Trau, SCHAU, wem? – V-IDS oder eine andere Sicht der Dinge

Björn Scheuermann Andreas Lindenblatt
Daniela Lindenblatt Benjamin Guthier
Solution – The Computer People e. K., Mannheim, Germany
{bjoern,azrael,tila,benjamin}@solution.de

Abstract: Die ständig wachsende Flut der in einem Netzwerk anfallenden sicherheitsrelevanten Daten macht in zunehmendem Maße neue Darstellungsformen notwendig. Nur so können diese Daten ausreichend schnell und in angemessenem Umfang erfassbar und beherrschbar bleiben. Wesentlich schneller und intuitiver als reinen Text können wir den Inhalt von Bildern erfassen, grafische Darstellungen machen Geschehnisse in der Regel leichter erfassbar. Informationen können zusätzlich stärker verdichtet dargestellt werden, ohne dass der transportierte Inhalt darunter leidet. Die Darstellung von Sicherheitsdaten in grafischer Form steht derzeit noch sehr am Anfang, es gibt wenig Erfahrung, welche Darstellungen mehr und welche weniger geeignet sind. V-IDS soll Grundlagen legen für eine dynamische, dreidimensionale Darstellung solcher Daten. Es soll ein einfaches Experimentieren mit verschiedenen und neuartigen Darstellungen ermöglichen. Damit können dann vorhandene und zukünftige Ideen einfach und ohne längere Entwicklungszeit prototypisch umgesetzt und bewertet werden.

1 Motivation

„Man muss zum Auge reden, will man verstanden werden“
(Johann Gottfried Herder)

In den letzten Jahren hat sich der Bereich Informationstechnik stetig weiterentwickelt. Eine Veränderung tritt sicher mit am stärksten hervor: die Wandlung von kleinen Systemen, Insellösungen zu weiträumigen, weltweit miteinander vernetzten Systemen. Damit einher geht das Aufkommen völlig neuer Risiken, die Gefahr aus dem Netz ist real.

Durch diese veränderte Situation wird es immer schwieriger, die Netze zu administrieren, zu kontrollieren und zu sichern. Netzwerkadministratoren stehen vor einer immer größer und umfangreicher werdenden Aufgabe: In der teilweise sehr unübersichtlichen Datenflut müssen sie das für Ihre Belange Wesentliche rechtzeitig und genau erkennen und richtig einordnen.

Dies ist deshalb so wichtig, da der Mensch als urteilende und gegebenenfalls reagierende Instanz unabdingbar ist. Gerade bei neuartigen Vorkommnissen muss der Mensch erkennen und / oder entscheiden, wie ein Ereignis zu bewerten ist und ob es eine potentielle

Gefahr bedeutet. Aktuelle Sicherheitssysteme leisten hier einen erheblichen und wertvollen Beitrag. Sie konzentrieren sich aber in aller Regel auf das zuverlässige Erkennen von sicherheitsrelevanten Ereignissen und das Aggregieren von Daten aus unterschiedlichen Quellen. Die Übermittlung dieser Daten an den oder die Systembetreuer ist hier nicht als primäres Ziel vorgesehen. Die Meldungen erfolgen meist rein textbasiert, eine visuelle Übermittlung ist nicht oder nur eingeschränkt möglich.

Textbasierte Informationen lassen sich zwar prägnant und unmissverständlich weitergeben, der Mensch ist jedoch kaum fähig, größere Mengen Text oder Tabellen in kurzer Zeit aufzunehmen und auszuwerten. Wesentlich leichter und intuitiver als Text kann man Bilder erfassen. Grafische Darstellungen machen Geschehnisse leichter erkennbar und erfassbar, dreidimensionale Modelle machen sie sogar im wahrsten Sinne des Wortes „fühlbar“, wie mehrere Studien verschiedener Fachrichtungen belegen. Hierbei spielt die Art der Darstellung (Form, Farbe,. . .) eine wesentliche Rolle [Ja90, Ja94].

Die Fortschritte in der Technik zum Erzeugen von Bildern in dreidimensionalen, künstlichen Welten und zur Navigation in diesen sind – auch durch die Spieleindustrie – unübersehbar. Es liegt daher nahe, diese Fähigkeiten aktueller Technologie zum Veranschaulichen und Begreifbar-Machen abstrakter Ereignisse im Sicherheitsbereich zu nutzen.

Ergänzend trägt eine solche Darstellung zur Lösung einer anderen oft undankbaren Aufgabe von IT-Sicherheitsverantwortlichen bei. Denn diese müssen Kollegen und Vorgesetzten die Notwendigkeit von Einschränkungen vermitteln, die für die Sicherheit wichtig sind, aber als lästig empfunden werden. Nicht zuletzt müssen sie auch finanzielle Aufwendungen durchsetzen, um die Sicherheit im Netzwerk zu gewährleisten. In beiden Bereichen stoßen sie häufig auf Unverständnis und Widerstand. Für die „User“ und die „Entscheider“ ist es schwer nachzuvollziehen, warum bestimmte Beschränkungen bestehen oder weshalb finanzielle Aufwendungen zum Erhalt und zur Verbesserung der Sicherheit notwendig sind.

2 Verwandte Arbeiten

Viele der Grundlagen, auf denen V-IDS basiert, waren und sind Gegenstand vielfältiger Forschungsarbeiten. Sie wurden jedoch, soweit wir das wissen, bislang noch nicht in dieser Weise miteinander kombiniert. Im Bereich der Erkennung, (teil-)automatisierten Bewertung und Klassifizierung von sicherheitsrelevanten Ereignissen existiert eine unüberschaubare Fülle von Projekten. Zumindest zu denjenigen, die eine gewisse Verbreitung im praktischen Einsatz erreicht haben, gibt es oft auch Analyse- und Auswertungswerkzeuge. Diese erlauben es, mehr oder weniger flexibel, Anfragen in bestimmten Formen darzustellen, nach Bedarf aggregierte Darstellungen zu erzeugen und Vergleiche vorzunehmen. In der überwiegenden Mehrheit der Fälle sind dies tabellarische Auswertungsschemata oder zweidimensionale bildliche Darstellungen.

Dass grafische Darstellungen prinzipiell das Erkennen von Strukturen in Logdaten erleichtern bzw. gar erst ermöglichen können, zeigen die von Axelsson in [Ax03] dokumentierten Erfahrungen mit der Auswertung von Webserver-Logs über Trellis-Graphen.

Auch im Bereich der allgemeinen grafischen Datenvisualisierung – in zwei oder drei Dimensionen – existieren bereits einige viel versprechende Ansätze. Diese verzichten jedoch, zugunsten der Größe der zu verarbeitenden Datenmengen, fast immer auf dynamische und interaktive Elemente. Sie sind daher zur Visualisierung aktueller Vorgänge in Echtzeit ungeeignet, oder es fehlen wichtige Funktionen, die die Darstellung diskreter Ereignisse für die angestrebte Anwendung ausreichend erleichtern.

Beispielhaft seien hier der IBM Open Visualization Data Explorer (exemplarische Anwendungen z.B. in [IBM96]) oder auch die Visualisierungs-Tools Tulip (speziell für sehr große Graphen, [Da01]) und WilmaScope [DE01] genannt.

Mit der Darstellung von Graphen, speziell auch von Daten zur Struktur und Leistungsmerkmalen von Rechnernetzen, haben sich bereits mehrere Projekte der AT&T Labs beschäftigt. Aber auch hier liegt der Fokus hauptsächlich auf zweidimensionalen und statischen Darstellungen [ATT].

Das thematisch verwandte NIVA-Projekt [NCSLO02] möchte ebenfalls IDS-Daten dreidimensional aufbereiten. Hier steht die Integration von haptischem Feedback mit im Vordergrund. Es wurden einige Ansätze für konkrete Darstellungsformen entwickelt. NIVA ist jedoch auf die vorgegebenen grafischen Umsetzungen beschränkt.

3 Entwurfsprinzipien von V-IDS

Nach unserer Vorstellung soll mit V-IDS ein Werkzeug entstehen, das die Meldungen und Aufzeichnungen der Sicherheitssysteme in einer aussagekräftigen dreidimensionalen Darstellung vermittelt. Der Anwender soll die Möglichkeit haben, zu bestimmen, in welcher grafischen Form die Ereignisse dargestellt werden. Idealerweise mit dem Anspruch, keine unerwünschten Verluste bzgl. des Informationsgehaltes der Daten zu erzeugen.

Die Zielsetzung von V-IDS ist es, die tägliche Praxis des Erkennens von Netzwerkproblemen und auch deren spätere Analyse wirksam zu unterstützen. Auftretende Unregelmäßigkeiten sollen leichter und effektiver erkannt und erfasst werden, als positiver Nebeneffekt soll auf diese Weise auch ein besseres Verständnis für Abläufe in einem Netzwerk erreicht werden.

Es fallen eine Vielzahl unterschiedlicher Daten an, deren Menge, Form und Zusammensetzung zusätzlich im Einzelfall stark variiert. Ebenso erfordert die Praxis je nach gegebener Situation völlig unterschiedliche Zusammenstellungen und Auswertungen der vorhandenen Daten. Daher bedarf es eines extrem hohen Grades an Individualisierbarkeit der Darstellung. Eine derarige Flexibilität kann durchaus ganz allgemein als wichtige Anforderung an Visualisierungs-Toolkits angesehen werden [TFC+89].

Um viele unterschiedliche, auch gegenwärtig noch gar nicht in Betracht gezogene Einsatzmöglichkeiten unterstützen zu können, besteht V-IDS aus einer daten- und datenquellenunabhängigen Visualisierungs-Engine (VE) mit einer Schnittstelle für modulare Eingabefilter. Letztere stellen die Verbindung zu einer Datenquelle her und bringen deren Daten in ein einheitliches Format. Um die Struktur der Eingabefilter so einfach wie möglich hal-

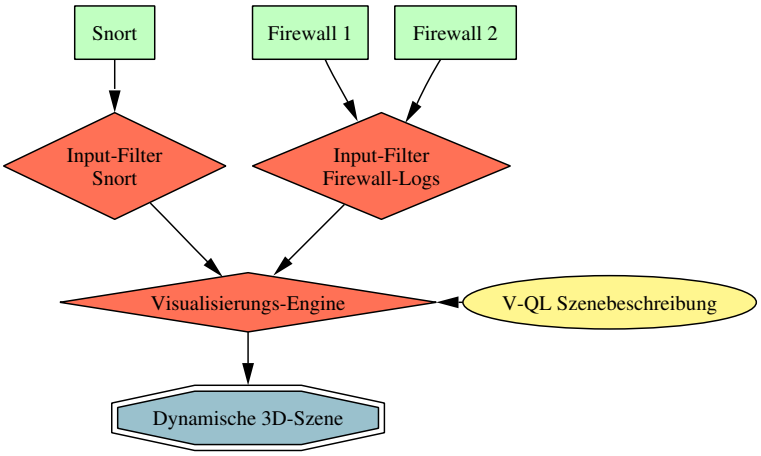

Abbildung 1: V-IDS-Architektur, hier exemplarisch mit einem Snort-IDS und zwei Firewall-Systemen als verwendete Datenquellen

ten zu können, umfasst die VE mit Abstand den größten Teil des Projektes. Für V-IDS kann so eine neue Datenquelle allein durch das Schreiben eines passenden Eingabefilters – oft nicht mehr als einige Zeilen in einer gängigen Skriptsprache – erschlossen werden. Eine Übersicht der Architektur zeigt Abb. 1.

Wie die Daten letztlich dann in einer 3D-Szene dargestellt werden, ist ebenfalls nicht vordefiniert. Stattdessen erfolgt die Umsetzung anhand einer Szenenbeschreibung. Eine solche Szenenbeschreibung ist in der speziell zu diesem Zweck entwickelten Sprache V-QL formuliert. Ein V-QL-Skript bestimmt, welche eingehenden Nachrichten welche Objekte und Eigenschaften der Darstellung wie betreffen. Die VE interpretiert die V-QL-Szenenbeschreibung und setzt die eingehenden Daten entsprechend um. Eine aus einem mehr oder weniger komplexen V-QL-Skript bestehende Anweisung zur Auswertung lässt sich so mit beliebigen Daten kombinieren, z. B. mit den Protokolleinträgen aus einem fix definierten Zeitfenster oder mit Echtzeitdaten.

V-IDS kann aufgrund dieser Architektur zunächst einen praktischen Einsatz bei der Echtzeitanalyse zur Problemerkennung finden und später auch bei der gezielten Auswertung und Beurteilung von Protokolldateien. Die Eingabefilter müssten nur im ersten Falle Echtzeitdaten, im zweiten die Einträge einer Logdatei an die VE übermitteln. Ebenso denkbar ist die Möglichkeit, V-IDS zu Demonstrations- und Dokumentationszwecken einzusetzen, mit dem Ziel, auch Nicht-Fachleuten die Ereignisse und Vorgänge zu veranschaulichen.

V-QL und die VE sind so angelegt, dass sie die häufig anfallenden Routineaufgaben übernehmen und den Anwender von Aspekten wie Datenhaltung- und -repräsentation oder Algorithmenwahl in hohem Maße abstrahieren lassen. Der Autor von Szenenbeschreibungen hat die Aufgabe, die Wahl für eine geeignete Darstellung zu treffen, deshalb benötigt er selbstverständlich ein grundlegendes Verständnis für die Umsetzung der „Rohdaten“ in die gewünschte Anzeigeform.

Der flexible Ansatz mit der per Skript frei konfigurierbaren VE zeigt gegenüber einer Herangehensweise mit einem statisch vordefinierten Satz von Darstellungen mehrere Vorteile:

- Darstellungen zum leichteren Auffinden bestimmter Regelmäßigkeiten sind beim Aufkommen von Verdachtsmomenten im Bedarfsfall schnell erstellt.
- Angreifern wird ein gezieltes Erzeugen von Fehlalarmen zur Verschleierung erschwert, ebenso das Umgehen von Aufmerksamkeit erregenden, auffälligen Darstellungen.
- Der Austausch bewährter Darstellungen zwischen V-IDS-Nutzern bietet sich an.

Gerade für Anwender, die sich nicht in die Details der Konfiguration für die Auswertungen einarbeiten können oder möchten, würde es eine aus vorgefertigten Darstellungsschemata bestehende Bibliothek ermöglichen, V-IDS dennoch gezielt einzusetzen. Vergleichbar ist dies mit dem derzeit gängigen Austausch von Signaturen zwischen Anwendern signaturbasierter IDS-Systeme. Auf diese Weise lassen sich die Vorteile eines gänzlich frei konfigurierbaren Darstellungssystems mit denen eines fertigen Satzes von grafischen Aufbereitungen der Daten kombinieren.

4 Die Szenenbeschreibungssprache V-QL

V-QL ist die Beschreibungssprache, in der der Anwender von V-IDS der VE mitteilt, welche Arten von Eingangsdaten sie zu erwarten hat und wie diese in eine dreidimensionale Szene umzusetzen sind. Für eine umfassende Darstellung von V-QL reicht der hier zur Verfügung stehende Platz nicht aus.

Szenenbeschreibungen werden nicht wie in einer imperativen Programmiersprache Zeile für Zeile abgearbeitet. Vielmehr besteht ein typisches Skript vorwiegend aus Definitionen von Zusammenhängen. Bei der Auswertung durch die VE entscheidet diese, welche Berechnungen wann tatsächlich durchgeführt werden. Ein Beispiel, das hier Schritt für Schritt aufgebaut wird, soll diesen Ansatz verdeutlichen.

In V-QL werden neben den grundlegenden Datentypen wie Ganz- und Fließkommazahlen, booleschen Werten und Strings auch Vektoren und spezielle Datentypen für die Grafikdarstellung (Texturen, Transformationen,. . .) direkt mit Sprachkonstrukten unterstützt. Das zentrale Element sind jedoch Mengen. Eine Menge in V-QL ist eine Definition für den Aufbau, die Eigenschaften, das Verhalten und nicht zuletzt das Aussehen einer Gruppe „gleichartiger“ Objekte. Alle Objekte einer Menge enthalten die gleichen Datenfelder und deren Abhängigkeiten sind gleich definiert. Die konkreten Werte können sich jedoch zwischen den einzelnen Mengenelementen unterscheiden.

Beispielsweise würden alle Hosts in einem Netzwerk, die auf die gleiche Art und Weise dargestellt werden sollen, eine Menge bilden. Jeder dieser Hosts hat Eigenschaften wie z. B. seine IP-Adresse, aber auch die Position, Form, Farbe etc. für die Anzeige.

```
hosts := SETOFSIZE(6);
EnumID hosts->hostNr;

innerRad := 6.0;

IN hosts {

  rotAngle := Pi/3 * hostNr;
  hostTrans := Rotate([0,1,0], rotAngle) * Translate([0, 0, <innerRad]);

  SHOW(Box, Color([1.0, 0.2, 0.0]), hostTrans);

};
```

Abbildung 2: Eine einfache V-QL-Szenenbeschreibung

Im ersten Beispielskript (Abb. 2) wird dies deutlich. Zu Beginn wird eine Menge `hosts` definiert, die hier der Einfachheit halber konstant sechs Elemente enthalten soll. Zunächst sind diese Elemente alle gleich und haben keinerlei Eigenschaften, da sie nicht auf irgendwelche Eingabedaten zurückgehen. Um sie unterscheiden zu können, soll eine eindeutige Nummer vergeben werden. Mit dem `EnumID`-Statement geschieht genau das: Die Elemente der Menge `hosts` erhalten nun ein Attribut `hostNr`, das die VE mit den Werten 0 bis 5 belegen wird. Somit ist es nun möglich, innerhalb von `hosts` neue Eigenschaften – wie im Beispiel die Position – zu definieren, deren Werte sich in Abhängigkeit dieser eindeutigen ID berechnen. Um dies zu tun, kann wie im Beispiellisting mittels des `IN`-Statements in den Kontext einer bestimmten Menge gewechselt werden.

Zentraler Punkt dieses einführenden Beipiels ist die `SHOW`-Anweisung. Sie legt fest, dass und wie die Elemente der Menge, in deren Kontext `SHOW` auftaucht, angezeigt werden. Ihre drei Parameter legen die geometrische Form (hier: ein Würfel), die Oberflächenbeschaffenheit (einfarbig in einem kräftigen Rot, der Vektor `[1.0, 0.2, 0.0]` repräsentiert eine Farbe in RGB-Darstellung) sowie eine geometrische Transformation fest, die Position und Größe bestimmt.

V-QL unterstützt gängige Transformationen wie Rotation (im Beispiel in Achse-Winkel-Repräsentation), Skalierung, Translation in verschiedenen Varianten direkt. Außerdem lassen sich Transformationen verketten, wobei von rechts nach links ausgewertet wird.

In der Definition der Translation im Beispiel fällt eine weitere Besonderheit auf: Der Verschiebungsvektor nimmt Bezug auf den Wert `innerRad`, der jedoch nicht im Kontext der Menge `hosts`, sondern im übergeordneten Kontext definiert ist. Für den Zugriff auf solche Werte aus „höheren Ebenen“ kommt der <-Operator zum Einsatz.

Ein Screenshot der erzeugten Szene findet sich in Abb. 3.

Dieses Beispiel verarbeitet noch keine dynamischen Daten. Um dies zu ermöglichen, ist die Definition einer Eingabemenge nötig. Sie stellt die Verbindung zu einem Input-Filter her, der dann zur Laufzeit Elemente hinzuzufügen oder entfernen kann. Dazu muss der VE mitgeteilt werden, welche Eigenschaften die eingehenden Datensätze haben. Wie das betrachtete Beispiel entsprechend erweitert werden kann, zeigt das zweite Listing in Abb. 4.

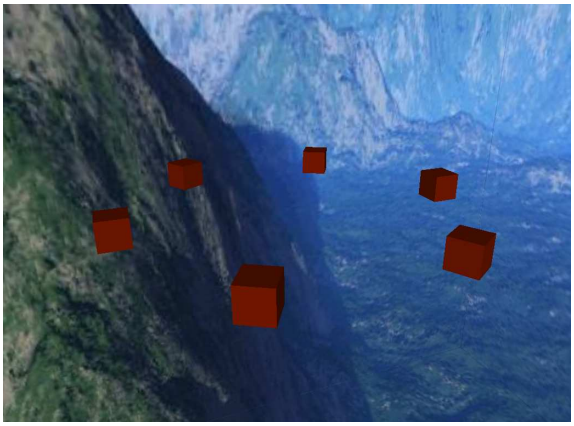

Abbildung 3: Screenshot der mit dem Skript aus Abb. 2 erzeugten Szene

Mit dem `INPUT`-Statement werden hier die Eingabemenge und ihre Attribute deklariert. So weiß die VE, dass sie zur Laufzeit Datensätze aus einer externen Quelle erhält, die sie als Elemente der Menge `snortIn` speichern wird. Die für `sourceIP`, `destIP` und `severityCode` angegebenen Werte sind lediglich Standardwerte, die nur angenommen werden, falls ein eintreffender Datensatz unvollständig spezifiziert wurde.

IPv4-Adressen werden in V-QL als vierdimensionale Ganzzahl-Vektoren behandelt; auf diese Weise ist für diesen häufig auftretenden Datentyp keine Sonderbehandlung notwendig.

Die vom Input-Filter in die Eingabe-Menge `snortIn` übergebenen Datensätze können nun als Grundlage für eine Darstellung dienen. Dazu sollen zunächst die eintreffenden Meldungen auf die sechs im Beispiel betrachteten Hosts aufgeteilt werden. Dies geschieht mit der `SELECT`-Anweisung, die aus der Eingabemenge die zum jeweiligen Host gehörenden Meldungen heraussucht. Für jeden Host enthält die Menge `specific` Datensätze mit der passenden `destIP`.

Jedes Element aus der Teilmenge `specific` entspricht einer Meldung, die aus einer Datenquelle von aussen in die VE eingegangen ist und angezeigt werden soll. Wiederum mittels eines `IN`-Statements kann nun noch eine Ebene tiefer in den Kontext dieser Menge abgestiegen werden, um dort Definitonen vorzunehmen. Position und Farbe für jede einzelne dieser Meldungen werden auf diese Weise bestimmt, ebenso wie ihre Darstellung (hier als Kugeln). Diese Eigenschaften sind für jede einzelne Meldung verschieden, da sie aufgrund unterschiedlicher Eingangsdaten berechnet wurden. So bekamen zum Beispiel alle Elemente eine eindeutige, fortlaufende Nummer `id`, die später in der Definition der Position Verwendung findet.

Startet man die VE mit dem Skript aus dem Beispiel und einem Input-Filter, der Daten im erwarteten Format bereitstellt, wird unmittelbar der statische Teil der Szene – bestehend hier aus den sechs Würfeln – aufgebaut. Sobald der Input-Filter nun Datensätze einfügt,

```
// Deklaration des Eingabeformats
INPUT snortIn {

  sourceIP = [127, 0, 0, 1];
  destIP   = [192, 168, 0, 1];
  severityCode = 5;

};


hosts := SETOFSIZE(6);
EnumID hosts->hostNr;

innerRad := 6.0;
sphereDist := 5.0;
sphereRad := innerRad + sphereDist;
outerRad := 4.0;


IN hosts {

  rotAngle := Pi/3 * hostNr;
  hostTrans := Rotate([0,1,0], rotAngle) * Translate([0, 0, <innerRad]);

  // Würfel für die Hosts anzeigen
  SHOW(Box, Color([1.0, 0.2, 0.0]), hostTrans * Scale(2));

  // Host-spezifische Daten suchen
  specific := SELECT FROM <snortIn WHERE destIP == [192,168,0,<hostNr+1];
  EnumID hostSpecific->id;

  IN specific {

    // Farben für die Kugeln in Abh. von severityCode wählen
    color := IF severityCode > 5 THEN [1, (10-severityCode)/5, 0]
                                 ELSE [severityCode/5, 1, 0];

    // Position der Kugeln festlegen
    spherePos := [Sin(Pi/5*id), Cos(Pi/5*id), 0.0] * <<outerRad +
                 [0.0, 0.0, <<sphereRad + id/3];

    SHOW(Sphere, Color(color), <hostTrans * Translate(spherePos));

    // Verbindende Zylinder anzeigen
    SHOW(Cylinder, Color([0.2, 0.7, 1.0]), <hostTrans *
      FromTo([0, 0, 0], spherePos) * Scale([0.2, 1.0, 0.2]));

  };

};
```

Abbildung 4: Die Beispiel-Szenebeschreibung in der erweiterten Fassung

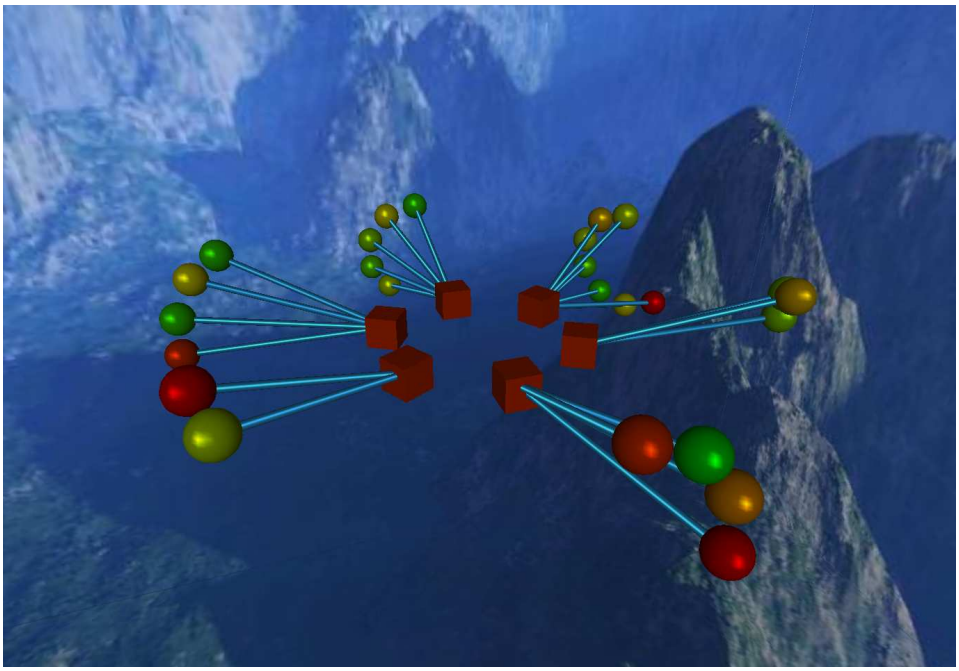

Abbildung 5: Screenshot der erweiterten Beispiels aus Abb. 4 nach Übergabe einiger Beispieldaten

verändert oder löscht, wird die Darstellung entsprechend aktualisiert. Die Ereignis-Verwaltung, die Ermittlung, welche Teile der Szene aktualisiert werden müssen usw., übernimmt die VE. Einen Screenshot zeigt Abb. 5.

Das hier gezeigte Beispiel kann bei weitem nicht alle Fähigkeiten von V-QL und der VE demonstrieren, auch ist eine Darstellung wie die gezeigte natürlich noch sehr rudimentär. Viele weitere Funktionen, die unter anderem Unterstützung für stetige Bewegungsabläufe, automatisches Layout von Elementen, komplexere Objekte, Aggregatfunktionen usw. bereitstellen, sind vorhanden und einsatzbereit. Ihre detaillierte Beschreibung würde jedoch diesen Rahmen sprengen.

5 Projektstatus und zukünftige Arbeit

Es existiert eine grundlegende Implementation der VE. Des Weiteren sind einige Eingabefilter einsatzbereit, unter anderem für die Übergabe von Snort2-Meldungen in Echtzeit.

Ein Schwerpunkt der gegenwärtigen Arbeit ist die Darstellung textueller Informationen in der 3D-Szene. So sollen ergänzende Informationen auf oder an den 3D-Objekten dargestellt bzw. interaktiv abrufbar gemacht werden. Dies ist ein Berührungspunkt mit dem

zweiten Bereich, der momentan erweitert wird: der direkten Interaktion des Benutzers mit der 3D-Szene. Dies betrifft insbesondere Mechanismen, um die Behandlung von Benutzereingaben in V-QL formulieren zu können.

6 Schlussbetrachtung

Vieles deutet darauf hin, dass sich die Art der Darstellung von Abläufen in einem Netzwerk und das Auftreten sicherheitsrelevanter Ereignisse gegenüber den momentan gängigen und eingesetzten Verfahren noch deutlich verbessern lässt. Bislang sind die Erfahrungen hinsichtlich besser geeigneter, grafischer Darstellungsformen noch sehr eingeschränkt. Ein flexibel konfigurierbares Darstellungssystem wie V-IDS kann daher dazu dienen, hier mehr Praxiserfahrung zu sammeln und realistische Einsatzszenarien in größerem Kontext zu entwickeln.

Literatur

[ATT] At&t information visualization research group, webpage. http://www.research.att.com/areas/visualization/.

[Ax03] Axelsson, S.: Visualization for intrusion detection hooking the worm. In: *Computer Security – ESORICS 2003 – 8th European Symposium on Research in Computer Security, Gjørvik, Norway*. S. 309–325. Springer Verlag. 2003.

[Da01] David, A.: Tulip. In: *Graph Drawing: 9th International Symposium, 2001, Vienna, Austria*. volume 2265 / 2002 of *Lecture Notes in Computer Science*. S. 435. 2001.

[DE01] Dwyer, T. und Eckersley, P.: Wilmascope – an interactive 3d graph visualisation system. In: *Graph Drawing: 9th International Symposium, 2001, Vienna, Austria*. volume 2265 / 2002 of *Lecture Notes in Computer Science*. S. 442. 2001.

[IBM96] IBM Corporation: *Proceedings of the 1996 IBM Visualization Data Explorer Symposium*. 1996.

[Ja90] Jacobs, B.: Ein vergleich der auswirkungen graphischer und tabellarischer präsentationsformen auf die schnelligkeit und genauigkeit beim erkennen und interpretieren statistischer daten. *Arbeitsberichte des Medienzentrums der Universität des Saarlandes*. 3. 1990.

[Ja94] Jacobs, B.: Graphische vs. tabellarische präsentation von statistischen daten. *Zeitschrift für Pädagogische Psychologie*. 8:73–84. 1994.

[NCSLO02] Nyarko, K., Capers, T., Scott, C., und Ladeji-Osias, K.: Network intrusion visualization with niva, an intrusion detection visual analyzer with haptic integration. In: *Proceedings of the 10th Symposium On Haptic Interfaces For Virtual Environment and Teleoperator Systems*. S. 277. IEEE Computer Society. 2002.

[TFC+89] Treinish, L. A., Foley, J. D., Campbell, W. J., Habor, R. B., und Gurwitz, R. F.: Effective software systems for scientific data visualization. In: *ACM SIGGRAPH 89 Panel Proceedings*. S. 111–136. ACM Press. 1989.