Integrated Visualization of Network Security Metadata from Heterogeneous Data Sources

Volker Ahlers(✉), Felix Heine, Bastian Hellmann, Carsten Kleiner, Leonard Renners, Thomas Rossow, and Ralf Steuerwald

Faculty IV, Department of Computer Science, University of Applied Sciences and Arts Hannover, P.O. Box 920251, 30441 Hannover, Germany
volker.ahlers@hs-hannover.de, trust@f4-i.fh-hannover.de
http://trust.f4.hs-hannover.de/

Abstract. In computer networks many components produce valuable information about themselves or other participants, especially security analysis relevant information. Although such information is intrinsically related as components are connected by a network, most of them still operate independently and do not share data amongst each other. Furthermore, the highly dynamic nature of a network hampers a profound understanding of security relevant situations, such as attack scenarios. Hence, a comprehensive view of the network including multiple information sources as well as temporal network evolution would significantly improve security analysis and evaluation capabilities. In this paper, we introduce a comprehensive approach for an integrated visualization, covering all aspects from data acquisition in various sources up to visual representation of the integrated information. We analyze the requirements on the basis of an exemplary scenario, propose solutions covering these demands based on the IF-MAP protocol, and introduce our software application VisITMeta as a prototypical implementation. We show how the graph-based IF-MAP protocol provides a graphical model for an integrated view of network security.

1 Introduction

In recent years several visualization approaches have been proposed for network security components like Intrusion Detection System (IDS and flow controllers, which monitor different aspects of network traffic or the behavior of systems and users [11]. For a comprehensive view on the overall network state, however, an integrated visualization of security information gathered from multiple separate components is desirable. In contrast to existing dashboard user interfaces, which visualize information from different sources in separate views on the same screen, we aim at the visualization of homogenized data within a single representation to emphasize their interrelations. In this way, the user can get a thorough understanding of well-defined aspects of the network by focusing on its immediate surroundings, while still being able to gain a broad overview of the network by including more generally related data, e.g. infrastructure information. An integrated visualization thereby facilitates a detailed assessment of the security state and the detection of potential security threats or attacks.

© Springer International Publishing Switzerland 2016
S. Mauw et al. (Eds.): GraMSec 2015, LNCS 9390, pp. 18–34, 2016.
DOI: 10.1007/978-3-319-29968-6_2

1.1 Exemplary Real World Scenario

The following scenario clarifies the necessity of those advanced monitoring and visualization approaches and serves as a basis for the requirements we derive for our approach. Given is a typical enterprise environment with several employees, each with a personal computer or smartphone. They are part of a network, which includes services, like mail and SSH servers, internal storage devices, and databases. An IDS is used to detect unwanted behavior and firewalls regulate the traffic and enforce the security policy.

One possible attack scenario is a user downloading an infected file (not recognized by the IDS), which results in compromising his account and device. The malware not only starts to spread across the internal network, but also tries to compromise further accounts by conducting brute-force login attempts onto available services within the network (this finally is detected by the IDS). Subsequently systems successfully infected start doing the same.

The desired process for a network security administrator is to: (1) Quickly detect not only single incidents, but recognize the combination of failed login attempts on multiple services by the same users. (2) Find the sources of the attacks and thereby the infected systems and users (they might use different login information or spoofed addresses). (3) Discover the initial security breach by identifying the initially infected component and determine the way the malware entered the network. (4) React as fast as possible (e.g., shut down accounts or lock out devices) to prevent further harm, like successful brute-force attempts. (5) Return the network as well as its components and users towards a productive state after the problem has been handled appropriately.

1.2 Requirements

A tool which helps the security administrator perform tasks as in Sect. 1.1 has to fulfill three main requirements, each of which with suitable visual support.

(I) *Real-time Monitoring.* Due to the highly dynamic nature of networks, time is a crucial factor when analyzing the network state. Fast reactions can prevent other systems from getting corrupted or valuable information from getting stolen. Furthermore, analysis results may only be accurate for a certain time frame, and it is crucial to acquire knowledge before it is outdated.

(II) *Data Integration.* The different network components generally perform their respective tasks without communicating with each other or sharing their available information, most of them using proprietary data models. To overcome these isolated views, an integration of the different parts of information is fundamental. Nevertheless, the source of each datum should still be retained, in order to correlate achieved knowledge on a semantically higher level with the triggering low level events.

(III) *Retrospective Analysis.* In order to understand the current state of a network it is necessary not only to perceive and detect current situations, but

to retrace the events which have led to the current state. Hence, tracking the changes within the network is unavoidable and certain access mechanisms offer different advantages in the analysis. A playback of the events between two states enables an easier understanding of the transition from a desired towards an undesired state of a device or network, whereas replication of a single previous state can be used to review and restore an older configuration.

1.3 Contribution and Outline

In this paper we show how the Interface for Metadata Access Points (IF-MAP), a graph-based framework to manage metadata from multiple data sources in an integrated fashion, can be used as a foundation to collect information from different services, infrastructure components and endpoints for integrated visualization. While graphs provide natural representations of network topologies, IF-MAP also uses graphs to model relations between logical network components, like device addresses or user identities, and accompanying metadata, like capabilities, authentication information, or security-related events. Accordingly, the visualization of MAP graphs described in this paper allows a deeper view on the network state and enables correlation of security-relevant information on different semantic levels.

The remainder of this paper is organized as follows. Section 2 provides an overview of related work on the visualization of security information from multiple data sources. In Sects. 3 to 5 solution approaches for the requirements stated above are developed. Section 6 introduces the VisITMeta application, which implements the proposed solutions. Section 7 shows the benefits of the proposed systems and the challenges that were recognized. Section 8 gives a conclusion and an overview of future work.

2 Related Work

OSSIM[1] (Open Source Security Information Management, commercial version: USM) is an open-source Security information and event management (SIEM) system which integrates a collection of other well known open source security systems (e.g., OSSEC, OpenVAS, Snort and Nagios) into one unified system. OSSIM collects information from these systems in a central place for analysis purposes. The analysis results and the raw data are available for oversight by human security experts via a web interface. In addition OSSIM employs pattern and anomaly detection techniques as well as a general model for event correlation, which operates based on event streams or using heuristic algorithms [7].

VIAssist[2] is a commercial visual analysis platform which collects network flow and security data from different sources (e.g., Netflow, IDS) and uses a

[1] http://www.alienvault.com/open-threat-exchange/projects.
[2] http://securedecisions.com/products/viassist/.

dashboard with various visualization options [6]. The approach of VIAssist is to offer parallel views of the same or related data in order to facilitate threat detection by visual correlation. In contrast to our approach, both OSSIM and VIAssist use dashboards to visualize security information of different sources and analysis results, but no integrated graphical model. We consider such a model to be extremely important to gain deeper insight into more complex security threats.

Prelude OSS[3] (commercial version: Prelude Pro) is an open-source SIEM system. Prelude collects security-related information from various sources like network sensors, network infrastructure, service end-points and other security systems. Prelude is able to read alarm information in different formats (e.g. flat log files or syslog) and normalize this information to the Intrusion Detection Message Exchange Format (IDMEF) [5]. This normalized information is then stored in a central database and available for third-party software or via the Prelude web interface called *Prewikka*. Prelude includes a rule-based correlation engine called *Prelude-Correlator* which is able to detect security incidents and raise new IDMEF alerts [14]. In contrast to our approach Prelude has a focus on alert information, whereas we suggest to collect all relevant information, which includes alarm information but extends the set of relevant information to infrastructure data (e.g., location and configuration of devices) as well as high-level information like user login patterns.

In [10] a graph-based visualization of data stored in a SIEM system is described. While attack graphs are also used to visualize security incidents, the prototype in the paper neither uses an extensible graph model such as IF-MAP as foundation to store relevant data. Also, it does not provide the historical and progression views of the metadata graph as our approach does.

In summary, the described systems use dashboard-style visualizations of security data from different network components, while we propose an integrated, graph-based visualization. Furthermore we focus on the visualization of network history and temporal development, which none of these systems allows.

TVi [4] and ENAVis [8,9] are approaches to visualize network dynamics with regard to network security. TVi displays time lines of entropies of certain network features. ENAVis is a visual analytics tool that constructs different graphs from network data (e.g., host-host, user-user, host-user, or user-application interactions) and visualizes them using different layouts (force-directed, bipartite). ENAVis allows to find clusters and study similarities between network states at different times by means of visual analysis. While TVi visualizes the temporal evolution of network statistics (entropies, histograms), ENAVis uses an approach similar to ours by modeling different relations within a network as graphs (not restricted to device connections or packet flow) and allowing to review network changes by graph comparison. Our approach goes one step further by integrating different types of relations into a single graph, using metadata to differentiate between relation types. For the visualization of network history we use a

[3] https://www.prelude-ids.org/.

different approach based on graph deltas, i.e., sets of newly created and deleted nodes and edges within a selectable time interval.

In [1] we proposed a mechanism for the persistence of MAP data and the computation of graph deltas. A brief overview is given in Sect. 4 below. In the present paper we put this persistence mechanism in a wider context and focus on the integration of multiple data sources as well as visualization of MAP graphs.

3 Integration of Data Sources

As explained in the introduction, only the integration of multiple information sources—such as security-critical events, device configurations, and network topology—can provide a comprehensive view of the overall network state. In order to achieve a common knowledge base, a homogenization of these data sources is required.

Obviously, physical and logical network topology are of great interest for security analysis. Furthermore, configuration, state, and behavioral data of each participant in the network are essential to make a statement about the role of the devices and whether they behave in a desired manner. The collection of this information is not necessarily limited to end-user devices, like desktop computers or smartphones, but includes infrastructural components such as Network Access Control (NAC) systems, switches, or routers. A major challenge is posed by the heterogeneous and distributed nature of the data and its producers. Hence, it is important to combine all data into a common representation by performing homogenization as early as possible.

We suggest using IF-MAP as a protocol since it matches the given requirements for data integration (compare Sect. 1.2) by offering a standardized and extensible data model and a centralized communication model, which conserves the publisher of each information. By using a standardized data model, an implicit homogenization of data is already given and the process of homogenization is sourced out to a data-gathering client. Furthermore several of those clients for data acquisition already exist. In the following, we will provide a basic introduction to IF-MAP and describe its application in our architecture.

3.1 IF-MAP

IF-MAP is an open standard published as part of the Trusted Network Connect (TNC) framework by the Trusted Computing Group (TCG). It defines an XML-based client-server protocol for the exchange of *metadata* between many Metadata Access Point (MAP) clients and a central MAP server. The main specification defines a core data model with basic operations MAP clients and MAP servers have to support and their encapsulation within SOAP [12]. Metadata in IF-MAP are defined in an extensible way, offering the possibility to use IF-MAP for arbitrary domains. Additional specifications describe metadata for, e.g., the domains of network security or security in industrial control systems.

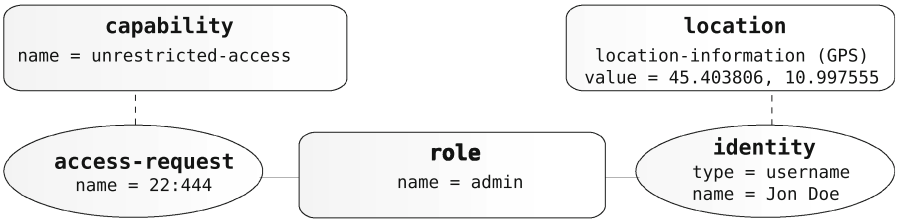

Fig. 1. Example MAP graph: ellipses represent identifiers, rectangles represent metadata, both of which are modeled as different types of nodes. Edges represent relations between identifiers and metadata. In this representation, links are implicitly shown via their attributed metadata connecting two identifier nodes, such as *role*.

IF-MAP Data Model. The data model of IF-MAP consists of an undirected graph that allows cycles and loops. There are three basic data types in IF-MAP: (1) Entities in the domain are described as *identifiers* represented by nodes in the graph, (2) Relations between these entities are called *links* and are represented by edges, (3) Any additional information for an entity or a relation is described as *metadata* and can be attached to both identifiers or links. Identifiers and metadata have a *type* definition, such as `identity` or `location`. Furthermore they potentially have different *attributes* like `name` or `value`. Metadata additionally have a mandatory *cardinality*, which expresses whether one type of metadata can be attached exactly once (`singleValue`) or multiple times (`multiValue`) to the same identifier or link. Figure 1 depicts a simplified example graph using some of the standard elements for network security [13]. Management information like the cardinality, timestamp, or publisher-id are omitted for reasons of clarity.

IF-MAP Communication Model. The communication model of IF-MAP is a content-based publish-subscribe model. Both publishers and subscribers are MAP clients that exchange information with a MAP server. To attach data to the graph structure in the MAP server, clients use the *publish update* operation providing either an identifier and the metadata that is to be attached to it or two identifiers and the metadata to be attached to the link between them. Clients can also delete data from the graph using the *publish delete* operation and similar parameters. Additionally, metadata can be propagated using the *notify* mechanism. Metadata propagated by this mechanism are not attached permanently to the graph structure but only forwarded to the clients holding a subscription that matches the identifier involved. Subscriptions themselves are carried out in an asynchronous fashion using a poll and notification scheme.

3.2 System Architecture

Figure 2 depicts the system architecture of our proposal for an IF-MAP based data integration and visualization. At the bottom, several examples for MAP clients are given: The Interconnected-asset Ontology (IO) framework provides

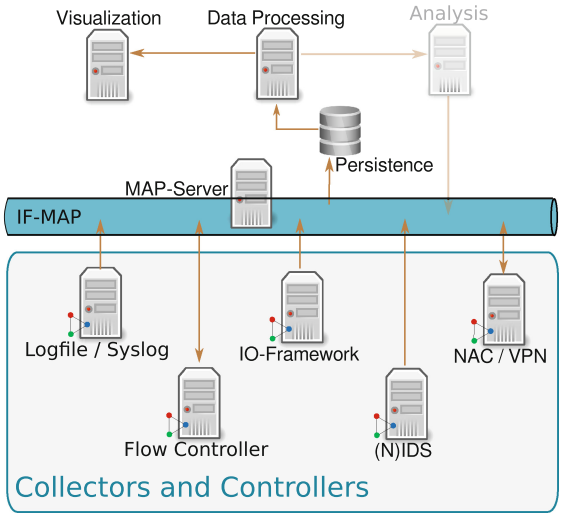

Fig. 2. System architecture overview - arrows depict the data flow

methods to extract the current state of an IT infrastructure [3]. A modified syslog server transforms syslog messages to corresponding IF-MAP metadata, e.g., failed login attempts at an SSH server. Network access mechanisms can publish their knowledge about connecting users and accordingly assigned addresses and accounts. An IDS monitors the network state and communicates incident reporting using IF-MAP. Finally, this information can be used by a flow controller or the network access components to react on specific situations, for example by shutting down a compromised user or device. After the first homogenization, which is implicitly performed by the clients when generating IF-MAP data, the information are gathered at the central MAP server. The MAP server only preserves the current state of the network, while historical information or temporal progress are not recorded. Therefore, a persistence component is needed to gain insight in progression of network state. An analysis component like an anomaly detection engine as in [2] can use the historical information to make policy-based decisions if an network threatening event exists and can propagate this by publishing new metadata. A visualization component renders the IF-MAP graphs to enable a better understanding of the network states.

3.3 Application Level Data Model

With the application of IF-MAP a first homogenization is implemented and a first mutual data model is given. Nevertheless, requirement III in Sect. 1.2 shows that the progression of the graph over time has to be taken into account for an appropriate data analysis. Since IF-MAP is designed to only cover the current situation and no historical information, we utilize a slightly varying application level data model to enable a persistent storage and restoration

of past graphs' states. A simplified Unified Modeling Language (UML) representation of this data model is depicted in Fig. 3. The IF-MAP data types (`Link`, `Identifier`, `Metadata`) are merely extended with a method yielding a boolean value whether the datum is valid with regard to a particular timestamp (`isValidAt()`). Another deviation from the IF-MAP model is the introduction of a dedicated data structure for the representation of a pair of identifiers (`IdentifierPair`) for implementation reasons.

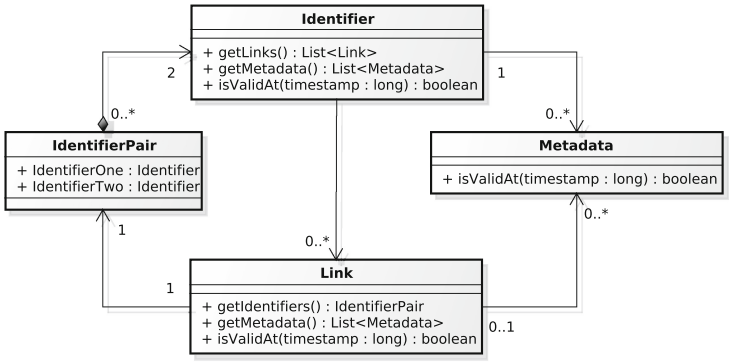

Fig. 3. Application level data model.

4 Persistence Layer Requirements

The requirements for a retrospective analysis also lead to certain consequences for the persistence layer. For example, in order to support *playback* of events, the persistence layer has to provide a query method which yields the graph history step-by-step. In addition, a query mechanism which returns complete snapshots of the graph at any given time is needed for the *replication* part.

In short, the following specific requirements for the graph persistence layer can be directly derived from the need for different forms of retrospective analysis:

(1) *Continuous Recording.* All information related to changes in the graph must be preserved. A snapshot approach which only saves the graph state at constant points in time (and thus aggregates different changes) is not desired.
(2) *State Queries.* It must be possible to query snapshots of a graph at any given time in the history of the graph. The result of such a query must contain the complete graph at that time.
(3) *Change Queries.* It must be possible to query the differences of the graph between two arbitrary points in the graph history.
(4) *Scalability.* Because of the high frequency of changes in a modern network it is necessary that the persistence layer can handle massive amounts of read and write operations. Therefore a solution which can be distributed over multiple hosts and easily scales horizontally is desirable.

To satisfy these demands, we implemented the data acquisition for the persistence as a subscriber to the MAP server, which thereby gets updated on demand and not in fixed intervals. Furthermore we developed algorithms to reconstruct a graph at a given time and to calculate changes between graphs. We chose to model the changes that the graph underwent as an explicit part of the query result. This relieves clients from calculating the actual change between two graph snapshots and directly supports the implementation of such features as "change highlighting". Moreover, this abstract interface can be used as a foundation for the step-by-step query method and can also be used as a building block for other kinds of change-related query or visualization features. For a detailed description of the construction of a graph valid at a given time and how we define and calculate the changes between two points in time (*graph deltas*) see [1].

5 Visualization Concepts

This section introduces the concepts to visualize homogenized data from multiple sources, based on the model described in the previous sections and the requirements stated in Sect. 1.2. Visualizing homogenized and historical information gathered from different data sources allows users to gain insight into the network state and the relationships of logical and physical components. Thus some requirements on the visualization itself have to be fulfilled.

(1) *Representing the Data Model.* The data model as described in different degrees of abstraction in the previous sections forms a graph model with nodes, edges, and metadata attached to both nodes and edges. A visualization of a concrete occurrence of such data must include the entities and their relationships in a distinguishable manner, so that a user can easily separate between them.
(2) *Publisher Distinction.* The possibility to trace the origin of the data is important for the user. As the data itself is homogenized by transmitting it via IF-MAP and arbitrary MAP clients can publish data of the same type, the source of data can only be determined by using the information about the original data source, i.e. the publisher-id of the corresponding MAP client. This information can be used to alter the appearance of the visualized data. It allows to focus on data from a single data source, or to inspect how data from different sources creates a more detailed overview when combined in a single graph, showing relationships that are invisible if the original data is viewed independently.
(3) *History Navigation.* The architecture to persist the history of gathered data and the possibility to retrieve the current state at a given timestamp as well as querying graph deltas between two timestamps requires a proper functionality to navigate in time and to select the mode of data retrieval.
(4) *Clustering of Data.* The integration of formerly independent data sources most likely shows unknown dependencies and relationships between the data. To emphasize them to the user, data should be grouped and arranged in a way that utilizes the semantics of the data, e.g., subgraphs that represent all information about one specific endpoint.

5.1 Graph Visualization

By matching the data model itself general mechanisms to visualize graphs can be used. Identifiers and metadata can be rendered as arbitrary shapes whereas the relationships between identifiers can be rendered as straight lines or curves. The layout of the overall graph and its subgraphs should allow an easy understanding of connections and give a quick glance of the global network state.

5.2 Configurable Color Schemes

One possible way to distinguish data gathered by different data sources in the visualization is to use the information about the publisher of the data itself. By assigning a color value to each publisher—automatically or user defined—the origin of data can be easily determined by the user. Figure 4 shows an MAP graph with colorization according to the publisher ID of the metadata.

In some cases this will lead to equally colored clusters of data, representing data of a single data source that possibly describes a very specific type of information about the network. The total number of different colors shows the amount of data sources involved and therefore visualizes the significance of the data base itself, where fewer data sources could mean a less reliable overview.

All possible IF-MAP identifiers exist at all times, but are seen as valid only when metadata—published by a MAP client—are attached to them. Thus for visualization purposes they can either be colored identically (to distinguish them by color from metadata) or according to their types (e. g. ip-address, mac-address).

5.3 History Navigation

To navigate data in time, the user can switch between three tabs within the graphical user interface: the first one called *live view* always shows the current state of the graph, as stored in the persistence layer. The second tab, *history view*, allows the user to select one of all recorded timestamps including the newest one. A third tab called *delta view* (as shown in Fig. 4) can be used to display the differences in the graph data between two timestamps, as depicted in Sect. 4. Here the user can select both the start and end timestamp for the delta calculation. The graph delta for these two timestamps is retrieved and visualized, with updates and deletes colored or highlighted differently.

Furthermore, the selected interval between start and end timestamp can be moved forward and backward. This is especially useful when selecting a start and end timestamp with an interval of exactly one timestamp, thus showing the changes between two consecutive timestamps; when moving this interval, the user can easily recognize the changes in the graph over time.

The mechanisms to select single timestamps can be implemented in various ways. The simplest way can be a single set of buttons to move forward or backward in time, jumping from one recorded timestamp to the next (or previous).

A timeline with a movable knob can be used to let the user move quickly along time and visualize the instant to which he navigates.

To enhance the selection, the user can be allowed to select a coarse time interval by using input fields or drop-down lists, showing only timestamps at which changes occurred within the persisted data, and then using the timeline to navigate only in the timestamps of that chosen interval. This reduces the amount of snap-in points on the timeline and therefore facilitates navigation.

5.4 Search Functionality and Filtering

To further reduce the amount of data and to only visualize the data relevant to the user, a search functionality as well as filtering is needed. This allows the user to pinpoint a single node, i.e. an identifier or a metadatum, or a selection of nodes with similar features.

The results of the search can be shown by simply highlighting or coloring the matching nodes in a different way than all non-matching nodes. When rendering all non-matching nodes in a lucent way, the user can both easily detect the matching nodes as well as retain the overall graph structure in view.

6 VisITMeta Application

This section introduces the open-source software application VisITMeta (*Visualizing the Security of Modern IT Environments Using Metadata*),[4] which is developed within a research project of the same name with the aim of implementing the concepts described in the previous sections.

6.1 Architecture

The software consists of two different applications. A component called *dataservice* is working as a MAP client and is thus responsible for retrieving information from the MAP server. The data is then persisted in a Neo4j database, using the internal Neo4j data model adjusted to the requirements of storing historical IF-MAP graph data. A REST-like web service interface allows to query the *dataservice* application for the graph at different timestamps, including the current state as well as a list of all timestamps that include changes (updates and deletes), and for graph deltas between two timestamps. The *dataservice* application resembles both the persistence and data processing components of the architecture shown in Fig. 2. The second component called *visualization* uses this interface to fetch graph data and visualizes it by employing the concepts of Sect. 5. It features a graphical user interface that enables the user to navigate through a MAP graph and its history.

[4] https://github.com/trustathsh/visitmeta.

6.2 Implemented Features

The VisITMeta application provides the following features to the user:

(1) *General Visualization and Navigation.* The graph layout is performed by the JUNG2 library[5], except for the bipartite layout. In Sect. 6.4 we show some specifics of IF-MAP graph layouting as well as details about the bipartite layout. The results are then rendered via Piccolo2D[6].
(2) *Color-coded Publisher Distinction.* The user can select colors for the originating IF-MAP publishers, which are then used to colorize the shapes used for rendering metadata nodes.
(3) *History Navigation.* Navigation in time is realized via multiple tabs as described in Sect. 5.3; it supports a live view that resembles the current state of the MAP server, as well as views on the history at a given time and graph deltas between two timestamps.
(4) *Highlighting Changes.* When changes in the data occur, updates and deletes are highlighted with colored halos to be recognizable by the user.
(5) *Search Functionality and Filtering.* Search functionality is given by using a textfield and a simple search language, combined with the ability to render non-matching elements in a lucent way.
(6) *Detailed Node Information.* The information of each identifier or metadata node is shown in a separated view on the bottom of the main graph screen. All attributes and elements of the XML data structure underneath each node can be seen, and in case of so called *extended identifier*, a mechanism of IF-MAP to define new identifier types, even the inner information can be seen in a structured way.
(7) *Motion Control.* The visualization application supports external control devices, e.g., the LeapMotion gesture controller.[7]

6.3 Example Illustration

Figure 4 shows the VisITMeta application providing a delta view of an excerpt of the earlier introduced example scenario introduced in Sect. 1.1. In the selected time interval, an external analysis component detected the (not shown) failed login attempts and created an event metadatum indicating a brute force attack. The PDP therefore enforces the disconnection of the aforementioned user, which results in the enforcement-report metadatum.

The green and red halos mark nodes that have been created and deleted, respectively, within the selected time interval, thus showing the creation of the event and enforcement-report and deletion of the subgraph representing the device information and user credentials of the now disconnected endpoint.

The user sees the information gathered by the three components (analysis component, PDP and DHCP server) and how they are related to each other.

[5] http://jung.sourceforge.net/.
[6] https://code.google.com/p/piccolo2d/.
[7] https://www.leapmotion.com/.

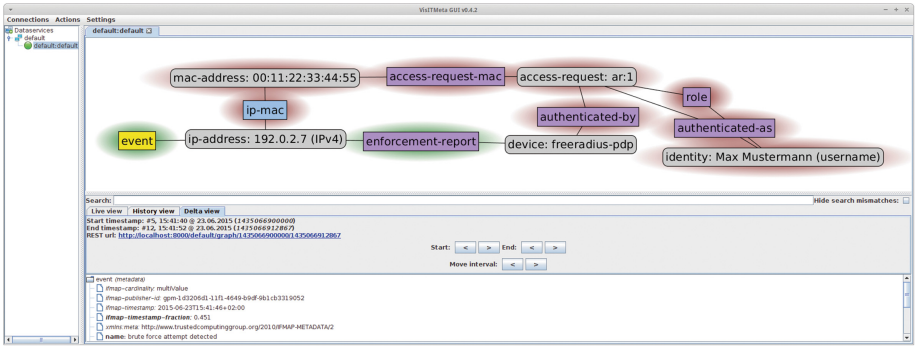

Fig. 4. VisITMeta application showing the example MAP graph with force-directed layout, MAP server connection tree (left), delta navigation (center), and metadata details (bottom). Identifier nodes are depicted by gray rounded rectangles, metadata nodes by rectangles with color coding for publisher distinction (blue: DHCP, violet: PDP, yellow: analysis component) (Color figure online).

Therefore—in this excerpt—the process of detecting malicious behaviour, reacting to it and the result of the reaction can be seen.

The other information are omitted for visibility reasons. VisITMeta could further be used to navigate through the history and track down the user which first started brute forcing (since multiple users might have been affected due to the malware spreading).

The IF-MAP data has been created by the irondemo test suite.[8]

6.4 Specifics of IF-MAP Graph Layouting

A MAP graph in the representation of Fig. 1 consists of two different types of nodes, i.e. nodes for identifiers and metadata. Furthermore, identifier nodes are only connected to metadata nodes, and vice versa. This distinction between graph nodes can be utilized when calculating layouts for the graphical representation of MAP graphs in such a way, that the layout can emphasize their different meaning in a MAP graph. Common layout algorithms could be adjusted to create good distributions of identifier nodes, while clearly positioning metadata nodes belonging to an identifier node near it.

Bipartite Layout. The two distinct types of nodes for identifiers and metadata can be considered as a bipartite graph. VisITMeta makes use of this property by offering a structured—and potentially clearer—graph layout, hereafter named *bipartite layout*. Regarding the specific nature of MAP graphs, i.e., having metadata nodes either connecting two identifier nodes via links or being attributed to single identifier nodes, a variation of the classical two-column (or two-row) layout for bipartite graphs is applied.

[8] https://github.com/trustathsh/irondemo.

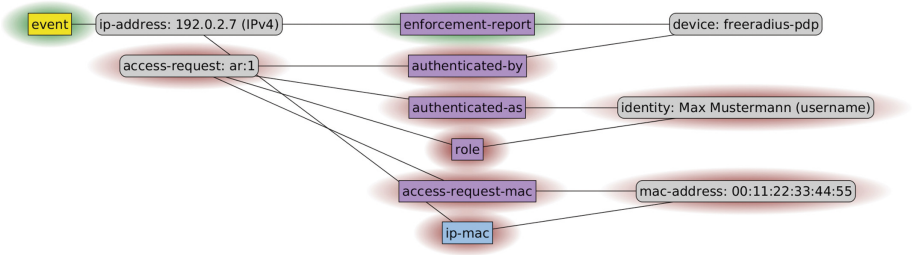

Fig. 5. Bipartite layout applied to the MAP graph of Fig. 4.

It consists of five columns, numbered 1 to 5 for the following explanation. Columns 2 and 4 are designated for identifier nodes. Column 3 is designated for metadata nodes attached to links, thus connecting two identifier nodes.
Columns 1 and 5 are designated for metadata nodes attached to single identifiers that are drawn in column 2 and 4, respectively. Starting from the first identifier node of the current MAP graph, the graph is traversed in depth-first order. Identifier and metadata nodes are drawn from top to bottom in their respective columns, switching from column 2 to 4 (and vice versa) with each metadata node attached to a link (i.e., connecting two identifier nodes).

The example MAP graph from 6.3 with bipartite layout instead of a force-directed layout is shown in Fig. 5. In this case the bipartite layout can be used to recognize the update and delete operations more easily, as they only affect metadata in IF-MAP and the bipartite layout shows them in distinct columns.

7 Discussion of Results

In this section we show some of the benefits our proposed concept of an integrated visualization with a continuous recording might bring the user. We also present some of the challenges we identified during implementing our concept or by findings of using the prototype itself.

7.1 Benefits of an Integrated Visualization with Continuous Recording

(1) *Homogenization.* The graph-based data model of IF-MAP and its acquisition of data helps to homogenize the data of arbitrary and different network components. The components must therefore only be extended to publish their information to the central MAP server (MAPS).
(2) *Data Context.* Using IF-MAPs *links* to connect information like network addresses, user credentials and services as well as (automatically) detected high-level events provides a benefit to the user, as this information has not to be connected manually from separate views in a graphical software or even different analysis tools.

(3) *Interoperability.* Another benefit by using IF-MAP as the underlying technique is the ability to use the proposed software tool VisITMeta in any network that facilitates an IF-MAP environment.
(4) *Continuous Recording and Retrospective Analysis.* By collecting all information and persisting them as they are processed by the MAPS, a retrospective analysis can be done, where even the differences in the graph between two given timestamps can be inspected. If something happened in the network, the changes in the state of all involved components can be reconstructed step by step in the same order as these changes were registered by the MAPS.

7.2 Identified Challenges

(1) *Visual Scalability.* Big graphs get cluttered really quick and loose the possible insight gain for the user. Techniques to reduce the size of the graph data shown to the user at a time have to be added to the visualization. Mechanisms like a level of detail approach, where parts of the graph were collapsed into single abstract nodes when zooming out would help retain the overall view on the graph.
(2) *Visual Dynamics.* Networks characterized by fast and frequent changes also lead to many changes in the visualization. This makes it difficult for the user to track single changes in a live view of the network. Besides from techniques on the visualization side to reduce the visual changes, an approach to reduce the incoming data could also be useful. By removing low-level data from the graph and relying on high-level data of detection components that abstract the low-level data could also lead to fewer but more meaningful changes in the visualized data.
(3) *Recording of All Data.* The mechanism to fetch the data from the MAPS has a basic shortcoming, as only connected graphs can be observed via the subscribe operation. Multiple subscriptions for every disconnected graph minimize the problem, but to get to know if there are (new) disconnected graphs not observed by running subscriptions is not supported via standard means by a MAPS.

8 Conclusion and Future Work

In this paper we have argued that an integrated and extensible monitoring system that gathers information from various network components is required to quickly and accurately detect and analyze security-related events. In the visualization of such integrated data, historical analysis and a suitable representation are needed to support a deeper understanding and invoke necessary measures. Thus we derived different requirements for a data model, its persistence and specifically its visualization. We presented IF-MAP as a suitable approach to integrate data from different heterogeneous data sources and extended its data model for the persistence of MAP graphs. Furthermore, we defined requirements regarding the visualization of such integrated data. Based on these requirements

we have presented concepts for an advanced visualization component based on IF-MAP. Our prototypical implementation of these concepts, the software application VisITMeta, targets the requirements on two different levels, operating as a data repository and providing a useful visualization with the possibility for further extensions.

Future work in terms of extending the VisITMeta software include the implementation of multi-layouts allowing different layouts for subgraphs—e.g., by combining a force-directed algorithm for the overall layout with the bipartite layout for subgraphs—as well as mechanisms within the GUI to filter and search the displayed graphs with more options. Another field of work is to handle extremely large graphs in their visualization by abstracting groups of graph elements into single meta-nodes that can be expanded on demand by the user (generalization).

Beside the extension of VisITMeta as a tool itself, we are also looking towards its integration into a bigger environment. Although VisITMeta already allows for live and historical analysis of MAP data, it gets difficult to manually analyze and track the network state. Especially in larger scenarios the full graph becomes very complex and confusing. Time-critical decision making and the detection of complex scenarios therefore require an automated analysis of the integrated data. We pursue such an approach in another current research project[9]. Both aspects of the VisiITMeta application are continued to be used: the data persistence with its interface as input to a graph-pattern-based detection component as well as the visualization as an integrated tool in a SIEM-like GUI. The VisITMeta visualization can be utilized to take a detailed look at the graph state when incidents in the network were monitored. This allows for a focused analysis to gain a deeper understanding of smaller parts of the network (graph) and also allow for a retrospective auditing.

Acknowledgements. The fruitful collaboration with Gabi Dreo Rodosek, Josef von Helden, Frauke Sprengel, and our students is gratefully acknowledged. This work is financially supported by the German Federal Ministry of Education and Research (BMBF) within the projects VisITMeta (grant no. 17PNT032) and SIMU (grant no. 16KIS0045).

References

1. Ahlers, V., Heine, F., Hellmann, B., Kleiner, C., Renners, L., Rossow, T., Steuerwald, R.: Replicable security monitoring: Visualizing time-variant graphs of network metadata. In: CEUR Workshop Proceedings of the Joint Proceedings of the Fourth International Workshop on Euler Diagrams and the First International Workshop on Graph Visualization in Practice co-located with Diagrams, vol. 1244, pp. 32–41 (2014)
2. Bente, I., Hellmann, B., Vieweg, J., von Helden, J., Dreo, G.: TCADS: trustworthy, context-related anomaly detection for smartphones. In: Barolli, L., Taniar, D., Enokido, T., Rahayu, J.W., Takizawa, M. (eds.) 15th International Conference on Network-Based Information Systems, NBiS, pp. 247–254. IEEE (2012)

[9] http://simu-project.de/.

3. Birkholz, H., Sieverdingbeck, I., Sohr, K., Bormann, C.: IO: an interconnected asset ontology in support of risk management processes. In: Proceedings of the Seventh International Conference on Availability, Reliability and Security, ARES 2012, pp. 534–541. IEEE (2012)
4. Boschetti, A., Salgarelli, L., Muelder, C., Ma, K.-L.: TVi: a visual querying system for network monitoring and anomaly detection. In: Proceedings of the 8th International Symposium on Visualization for Cyber Security, VizSec 2011, pp. 1–10. ACM (2011)
5. Debar, H., Curry, D., Feinstein, B.: The intrusion detection message exchange format (IDMEF), RFC 4765 (Experimental), March 2007
6. Goodall, J., Sowul, M.: VIAssist: visual analytics for cyber defense. In: Proceedings of the IEEE Conference on Technologies for Homeland Security, HST 2009, pp. 143–150. IEEE (2009)
7. Karg, D., Muñoz, J.D., Gil, D., Ospitia, F., González, S., Casal, J.: OSSIM: open source security information management, general system description, version 0.18, November 2003. http://www.alienvault.com/docs/OSSIM-desc-en.pdf
8. Liao, Q., Blaich, A., Striegel, A., Thain, D.: ENAVis: enterprise network activities visualization. In: Proceedings of the 22nd Large Installation System Administration Conference, LISA, pp. 59–74. USENIX Association (2008)
9. Liao, Q., Striegel, A., Chawla, N.: Visualizing graph dynamics and similarity for enterprise network security and management. In: Proceedings of the Seventh International Symposium on Visualization for Cyber Security, VizSec 2010, pp. 34–45. ACM (2010)
10. Novikova, E., Kotenko, I.: Analytical visualization techniques for security information and event management. In: 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, PDP, pp. 519–525. IEEE Computer Society Press (2013)
11. Tamassia, R., Palazzi, B., Papamanthou, C.: Graph drawing for security visualization. In: Tollis, I.G., Patrignani, M. (eds.) GD 2008. LNCS, vol. 5417, pp. 2–13. Springer, Heidelberg (2009)
12. Trusted Network Connect Working Group. TNC IF-MAP binding for SOAP, version 2.1, Revision 15, May 2012. http://www.trustedcomputinggroup.org/resources/tnc_ifmap_binding_for_soap_specification
13. Trusted Network Connect Working Group. TNC IF-MAP metadata for network security, version 1.1, Revision 8, May 2012. http://www.trustedcomputinggroup.org/resources/tnc_ifmap_metadata_for_network_security
14. Yasm, C.: Prelude as a hybrid IDS framework. Technical report, SAMS Institute (2009). http://www.sans.org/reading-room/whitepapers/awareness/prelude-hybrid-ids-framework-33048

http://www.springer.com/978-3-319-29967-9

Graphical Models for Security
Second International Workshop, GraMSec 2015,
Verona, Italy, July 13, 2015, Revised Selected Papers
Mauw, S.; Kordy, B.; Jajodia, S. (Eds.)
2016, IX, 103 p. 44 illus. in color., Softcover
ISBN: 978-3-319-29967-9