IO: An interconnected asset ontology in support of risk management processes

Henk Birkholz
Universität Bremen TZI
Germany

Ingo Sieverdingbeck
Universität Bremen TZI
Germany

Karsten Sohr
Universität Bremen TZI
Germany

Carsten Bormann
Universität Bremen TZI
Germany

***Abstract*—Asset information obtained via infrastructure analysis is essential for developing and establishing risk management. However, information about assets acquired by existing infrastructure analysis processes is often incomplete or lacking in detail, especially concerning their interconnected topology. In this paper, we present the Interconnected-asset Ontology, IO, as a step towards a standardized representation of detailed asset information. The utilization of an asset ontology as a machine-readable representation supports the automation of risk management processes and the standardization of asset information reduces redundant acquisition processes that are often found in practice.**

***Keywords*—Risk Management, Asset Management, Onotology, Security, Automation, Network**

I. Introduction

IT infrastructure analysis is a crucial part of every risk management (RM) methodology [1], [2]. As a part of the German IT-Grundschutz methodology, for example, *structure analysis* provides the basis for additional procedures when creating an organization's security concept according to IT-Grundschutz [3]. Unfortunately, an infrastructure analysis is not only a complex and time-consuming task, but is often performed redundantly to already existing asset management (AM) processes. The typical focus of already implemented AM processes, e.g. cost center and warranty terms, can result in a lack of detail regarding information that is mandatory to introduce and support new RM processes. This necessitates further—a potential redundant or even manual—information acquisition. Consequently, the increased amount of information is often stored in separate databases because integration in existing databases that support AM processes is time consuming as well. While this solution is simple, it introduces additional redundancies that, over time, can result in discrepancies between databases.

To remedy this situation we propose IO—the Interconnected-asset Ontology—as a step towards a standardized format to store heterogeneous infrastructure information including interconnected relationships for application in the security domain. The contributions of IO and its corresponding framework are: efficient provision of a central knowledge base by utilizing the taxonomy of an ontology, eliminating the need for redundant acquisition procedures by defining a standardized and automated information acquisition, and a high level of detail from which more abstract subsets can be derived for specific applications.

An ontology [3], [4] was chosen as a structured representation because recent work shows that security-related ontologies can provide an appropriate, unified and formal knowledge model to support security RM processes [5]. Ontologies are also suitable for integrating information from heterogeneous sources [6], and it is well understood how to retrieve specialized views and infer context information from them [7], [8], [9]. Furthermore, modern approaches to security management, e.g. intelligent *Security Information and Event Management* (SIEM) systems, are already utilizing domain knowledge represented by ontologies (e.g. [10], [11]). Both risk- and asset management are time- and resource-consuming tasks, making the automation of these processes desirable [12]. Roughly a third of the controls defined in the information security standard ISO 27001 [13] can be automated [14]. As a part of asset management, the *Inventory of Assets* is one of these controls.

IO is currently deployed to collect asset information in a production network composed of approximately 400 centralized managed network components, 1500 decentralized managed network components, 650 WLAN access points, 3500 VoIP endpoints and about 10k miscellaneous endpoints. Raw information is automatically extracted from the centralized, managed production network components. Network scans (e.g., nmap) and other active probing mechanisms (e.g., nessus) can be subsequently used to further extend the level of information detail and to detect inconsistencies between known asset configuration and actual asset behavior in an interconnected network. Processed information includes static configuration (e.g., virtual-lan-id associations or access lists) and real-time information (e.g., neighborhood relationships or known MAC addresses). Until now, most of IO's basic asset concepts are network related. This proves to be an advantage, because almost every kind of asset information can be acquired over the network directly or indirectly. Most people employ networks to perform their work these days; this makes networks a good starting point to model the assets involved in their work.

IO began as part of the research project FIDeS [16] and its development adopted some of the FIDeS requirements in its initial stages. FIDeS employs a SIEM component as a central hub for processing security alerts produced by heterogeneous IT infrastructure, especially Intrusion Detection Systems (IDS). In order to effectively assess the potential

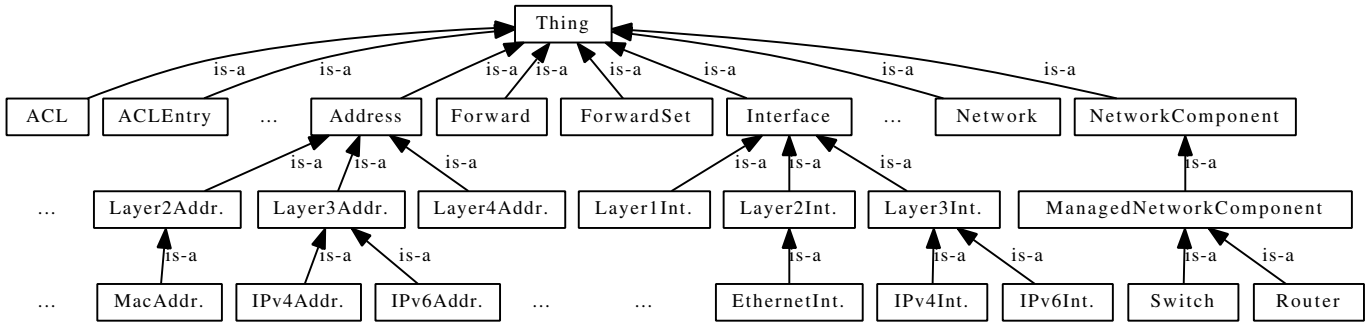

Fig. 1. Excerpt of IO's class taxonomy, a complete overview can be found at [15]

impact of e.g. a detected network-based attack on the infrastructure, context information has to be identified and retrieved. Examples for typical context information about assets are: placement of an asset in the interconnection topology of a network, annotated zones based on the subnetting topology, such as a Demilitarized Zone (DMZ), or layer-specific end-to-end reachability.

The remainder of the paper is organized as follows. Section II discusses related work, whereas Section III describes the ontology design of IO. In Section IV, we present the IO-Framework utilizing the ontological representation. The use cases described in Section V provide the basis for the evaluation in Section VI. Section VII concludes the paper and gives an overview of future work.

II. Related work

Ontologies: A variety of security ontologies have been proposed, which already include one or more *concepts* for IT assets (e.g. [17], [18], [19]). Each version of these asset concepts has a very specific and slightly different use in the corresponding ontological structured security domain knowledge. Concepts representing assets are often in the center of an object property chain, providing a link between other security concepts. Prominent examples are: linking actual vulnerabilities to threats, threats to impacts or even to security controls. There are cascading relationships which can be resolved by a single asset individual, e.g. a local security control, countering a vulnerabilty associated with an asset. There are others, however, that need asset information resulting from a topological interconnection of assets, e.g. an external threat, exploiting a vulnerability associated with an asset. Presenting their enhanced vulnerability ontology, Aime et al. discuss this integral need for information not only about assets, but also about their physical and logical interconnection topology [20]. They describe a top-level view of an asset ontology, which introduces two general communication concepts between assets (messages and connections). While these concepts are useful to represent interconnected relationships between individual assets, they are already quite abstract, considering the complexity of interconnected infrastructure in a real network. Various configuration and neighborhood information have to be acquired in detail before it is possible to create appropriate individuals matching these concepts. The IO framework (IO-F) in our approach has a strong focus on the automatic acquisition of asset information with a high level of detail. This level of detail is not only used to create individuals according to abstract concepts, but preserved in IO's ontological representation in equivalent detail.

Asset Information: Syalim et al. show that major risk analysis methods depend on the availability of asset information [21]. In their comparison of risk analysis methods, they also stress the fact that IT asset information is crucial for the assessment of potential threats and impacts (magnitude of harm) associated with a threats exercise of a vulnerable IT asset. Ekelhart et al. highlight the dependency on this detailed knowledge about the IT security domain, in a real production environment [22]. They propose the AURUM methodology, which implements a highly granular physical infrastructure model. Their focus includes physical infrastructures hierarchies including buildings, areas, rooms or doors and related target objects, similar to the ones defined in the German IT-Grundschutz standard [23]. Related to their AURUM methodology, Ekelhart et al. introduce an intuitive inventory solution to acquire IT asset information with a special focus on network endpoints [24]. Host Inventory regarding network endpoints is produced by host-based sensor agents and is then aggregated in an XML structure. Further details acquired by nmap, ping or ARPPing are merged into this information. This is also often encountered in security audit methods [25], [26]. The third kind of sources is potentially available software inventory solutions that provide externally maintained infrastructure information.

Generally, it is a good idea to include as many information sources as are available in order to enhance the level of detail of the current asset topology. Unfortunately, manually maintained pools of information tend to become outdated rapidly and may be incomplete, which results in a degradation of quality regarding security processes such as risk management [27]. Therefore, IO has a strong focus on updating the derived topology periodically to keep up with changes in the infrastructure.

Security Automation: One objective of acquiring detailed up-to-date asset information is to support the automation of

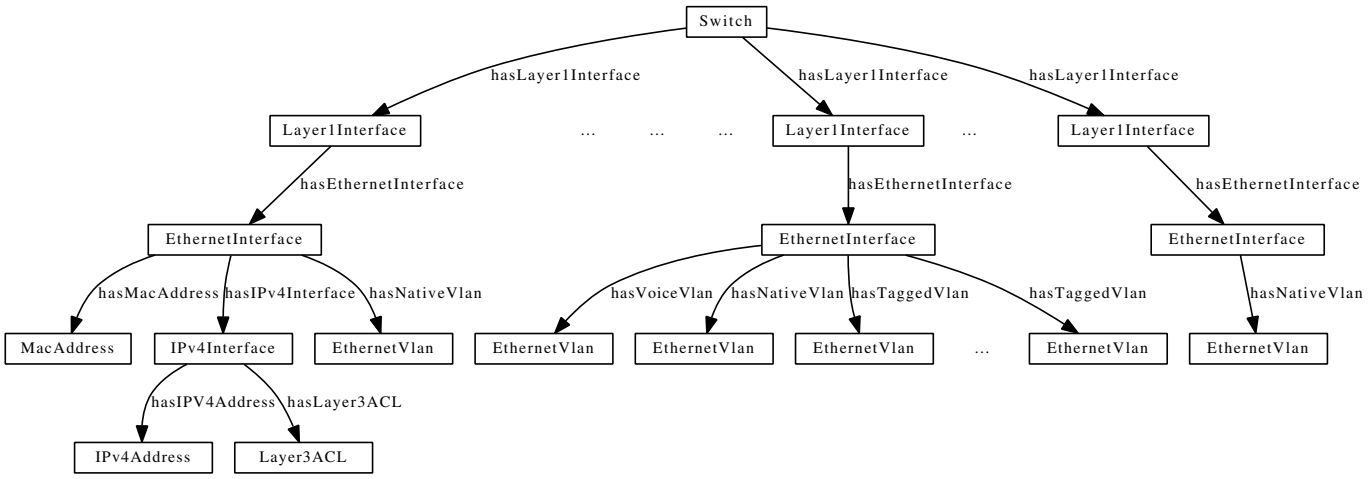

Fig. 2. Subset of asset relationships representing a switch

IT-related security processes [28]. The National Institute of Standards and Technology (NIST) in cooperation with the Massachusetts Institute of Technology Research Establishment (MITRE) provide the Security Content Automation Protocol (SCAP), a multipurpose approach which enables e.g. automated vulnerability checking employing security configuration checklists [29]. SCAP content benefits from multiple sources, among them the *National Vulnerability Database* (NVD) or the MITRE *Open Vulnerability and Assessment Language* (OVAL) Database [30]. SCAP also defines the *Asset Identification* (AI) specification [31]. Its primary use is better correlation, assessment and management of every asset-related piece of information (vulnerabilities, weaknesses, alerts, system events). The top-level design goals therefore are similar to IO's, but are basically driven by a communities' interest in better structuring known vulnerabilities. While this aligns with our long term goal, the description in SCAP AI is already abstract and generalized. This helps in matching potential CVE entries to assets, but does not leave detailed information to describe a topological interconnection of the assets itself. The development of IO is aiming at a continuous compatibility with the AI specification, so that AI-conforming output can be inferred and retrieved as one application of IO.

An approach for automatic management of network security policies was already introduced by Burns et al. in 2001 who presented a Prolog prototype [32]. This required a topology model; properties of such a model were summarized: formal and machine-readable, allowing automated reasoning, composability of devices and vendor-independence. Since the aim of our contribution is to support security automation with machine-readable and machine-interpretable knowledge, we adopted all of the properties introduced by Burns et al. into the IO framework.

Typically, evaluation of automation takes place in small networks [33] or randomly generated network topologies based on specific assumptions [34]. Evaluation in large networks is more difficult and rare, especially if manual annotation is necessary. An approach evaluated in a network even larger than ours is PRESTO [35]. The goal of this configuration management system is to ease the deployment of a configuration in large networks, by utilizing generalized *configlets* from which a specialized configuration can be derived. Both PRESTO and IO store detailed information, but PRESTO focuses on its deployment not its acquisition. IO's application aims at institutions where configuration management is not obligatory. In these cases, understanding the actual state of the network is often the first step towards implementing further measures such as configuration management.

Reachability: One primary application of IO in the context of SIEM systems is the inference of end-to-end reachability on layer 1 to layer 4. A similar goal is pursued by using model checking methods applied on aggregated configuration of managed network components. Al-Shaer et al. present an approach that models the global end-to-end behavior of the access control configurations of a network [34]. The network is represented as a state machine; packet headers and their location thereby define the state. By checking future and past states of the packets in the network, reachability can be verified. This is very fast but is restricted to layer 3 and higher. Noel and Jajodia present a method to pinpoint optimal placements for Intrusion Detection Systems (IDS) in a network [36]: A minimal number of IDS sensors have to cover every possible communication path (end-to-end reachability). The goal of their Topological Vulnerability Analysis (TVA) is to improve the assessment of IDS alerts by taking into account the actual network configuration, vulnerabilities, and thereby potential mission impact of a recognized attack signature. The TVA is based on a network-specific attack graph including asset information and is comparable to the attack graphs produced by NetSPA [37].

III. The Interconnected Asset Ontology IO

The basis of our approach is the Interconnected-Asset Ontology, IO. The ontology design is influenced by the Network

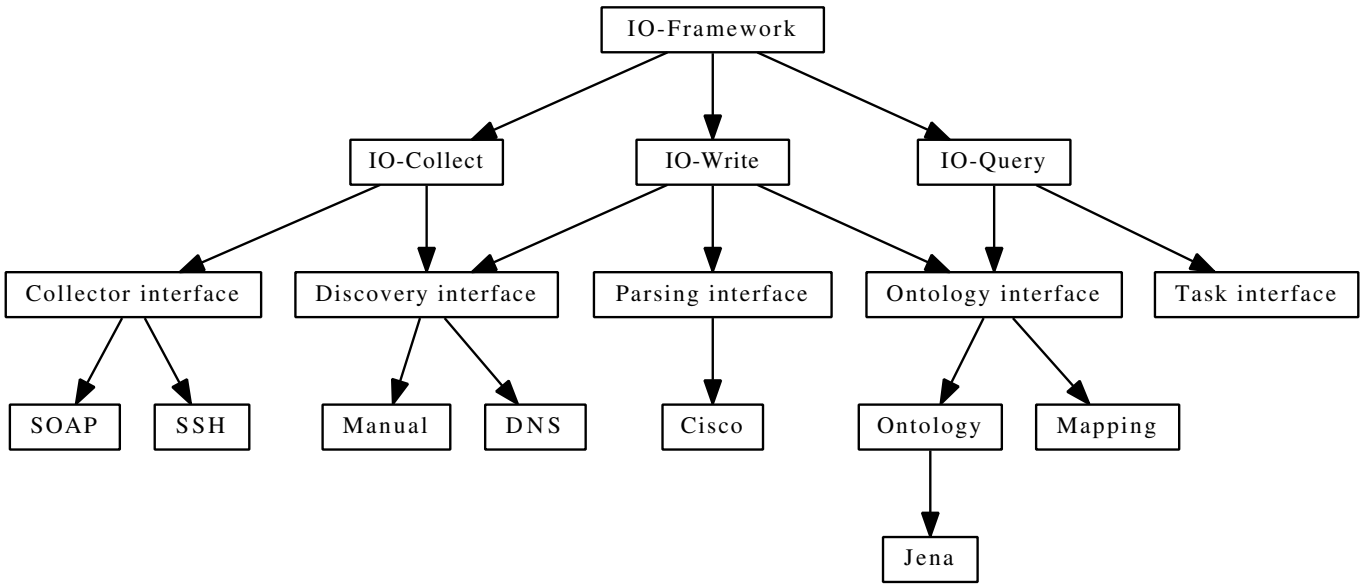

Fig. 3. Overview of modules in the IO framework

Model White Paper of the Distributed Management Task Force (DMTF) Common Information Model v2.7 [38] not unlike the security ontology presented in [26]. We adopted the intuitive layout of *local ProtocolEndpoint relationships* and the corresponding principles of *logical and physical device modeling*. The development of the ontology is based on an artifact-building research approach [39] and was affected by its practical application in the production network and by adherence to the design principles proposed in [40], [41] and [42]

To give an overview, Fig. 1 shows a simplified taxonomy of basic concepts in the IO (the complete TBox can be found at [15]). The taxonomic structure of basic concepts allows retrieving abstract information about a group or a category of individuals from IO, which e.g. satisfies requirements coming from the SIEM domain. Fig. 2 highlights common object properties used to represent a managed network component. The IO is modeled in W3C's OWL [43] with the help of Protégé, a widely used ontology development platform for which an OWL plugin is available [44]. For productive application, we use the Pellet OWL-DL Reasoner [45], which is open source and has proven to be fast applied in realistic ontologies [9].

OWL uses RFC 3987 *Internationalized Resource Identifiers* (IRI), which must be unique. The difficulties with acquiring unique identifiers from assets are highlighted in [46]. The naming policy in our approach differentiates between identifiers for higher and lower level individuals. Higher level identifiers can include a DNS (or host) name: e.g., a switch interface identifier includes the DNS name of its parent switch. In case of insufficient information or lower level individuals (e.g. access list entries), uuids (RFC 4122) are generated. Associated information from any source, such as version and serial numbers, addresses, host and fully qualified domain names are aggregated as data and object properties (attributes and relationships, respectively) in the identified unique individuals (objects).

IV. The IO framework

Currently, IO uses three components: the universal collection framework IO-C, the network ontology writer IO-W and the ontology query module IO-Q. The whole framework is written in Ruby 1.9 to facilitate deployment and provide high flexibility as well as long-term continuity in development. The only non-Ruby part is the internal API for handling OWL ontologies or a SPARQL query engine, provided by Jena [47].

IO-Collect: IO-C's sole function is to collect and aggregate raw asset information employing standardized acquisition procedures. Raw asset information are very heterogeneous, e.g. a configuration dump, an output of a SOAP real-time information service or a text file containing static associations between layer-2 and layer-3 addresses. IO-C provides a *discovery interface* (DI) to locate managed network components as an initial source of information, potentially cascading the acquisition process in further steps by making use of discovery methods available on initially accessed assets. The interface can be freely adapted to domain specific characteristics: for example, the evaluation of appropriate fully qualified domain names by DNS zone transfer or simply the manual insertion of management interface addresses. A modular *collector interface* (CI) can acquire asset information using many available protocols such as SNMP, telnet, SSH or SOAP. Other approaches, such as network based inventory-tools [25] or simply nmap, can also be used to acquire additional information.

IO-Write: IO-W processes raw information aggregated by the IO-C and then stores it in IO. Although named Writer, IO-W also parses the raw input. This task is executed by

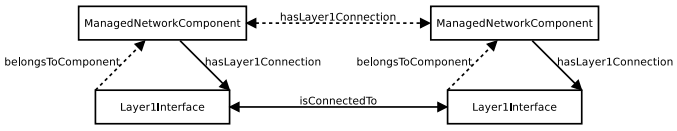

Fig. 4. Abstraction of connections between ManagedNetworkComponents

modules connected through a generic *parsing interface* (PI). The selection of parsing modules can be commenced specifically by IO-C or can be performed automatically if IO-W is able to detect certain features in the raw asset information (e.g. vendor-specific strings). Multiple PI modules can be necessary for a single vendor due to format variations. While the vendor-specific network topology configuration tends to be homogeneous in its representation, raw information to uniquely identify an asset (such as device type or stepping) can be very heterogeneous. The selection of a specific parsing module implicitly provides context information, which is used by IO-W to map identified data to appropriate classes in the IO.

The IO-W is aware of all domains, ranges and data types as defined by IO (in OWL): Every concept in the IO is represented by a corresponding class in IO-W. While this necessitates a fundamental understanding of IO, it supports strictly formalized acquisition procedures for adding or updating individuals and ensures consistency even without using a reasoner, which might consume a significant amount of time. Hence, being able to omit the use of a reasoner provides an additional benefit; especially if the consistency of the whole ontology with all its individuals has to be checked repetitively during update procedures.

Depending on the parsing module selected in the *parsing interface*, a corresponding class is identified by the *ontology interface* (OI). Up to now, usage of the *ontology interface* initiates a two-step process. First, the taxonomy of related individuals of an asset is added in the IO recursively. In the second step, all neighborhood relationships are added. This includes partial information about individuals relating to other assets, such as a corresponding physical switch port on another managed network component (which may not already be added).

IO-Query: The IO-Q provides the general *query interface* (QI) which is used, for example, by our basic command line interface implemented in an interactive Ruby (IRB) shell. Its primary task is not only to handle SPARQL queries, but also to maintain a number of pre-computed lookup tables (indexes), which speed up recurring or cascading queries. In a productive application such as providing context information for a SIEM system, one needs to retrieve context information fast. We use, for example, lookup tables which contain the association of every known internal IPv4 address with its subnet (pre-calculated with bit-shifting), or every known layer-3 subnet attributed to layer-2 interfaces (inferred by virtual-lan-id association). Specifically, the retrieval performance of reachability information is significantly increased by using a lookup table containing neighborhood information inferred by the reasoner beforehand. Lookup tables are registered by the *task interface* (TI). Modules for the *task interface* retrieve information via the *query interface* or from the defined pool of lookup tables. Most of the time, basic information is retrieved from lookup tables to support more complex SPARQL queries to the QI. Lookup tables are generated on demand or (more preferably) automatically when IO-Q is triggered by IO-W after a successful IO update.

LISTING 1: Layer-3 reachability between two ManagedNetworkComponents

```
1:  s_a ← Source ip address
2:  d_a ← Destination ip address
3:  s_n ← get_ip_subnet(s_a)
4:  d_n ← get_ip_subnet(d_a)
5:  s_i ← get_component_interface(s_a)
6:  d_i ← get_component_interface(d_a)
7:  s_s ← get_switch_interface(s_i)
8:  d_s ← get_switch_interface(d_i)
9:  s_r ← get_router_interface(s_n)
10: d_r ← get_router_interface(d_n)
11: s_v ← get_vlans(s_i, s_s, s_r)
12: d_v ← get_vlans(d_i, d_s, d_r)
13: routing ← ((s_v ∪ d_v) = ∅) ∧ s_n ≠ d_n
14: path ← get_path(s_i, s_s, s_r, d_r, d_s, d_i)
15: acls ← new(Array)
16: for p in path do
17:   acls.append(get_acl(p)
18: end
19: pass ← TRUE
20: for a in acls do
21:   pass ← pass ∧ eval_acl(s_a, s_n, s_i, d_a, d_n, d_i)
22: end
23: if pass then
24:   print(Reachabilty from s_a to d_a)
25: else
26:   print(No reachabilty from s_a to d_a)
27: end
```

V. IO USE CASES

We selected four current applications of IO to be presented as use cases in this paper. The goal is to give an impression about the scope of applications an interconnected asset ontology can already be used for. Selected characteristics of the presented use cases will provide the basis for the evaluation scenario in Sect. VI. The evaluation is based on an artifact-evaluating research approach [39]. To enhance readability pseudo-code is used to better visualize selected information retrieval procedures.

Use case 1: Automatically generated security alerts are often false alarms (false positives). Usually, IDS sensors are placed at vital points in the network [36]. A potentially harmful signature detected in a packet stream, which cannot reach its destination from the vital point where the packet was detected, has a significantly reduced likelihood of succeeding in exploiting a given vulnerability [21]. Reachability could be foiled, for example, by a lack of routing between subnets or due to packet filter policies defined in an access list (ACL). To retrieve the information whether a packet stream can traverse a certain path towards a destination, we need to take into account: physical cabling, virtual-lan (VLAN) ids, subnet configuration, routing tables, ACLs and relationships

LISTING 2: SPARQL-Query for neighborhood retrieval

```
SELECT ?sw2 ?i1 ?i2 WHERE {
 ?sw2 a io:Switch
 . io:node io:hasLayer1Connection ?sw2
 . io:node io:hasLayer1Interface ?i1
 . ?sw2 io:hasLayer1Interface ?i2
 . ?i1 io:isConnectedTo ?i2 }
```

LISTING 3: Identify VoIP-switches

```
1:  vlans ← new(Array)
2:  switches ← new(Array)
3:  vlans.append(get_voice_vlans_on_phones)
4:  vlans.append(get_voice_vlans_on_switches)
5:  vlans.sort
6:  vlans.unique
7:  for v in vlans do
8:    switches.append(get_switches_in_vlan(v))
9:  end
10: switches.sort
11: switches.unique
```

such as neighborhood. To speed up retrieval, IO-Q uses two lookup tables: One for "managed network components and all their neighbors" inferred by Pellet and one for "all pair shortest paths in the network" computed by a breadth first search (bfs). In our network, routing is only restricted by ACLs, so we omit creating a routing lookup table. Listing 1 summarizes the procedure in pseudo-code.

Use case 2: It is desirable to have abstract or specialized overviews about the current network configuration. Creating a visual representation of the connections between switches and routers is a common step in any *structure analysis* process. Topological interconnection represented by formal graphs is also often necessary for research in the domains of network modeling [32], [48] or attack graph generation [49], [37]. "A connection" between two switches, for example, is represented as a property chain between two individuals of the managed network component concept in the IO (see Listing 2). A more abstract relationship can be inferred automatically by the definition of a corresponding property chain (see Fig. 4).

Use case 3: One important step in the IT-Grundschutz methodology [23] is the differentiation of categories of device types (target objects) as part of the *structure analysis*. Categories are implicitly defined by the BSI modules (bundles of security controls) associated with them. E.g., privacy compliance may require knowing whether a managed network component can come in contact with sensitive VoIP packet streams. Essentially, this is a two-step process: To retrieve all managed network components which could be involved in a VoIP communication, first every VLAN id to which a VoIP device is connected has to be inferred. Secondly, every managed network component associated with this set of VLAN ids can be retrieved. Listing 3 represents the procedure in pseudo-code.

Use case 4: Aime et al. describe a situation in which an assessment of a risk to the IT infrastructure depends on the interaction of other assets [20]: "botnet's slaves may compromise unprotected communication across their hosting subnet". Every host and its corresponding physical switch port can be inferred by first retrieving every VLAN id related to the subnet with the attacker in question and then using the reasoner to collect the appropriate switch ports and their known MAC and IP addresses. Resulting hosts might be at a higher risk of being added to a botnet and temporary ACLs can be brought in place with high precision.

TABLE I
PERFORMANCE DATA FOR ACQUISITION AND STORAGE OF INFORMATION

Information acquisition or storage procedure	Runtime (s)
Acquisition of raw asset information (CI):	933.02
Parsing of raw asset information (PI)	33.39
Initial adding of individuals to the IO (OI)	264.41
Updating individuals in the IO (OI)	218.32
Initializing reasoner & committing a simple sanity-check query (OI)	85.30

VI. EVALUATION

IO is currently deployed and evaluated in a live network. In this section, we describe the evaluation environment and present an evaluation scenario based on currently implemented SIEM processes. After that, we present performance results of various benchmarks that show that our approach is working (proof of concept). The benchmarks are based on formal competency questions addressed at IO-Q using the Pellet Reasoner. Due to the large number of individuals in IO, a comparative evaluation utilizing the Protégé OWL API had to be omitted. Finally, we show additional benefits and improvements which arose with the deployment of the IO framework in a live network environment.

The IO framework is deployed on a Sun Fire X4150 with 8 physical cores and 16GB RAM. One focus of our evaluation scenario is the processing of security alerts as a common task for SIEM systems. We pre-recorded roughly 1 M IDS alerts containing about 18 k unique tuples of source and destination IP addresses in these alerts. The alerts are produced by a *Snort* IDS in version 2.1.9.2 including the open emerging threats rule set. Alerts are generated from traffic in 24 /24 subnets relayed by a monitoring session directly out of the production network (/16). Alerts are post-processed in a manner that mimics the information available to SIEM systems. The usage scenario does not include a real SIEM system, to better isolate the performance of the IO framework. All given performance values are averages of 240 hourly module executions.

Acquisition & Update (see Table I): Asset information is obtained from 370 initially accessed managed network components via SSH. An update of the IO iterates the complete acquisition process. The resulting IO contains on average: 242,354 individuals, 242,983 object properties and 400,182 data properties and takes up about 120 MB written in OWL format.

Retrieval (see Table II): Retrieval procedures are accelerated by generating lookup tables every time the IO is initialized or updated. More complex queries are composed of lookup table and SPARQL queries. All use case queries are conducted by modules registered at the TI.

TABLE II
PERFORMANCE DATA FOR RETRIEVING INFORMATION FROM THE ONTOLOGY

Information retrieval procedure	Runtime (s)
Listing IPv4 subnets	0.04
Listing managed network components	0.05
Mapping IPv4 addresses to IPv4 subnets	8.38
Mapping IPv4 addresses to physical ports	2.62
Mapping IPv4 subnets to physical ports	2.34
Mapping MAC addresses to physical ports	4.56
Mapping managed network components to all neighbors	68.31
Listing "all pair shortest paths" by bfs (use case 1)	1.63
Listing all switches, neighboring switches and paths (use case 2): Selecting 406 switches and routers connected by 895 edges	74.11
Listing "VLAN id with VoIP phones" (use case 3, step 1)	1.44
Listing "VoIP managed network components" (use case 3, step 2)	77.52
Listing all MAC & IPv4 addresses threatened at switch ports with VLAN id of source port (use case 4)	5.35

TABLE III
PERFORMANCE DATA FOR ANNOTATING IDS ALERTS (IN ALERTS/S)

Set	Annotation w/o result caching	Annotation with result caching
I	569.95	14,346.79
II	552.08	14,105.20
III	145.62	4,301.61

Annotation of IDS alerts (see Table III): In the following evaluation, we annotate the alerts generated by the generic IDS *Snort* with information that enables assessing the impact of the alert in our specific network. Annotation set I retrieves the associated VLAN id, the subnet and its gateway address. Set II adds a path of managed network components the attack packets would traverse. Set III adds all ACLs placed on the managed network components of the inferred path (to support further potential actions in an escalation process). In the attack mix we see that the performance can be improved drastically by grouping alerts with respect to source and destination addresses (result caching).

Verification of consistency: The use of IO has additional benefits. Raw asset information being parsed and then mapped to concepts in IO is rejected by the ontology interface (sometimes even in the parser interface) if there are inconsistencies to the modeled concept layout. We were able to identify and eliminate several minor violations in the production network, including: DNS names violating a naming policy, discrepancies between DNS name and host name, misconfigured round robin DNS entries, redundant DNS names for IP addresses, violations of configuration policies, VoIP phones without voice VLAN on the switch interface and VoIP phones with broken CDP name propagation.

VII. CONCLUSION & FUTURE WORK

We have shown that IO is able to provide knowledge in support of tasks performed in the IT-Grundschutz and SIEM domain, while eliminating the need for further information acqusition. Performance regarding the information acquisition and retrieval procedures is already in a scope that makes a productive use feasible. The design of the IO framework entails a standardized information acquisition according to the concept- and relationship-layout in IO. This approach enables asset information acquisition with a high level of detail across different vendors or configuration formats.

In a future project, a fully deployed configuration management system in the benchmark network will enable IO to differentiate between assets with configuration changes from assets without. Event dispatcher mechanisms in managed network components that can push changes regarding real-time information, such as neighborhood relationships, will reduce the need of polling mechanisms in IO-C. This again will enable incremental updates of the ontology and thereby reduce acquisition runtime. For now, IO was developed and deployed in the benchmark network only. To continue with the generalizing of concepts, we will deploy IO in additional networks of similar and greater size. Future development will include representation for asymmetric routing and intentionally redundant routes (e.g., as part of load balancing).

By including *Information Securtiy Management System* (ISMS) concept hierarchies for ISO 27001 controls and IT-Grundschutz countermeasure modules or protection requirements we will be able to infer the importance of IT assets and match IT-Grundschutz modules to appropriate assets automatically (analogously to use case 3). Adding context information about a telephone infrastructure, such as VoIP-server configuration (acquired via SOAP) and phone location (acquired by LDAP), is already in first stage of testing in the current network. In general, we plan to add more concepts related to the lifecycle of IT assets, e.g. the relocation history of VoIP phones or the availability history of WLAN access points. To provide better performance, for e.g. computation of reachability, IO could be used to infer a network model used to build a state machine as presented in [34].

Our final observation is: While we try to provide a bigger picture about the security implications of an interconnected asset topology and identify existing threats [50], new threats are introduced by the possibility of theft or malicious manipulation of IO itself. In any case, the framework design must continue to incorporate sophisticated measures to ensure availability, integrity, and confidentiality to be useful in a production environment.

REFERENCES

[1] R. Baskerville, "Information systems security design methods: implications for information systems development," *ACM Comput. Surv.*, vol. 25, no. 4, pp. 375–414, 1993.

[2] C. Jung, I. Han, and B. Suh, "Risk analysis for electronic commerce using case-based reasoning," in *Reasoning, International Journal of Intelligent Systems in Accounting, Finance & Management*, 1999, pp. 61–73.

[3] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowl. Acquis.*, vol. 5, pp. 199–220, 1993.

[4] H.-J. Happel and S. Seedorf, "Applications of Ontologies in Software Engineering," in *SWESE 2006, held at ISWC 2006*, 2006.

[5] A. Herzog, N. Shahmehri, and C. Duma, "An ontology of information security," *International Journal of Information Security*, vol. 1, no. 4, pp. 1–23, 2007. [Online]. Available: http://portal.acm.org/citation.cfm?id=508171.508180

[6] M. Schumacher, "3. Ontologies," in *Security Engineering with Patterns*, ser. LNCS. Springer Berlin / Heidelberg, 2003, vol. 2754, pp. 29–44.

[7] T. Gardiner *et al.*, "Framework for an automated comparison of description logic reasoners," in *Proc. of the ISWC 2006*. Springer, 2006, pp. 654–667.

[8] E. Kaufmann and A. Bernstein, "How useful are natural language interfaces to the semantic web for casual end-users?" in *Proc. of the ISWC'07/ASWC'07*. Berlin, Heidelberg: Springer, 2007, pp. 281–294.

[9] Z. Pan, "Benchmarking DL Reasoners Using Realistic Ontologies," *Proc. of the OWLED*, 2005.

[10] C. Elfers *et al.*, "Typed linear chain conditional random fields and their application to intrusion detection," in *IDEAL 2010*, ser. LNCS. Springer Berlin / Heidelberg, 2010, vol. 6283, pp. 13–20.

[11] M. C. Parmelee, "Toward the semantic interoperability of the security information and event management lifecycle," in *Work. N. SecArt 2010*, 2010, pp. 18–19.

[12] S. Fenz *et al.*, "Information Security Risk Management: In which security solutions is it worth investing?" *Communications of the Association for Inf. Systems*, 2011.

[13] *ISO/IEC 27001:2005 - Information technology – Security techniques – Information security management systems – Requirements*. ISO, 2005.

[14] R. Montesino and S. Fenz, "Information Security Automation: How Far Can We Go?" in *ARES 2011*, 2011, pp. 280 –285.

[15] "IO development website." [Online]. Available: http://dev.tzi.org/io

[16] "FIDeS development website." [Online]. Available: http://www.fides-security.org

[17] J. An Wang *et al.*, "An ontological approach to computer system security," *Inf. Sec. J.: A Global Perspective*, vol. 19, pp. 61–73, 2010.

[18] S. Fenz and A. Ekelhart, "Formalizing information security knowledge," in *Proc. of ASIACCS 2009*. New York, NY, USA: ACM, 2009, pp. 183–194.

[19] B. Tsoumas and D. Gritzalis, "Towards an ontology-based security management," in *AINA '06*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 985–992.

[20] M. D. Aime and F. Guasconi, "Enhanced vulnerability ontology for information risk assessment and dependability management," *DEPEND*, pp. 92–97, 2010.

[21] A. Syalim *et al.*, "Comparison of risk analysis methods: Mehari, magerit, nist800-30 and microsoft's security management guide," in *ARES 2009*, 2009, pp. 726 –731.

[22] A. Ekelhart *et al.*, "Ontology-based decision support for information security risk management," in *Proc. of the 4th IEEE SMC 2011*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 80–85.

[23] BSI, "German IT Baseline Protection Manual," 2011.

[24] A. Ekelhart *et al.*, "Formal threat descriptions for enhancing governmental risk assessment," in *Proc. of the ICEGOV 2007*. New York, NY, USA: ACM, 2007, pp. 40–43.

[25] F. Cheng *et al.*, "An integrated network scanning tool for attack graph construction," in *Proc of the GPC 2011*, ser. LNCS, no. 6646. Oulu, Finland: Springer, 2011, pp. 138–147.

[26] B. Tsoumas *et al.*, "An ontology-based approach to information systems security management," in *Computer Network Security*, ser. LNCS. Springer Berlin / Heidelberg, 2005, vol. 3685, pp. 151–164.

[27] C. Pak and J. Cannady, "Asset priority risk assessment using hidden markov models," in *Proc. of the SIGITE 2009*. New York, NY, USA: ACM, 2009, pp. 65–73.

[28] S. Radack and R. Kuhn, "Managing security: The security content automation protocol," *IT Professional*, vol. 13, no. 1, pp. 9 –11, 2011.

[29] R. Montesino and S. Fenz, "Automation possibilities in information security management," in *EISIC 2011*, 2011, pp. 259 –262.

[30] C. Johnson, *The technical specification for the Security Content Automation Protocol (SCAP)*. U.S. DoC, NIST, Gaithersburg, MD, 2009.

[31] J. Wunder *et al.*, *Specification for Asset Identification 1.1*. U.S. DoC, NIST, Gaithersburg, MD :, 2011.

[32] J. Burns *et al.*, "Automatic management of network security policy," in *DISCEX II 2001*, 2001, p. 1012.

[33] K. Ingols *et al.*, "Modeling modern network attacks and countermeasures using attack graphs," in *ACSAC 2009.*, 2009, pp. 117–126.

[34] E. Al-Shaer *et al.*, "Network configuration in a box: towards end-to-end verification of network reachability and security," in *ICNP 2009*, 2009, pp. 123 –132.

[35] W. Enck *et al.*, "Configuration management at massive scale: system design and experience," *IEEE J.Sel. A. Commun.*, vol. 27, pp. 323–335, 2009.

[36] S. Noel and S. Jajodia, "Optimal IDS Sensor Placement and Alert Prioritization Using Attack Graphs," *J. Netw. Syst. Manage.*, vol. 16, pp. 259–275, 2008.

[37] K. Ingols *et al.*, "Practical attack graph generation for network defense," in *ACSAC 2006*, 2006, pp. 121–130.

[38] DMTF, "Cim network model white paper," 2003.

[39] P. Järvinen, "Research Questions Guiding Selection of an Appropriate Research Method," in *European Conference on Information Systems*, Hansen, Bichler, and Mahrer, Eds., Vienna University of Economics and Business Administration, 2000, pp. 124–131.

[40] T. R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing," *Int. J. Hum.-Comput. Stud.*, vol. 43, pp. 907–928, 1995.

[41] D. L. M. Natalya Fridman Noy, "Ontology development 101: A guide to creating your first ontology," Knowl. Syst., AI Lab., Stanford U, Tech. Rep. KSL-01-05, 2001.

[42] I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen, "From shiq and rdf to owl: the making of a web ontology language," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 1, no. 1, pp. 7 – 26, 2003.

[43] W3C, "OWL 2 Web Ontology Language Document Overview," Tech. Rep., 2009.

[44] H. Knublauch *et al.*, "The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications," in *ISWC 2004*, ser. LNCS. Springer Berlin / Heidelberg, 2004, vol. 3298, pp. 229–243.

[45] E. Sirin *et al.*, "Pellet: A practical OWL-DL reasoner," *Web Semant.*, vol. 5, pp. 51–53, 2007.

[46] DMTF, "Cim core model white paper," 2000.

[47] J. J. Carroll *et al.*, "Jena: implementing the semantic web recommendations," in *Proc of the WWW Alt. 2004*. New York, NY, USA: ACM, 2004, pp. 74–83.

[48] J. Govaerts *et al.*, "A formal logic approach to firewall packet filtering analysis and generation," *In: Artif. Intell. Rev.*, vol. 29, pp. 223–248, 2008.

[49] H. Birkholz *et al.*, "Efficient automated generation of attack trees from vulnerability databases," in *Work. N. SecArt 2010*, 2010, pp. 47–55.

[50] M. Chmielewski and P. Stąpor, "Protégé; based environment for DL knowledge base structural analysis," in *Proc. of the 3rd ICCCI2001*. Berlin, Heidelberg: Springer, 2011, pp. 314–325.