer freien Lizenz steht und wir daher nicht nach Ende der Arbeit davon "ausgeschlossen" werden wie dies bei properitären Anwendungen der Fall wäre.
Bedanken möchte ich mich bei Herrn Glatz für die ausgezeichnete Betreuung. Die wöchentlichen Sitzungen sowie die Möglichkeit, jederzeit per E-Mail Kontakt aufzunehmen hat unserem Projekt sehr viel geholfen. Ebenfalls positiv erwähnen will ich die Zusammenarbeit mit Peter, welcher uns, als wir dann endlich die Sitzung abhalten konnten, wie schon im letzten Semester zahlreiche gute Inputs gegeben hat.
Nach Ende dieser Arbeit haben Sebastian und ich vor, zusammen mit Peter das HAP4NfSen-Plugin inkl. Afterglow-Einstiegspunkt so weit zu bringen, um es an Tester verteilen zu können.

8.2 Sebastian Hügli

8.2.1 Projektverlauf

Bereits während unserer Studienarbeit im letzten Semester schlug Herr Glatz uns vor, eine Nachfolgearbeit zu einem verwandten Thema zu schreiben. Da das Gebiet sehr interessant ist und wir bereits einige Einblicke erhalten haben, sagten wir sofort zu. Wir hatten das Glück, bei der Festlegung der Ziele eigene Ideen einbringen zu können. So entstand die Aufgabenstellung, welche die Software der Studienarbeit erweitert, aber gleichzeitig neue Herausforderungen mit der Anpassung des HAPviewer bringt.
Dadurch, das wir bereits ein eingespieltes Team waren und Vorkenntnisse zu den Applikationen hatten, konnten wir gut ins Projekt starten. Nachdem wir das Build System aufgesetzt hatten, wurden wir durch die Erweiterung der Addressgrösse von IPv4 auf IPv6 erstmals vor eine grosse Herausforderung gestellt. Grund dafür war, das wir aufgrund der Grösse einen eigenen Datentyp verwenden mussten. Da IP Addressen in allen Bereichen des HAPviewer, inklusive der gelesenen und geschriebenen Dateiformate, verwendet werden, dauerte dieser Umbau mehrere Wochen.
Danach arbeiteten wir an den anderen Aufgaben, wobei die partielle Desummarisierung und die Auswertung von Klick-Ereignissen auf das Grahlet zu den grössten Herausforderungen gehörten. Bei der Desummarisierung war dies der Fall, da wir uns etwas vom bisherigen Konzept der vier Rollentypen lösen und die Handhabung von generischen Sub-Rollen einbauen mussten. Die Erkennung von Klicks auf Summarisierte Knoten war daher schwierig, da die aktuelle Version nur von GraphViz generierte GIF Graphiken anzeigt. Der erste Ansatz war die Verwendung von SVG, wie wir es in der Plugin Version des Tools verwenden. Allerdings ergaben sich dadurch zu grosse Abhängigkeiten, worauf wir uns nach einigen anderen Versuchen darauf einigten, Graphlets selbst mit Hilfe einer von GraphViz generierten Graphenbeschreibung zu zeichnen.
Gegen Ende der Arbeit trafen wir uns nochmals mit Peter Haag, dem Entwickler von NfSen um unsere Erweiterungen zu präsentieren. Durch seine Rückmeldungen

konnten wir zusätzlichen Input für die Erweiterungen unseres Plugins sammeln.

8.2.2 Rückblick

Ich bin mit dem Ergebnis unserer Arbeit sehr zufrieden. Wir konnten alle geforderten Ziele erreichen und hatten zudem noch Zeit, die optionalen anzugehen. Zu den Punkten, welche nicht wie geplant verlaufen sind, gehört sicher die Zeitplanung. Wir haben den Aufwand einiger Punkte wie etwa den Erweiterungen für IPv6, der partielle Desummarisierung oder der Klick-Erkennung auf Graphlet-Knoten komplett falsch eingeschätzt. Beide haben viel mehr Zeit als ursprünglich vorgesehen eingenommen, was die ursprüngliche Planung durcheinander brachte. Trotzdem haben wir am Ende des Projekts unsere Ziele erreicht.

8.2.3 Gelerntes

Zu Beginn dieses Projekts hatte ich nur wenig Erfahrung mit C und C++. Durch die Arbeit am HAPviewer hatte ich erstmals die Möglichkeit, mich mit einer grossen und komplexen C++ Applikation auseinanderzusetzen, was sehr lehrreich war. Zudem konnte ich meine Kenntnisse in Perl und PHP, welche ich im Laufe der vorangegangenen Studienarbeit erworben habe, durch die Erweiterungen am NfSen Plugin vertiefen. An neuen Tools lernte ich ich Afterglow, CMake, Argus und Valgrind kennen.
Auch in der Planung grösserer Projekte und insbesondere in der Abschätzung des Aufwands einzelner Arbeiten konnte ich zusätzliche Erfahrungen sammeln.

8.2.4 Fazit

Die Erweiterung des HAPviewer war sehr interessant und lehrreich. Das Projekt bot die Möglichkeit, sich in eine grössere bereists bestehende Applikation einzuarbeiten und Erweiterungen vorzunehmen. Die daraus gewonnen Erfahrungen werden mir bestimmt bei ähnlichen Situationen im Berufsalltag nützlich sein. Was mich besonders gefreut hat, ist, das wir mit unseren Erweiterungen etwas erstellen konnten, was nach dem Abschluss dieser Arbeit wirklich eingesetzt wird.
Ich möchte mich bei Herrn Glatz für die Betreuung der Arbeit bedanken. In den Sitzungen nahm er sich jede Woche viel Zeit, um uns zu beraten und die Fortschritte und Probleme unserer Arbeit zu besprechen, was uns sehr geholfen hat. Zudem möchte ich mich ebenfalls bei Peter Haag bedanken, der uns bereists seit unser Studienarbeit unterstützt. Er nahm sich Zeit, die Anpassungen am HAP4NfSen Plugin zu besprechen und konnte uns mit seinem Input sehr weiterhelfen.
Nach dieser Arbeit werde ich zusammen mit Reto das Plugin fertigstellen, damit es an die NfSen Benutzer weitergegeben werden kann.

9 Glossar

Begriff	Beschreibung
AJAX	AJAX(Asynchronous JavaScript and XML) ist ein Konzept, welches asynchronen Datenaustausch zwischen Server und Webbrowser erlaubt.
Argus	Argus(Audit Record Generation and Utilization System) ist ein Open Source Tool zum aufzeichnen und auswerten von Netzwerkverkehr
AS	Autonomes System, eine Gruppe von IP-Netzen
C	Eine weit verbreitete prozedurale Programmiersprache.
C++	Eine Programmiersprache mit prozeduralen und objektorientierten Elementen.
Cairo	Eine 2D Graphik Bibliothek
CDash	Ein webbasierender Softwaretesting Server
CMake	Ein Buildsystem
DOT	DOT language, eine Graphenbeschreibungssprache
XDOT	Erweiterte Version von DOT, Layout bereits berechnet
Graphlet	Eine vom HAP Viewer generierter Graph, welcher aus 5 Partitionen(Host IP, Protokoll, Host Port, Destination Port und Destination IP) besteht.
GraphViz	GraphViz(Graph Visualization Software) ist eine Software, welche aus Graphenbeschreibungen Bilder generiert.
GTK/GTKMM	Toolkit für das Erstellen von graphischen Userinterfaces
HAPviewer	Der HAPviewer(Host Application Profile Graphlet Viewer) ist eine Applikation zur Analyse von Netzwerkverbindungen.
HAP4NfSen	Das im Rahmen dieser Arbeit erstellte Plugin für NfSen

hpg	**H**ost **P**rofile **G**raphlet - Ein properitäres Format von Prof. Eduard Glatz um Graphen platzsparend zu speichern. Wird vom HAPviewer intern verwendet.
HTML	HTML(HyperText Markup Language) ist eine Sprache, die die Darstellung von Webseiten beschreibt.
JavaScript	Die Skriptsprache Java Script wird hauptsächlich innerhalb von Web Browsern verwendet, um Webseiten dynamischer zu gestalten.
Netflow	Ein Tupel von Verbindungsdaten, das Informationen wie die von beiden Seiten verwendeten Ports enthält.
NfDump	NfDump(Netflow Dump) ist eine Kommandozeilen Applikation, mit welcher man Netflow Dateien Filtern sowie statistische Auswertungen erstellen kann.
NfSen	NfSen(Netflow Sensor) ist eine Applikation zum sammeln und analysieren von Netflows
Perl	Eine Skriptsprache.
PHP	PHP(PHP: Hypertext Preprocessor) ist eine Skriptsprache welche zur erzeugung von dynamischen Webseiten verwendet werden kann.
SVG	SVG(Scalable Vector Graphics) ist ein XML basiertes Format für Vektorgraphiken.
SVGPan	Eine JavaScript Library mit der eine SVG Graphik um Zoom und Pan Fumktionalität erweitert werden kann.
SWIG	**S**implified **W**rapper and **I**nterface **G**enerator) ist ein Tool, um im C/C++ geschriebene Programme in anderen Programmiersprachen (in unserem Fall Perl) verfügbar zu machen.
Valgrind	Eine Sammlung von Tools, welche es unter anderem ermöglichen, Memory Leaks zu erkennen

List of Figures

References

[1] http://eprints3.hsr.ch/168/, 21.2.2011

[2] http://hapviewer.sourceforge.net/, 21.2.2011

[3] http://nfdump.sourceforge.net/, 21.2.2011

[4] http://www.qosient.com/argus/, 8.4.2011

[5] http://tools.ietf.org/html/rfc4893, 15.6.2011

[6] http://argus.tcp4me.com/, 20.2.2011

[7] www.vizsec2010.org/files/Eduard%20Glatz.pdf, 10.7.2011