Replicable Security Monitoring: Visualizing Time-Variant Graphs of Network Metadata

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

University of Applied Sciences and Arts Hannover, Faculty IV,
Department of Computer Science, P. O. Box 920251, 30441 Hannover, Germany*

Abstract. Monitoring a computer network's security state is a difficult task as network components rarely share their information. The IF-MAP specification defines a client/server-based protocol that enables network components to share security information among each other, which is represented in a graph structure. Visualization of this data is challenging due to the highly dynamic topology and the mapping of logical nodes onto physical devices. Furthermore, data in a MAP server is volatile and there is no standardized way to preserve and review changes or previous states of a MAP graph. The evolution of such a graph, however, embodies valuable information for the analysis of past incidents and attacks on the network infrastructure. In this paper we introduce a software framework to visualize MAP data and propose a solution for the efficient long-term storage and replication of MAP graphs. We demonstrate how changes in the graph structure between given points in time can be computed and visualized.

1 Introduction

Within enterprise networks, many components like Intrusion Detection Systems (IDSs) or Flow Controllers monitor different aspects of the traffic or the behavior of the participants and are responsible for enforcing security-related decisions. In most cases, however, these components work independently, not sharing information with each other. For most of these separate components, different visualization approaches have been proposed, many of which employ graph drawing methods [6,9].

An aspect to consider in the visualization is that most computer networks are not static, e.g., with users logging in and out or devices being connected to and disconnected from the network. In recent years, the analysis and visualization of dynamic networks has attracted much interest, e.g., [4] and references therein. Since many real-world applications – including computer network security – are characterized by large-scale networks, efficient storage concepts for time-dependent network data are required.

The Interface for Metadata Access Points (IF-MAP) protocol allows to collect information from different services, infrastructure components and endpoints on

* Email: trust@f4-i.fh-hannover.de, WWW: http://trust.f4.hs-hannover.de/

a central Metadata Access Point (MAP) server in a time-variant graph data structure. IF-MAP therefore has the potential to provide a foundation for an integrated and comprehensive view on a network's overall state and security for both automated and human monitoring. As of now, data on the MAP server is volatile, i.e., only the present state of the graph is made available via IF-MAP.

The evolution of such a graph, however, clearly embodies valuable information, e.g., for data mining purposes since changes in the graph directly relate to changes in the network and the state of its security. In fact, changes in the graph themselves might be security-related incidents. An intuitive graphical representation of the changes of the MAP graph would greatly support a security officer in (a) assessing how the overall security of the network has developed over time or due to changes to the network infrastructure or services, (b) analyzing past incidents and therefore greatly improving the process of human network security monitoring.

In this paper, we introduce a system for the visualization of network security metadata driven by the following requirements:

Data dynamics: Due to frequent changes within a computer network, a continuous recalculation of the graph layout is necessary with the constraint that dramatic changes in the general visual representation should be avoided. Changes should furthermore be easily recognizable by the user.

Data semantics: As the data itself can have different semantics, this information can be used to improve the layout. Sub-graphs that feature a strong hierarchical structure should use different layout algorithms than sub-graphs with a seemingly random structure. Semantically cohesive sub-graphs should also be detected and displayed in generalized form to reduce the amount of graph elements to be presented to the user, as in a level of detail mechanism.

Data history: Both the present state as well as past states of the MAP server's data must be accessible through the Graphical User Interface (GUI). The user should also be able to obtain a graphical representation of the changes that occurred between two supplied points in time. Since data within a MAP server (MAPS) only represents the current state of the network, a proper storage mechanism has to be established.

The main focus of this paper thus is on the data history aspect. We propose a timestamp-based and storage-efficient model to persist MAP graphs and suggest algorithms to restore past graphs' states and calculate cumulative changes between two points in time. To do so, we use a combination of currently popular edge-centric and vertex-centric models. We show how the graph changes are visualized and how the user can interact with the graph history database.

The remainder of this paper is organized as follows: After reviewing related work in section 2, the technical background of these topics is outlined in section 3. Our concept for an efficient long-term storage of graph data and the corresponding algorithms are described in section 4. Visualization and GUI aspects of handling time-variant MAP graphs are discussed in section 5. Section 6 concludes this work with a summary of the findings and an outlook on future lines of research.

2 Related Work

IF-MAP, specifically applied to the security domain, has been a topic of recent and ongoing research. The secure integration of smartphones into corporate networks has been addressed by our approach called TCADS, which uses IF-MAP as the base protocol to share security-related information between various network entities [1,2].

For the visualization of integrated network security data, a commercial solution called IPSonar exists. It supports IF-MAP by being able to publish certain pieces of information to a MAP server[1]. IPSonar, however, does not rely on an openly specified protocol such as IF-MAP for data acquisition. Furthermore, IPSonar does not offer the visualization of graph changes.

To the best of our knowledge there is no solution providing the visualization of MAP data, especially regarding dynamic changes and historical development of data. A first effort to visualize the current state of a MAP graph has been made with the *irongui*[2] project, which can be understood as an initial exploration of the problem domain for our current work.

The analysis of time-variant systems is part of a vast amount of different areas of science. Different solutions have been proposed to address the complex task of modeling these time-variant systems as graph data structures.

Casteigts et al. introduced the concept of time-varying graphs [3]. They defined three ways to represent the dynamic history of a graph: the edge-centric evolution of a graph provides information about the presence of edges at a specific point in time, the vertex-centric evolution describes the same as the edge-centric view but for each vertex of the graph. The third view is described as a graph-centric evolution which represents each state of the graph as a static snapshot.

Ren et al. [8] defined so-called evolving graph sequences (EGS). An evolving graph sequence represents changes of the graph's structure as distinct snapshots. Each of the snapshots represents the graph's state at a given point in time. These ordered snapshots form an EGS, which represents all changes to the graph over time. Ren et al. also proposed a framework to query an EGS. Algorithmic examples include finding the shortest path between two vertices that ever existed in the history of the graph. Ren at al. further showed that their solution performs well for large datasets containing social network information. They address issues of large graph instances often associated with snapshot-based data models with a storage model that groups multiple graphs into a cluster, that can be compressed to fit into memory.

Holme and Saramäki [5] suggest to include the dynamic changes of a graph data structure directly in the data model, rather than defining a dynamic system which operates on discrete time-dependent instances of the graph. Such graphs are known as temporal networks, where each edge may be active for some timespan in the graph's overall history. This approach can be viewed as an

[1] `http://www.lumeta.com/solution/trusted_computing.html`
[2] `https://github.com/trustathsh/irongui`

interpretation of edge-centric models. The advantage of temporal networks over traditional dynamic systems arises especially if the typical operation or query in the problem domain focuses on temporal rather than pure topological features. Holme and Saramäki showed different problem domains where temporal networks can be useful such as models for the spread of diseases where nodes represent persons and edges the contact of two persons at some point in time.

Dutot et al. [7] introduced the Java-based GraphStream library, which can be used to develop models for different problem domains based on dynamic graphs. The basis of GraphStream is formed by an event stream model, where each event represents a change of the graph. They stated that the event stream model allows for efficient in-memory processing of big graphs, because there is no need to hold the complete graph in memory. Dutot et al. also developed a file format for the preservation of a graph's evolution. They have, however, not published any details about it.

With respect to our problem domain, the three different views proposed in [3] are not directly applicable. In fact, we choose to combine the edge-centric and vertex-centric views to be able to preserve changes associated with edges and vertices. We opt against the snapshot based graph-centric view to store only the minimal amount of information needed to represent a changing graph. The inclusion of temporal information directly into the graph data structure is very similar to our approach, with the difference that in our approach validity information is contained in the vertices rather than the edges. Also, the general research focus of the work discussed above lies on time aware query methods and the application of classical graph algorithms on time-variant graphs whereas our approach targets efficient long term storage of changing graph data structures and lightweight query methods which can be used to build more sophisticated queries to support various different use cases.

3 Technical Background

IF-MAP The term IF-MAP refers to a set of specifications published by the Trusted Computing Group (TCG) as part of the Trusted Network Connect (TNC) framework. IF-MAP defines an XML-based network protocol for exchanging so-called metadata among an arbitrary number of MAP clients via a central MAP server. The main motivating use case for IF-MAP is the distribution of security information within a network in a standardized and interoperable way. Since the specifications include a flexible extension mechanism, IF-MAP can be customized to virtually any use case – even beyond the classical network security domain.

The main specification document defines the core data model, the basic operations MAP clients and MAP servers must support and their encapsulation within SOAP [10]. Additional documents specify metadata for specific domains. As of now, there is a dedicated specification addressing metadata for the domain of network security [11] and one for security in industrial control systems [12].

Data Model The data model of IF-MAP is represented by an undirected graph which allows cycles and loops. There are three fundamental data types: (1) *identifiers*, which describe entities in the network, are represented by the nodes of the graph, (2) *links*, which describe relations between entities, are represented by the edges of the graph, (3) *metadata*, which describe additional information for an entity or a relation, can be attached to both identifiers and links.

There are different *types* of identifiers and metadata such as `identity` or `location`, each with potentially different `attributes`, e.g., `name` or `value`. Metadata types also have a certain *cardinality* expressing whether exactly one metadatum (`singleValue`) or an arbitrary number of metadata (`multiValue`) of the given type can be attached to a single identifier or link. An example graph using some of the standard identifiers and metadata is depicted in Fig. 1.

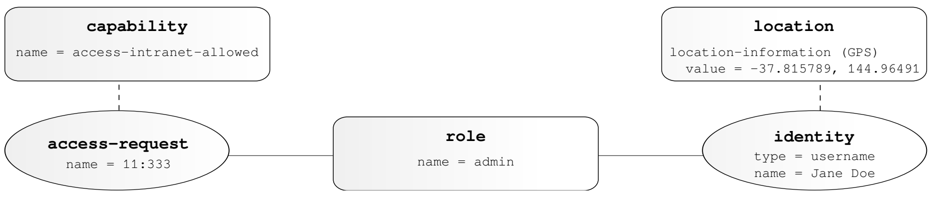

Fig. 1. Example MAP graph. Ellipsoids represent identifiers, rectangles metadata.

Communication Model The communication model of IF-MAP is a content-based publish-subscribe model. Both publisher and subscriber are MAP clients connected to a single MAP server. A publisher can insert new and update existing information (*publish update*) or delete data from the graph (*publish delete*). Subscriptions are handled asynchronously. The subscriber is notified whenever changes to the subscribed information occur. Metadata can also be propagated using the *notify* mechanism. Notify data is only sent to current subscribers and never added to the graph structure in the MAP server itself.

Furthermore, IF-MAP specifies a search functionality, that allows the MAP client to query and search for information with an immediate result. Searches follow the same pattern as subscriptions. They can range from simple queries for a specific metadatum, towards more complex patterns within the graph, e.g., only following specific links.

4 Concept for Change Tracking of IF-MAP Graphs

The following section describes our concept for change tracking of MAP graphs. This includes the extensions needed to store changes as well as the algorithm for restoring past graphs' states and the calculation of changes between two points in time.

Change Tracking Extension In the IF-MAP data model, only metadata is volatile, i.e., has a certain lifetime of its own. Identifiers are never created nor deleted, but (at least conceptually) always exist as globally unique entities. Links exist as relationships between two identifiers "indicated by metadata" [10]. From an application level perspective, of course, identifiers have a lifetime – hence, only identifiers that have valid links or metadata attached to them are considered to be valid (or existent for that matter).

Metadata instances are provided with two pieces of additional administrative information in order to examine their validity for a given point in time: (1) the IF-MAP publish timestamp to mark the start of their validity (2) a delete timestamp that marks the end of the metadata's validity.

A metadatum m is considered valid for time t if $t_{publish}(m) <= t < t_{delete}(m)$. The validity of links and identifiers is derived from the validity of metadata as follows: A link l with an arbitrary set of metadata M_l connected to it, is considered valid at time t if $\exists m : m \in M_l \wedge isValid(m, t)$. An identifier i with an arbitrary set of metadata M_i and an arbitrary set of links L_i connected to it, is valid at time t if $(\exists m : m \in M_i \wedge isValid(m, t)) \vee (\exists l : l \in L_i \wedge isValid(l, t))$. Using this understanding of validity, algorithm 1 can be used to restore a graph's past state using the corresponding timestamp and a (random) identifier as a starting point.

```
Algorithm: BuildGraph(Identifier currentId, List<Identifier> seenIds,
timestamp t)
if !seenIds.contains(currentId) and isIdentifierValid(currentId, t) then
    result.add(currentId);    /* Add identifier and connected metadata */
    seenIds.add(currentId);
    for Identifier nextId in all linked identifiers of currentId do
        if seenIds.contains(nextId) and isLinkValid(currentId, nextId, t) then
            result.addLink(currentId, nextId);    /* Add Link and connected
            Metadata */
        else
            BuildGraph(nextId, seenIds, t);
        end
    end
end
```

Algorithm 1: Graph construction for valid identifiers at time t.

Calculation of Graph Deltas The query for a graph delta takes two input parameters: t_s is the first point in time – "the starting point of the query" – and t_e is the second point in time – "the ending point of the query". The query returns a graph tuple $(U_{[s,e]}, D_{[s,e]})$ where the graph $U_{[s,e]}$ contains all identifiers and links which happen to have new or updated metadata attached to them. In this context new or updated means that the metadata attached to an identifier or link was not present (or had different attributes) at t_s but is valid at t_e. If some metadatum was valid at t_s but is no longer valid at t_e, the identifier or link attached to this metadatum will be included in the $D_{[s,e]}$ graph.

To calculate $U_{[s,e]}$ we take the current graph G_e which represents the state of the graph at time t_e and the graph G_s which is the graph state at time t_s.

We now check for each metadatum in G_e whether it is present in G_s, if we find such metadata we remove it from G_e. If this leaves an identifier or link without any metadata attached to it, we remove the identifier or link as well. We are left with the graph $U_{[s,e]}$ that only contains identifiers or links that have not been present at the start time of the query, but were present at the end time of the query. In order to calculate the graph $D_{[s,e]}$, we apply the same operation, but swap G_s and G_e: for each metadatum in G_s we check whether it is also present in G_e. If we find such a metadatum, we remove it from G_s. Identifiers and links without any metadata are dropped as described above.

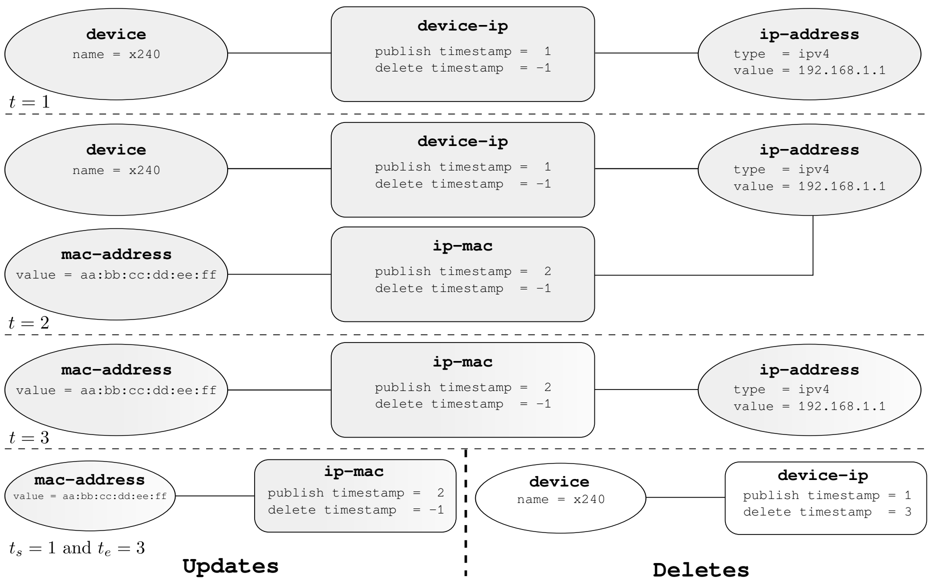

Fig. 2. Example delta calculation

An example for the calculation of deltas is depicted in Fig. 2. At $t = 1$ a `device-ip` metadatum is published, the negative delete timestamp indicates that this metadatum is still valid. At $t = 2$ an `ip-mac` metadatum is published to the already known `ip-address` and at $t = 3$ the `device-ip` that had been published at $t = 1$ is deleted. The query for the delta from $t_s = 1$ to $t_e = 3$ yields the tuple depicted at the bottom of Fig 2. $U_{[e,s]}$ is shown on the left, $D_{[e,s]}$ shown on the right hand side.

As figure 2 shows, this algorithm may yield a $U_{[s,e]}$ or $D_{[s,e]}$ that contains edges without identifiers or “half edges” with only one identifier present. In the implementation, one might choose to either use the algorithm as is and assign unique identities to links or to always return an identifier-link-identifier triple to identify links unambiguously, even if one or both of the identifiers have not changed.

5 Visualization Concept and System Architecture

In this section we show the concept and architecture of the VisITMeta software system that combines persistence of IF-MAP data and its visualization. The retrieval of and navigation within time variant data are also described.

Software architecture and data retrieval The system architecture of VisITMeta consists of two strongly separated applications. All application layers are designed as independent from each other as possible, so that libraries and algorithms can easily be exchanged.

The *dataservice* collects metadata from a MAP server (MAPS) as a regular MAP client (MAPC), stores it inside a Neo4j[3] graph database and also provides access to the stored metadata via a REST-like interface.

The *visualization* application fetches metadata from the dataservice via the REST interface and converts the data into graph elements. Layouts are generated with the JUNG2[4] library and the results are rendered with Piccolo2D[5]. A Java Swing GUI allows for navigating through the graph history and editing the underlying connections to one or multiple dataservices.

To retrieve data from the *dataservice*, the *visualization* application – and any other possible application – can use methods of the REST-like interface to request (a) a map of all timestamps at which changes occurred in the graph, (b) a delta by specifying a start and an end time, (c) the graph at a given timestamp which contains all valid identifiers, links, and metadata, or (d) the graph at the latest timestamp, i.e., the current state of the graph.

Graph and delta visualization Figure 3 shows a small example of a layouted MAP graph within VisITMeta's GUI application. Identifiers and metadata are both visualized as nodes, whereas links are only shown implicitly: a link exists between two identifiers which are connected by one or more metadata, such as *enforcement-report* or *device-attribute* plus *access-request-device*.

The user controls the time-variant view on the MAP graph by using a slider mechanism with two knobs that can be moved independently. This allows to select (a) a single point in time to get the graph associated to that timestamp by only moving the right knob or (b) a time interval for viewing a graph delta by moving both knobs. Alternatively, the current state of the graph can be displayed ("live view").

After selecting a time interval via the slider, the resulting delta is visualized as a graph that contains all updates and deletes. All updated and deleted metadata are highlighted in the corresponding user-defined colors. By selecting two succeeding timestamps, changes in the MAP data can directly be observed.

[3] `http://www.neo4j.org/`
[4] `http://jung.sourceforge.net/`
[5] `http://www.piccolo2d.org/`

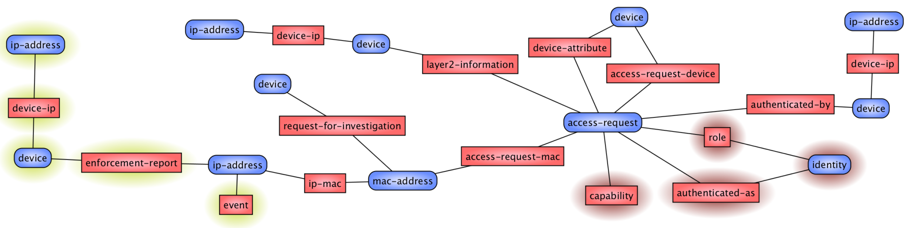

Fig. 3. A simple example of visualizing a MAP graph history. Identifiers are depicted by blue rounded rectangles, metadata entities by red rectangles. Links are only shown implicitly (cf. text). The green and red glow effects highlight identifiers and metadata that are created and deleted within the selected time interval, respectively.

6 Conclusion and Future Work

In this paper we have introduced a system for the visualization of highly dynamic network security metadata represented in a graph structure. An early prototype of this work is available via Github.[6] Our system is able to display both the current state and past states of the metadata graph as well as deltas between two given points in time. We have proposed a timestamp-based model to persist time-variant MAP graphs and developed algorithms to restore past graphs' states and calculate deltas. In contrast to other approaches related to time-variant graphs, our work primarily targets long-term storage and reproduction of graphs using a combination of edge-centric and vertex-centric methods with minimal storage overhead.

At the time of this writing our work is subject to the following limitations: (a) only metadata made available via IF-MAP publish update operations is made persistent within the database, data published via notify is discarded, (b) reconstruction of a graph's past state is only guaranteed to be idempotent, if the subscription used to receive data from the MAP server retrieves all data from the map server, i.e., each identifier and each metadatum in the MAP server can be reached using the subscription traversal at all times.

Visualization of dynamic, time-variant and potentially very large graphs still needs research, even more so in the area of IF-MAP where both edges and vertices can be tightly packed with information. Future work will have strong focus on new concepts for IF-MAP graph visualization. The next tasks will be to develop and implement multi-layout algorithms and find ways to detect and display semantically cohesive sub-graphs.

Apart from visualization, historic IF-MAP data is also useful for mining patterns that capture the network's behavior. These patterns could be used to detect outliers that might indicate incidents. We plan to build a data model that captures the graph history for this use case. We assume that this model will be different from the presented model as it targets different types of access.

[6] `https://github.com/trustathsh/visitmeta`

7 Acknowledgements

The fruitful collaboration with J. von Helden, T. Ehlers, J. Fuchs, B. Merieau, S. Misztal, and F. Sprengel is gratefully acknowledged. This work is financially supported by the German Federal Ministry of Education and Research (BMBF), projects VisITMeta (grant no. 17PNT032) and SIMU (grant no. 16KIS0045).

References

1. Bente, I., von Helden, J., Hellmann, B., Vieweg, J., Detken, K.O.: ESUKOM: Smartphone Security for Enterprise Networks. In: Pohlmann, N., Reimer, H., Schneider, W. (eds.) ISSE 2011, Securing Electronic Business Processes. pp. 371–382. Vieweg+Teubner, Wiesbaden (2011)
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 2012. pp. 247–254. IEEE (2012)
3. Casteigts, A., Flocchini, P., Quattrociocchi, W., Santoro, N.: Time-varying graphs and dynamic networks. In: Frey, H., Li, X., Ruehrup, S. (eds.) Ad-hoc, Mobile, and Wireless Networks, Lecture Notes in Computer Science, vol. 6811, pp. 346–359. Springer Berlin Heidelberg (2011)
4. Federico, P., Aigner, W., Miksch, S., Windhager, F., Zenk, L.: A visual analytics approach to dynamic social networks. In: Proceedings of the 11th International Conference on Knowledge Management and Knowledge Technologies (i-KNOW '11). pp. 47:1–47:8 (2011)
5. Holme, P., Saramäki, J.: Temporal networks. Physics Reports 519(3), 97–125 (2012)
6. Marty, R.: Applied Security Visualization. Addison-Wesley, Upper Saddle River, NJ (2008)
7. Pigné, Y., Dutot, A., Guinand, F., Olivier, D.: GraphStream: A Tool for bridging the gap between Complex Systems and Dynamic Graphs. In: Emergent Properties in Natural and Artificial Complex Systems. Satellite Conference within the 4th European Conference on Complex Systems (ECCS'2007). pp. 63–72 (2007)
8. Ren, C., Lo, E., Kao, B., Zhu, X., Cheng, R.: On querying historical evolving graph sequences. Proceedings of the VLDB Endowment 4(11), 726–737 (2011)
9. Tamassia, R., Palazzi, B., Papamanthou, C.: Graph drawing for security visualization. In: Tollis, I.G., Patrignani, M. (eds.) Graph Drawing, 16th International Symposium, GD 2008, LNCS, vol. 5417, pp. 2–13. Springer, Berlin (2009)
10. Trusted Network Connect Working Group: TNC IF-MAP Binding for SOAP, Version 2.1, Revision 15. http://www.trustedcomputinggroup.org/resources/tnc_ifmap_binding_for_soap_specification (May 2012)
11. Trusted Network Connect Working Group: TNC IF-MAP Metadata for Network Security, Version 1.1, Revision 8. http://www.trustedcomputinggroup.org/resources/tnc_ifmap_metadata_for_network_security (May 2012)
12. Trusted Network Connect Working Group: TNC IF-MAP Metadata for ICS Security, Version 1.0, Revision 44. http://www.trustedcomputinggroup.org/resources/tnc_ifmap_metadata_for_ics_security (May 2014)