Using Ontologies to Quantify Attack Surfaces

Michael Atighetchi, Borislava Simidchieva, Fusun Yaman
Raytheon BBN Technologies
Cambridge, MA 02138 USA
{matighet | simidchieva | fusun}@bbn.com

Thomas Eskridge, Marco Carvalho
Florida Institute of Technology
Melbourne, FL 32901 USA
{teskridge | mcarvalho}@fit.edu

Captain Nicholas Paltzer
Air Force Research Laboratory
Rome, NY 13441 USA
nicholas.paltzer@us.af.mil

***Abstract*—Cyber defenders face the problem of selecting and configuring the most appropriate defenses to protect a given network of systems supporting a certain set of missions against cyber attacks. Cyber defenders have very little visibility into security/cost tradeoffs between individual defenses and a poor understanding of how multiple defenses interact, which, in turn, leads to systems that are insecure or too overloaded with security processing to provide necessary mission functionality. We have been developing a reasoning framework, called Attack Surface Reasoning (ASR), which enables cyber defenders to explore quantitative tradeoffs between security and cost of various compositions of cyber defense models. ASR automatically quantifies and compares cost and security metrics across multiple attack surfaces, covering both mission and system dimensions. In addition, ASR automatically identifies opportunities for minimizing attack surfaces, e.g., by removing interactions that are not required for successful mission execution. In this paper, we present the ontologies used for attack surface reasoning. In particular, this includes threat models describing important aspects of the target networked systems together with abstract definitions of adversarial activities. We also describe modeling of cyber defenses with a particular focus on Moving Target Defenses (MTDs), missions, and metrics. We demonstrate the usefulness and applicability of the ontologies by presenting instance models from a fictitious deployment, and show how the models support the overall functionality of attack surface reasoning.**

I. Introduction

Cyber security remains one of the most serious challenges to national security and the economy that we face today. Systems employing well known but static defenses are increasingly vulnerable to penetration from determined, diverse, and well resourced adversaries launching targeted attacks such as Advanced Persistent Threats (APTs).

Due to the heavy focus on cyber security technologies in both commercial and government environments over the last decade, an overwhelming array of cyber defense technologies have become available for cyber defenders to use. As the number and complexity of these defenses increase, cyber defenders face the problem of selecting, composing, and configuring them, a process which to date is performed manually and without a clear understanding of integration points and risks associated with each defense or combination of defenses.

As shown in Figure 1, the current state-of-the-art approach for selecting and configuring cyber defenses is manual in nature and is often done without a clear understanding of security metrics associated with attack surfaces. Due to the talent shortage in cyber security Subject Matter Experts (SMEs) [9], this introduces significant delays and cost.

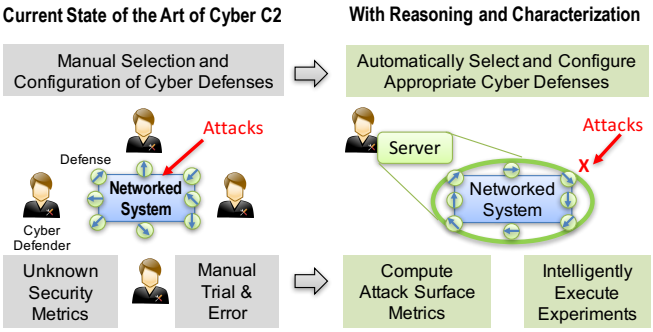

Fig. 1. The proposed approach computes attack surface metrics, provides structured support for deployment of (and experimentation with) wrapped defenses, and automates the defense selection and configuration process

The reasoning framework presented in this paper aims to significantly improve the level of rigor and automation associated with selection and configuration of cyber defenses. Using an ontologically grounded definition of an attack surface, the framework contains algorithms to find all applicable attack vectors and compute metrics for the security and cost impact of adding cyber defenses to target systems. Using models of key mission processes and their interactions, the analysis extends observations about system-level components to the resulting impact on execution of mission critical workflows. Finally, the framework combines measurement, modeling, and analysis with testing of software artifacts through the use of a virtualized test infrastructure [1]. Experimental validation of analysis results on real systems with real defense implementations establishes the usefulness and validity of the approach.

Figure 2 illustrates how the Attack Surface Reasoning (ASR) framework captures models of underlying systems, cyber defenses, and missions in the form of unified models. These models are augmented by other models that describe adversary constraints, potential attack steps, and definitions of security and cost metrics. ASR provides two categories of algorithms: attack surface characterization and minimization. The characterization algorithm constructs attack vectors and calculates security and cost metrics. The minimization algorithm uses system and mission information to identify opportunities for pruning unnecessary access paths to reduce

Distribution Statement "A" (Approved for Public Release, Distribution Unlimited). This material is based upon work supported by the Air Force Research Laboratory under Contract No. FA8750-14-C-0104.

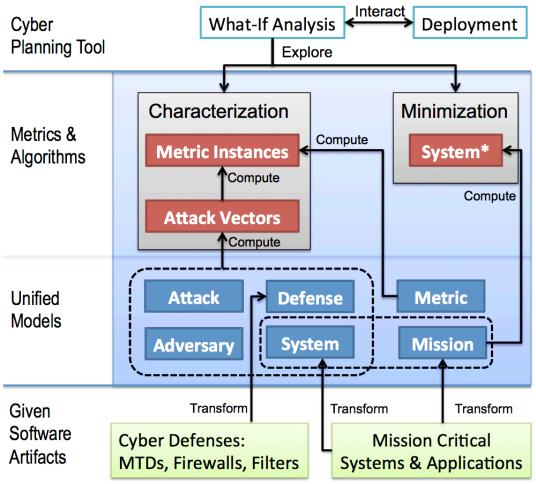

Fig. 2. The Attack Surface Reasoning (ASR) framework

the attack surface. Using the models, algorithms, and metrics, cyber defenders can compare various deployments of proactive cyber defenses in a quantitative manner and contrast tradeoffs between security benefits and performance overhead. As such, ASR provides a foundational capability in support of an envisioned cyber planning tool that automatically suggests and configures defenses given mission executions over systems.

This paper describes the ontologies used to model systems, cyber defenses, adversarial capabilities, and mission constraints. Validation of the approach focuses on a specific class of proactive cyber defenses, Moving Target Defenses (MTDs) [7], [11]. MTDs claim to make entry points into networks and systems harder to detect, thereby reducing vulnerabilities and making the exposure to those vulnerabilities that remain more transient. This introduced dynamism ought to render attacks against MTD-protected systems less effective, but few quantitative results are available to date, which makes MTDs a prime choice for quantification.

The rest of the paper is organized as follows. Section II describes related work in threat modeling and analysis. Section III describes the set of ontologies we developed to support attack surface reasoning. Section IV reports on the validation results of applying the ontologies to cyber defense operations of a small enterprise network. Section V concludes the paper.

II. Related Work

The ontologies presented in this paper relate to several approaches for modeling cyber security systems and observables.

A. Security Standards

A number of different taxonomies exist for describing cyber security related information. For threat information, this set of standard includes the Common Vulnerabilities Enumeration (CVE), Common Weakness Enumeration (CWE), Common Vulnerability Scoring System (CVSS), Malware Attribute Enumeration and Characterization (MAEC), Structured Threat Information eXpression (STIX), and Common Attack Pattern Enumeration and Configuration (CAPAC). Taxonomies in use for system modeling include the Cyber Observable eXpression (CybOX) and the Common Information Model (CIM). These standards focus on capturing detailed information about system observables, cyber security events, indicators of compromise, and vulnerabilities for the purposes of sharing specific threat information (to yield enhanced intrusion detection) and eliminating existing vulnerabilities (through continuous patching). In contrast, the ASR ontologies are expressed at a higher level of abstraction and focus on design-level assessments of attack surfaces. Another difference is that the ASR ontologies are expressed in OWL, while the community standards mentioned above are prescribed in XML. Finally, the above-mentioned standards focus on system and adversary modeling, but provide no structured means for representing cyber defense capabilities. In contrast, ASR contains a specific defense ontology describing the protection provided by defenses and the cost associated with various defense configurations.

B. Security Ontologies

A number of different ontologies exist for expressing security-related properties, including [6] and [4], as summarized in [13]. [5] applies semantic threat and defense modeling to identify proper firewall configurations. [14] develops an ontology for the HTTP protocol as well as attacks against web applications (using HTTP), and then uses a separate ontology for finding attack vectors. [10] focuses on a review of existing cyber security taxonomies and ontologies and points out several existing models. However, the review does not list any ontologies for cyber defenses. [15] describes an extensive ontology supporting forensic activities across disparate data sources. Finally, work on modeling cyber defense decision processes [3], [12] provides ontology support for learning and extracting cyber defense workflows and decision procedures.

The ASR ontologies are in large inspired by the STRIDE threat-modeling approach [16] used by Microsoft. One key difference to existing ontologies is the focus on abstract architectural concepts and high-level adversarial objectives.

III. Ontologies

The attack surface reasoning algorithms operate over a set of models that together describe the system under examination, its defenses, the assumed capabilities and starting point(s) of the adversary, and optionally a mission or set of missions which may operate over the defined system. In addition, the set of metrics to be computed is itself described in a model to allow for easy extension and modification by the user.

ASR models are defined in the WorldWideWeb Consortium (W3C) semantic Web Ontology Language (OWL). Using a semantic web substrate provides a number of benefits, including:

- Scalability: the OWL language and supporting tools allow for scaling to very large models;
- Inference: OWL ontologies encode meaning in a formal way, which enables inferring new facts from existing data;
- Cross-domain integration: OWL ontologies can connect disparate domains without contaminating the sources;

- Standards and community: OWL and associated languages such as Resource Description Framework (RDF) and SPARQL Protocol And RDF Query Language (SPARQL) provide interoperable libraries and tooling, and active practitioner communities; and
- Relative maturity: semantic web languages provide tested algorithms, established terminology, and relatively mature libraries. Tooling with predictable performance both within and beyond the laboratory setting is also available.

One of the key challenges of modeling distributed systems is to identify the level of abstraction most appropriate for the modelers who will create the models, the algorithms that will operate over them, and the results that are provided to stakeholders. Modeling at the extreme of precision allows exact answers to be derived, but creates models that are difficult to accurately create and to keep up to date, and leads to analysis outcomes that are brittle as the system changes. On the other hand, modeling at too coarse of a level of abstraction leads to easily created models, but models that can tell little to interested parties about questions of importance.

We took a middle road with ASR. A number of the concepts, and the level of granularity, were modeled after the Microsoft STRIDE [8] threat-classification framework and related modeling languages described in [16]. STRIDE expresses system concepts through abstract concepts including processes, data flows, boundaries, external entities, and data stores. We model the different aspects of an attack surface separately in order to facilitate modularity and extensibility. Table I lists the six ontological models used in ASR and summarizes their content.

TABLE I
ASR USES A COLLECTION OF MODELS TO QUANTIFY ATTACK SURFACES

Model	Concepts
System	System components and their relationships; e.g., computational entities, boundaries, and data flows
Attack	Generic attack logic as individual steps, vectors, and templates
Adversary	Adversarial starting position and goal
Mission	Mission relevant system elements and key performance metrics
Defense	Cyber defense capabilities in terms of protections provided plus associated costs
Metric	Metrics for security, cost, and mission impact

A. System Model

System models describe the business system against which attacks can be executed and within or around which defenses can be deployed. These models detail the hosts in the system, the networks that connect these hosts, and the processes that run on them. Data flows are modeled here at three different layers: process, network, and physical. The three layers are interconnected in the model such that one can determine for a given process-layer data flow that the described data is sent out through a given endpoint at the network layer, which in turn is bound to a particular network interface card (NIC) at the physical layer. Table II describes the main resource types associated with the system model ontology.

TABLE II
MAIN SYSTEM MODEL CONCEPTS

Resource	Description
Entity	General concept
Boundary	Trust realm for unrestricted access within a boundary
Vertical Boundary	subclassOf Boundary describing realm cross layers
Horizontal Boundary	subclassOf Boundary describing realm on a single layer
Host	subclassOf Vertical Boundary representing a computer system
WAN	subclassOf Horizontal Boundary representing a wide area network
VLAN	subclassOf Horizontal Boundary representing a wide area network
Layer	Logical layering of functionality into three main layers
NetworkLayer	subclassOf Layer describing network entities and interactions
PhysicalLayer	subclassOf Layer describing physical entities
ProcessLayer	subclassOf Layer describing application-level components and interactions
DataFlow	Flow of bits between two entities
DataStore	Persistent store of information
External	An entity that is external to the system
User	subclassOf External describing human actors
NetworkEndpoint	Sockets used in network connections
NIC	Network Interface Card
Process	Operating System process
Resource	Shared resource with certain capacity

The following properties have specific meaning:

- contains: expresses membership relationship between two Entities. For instance, a Host contains Processes and a VLAN contains NICs.
- connectsTo: expresses a data or control flow link between two Entities. For instance, a User connects to a Process, a Process connects to a NetworkEndpoint, and a NetworkEndpoint connects to a NIC.
- via: expresses a link between hierarchical data flows. For example, a process-layer flow is realized via a network-layer flow, which itself happens via a physical-layer flow.

B. Attack Model

The attack model describes the generic activities performed by adversaries as a collection of potential attack steps. Table III describes the main resource types associated with the attack model ontology. Each attack step definition comprises a number of attributes that specify an attack type (modeled via the six high-level types of attacks whose initials define STRIDE), the pre-conditions necessary for the attack step to execute, and the post-conditions that holds once the attack step executes successfully. Figure 3 shows an example of an attack step definition that represents network sniffing, and Table IV shows the set of attack step definitions that are currently modeled in ASR, using the STRIDE attack types from Table III.

C. Adversary Model

The adversary model contains the following information:

- Starting Position: A reference to an entity in the system model that describes the starting privilege an adversary has for the purpose of a specific assessment.
- Target Goal: Information about the type of attack and the intended target of the attack.

TABLE III
MAIN ATTACK MODEL CONCEPTS

Resource	Description
AttackStep	A specific instance of adversarial activity. Attack vectors consists of a collections of linked attack steps.
AttackStepDefinition	A reusable generic description of an adversarial activity. Attack steps are derived from definitions
AttackVectorElement	Ordering and context around an AttackStep to form an AttackVector
AttackVector	Ordered execution of AttackSteps
AttackTemplate	A templatized version of an attack vector
Attacker	Captures aspects of the expected adversary, including the starting position
SideEffect	As part of executing this attack, these specific facts are added to the model
AttackType	**The type of attack being executed**
Spoofing	subclassOf AttackType. Illegally accessing and then using another user's authentication information.
Tampering	subclassOf AttackType. Malicious modification of data
Repudiation	subclassOf AttackType. Deny performing an action without other parties having any way to prove otherwise
InformationDisclosure	subclassOf AttackType. Exposure of information to individuals who are not supposed to have access to it
DenialOfService	subclassOf AttackType. Deny service to valid users
ElevationOfPrivilege	subclassOf AttackType. An unprivileged user gains privileged access and thereby has sufficient access to compromise or destroy the entire system

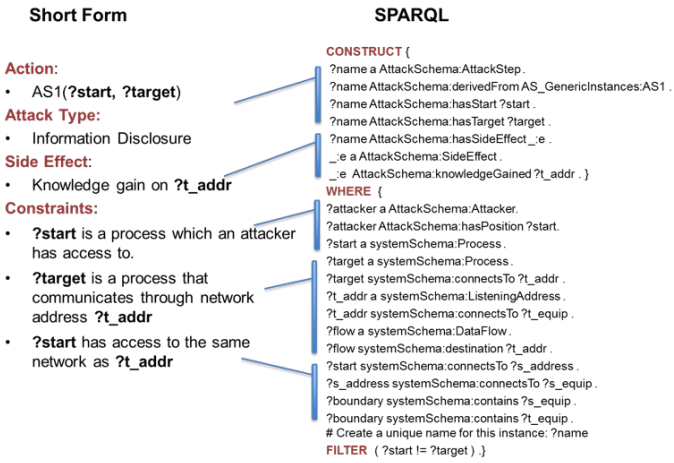

Fig. 3. Example of an attack step that performs a network sniffing action

TABLE IV
ATTACK STEPS CURRENTLY MODELED IN ASR

Name	Type	Pre-Condition	Post-Condition
Sniff	Information Disclosure	Access to network	Knowledge about observed network flows
PortScan	Information Disclosure	Network reachability	Knowledge about listening sockets
TCPConFlood	Denial of Service	Network reachability & Knowledge about the target endpoint	Depletes file descriptors at a given rate
OSFingerPrint	Information Disclosure	Knowledge on listening socket on a host	Knowledge about host OS specifics
GetRoot	Elevation of Privilege	Knowledge on host OS and listening socket	Root privilege on host
ShutDownServer	Denial of Service	Knowledge on host OS and listening socket Root privilege on host	Server unavailable

- Attack Vector Template: Preconceived structure of attack vectors specifying sequences of types of attack steps that have not been bound to specific instances.

Given these assumptions about the adversary, ASR will automatically identify all applicable attack vectors as a partially ordered sequence of bound attack steps.

D. Mission Model

Mission models describe mission-critical flows between actors and services at the application layer. The mission models are a strict subset of process-layer system entities and data flows contained in the system model. Table V shows the main concepts in the ASR mission models.Mission metrics evaluate the fitness of a specific mission within the context of a collection of other models. Like system metrics, mission metrics are evaluated along the two dimensions of cost and security, and mission-critical flows can specify requirements on the cost and security of information exchanges. Most mission metrics are rated on a *normal*, *degraded*, *fail* scale. To allow for quick and easy comparison of mission metrics among multiple configurations, we provide a mission aggregate cost index (ACI) and a mission aggregate security index (ASI), which return the minimum score along all cost or security concerns, respectively (i.e., if a single data flow fails a cost or security requirement, the mission aggregate cost or security index indicates a *fail* also). The individual metrics are provided for comparison purposes so that it is easy for the user to distinguish between a configuration that only has one or two poorly performing components for this mission, and an overall equally rated configuration whose every component is rated *degraded* or *fail* for this mission. Finally, the mission security and cost metrics are folded into an aggregate mission index (AMI), similar to the ACI and ASI. The value of the AMI

is *fail* if either the mission aggregate security or cost indices evaluates to *fail*, and equals the mission aggregate cost rating otherwise (this is because security is evaluated on a *pass/fail* scale, while cost follows the user-defined three-band ranking explained in detail below).

Mission performance is constrained through four threshold values, $p1_{latency}, p2_{latency}, p1_{throughput}, p2_{throughput}$, that describe lower and upper allowable thresholds for percentage overhead rates on latency and throughput. Not all mission-critical data flows must specify a lower and upper threshold, and, if there is no requirement on a data flow, user-configurable default threshold values will be used. These thresholds are used to define the following three bands:

- Normal ($p_{latency} < p1_{latency}$): The mission operates within normal parameters, i.e. the greatest latency penalty incurred is still less than the lower threshold.
- Degraded ($p1_{latency} <= p_{latency} < p2_{latency}$): The mission can continue, though with sub-optimal outcomes, i.e. the greatest latency penalty incurred is more than the lower threshold but less than the maximum allowable.
- Fail ($p2_{latency} < p_{latency}$): The mission cannot continue and misses objectives, i.e. the greatest latency penalty incurred exceeds the maximum allowed and the mission performance will be unacceptable.

For example, the user can specify that a latency penalty of up to 10% is acceptable if it allows for a more sophisticated defense to be deployed with a mission, but a latency penalty of 40% or more leads to unacceptable delays and jeopardizes the mission. In this case, if the cumulative latency along some mission-critical data flows does not exceed 110% of the normal value, these data flows are rated as *normal*; if the latency exceeds 110% but is below 140%, corresponding data flows are rated as *degraded*; and if the latency is over 140% of the original value, those data flows are rated as *fail*. The throughput calculations are analogous, with the exception that a penalty means a decrease, not an increase, in throughput.

Mission security requirements specify any required security attributes, which are delineated among confidentiality, integrity, and availability. Not all mission-critical data flows must specify a security requirement and if no requirement is specified, the data flow is not considered when evaluating mission security. Security metrics are evaluated on a binary scale where a data flow either meets its security requirement or violates it. A data flow is considered to violate a security requirement if an attack step can compromise that requirement.

For example, since all attack steps are categorized using STRIDE, if an attack step contributes to a denial of service on a data flow and that data flow has an availability requirement, the requirement is violated. If the same data flow also has confidentiality or integrity requirements, those are evaluated separately with respect to other attack steps that might compromise them. If at least one mission-critical data flow is found to violate a security-related requirement, that requirement is rated as *fail* for the entire mission. For example, if there are three data flows with integrity requirements and only one of them violates a requirement, then the mission still gets a *fail* score for integrity. If any of the individual percentages of data flows that fail for confidentiality, integrity or availability are greater than zero, the mission aggregate security index consequently evaluates to a *fail* score on security overall.

TABLE V
MAIN MISSION MODEL CONCEPTS

Resource	Description
Mission	Description of mission requirements over data flows
Requirement	Specifies thresholds for cost and minimum security requirements for a data flow
MetricType	Type of mission metrics
Integrity	$\subset$ MetricType. Security constraint
Availability	$\subset$ MetricType. Security constraint
Confidentiality	$\subset$ MetricType. Security constraint
Latency	$\subset$ MetricType. Cost constraint via performance impact
Throughput	$\subset$ MetricType. Cost constraint via performance impact

E. Defense Model

The defense models describe which static and dynamic defenses are in place, what elements of the system they protect, what types of coverage they provide, and what cost is incurred. A single defense model can incorporate multiple defenses. Table VI shows the main concepts associated with models of cyber defenses. Different defenses operate over different types of nodes and thus the coverage relationship from a defense has a range of type Entity, which in the ASR ontologies inheritance hierarchy is the parent of all system-level nodes (processes, hosts, NICs, etc.). In this way, MTDs from Address Space Layout Randomization (ASLR) to IP Hopping can all integrate with the system model in a uniform manner, despite the fact that they protect very different elements. Defenses can be modeled both abstractly, such as a generic definition for a firewall, and at the specific implementation level (e.g., IPTables).

Thanks to the ability of OWL to incorporate inheritance, we can reap the benefits of reuse. We can define a generic IP hopping MTD that describes the capabilities and requirements common to all IP hopping defenses, and extend this definition to minimize the effort needed to model any specific implementations of an IP hopping defense. We can even

TABLE VI
MAIN DEFENSE MODEL CONCEPTS

Resource	Description
Defense	Description of cyber defense mechanism
DefenseType	Categorization into different types of defenses
Cost	Characterization of the overhead defense incurred
Degradation	$\subset$ Cost. Reduction in metric.
Requirement	Prerequisite requirements for installing the defense
Setup	Description of the defense's configurable items
Protection	Security guarantees provided by the defense
Reconfiguration-Detail	Description of dynamic behavior associated with MTDs
ProtectionDetail	Description of target entities being covered by defense
Randomization-Detail	Description of the randomization space

analyze this generic instance without reference to a specific implementation to provide insight into how the entire class of defenses operates. In order to support the dynamic nature of MTDs, the defense model provides support for the proactive elements of a defense to be described. An IP hopping MTD may be configured to change IP addresses of the included NICs every 5 minutes, for example.

Our current approach divides MTDs into three main kinds, and Table VII shows the set of proactive defenses currently modeled in ASR that cover two of the three categories:

1) Time-bound observable information on targets. In this category, MTDs place limits on the useful life of information obtained in an execution step for use in a later execution step. IP Hopping in the context of TCP Connection flooding is an example of this.
2) Masquerade targets. MTDs in this category make a target look like another kind of target, causing an adversary to spend extra cycles. OS masquerading is an example of this effect.
3) Time-bound footholds. MTDs in this category reset the escalated privileges that an attacker has built up along the middle of an attack path. An example of this is the use of virtualization and watchdogs to proactively and continuously restart VMs to clear out corruption.

TABLE VII
DEFENSES CURRENTLY MODELED IN ASR

Name	Kind	Requires	Side Effect
IPHopping	Time-bound observable	Network Endpoints	IP changes at fixed intervals
OS Masquerading	Masquerade	Host OS image	Host OS image fake
OS Hopping	Time-bound observable	Multiple OSs compatible with applications	Host changes at fixed intervals

F. Metrics Model

The metrics model enumerates all ASR metrics and defines each metric's name, the domain over which it is executed, and the SPARQL query used to compute it. ASR computes a diverse set of both system- and mission-based metrics over a configuration. Most metrics are computed by querying other models (e.g., to count the total number of listening endpoints or of attack vectors found). Some metrics are post-processed to compute statistical attributes such as mean (e.g. to compute the average estimated duration of an attack vector) or maximum or minimum values (e.g., to find the shortest attack vector).

These metrics are meant to give the user an overview of how well a system is protected against a set of attacks executed by a modeled adversary, as well as what costs (in terms of latency and throughput) are incurred by the modeled defenses. To facilitate this cost-benefit analysis, ASR provides users with some index metrics that can be used to judge a configuration's fitness at a glance, and compare fitness between alternative solutions. Figure 4 illustrates how the

	Security	Cost
System	Aggregate Security Index summarizes security metrics	Aggregate Cost Index summarizes cost metrics
	Individual concerns such as length of attack vectors are easily viewed	Individual concerns such as latency penalty of defense can be examined
Mission	Missions are ranked as *pass*, *degraded*, or *fail*. Lowest score prevails for aggregate mission scores along security, cost, or both	
	Individual concerns such as confidentiality still accessible	Individual concerns such as percent of dataflows that fail latency viewable

Fig. 4. High-level ASR metrics

metrics are separated into security- and cost-related concerns along one axis, and along system- and mission-wide metrics along the other axis. Security and cost are frequently at odds, with higher security necessitating a more expensive defense. A single value may therefore be misleading to a user because it could either represent the ideal case of high security and low cost, or the clearly undesirable outcome of low security and high cost. For these reasons, ASR provides the user with a separate single-value index reflecting the cost of any deployed defenses (the Aggregate Cost Index, ACI) and another single-value index reflecting the security score of the current configuration (the Aggregate Security Index, ASI). If a mission model is specified, a third index reflecting the fitness of the configuration with respect to mission goals is also computed (the Aggregate Mission Index, AMI). The index metrics are composed of several lower-level metrics, as shown in Figure 5. The desired metrics are specified in an OWL ontology, which is user-extensible and customizable. The metric computation is done through SPARQL queries for both simple and aggregate metrics, and the Jena API is used to invoke the metric computation from the ASR server and store the results.

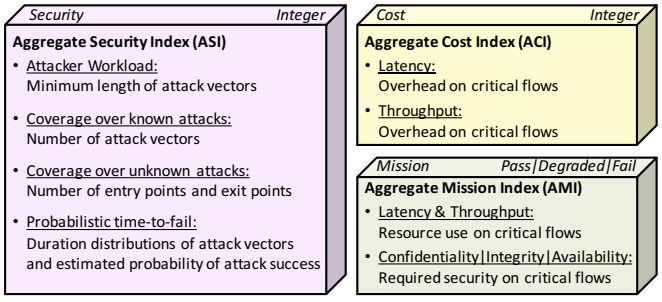

Fig. 5. The ASR index metrics take into account many submetrics

IV. EXEMPLAR APPLICATION OF THE ONTOLOGIES

To evaluate the modeling and reasoning performed by ASR, we developed an enterprise information sharing scenario involving several servers and both mobile and wired networks. Figure 6 shows the main actors participating in the scenario together with their interactions. An InformationProducer (e.g., a web camera) is sending videos and still images to a Website,

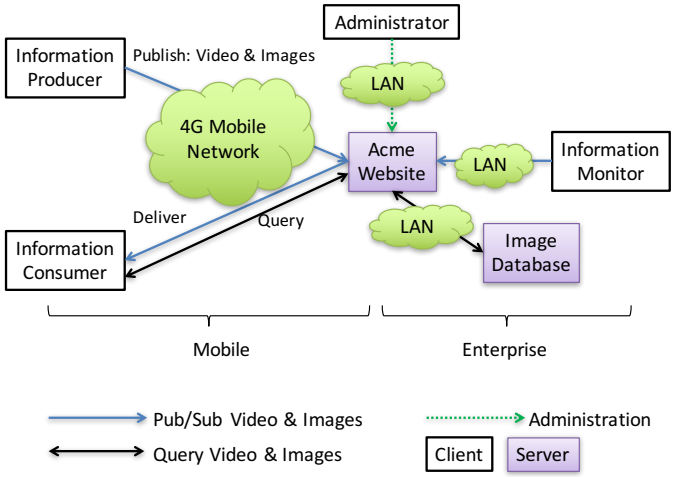

Fig. 6. Example information sharing scenario used to validate the approach

which in turn disseminates both video and images to two clients: an Information Consumer over a 4G mobile network and an Information Monitor over a Local Area Network. The Website is connected to an Image Database for persistence of images received. Finally, an Administrator can change settings on the Website through an administrative client.

A. Instance Models

Transcription of the components mentioned in the scenario involves creating instance models that are consistent with the ASR ontologies. To do this, we first define prefix shortcuts for name spaces as follows, using TURTLE:

```
@prefix demo1: <http://www.bbn.com/asr/demo1#> .
@prefix def: <http://www.bbn.com/asr/def#> .
@prefix sm: <http://www.bbn.com/asr/sm#> .
@prefix IPHop: <http://www.bbn.com/asr/iphop#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
```

The “Acme Website” host and its components can be expressed as:

```
% Acme Website from Figure 6
demo1:AcmeServer1
  rdf:type sm:Host ;
  rdf:type owl:Thing ;
  sm:contains demo1:Endpoint2 ;
  sm:contains demo1:ACME1 ;
  sm:hasImage demo1:OperatingSystem_1 .

 % Process running on the Acme Website Server
 demo1:ACME1
  rdf:type sm:Process ;
  rdf:type owl:Thing ;
  sm:connectsTo demo1:Endpoint2 .

 % NetworkEndpoint that ACME1 process connectsTo
 demo1:Endpoint2
  rdf:type sm:ListeningEndpoint ;
  rdf:type sm:NetworkEndpoint ;
  rdf:type owl:Thing ;
  sm:connectsTo demo1:MNE2 ;
  sm:hasResource sm:FileDescriptorPool_1 .

% Acme Website's MNE on the 4G Mobile Network
demo1:MNE2
  rdf:type sm:MNE ;
  rdf:type owl:Thing .
```

The MNE is plugged into a Mobile Network and there is a network flow coming in over that network that is expressed at three distinct layers that are linked through the “via” property.

```
% 4G Mobile Network from Figure 6
demo1:MobileNetwork1
  rdf:type sm:WAN ;
  rdf:type owl:Thing ;
  sm:contains demo1:MNE1 ; % Information Producer's MNE
  sm:contains demo1:MNE2 . % Acme Website's MNE

% Process-layer data flow from IP1 to ACME1
demo1:pDataFlow1
  rdf:type sm:DataFlow ;
  rdf:type owl:Thing ;

% Process on Acme Website defined above
  sm:destination demo1:ACME1 ;

% Process on Information Publisher from Figure 6
  sm:source demo1:IP1 ;
  sm:via demo1:nDataFlow1 .

% Underlying network-layer data flow
demo1:nDataFlow1
  rdf:type sm:DataFlow ;
  rdf:type owl:Thing ;
  sm:destination demo1:Endpoint2 ;
  sm:source demo1:Endpoint1 ;
  sm:via demo1:gDataFlow1 .

% Underlying physical-layer data flow
demo1:gDataFlow1
  rdf:type sm:DataFlow ;
  rdf:type owl:Thing ;
  sm:destination demo1:MNE2 ;
  sm:source demo1:MNE1 .
```

An Internet Protocol Address randomization (IP Hopping) defense is installed to cover the data flow between Endpoint 1 (the Information Producer) and Endpoint2, the Acme Website. The defense adds an additional data flow and processes for key synchronization. It also specifies necessary setup and configuration details and the incurred costs.

```
def:IPHopping1
  rdf:type def:Defense ;
  def:adds IPHop:DataFlow_pKeySharing ;
  def:adds IPHop:IPHoppingProcess_ACME ;
  def:adds IPHop:IPHoppingProcess_InfoProducer ;
  def:atCost IPHop:Cost_1 ;
  def:provides IPHop:Protection_1 ;
  def:requires IPHop:Setup_1 .

IPHop:Protection_1
  rdf:type def:Protection ;
  def:for demo1:Endpoint1 ;
  def:for demo1:Endpoint2 ;
  def:inSupportOf def:Confidentiality ;
  def:inSupportOf def:Discoverability ;
  def:through def:Randomization ;
  def:withSpecific IPHop:RandomizationDetail_1 .

IPHop:RandomizationDetail_1
  rdf:type def:RandomizationDetail ;
  def:disruptionLatency "5"^^xsd:float ;
  def:interval "10000"^^xsd:float ;
  def:space 6 .

IPHop:Setup_1
  rdf:type def:Setup ;
  def:includes demo1:Endpoint1 ;
  def:includes demo1:Endpoint2 .

IPHop:Cost_1
  rdf:type def:Cost ;
  def:impactOn IPHop:Latency_1 .

IPHop:Latency_1
  rdf:type def:MetricType ;
```

```
def:forProperty def:Latency ;
def:increase "0.3"^^xsd:float ;
def:on demo1:nDataFlow1 .
```

Further details and content for the remaining models, including attack steps, adversary, metrics, and mission, are included in the appendix to this paper and available at https://ds.bbn.com/projects/asr.html .

B. Quantification Results

To first step in quantifying an attack surface is creating a configuration containing the five model types and the metrics:

$C = (system, defense, attack, adversary, mission, metrics)$

The purpose of this evaluation was to study the impact of varying the hopping interval of one particular IP Hopping defense between slow and fast. To achieve this, we created three separate configurations where the only variable was the defense, as follows:

1) $C_{base} = (sm1, \varnothing, as_1, ap_1, mi_1, me_1)$
2) $C_{def1} = (sm_1, IPHopSlow, as_1, ap_1, mi_1, me_1)$
3) $C_{def2} = (sm_1, IPHopFast, as_1, ap_1, mi_1, me_1)$

Analyzing these three configurations using the ASR reasoning algorithms [2] yields the results shown in Table VIII. As a reminder, these index metrics are computed as weighted sums of several terms, as shown in Figure 5. Note that $IPHopSlow$ in C_{def1} and $IPHopFast$ in C_{def2} both add considerable cost compared to the base configuration, which contains no defense. This makes sense intuitively, since the latency penalty incurred by a defense with a shorter randomization interval (in this case, an IP Hopping defense that hops faster) is higher than the latency incurred by a defense with a longer randomization interval. The base configuration has no defenses deployed, so there is no latency penalty incurred and its ACI is therefore 0.

TABLE VIII
RESULTS OF ANALYSIS PERFORMED ON CONFIGURATIONS

Config	ASI	ACI	AMI
C_{base}	49.55	0	FAIL
C_{def1}	51.03	15.0	FAIL
C_{def2}	121.4	21.25	FAIL
C_{min}	MAX	21.25	DEGRADED

Also note that as $IPHopSlow$ in C_{def1} does not offer a significant security gain over the base configuration whereas $IPHopFast$ in C_{def2} doubles the ASI with respect to the base model. This is because in addition to submetrics that are computed over the base ontological models and do not change between the two configurations (such as the number of entry and exit points), the ASI also takes into account the probabilistic vector impact, which consists of vector duration distributions and their estimated probability of success. Intuitively, it makes sense that an IP Hopping defense that hops more frequently would provide better protection against a comparable adversary, since the adversary would have less time to complete a successful attack and would therefore be less likely to succeed. Figure 7 gives a primer on how the probability of success of attack steps and vectors is computed using the underlying ontologies.

For this example, suppose an attack step requires from 1 to 4 seconds to be successful (the duration distribution is part of the attack model) and we have a defense that hops every 1 to 3 seconds (this information is in the defense ontology). If the defense hops before the attack finishes, then the defense wins, else the attacker wins. Let us assume (for ease of computation) that both the attack step duration and the defense hopping interval are uniform random variables, which means that any number in the stated time range is equally likely and this will be captured in the sample data points. We also assume that these random variables are independent; intuitively this means that the attacker cannot detect when a hop has occurred and launch the attack immediately after the hop (which would give the attacker an unfair advantage). For this example, the probability density function for attack time needed will be

- $p_{attackDuration}(x) = \frac{1}{3}\ \forall x \mid 1 \leq x \leq 4$, and
- $p_{attackDuration}(x) = 0\ \forall x \mid x > 4$ or $x < 1$.

Similarly for defense we approximate

- $p_{defenseHoptime}(y) = \frac{1}{2}\ \forall y \mid 1 \leq y \leq 3$, and
- $p_{defenseHoptime}(y) = 0\ \forall y \mid y > 3$ or $y < 1$.

Lastly, the probability that the defense wins is computed as: $\sum(p_{attackDuration}(x) \times p_{defenseHoptime}(y)), \forall x, y \mid x > y$, which equals %66.7. Graphically, this is the normalized area to the right of the line $y = x$ in Figure 7, which represents the probability that the defense hops faster than attacker is able to successfully complete his attack.

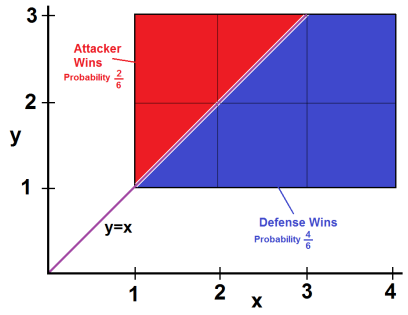

Fig. 7. A graphical representation of probability reasoning in ASR. The x axis represents the randomization interval of the defense. The y axis represents the duration distribution of an attack step that the defense is protecting against.

In addition to computing metrics, the ontologies are pivotal for another important innovation of ASR, its ability to semi-automatically minimize attack surfaces [2]. Minimization is supported through inspection and inference over all ontologies in a configuration. Two different modalities of attack surface minimization are supported:

- *System minimization* can find either entities that are not used within a system model (for instance an extraneous listening endpoint that no other endpoint connects to).
- *Mission minimization*, if a mission model is specified for a configuration, can find entities that are not defined to be mission-critical (e.g., an administrative interface that

is only used for the initial configuration of the system and never used during a mission).

Using the ontological models comprising a configuration and these two minimization modalities, ASR identifies all entities that can be safely removed and presents them to the user for selection. The user can select any or all of these entities to remove, and can save the minimized configuration for further inspection and analysis. Because removed entities may connect to other entities within the ontologies (e.g., an unused endpoint that is removed may result in an unnecessary process and its containing host, if they are not used for any other purposes), a second round of minimization may be necessary to remove all extraneous entities. The fourth configuration, C_{min}, in Table VIII is the fully minimized (i.e. with all extraneous and non-mission-critical entities removed) version of C_{def2}. Since the minimized configuration no longer contains all the entities that are not necessary (for instance, the Administrator host and associated processes, endpoints, and data flows), it has fewer entry points for an adversary to exploit and results in a higher security metric.

In all but the C_{min} configuration, the Aggregate Mission Index, AMI, is "FAIL." This is because none of them completely eliminate the attack vectors that threaten mission-critical resources. Only after minimization are all vectors are eliminated (thus the ASI score of "MAX"). The AMI is a single rating of mission health with respect to both security and cost and a single failing score on any requirement results in a failing score for the AMI. After minimization, the AMI improves from the initial "FAIL" score (initially the mission fails because of violated security requirements on mission-critical flows) to a "DEGRADED" score (the mission now passes all security requirements, but is "DEGRADED" on cost requirements). Intuitively, we have removed the security vulnerabilities that threatened the mission through deploying a faster defense and minimizing the attack surface. However, the improvement is only partial (the mission's rating is still "DEGRADED," not "PASS") due to the increased latency penalties incurred on mission-critical flows by an IP Hopping defense that hops more frequently.

We evaluated the runtime of the analysis algorithm with randomly generated models where the complexity of the models (i.e. number of hosts and other system entities and the number of available attack steps) vary in a controlled way. The points on the graph are averages of 5 runs for the same complexity configurations. The analysis time was measured on a MacBook Pro 2.8 GHz Intel Core i7 with 16 GB of RAM.

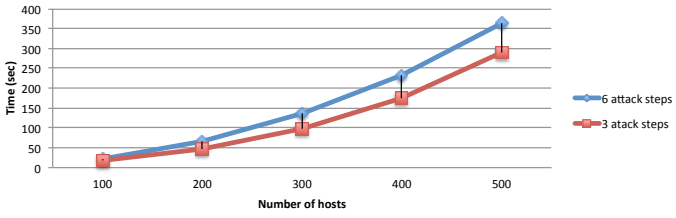

Fig. 8. ASR analysis runtime over system models of varying complexity

V. Conclusion

While it is common understanding that systems have attack surfaces and that those surfaces need to be minimized, the cyber security community has until now lacked a structured and generalizable approach for modeling attack surfaces and expressing associated security, cost, and mission impacts through concrete metrics. This paper presents ontologies including semantic models of attacks, systems, defenses, missions, and metrics, and supporting algorithms that quantify and minimize attack surfaces. An application of the ontologies on a concrete information-sharing demonstration scenario is also presented.

Next steps include extending coverage of the defense models beyond MTDs to include more traditional defenses, e.g., firewalls, VPNs, and host- and network-intrusion prevention systems. Furthermore, we plan to generate system models of realistic size systems, such as a model of the BBN network, which comprises hundreds of machines. Finally, we plan to improve the ontologies by including feedback provided by the cyber security research community.

References

[1] M. Atighetchi, B. Simidchieva, M. Carvalho, and D. Last. Experimentation support for cyber security evaluations. In *Proceedings of the 11th Annual Cyber and Information Security Research Conference*, page 5. ACM, 2016.

[2] M. Atighetchi, B. Simidchieva, N. Soule, F. Yaman, J. Loyall, D. Last, D. Myers, and C. B. Flatley. Automatic quantification and minimization of attack surfaces. In *The 27th Annual IEEE Software Technology Conference (STC)*, October 2015.

[3] N. Ben-Asher, A. Oltramari, R. F. Erbacher, and C. Gonzalez. Ontology-based adaptive systems of cyber defense. In *STIDS*, 2015.

[4] S. Fenz, T. Pruckner, and A. Manutscheri. Ontological mapping of information security best-practice guidelines. In *Business Information Systems*, pages 49–60. Springer, 2009.

[5] S. N. Foley and W. M. Fitzgerald. Management of security policy configuration using a semantic threat graph approach. *Journal of Computer Security*, 19(3):567–605, 2011.

[6] A. Herzog, N. Shahmehri, and C. Duma. An ontology of information security. *International Journal of Information Security and Privacy (IJISP)*, 1(4):1–23, 2007.

[7] S. Jajodia, A. K. Ghosh, V. Swarup, C. Wang, and X. S. Wang. *Moving target defense: creating asymmetric uncertainty for cyber threats*, volume 54. Springer Science & Business Media, 2011.

[8] L. Kohnfelder and G. Praerit. The Threats To Our Products, Apr. 1999.

[9] M. Loeb. Cybersecurity talent: Worse than a skills shortage, its a critical gap. *The Hill*, Apr. 2015.

[10] L. Obrst, P. Chase, and R. Markeloff. Developing an ontology of the cyber security domain. In *STIDS*, pages 49–56, 2012.

[11] H. Okhravi, M. Rabe, T. Mayberry, W. Leonard, T. Hobson, D. Bigelow, and W. Streilein. Survey of cyber moving targets. *Massachusetts Inst of Technology Lexington Lincoln Lab, No. MIT/LL-TR-1166*, 2013.

[12] A. Oltramari, L. Cranor, R. Walls, and P. McDaniel. Building an ontology of cyber security. In *9th Conference on Semantic Technologies for Defense, Intelligence and Security*. Citeseer, 2014.

[13] S. Ramanauskaitė, D. Olifer, N. Goranin, and A. Čenys. Security ontology for adaptive mapping of security standards. *International Journal of Computers, Communications & Control (IJCCC)*, 8(6):813–825, 2013.

[14] A. Razzaq, Z. Anwar, H. F. Ahmad, K. Latif, and F. Munir. Ontology for attack detection: An intelligent approach to web application security. *computers & security*, 45:124–146, 2014.

[15] M. B. Salem and C. Wacek. Enabling new technologies for cyber security defense with the icas cyber security ontology. In *STIDS*, 2015.

[16] A. Shostack. *Threat modeling: Designing for security*. John Wiley & Sons, 2014.