Computer Standards & Interfaces 33 (2011) 2–12

Contents lists available at ScienceDirect

Computer Standards & Interfaces

journal homepage: www.elsevier.com/locate/csi

Enforcing security in semantics driven policy based networks

Palanivel Kodeswaran *, Sethuram Balaji Kodeswaran, Anupam Joshi, Tim Finin

Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County, 1000 Hilltop Circle, Baltimore, MD 21250, United States

ARTICLE INFO

Available online 8 April 2010

Keywords:
Policies
Semantics
Networks

ABSTRACT

Security is an important requirement in scenarios such as mobile computing that allow users to make meaningful ad hoc collaborations. Traditional security solutions are not feasible for these scenarios due to the varying nature of the collaborations. We propose an extensible framework that takes the semantics of the collaboration into account and uses semantics driven policies for enforcing security. Our policies are rooted in semantic web languages which make them amenable to interoperability and high level reasoning. We describe our policy based network that exploits packet content semantics to secure enterprise networks and the BGP routing process.

1. Introduction

Security is emerging as an important requirement for a number of distributed applications such as online banking, social networking etc. due to the private nature of the data being involved. Furthermore, the wide spread use of portable devices such as laptops, PDAs etc. allows users to make meaningful ad hoc collaborations. Traditional security solutions are not feasible for these scenarios due to the varying nature of the collaborations in terms of entities involved and their roles, available resources etc. Under these circumstances, we need generic solutions that take into account the semantics of the collaborations in determining the set of allowable operations. In this work, we use policies for enforcing security as policies provide a generic and flexible framework that can later be easily modified based on changing requirements. We propose an extensible framework that uses semantics driven policies for enforcing security. Given the dynamicity of emerging computing environments, we want to be able to specify our policies at a high level such that we can focus on the abstract conditions and constraints that need to be maintained in the system. Also, given the heterogeneity of available devices, we expect that policy specifications should be as device independent as possible. In these cases, to enforce policies, an adaptation layer would be used to translate high level policy specifications into low level device specific primitives. Allowing automated reconfiguration of devices on the fly would require that the system be able to reason about policies and adapt them based on the new requirements.

We further propose that policies specified in semantic web languages can satisfy the above requirements. In our system, policies are specified using a combination of OWL and SWRL. The combination of OWL and SWRL can be used to define ontologies using which one can declaratively define facts, policies and rules in terms of what needs to be true or false for a policy to hold. In our system, policy specifications are in terms of SWRL rules which use high level concepts defined in appropriate ontologies, thus making the policy specifications generic, device independent and extensible. We can also specify meta-policies for guiding the interaction among policies. For example, we can use meta-policies to prioritize policies when multiple policies are applicable in a context. Further, we envisage that different organizations would have different policies at different granularities for the same device. By specifying policies in semantic web languages, devices would be able to reason over the policies and arrive at a configuration that meets the overall combined requirements. Also, rooting policies in semantic web languages makes dynamic reconfiguration automatic and easy, as new facts can be inferred from the policies.

We present our policy based network that reasons over packet content semantics for handling network traffic and show our framework can be used to secure enterprise networks and the BGP routing process.

The rest of this paper is organized as follows. Section 2 describes our content based tagging scheme. In Section 3, we present our semantics driven policy based network. Section 4 describes the rationale behind using semantic web languages for policy specification. Section 5 describes how security policies can be enforced in our framework. In Section 6, we present related work and finally we conclude in Section 7.

2. Content based semantic tagging

This section presents our packet level semantic tagging framework that enables intermediary routers to reason over the tags to determine how to best handle the data streams flowing through them. The idea

* Corresponding author.
E-mail addresses: palanik1@cs.umbc.edu (P. Kodeswaran), kodeswar@cs.umbc.edu (S.B. Kodeswaran), joshi@cs.umbc.edu (A. Joshi), finin@cs.umbc.edu (T. Finin).

0920-5489/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.csi.2010.03.010

behind our approach is to provide the routers visibility into the semantic content of packet data streams flowing through them. This content level information can then be used by the routers to enable intelligent routing and data handling decisions. Our approach differs from active networks in that the data streams merely provide additional meta-data while the network has complete control on how to use this meta-data. Thus network operators still retain complete over their network operations.

In our framework, we use RDF for labelling the semantic content. We choose RDF as it is very flexible, generic and its growing acceptance as the de-facto standard for meta-data markup. By utilizing RDF as the mechanism to markup flows/packets, intermediary intelligent routing entities can use this meta-data to reason over their knowledge base to determine how best to handle a given flow. Also, inferences can be made to generalize or specialize a given flow to best meet its demands. Furthermore, the use of a standard such as RDF enables interoperability among routers of different organizations since organizations may differ in their implemented mechanisms for providing the same service.

2.1. System architecture

We break our system architecture into two components: A node level and a System level that spans the network.

Fig. 1 shows the node level architecture of our framework. At the node level, we introduce an additional layer called the CoCoNet layer between the application and the transport layer. This layer is responsible for intercepting socket calls made by applications to the transport layer. The API is enhanced to allow the application to provide semantic level information for messages transmitted over this interface. A Local Policy Decision Point (LPDP) is used to determine what policies to enforce based on the content. In our framework, each Policy Enforcement Point (PEP) is at every layer in the networking stack while [1] treats the PEP at a node level. Placement of the PEP at every level of the stack allows us to implement coordinated cross-layer interactions initiated and controlled by our framework. The PEP exposes the interlayer optimization points that any particular layer supports. The framework utilizes the policies stored in the LPDP to drive the settings to be applied to each of the PEPs in the stack. Essentially, we are proposing to expose a network stack as a collection of switches and dials and allow an external policy to determine the exact settings of each of these dials (based on content and context). We want to expose functionality, not necessarily the mechanism of how it is achieved (this falls under intra-layer optimization). For example, a MAC can advertise two different data rates and their associated packet error probabilities without exposing the FEC scheme used to achieve these rates. The policies can be specified as production rules (if (condition) then (action)) or event–condition–action rules (on event if (condition) then (action)). In essence, the Node Framework provides a rich, extensible option for realizing policy controlled cross-layer interactions within a node's network stack. By parameterizing the possible set of interactions that are permissible, the cross-layer interactions are kept tractable without making the implementation overly convoluted [2].

Fig. 2 shows the network level architecture of our framework. At the network level, we envision that there will be an overlay network composed of routers that run the CoCoNet Router Framework. Client machines running our Node Framework communicate over this overlay. The overlay comprises of the following two components:

- A control plane component that involves interactions among the CoCoNet Router Layers at the routing elements.
- A data plane component through which the data packets are flowing.

Routers exchange traditional management information such as link states, buffer lengths, available bandwidth etc. in the control plane. Furthermore, additional information such as content types currently being handled and adaptations available can be advertised. An additional key piece of information that is exchanged is the local policies that are currently being applied to a data stream that is being routed. Local PEP settings for a given stream or flow have global implications. For example, unless every hop is reliable, a data packet cannot be reliably routed through a network.

The data plane can be implemented as either:

- A UDP connection between two routers.
- A TCP connection between two routers.
- An IP-in-IP tunnel between two routers.
- A layer 2 LSP.
- A DiffServ aware network.
- An IntServ aware network.

The CoCoNet Router Framework will perform the necessary mapping based on policy, content and context. For instance, suppose a packet arrives at a router indicating that it requires reliable transfer

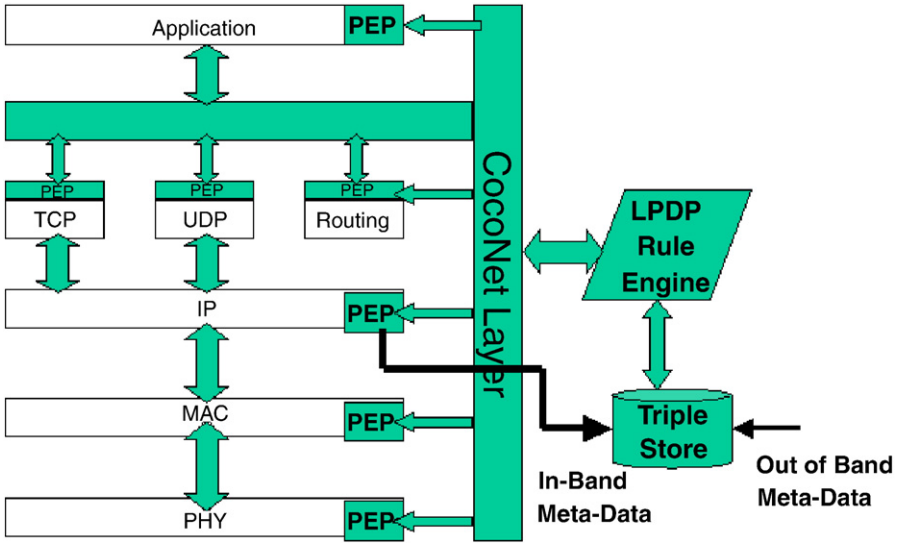


Fig. 1. CoCoNet Node Framework.

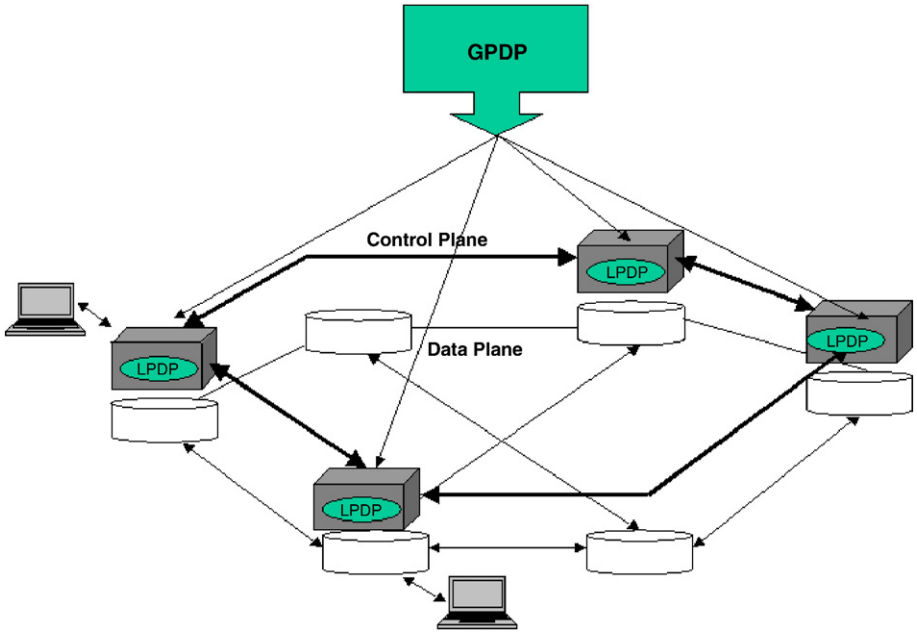


Fig. 2. Overlay network.

semantics. The data plane chosen to the next hop, in this case, could be over a TCP connection. Likewise, a data packet indicating that it is sensitive information (telnet logins for example) but currently not encrypted can be routed to the next hop over an IPSEC tunnel or dropped if none is available (if that is the policy). The choice of where a CoCoNet Router Framework runs is very implementation dependent. For example, in case of a wireless ad hoc network, every host is a router and hence can potentially run a (albeit simplified) CoCoNet Router Framework. Likewise, in an enterprise setting, the host machines within the enterprise will likely run only the Coconet Node Framework with only the exterior gateway routers running the Coconet Router Framework. A network service provider will most likely have only edge routers run the Coconet Router Framework leaving the core optimized for fast data flow handling.

The role of the Global Policy Distribution Point (GPDP) is to disseminate any network wide policies that need to be enforced. This can include items such as preferential treatment that needs to be given to content originating from a particular domain, preferential treatment for a particular type of content, any content based adaptation techniques that need to be employed in the network etc. It is envisioned that the GPDP is controlled by the ISP to set forth global rules while the LPDP hosted at an enterprise location is possibly shared between the ISP and the enterprise. This can further be extended to say that the LDPD is under local user control (based on user policies and preferences) and can additionally, host user preferences.

In order to propagate content level information for packets and flows, we propose to take one of the following approaches.

- The meta-data can be directly encoded into the IP options field of an IP packet (size is an issue). We refer to this as "in-band tagging."
- IP packets use the IP options field to carry a special key. This key is looked up in a directory service to identify the meta-data describing that packet or flow. We refer to this as "out-of-band tagging." In this approach, we will need to use a structured Peer-to-Peer overlay to enable key lookups. Before a client starts a flow through a network, it registers its content meta-data with the first hop router and generates a key. This key is carried in the IP packets and is available to any intermediary router. At any point along the data flow, this key can be used by intermediary routers to fetch the meta-data through an out-of-band mechanism.

The information conveyed in the meta-data is under the complete control of the application. For example, an MP3 stream may have the following description which can be used to differentiate between official and entertainment video streams.

```
<?xml version="1.0"?>
<rdf:RDF
 xmlns:rdf=
 "http://www.w3.org/1999/
                        02/22-rdf-syntax-ns"
 xmlns:mmschema=
  "http://www.mySchema.org/mms">
<rdf:Description
 rdf:about=
 "http://www.myContent.com/
                            SalesReport.mp3">
 <rdf:type rdf:resource=
  "http://www.mySchema.org/mmsaudio"/>
 <mms:LengthInMin>5</mms:LengthInMin>
 <mms:LengthInMB>4</<mms:LenghtInMB>
 <mms:technicalType>
  http://www.mySchema.org/mmsMP3
 </mms:technicalType>
 <mms:semanticType>
  http://www.mySchema.org/mmsLecture
 </mms:semanticType>
</rdf:Description>
```

Furthermore, providing content information so that a router can differentiate between, for example, video streaming from a surveillance camera and a streaming movie allows the network to make smart decisions on routing data streams across links with different reliability characteristics. Also, for our architecture, we are using RDF which provides a generic mechanism to convey meta-data which can be reasoned over. While we agree that semantic tagging could be an

expensive process, we would like to point to recent work [3] that shows that network level tagging can be performed efficiently.

3. Semantics driven policy based network

In this section, we present our policy based network built on top of the semantic tagging architecture.

Policy based networks employ mechanisms that allow network operators to specify rules at a high level, defining how packet flows are handled within a network, how network resources are allocated, access control restrictions and levels of service. The policies are enforced by configuring the network devices with the requisite primitives so that the appropriate actions are performed on the data streams. For example, one of the primitives may be routing matching flows through a middle box for security analysis.

One of the main challenges frequently faced is ensuring that network configuration settings are applied consistently throughout the network so that the correct actions are taken by the network devices; however, this is often error-prone and difficult to manage especially when there is a heterogeneity of network devices and management protocols. Additionally, policies that are commonly in use today are limited in their expressibility. Rules such as "allow traffic from A higher priority over B" and "permit user A" are easy to enforce but are limited in their expressibility. For networks to offer highly specialized services, administrators need to be able to specify more complex handling rules such as "allow security surveillance video streams higher priority than webcasts" (within an enterprise) or "downsample any video to user A so as not to exceed 128 kbps" (due to different levels of service or capabilities of the device associated with the user). For such policies, enforcement cannot be performed by packet header inspection alone as all the requisite details may not be directly accessible from the data packets as they are today.

To solve the above issues, we propose an alternate model to achieving policy based networks that provides fine grained services for network traffic as well as ease of network management. Our model employs the semantic tagging framework describe in the previous section within a formal framework for specifying rules that can be checked for consistency. The process of converting the rules to the lower level primitives understood by the network devices is also handled by the framework, thereby allowing the network administrators to focus only on defining the administrative policies. In our model, applications use the semantic tagger to encode data packets with descriptions conveying content semantics using the W3C Web Ontology Language (OWL) [4] as explained in the previous section. Ideally, the ontology used for this is provided by the network service provider. This description is encoded as a special header that is embedded into the data stream. Our motivation for using OWL (specifically, OWL-DL) is its capability to express formal semantics, define class hierarchies and their relationships, associated properties, cardinality restrictions while still retaining decidability and computational completeness. Using OWL for ontology specification makes the framework generic, flexible and more scalable than using proprietary labelling schemes that raise interoperability issues. For example, while one organization may invoke special procedures for handling surveillance video, another organization may have only a generic procedure for processing video traffic. In these cases, the class hierarchy offered by the ontology provides a clean interface for policy specification, where as flat labelling mechanisms may use a separate label for each different type of video traffic with no explicit relationship with the parent video traffic class. Furthermore, ontology based languages naturally support evolution, which makes them attractive for modeling network policies that change as enterprise goals change.

Our framework utilizes the W3C Semantic Web Rule Language (SWRL) [5] as the rule language which provides an easy to use mechanism for specifying event–condition–action rules which is the majority of rules envisioned for a typical network. Using our framework, content providers now provide meta-data to the network that can then be used by the network providers to determine how best to handle a given packet or flow that best suits that content. When a flow enters the network, CoConet routers running a reasoning engine reason over the OWL description of the incoming flow and apply appropriate network configurations to satisfy the matching policies.

An important corner piece of our framework is that operators retain control over the network and how traffic is handled within the network. This differentiates our approach from active networking [6,7] with respect to packet handling in that unlike active networks, the meta-data is not a contract on how the data should be handled but rather what the data is. The network provider retains complete control of how the packets are handled within the network and can fine tune policies to offer the best service for that type of content.

3.1. Policy architecture

We would like to note that we provide a generic framework where each node may have its own set of policies for handling different traffic types. Such policy differences may arise for a variety of reasons such as differences in node capabilities caused by the presence of hardware accelerators for certain operations, nodes belonging to different administrative domains following different policies etc. Therefore to ensure that network wide policies are properly enforced, the individual policies at each node need to be checked for consistency with the global policies. In this section, we describe how the distributed policies in our architecture are consistently maintained, thereby ensuring that network wide policies are properly enforced within the network.

The policy based networks managed using our framework are envisioned as a multi-tier system. A typical enterprise can be viewed as a collection of multiple Autonomous Domains (ADs) that may each be separately managed. Within each AD, there may be multiple sub-domains. We pursue a hierarchical policy architecture, in which policies can be classified as enterprise wide, specific to an AD or specific to a sub-domain within an AD. Policies in the system are distributed to the various ADs that are responsible for enforcement of those policies within that domain and all contained sub-domains. At the lowest level of this hierarchy is an adaptation layer that is responsible for translating the high level policies into low level protocol specific configuration routines that can be applied to the various network elements that are being managed.

Built into our policy architecture is a policy validation mechanism. All network management policy changes specifically, adding new policies, modifying existing policies and deleting policies are first examined for correctness and validity before being accepted into the system. This ensures the overall consistency of the system being maintained. The validation actions themselves are expressed through policies set forth typically by an enterprise network administrator. Fig. 3 illustrates the various components of our proposed architecture as applied to a generic enterprise comprised of several ADs managed by different administrators (this could be due to the departments being in different geographical locations, company policy etc.) and may potentially contain equipment from multiple vendors (our terminology is derived from [8]).

The *Enterprise Policy Data Store* (EPDS) is a central repository of all of polices that govern the enterprise. The EPDS contains a superset of all policies for each Policy Repository within the system. In addition, the EPDS stores any policies that govern the Enterprise Policy Arbitrator (EPA). Internally, the EPDS can be setup to be a hierarchical data store where for example, enterprise wide policies can be stored separate from departmental policies. Each departmental Policy Repository (PR), as part of its initialization, will contact the EPDS to obtain the set of policies that are relevant to this PR. This includes the enterprise wide policies and any department and sub-department

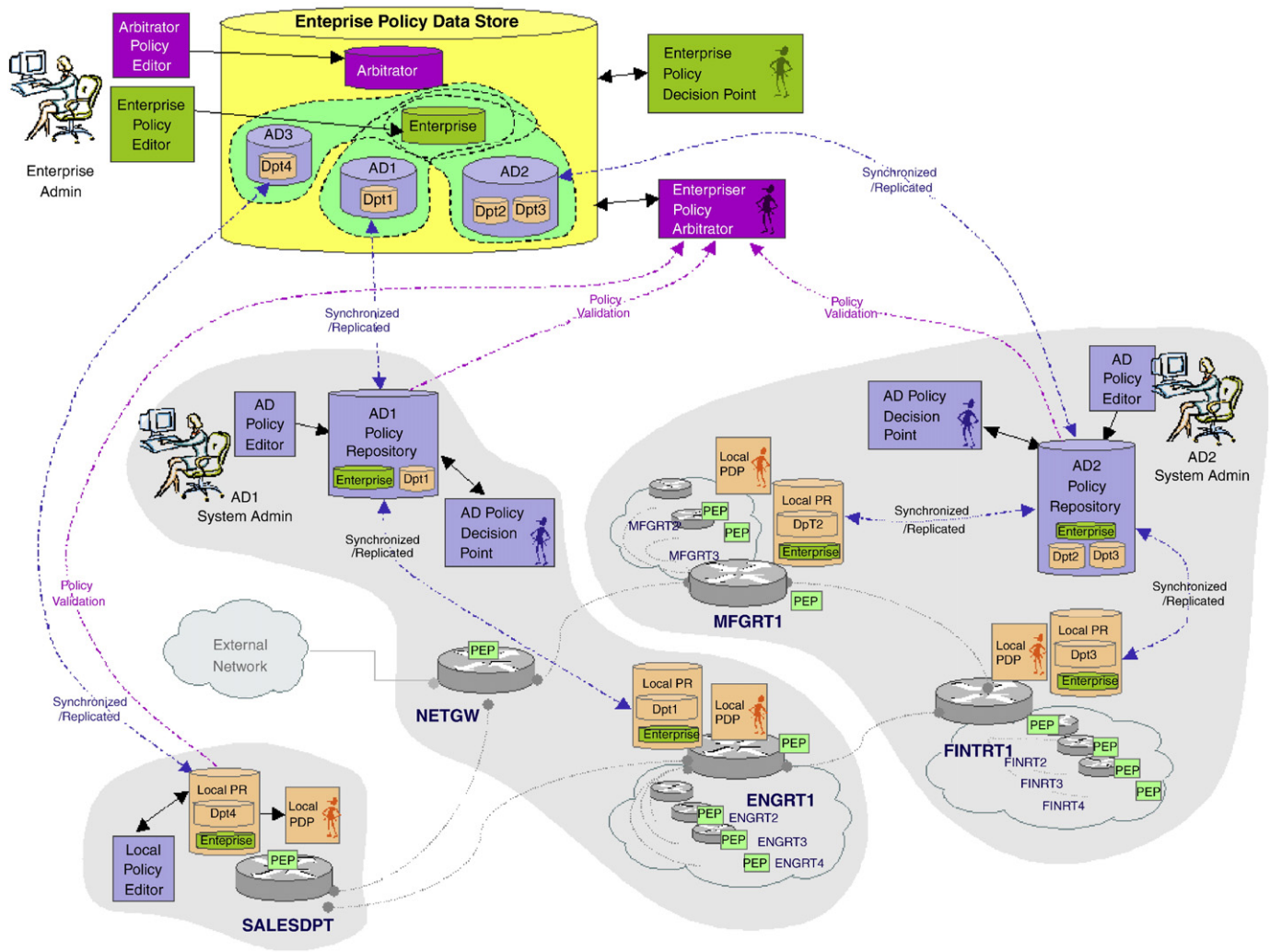


Fig. 3. Proposed architecture.

specific policies. The EPDS is constantly synchronized with the PRs in the system. Any policies approved for addition to a PR will be also forwarded to the EPDS so that if that PR were to crash, it can come back up and retrieve its original state from the EPDS.

The *Enterprise Policy Arbitrator* (EPA) validates any new policies that are being added/removed/modified to the system. The EPA is responsible for conflict resolution, dominance check, bounds check, relation checks, consistency checks, feasibility check etc. The EPA is governed by a set of its own policies that define how it does the arbitration, validation and other checks. In this manner, the EPA ensures that any policy entered into the system conforms to certain global system constraints. All policies that are submitted to a PR are first forwarded to the EPA for validation. Based on the response received from the EPA, the local PR either installs the new policy or rejects it.

The *Policy Repository* (PR) is a data store for a collection of policies. Typically this could be an AD specific PR (AD-PR), department or sub-department specific PR (referred to as local PR) etc. The AD-PR contains all policies specific to the enterprise and any policies specific to all departments (and sub-departments) that are part of this AD. Each PR in the system is synchronized with its parent PR and at startup, retrieves all its policies from this parent. In this hierarchy, the EPDS is the root parent. An AD-PR will retrieve all policies from the EPDS including all enterprise wide policies, all AD specific policies and all policies for any contained departments or sub-departments. Similarly, a departmental local PR on startup will contact an AD-PR to retrieve all relevant policies, including enterprise specific policies, AD specific policies and any department specific policies. Polices can be added, deleted or modified from a PR through a Policy Editor. Any such changes (regardless of whether the change is in an AD-PR or a Local PR) are first forwarded to the EPA for validation. If they are consistent, the EPA applies these changes into the EPDS. This addition will propagate to the PR chain so that all the PRs in the hierarchy are updated.

The *Policy Decision Point* (PDP) is the entity responsible for reasoning over the network traffic utilizing the content meta-data, network state and other contextual information available to it and determining the policies that need to be enforced. Each PDP operates with policies that are stored in a corresponding PR. PDPs can be at different levels, administrative domain (AD-PDP), department or sub-department specific PDPs (local PDP) etc. The PDP is responsible for reasoning over the policies (using its Configuration Reasoner) in its local PR and translating them into commands that can be sent to PEPs for enforcement (to the PEP's Configuration Conformance Enforcer). Additionally, the PDP is responsible for reacting to events coming from managed PEPs or subordinate local PDPs that cannot be resolved at the local PDP level. In this manner, the PDP acts as the decision making entity within the framework, the decisions being made at multiple levels depending on the severity of the trigger. The PDPs closer to the device deal with policies that are low level, fine grain and possibly device/protocol specific and the PDPs higher up the tree deal with more abstract and aggregate policies.

The *Policy Enforcement Point* (PEP) is the entity responsible for enforcing the policies at the device level. It resides on the managed devices and is responsible for installing and monitoring the health and status of a network device. PEP's main responsibilities and actions are:

- To request and store its configuration from the local PDP that is responsible for this device.
- To delegate any policy decisions to the local PDP by extracting content meta-data from data packets and adding to this description, any additional information that may be useful to the local PDP.
- Report errors and status updates to the local PDP.

The PEP is a vendor/device specific network management protocol. The PEPs collectively form the adaptation layer to abstract any device specific details into a normalized interface represented using the standard system ontology.

The *Network Ontology* (NetOnto) is the OWL ontology specified by the service provider or enterprise that is used to mark up the content of data packets conveying information such as application profiles (security requirements, delay, jitter etc.) and user profiles (customer paying more for service, end device capabilities). By using OWL rather than simple XML, the language is semantically richer and highly extensible which is very important especially when we have interdomain interactions (such as peering arrangements, SLAs etc). Policies are written using the concepts defined in NetOnto using SWRL as the rule language. A PEP extracts the content semantics description carried in the packet header and adds to it, any extra contextual information including aspects such as network state (congestion, link failures etc), network technology (wired, hybrid, MANET, cellular) etc. This information is then sent to the PDP and actions are invoked based on the response. The response back from the PDP will cause specific configuration to be installed by the PEP on the device (for example, updates to IPtables, addition of static routes and priorities, setup a label switched path etc).

The *Policy Editor* (PE) is the component that is used by a system administrator to view, add, modify and delete existing policies. The interface is typically a GUI allowing for ease of operation. PEs can be at different levels. The Enterprise Policy Editor (E-PE) provides a way for an enterprise system administrator to specify enterprise wide policies. An Arbitrator Policy Editor (A-PE) allows an enterprise system administrator to specify policies that drive the EPA. Similarly, an AD Policy Editor (AD-PE) allows an ADs system administrator to specify AD specific policies. In general, a user uses the PE to request changes to be made to a PR. This request will pass through the EPA validation and the user receives an acknowledgement of whether or not the requested change is allowed or rejected. Additionally, the PE also offers views of the managed network such as topology views, status of network devices and links etc.

Utilizing this framework, devices can now be deployed in a network and policies specified that can provide specialized services (content adaptation, forwarding priorities, resource reservations etc) to data packets as shown in Fig. 4. At each interim device, packets are inspected to see if they carry a semantic header. These packets are then offloaded to a separate forwarding path. The semantic header is extracted, any additional contextual detail available to the PEP is added to complete the OWL description and sent to the PDP for reasoning. We use OWL's abbreviated XML encoding format for this purpose. The PDP runs a reasoner that takes the OWL description and SWRL rules to determine the actions that need to be initiated. This information is then conveyed back to the PEP to be applied to the offloaded packets (and possibly to the express lane) to realize the necessary policies.

4. Design of policies rooted in semantic web languages

There are several reasons motivating us to root our policy specification and enforcement mechanisms in semantic technologies. Specific to the domain of networking, for any successful policy language, it must be universally interoperable considering the number of various organizations (enterprises, ISPs, networking vendors etc) that must interact to power a large scale network. In addition, if the system needs to be capable of automatically processing, reasoning over and responding as appropriate, the language must be machine-interpretable with understandable syntax and semantics for expressing data, rules and constraints on networks, networking devices, hardware components, software protocols, user applications and end users. Furthermore, it must be easy for users to specify policies. In this regards, a declarative policy language that enables each authority to draft abstract policies in a high level language to guide activities of the networking enterprise is a good candidate. Each authority can define only those objectives and constraints that are relevant to its needs. The policies represent rules and constraints that are necessary for the target network infrastructure to be valid. This information contained in the policies is defined in a manner that is as hardware, software, and protocol independent as possible. Therefore, the authorities need not focus on writing procedures for configuring a specific infrastructure; instead they can focus on describing a generic infrastructure and its features without needing to master and understand each of the various device/protocol/system specific mechanisms. The policy software components embedded or in the vicinity of each of the networked devices can convert the specified policy into device specific settings and configurations. Furthermore, the policy language must be capable of supporting changing network goals as well as run time policy interactions.

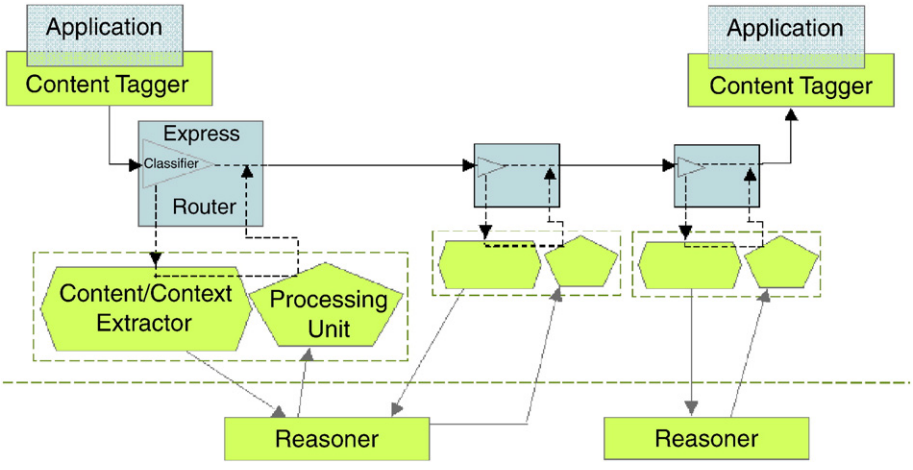


Fig. 4. Packet flow.

We believe that the combination of the W3C Web Ontology Language (OWL) and W3C Semantic Web Rule Language (SWRL) standards is applicable for policy control as it is machine understandable, sound, complete, extensible through additional ontologies, and supports heterogeneous application domains. OWL has axiomatic and model-theoretic semantics, which allows for verification of knowledge expressed in OWL constructs. In our work, we have chosen to use a subset of OWL, namely OWL-DL as it is complete and decidable. OWL + SWRL can be used to define ontologies, using which one can declaratively define facts, policies, and rules in terms of what needs to be true or false for a policy to hold. SWRL specifies OWL-based abstract syntax and vocabulary for representing Horn-like rules. SWRL defines a rule as an implication from a set of antecedent atoms to a set of consequent atoms. In our work, the policy language uses the antecedent atoms for representing policy constraints. The language uses the consequent atoms for defining directive actions that apply whenever the constraints are satisfied by evaluating information stored in a local knowledge base and by executing relevant attached procedures. The ontologies can be extended for declaratively capturing any concept or predicate without changes to the underlying system capable of processing OWL and SWRL. The language can be further extended by defining functions as procedural attachments and mapping them to predicates in OWL ontologies. This allows for the policy enforcement mechanisms to process functions thereby enabling the system to exploit low level, optimized implementations which is particularly important in the domain of networking. Extending on our current framework, in order to support multiple policies, we can also define a vocabulary for creating meta-policies. Meta-policies are used for guiding the interaction among policies. The meta-level vocabulary defines constructs for resolving conflicting, overlapping policies. For example, the meta-level vocabulary can be used to create a default conflict resolution rule such that a prohibitive policy overrides a permissive policy. At the same time, the meta-level vocabulary also allows one to define absolute and relative prioritization of policies, thus overriding the default rule. The meta-policies provide an automatic conflict resolution diagnosis in order to respond to situations when policies presented to a network impose conflicting conditions on the overall infrastructure or on one specific component. Additionally, the policy software components embedded or in the vicinity of each networked device can use this meta-information to automatically merge policies from multiple authorities and generate a target configuration that meets the combined requirements. The components follow the semantics defined by the policy language. Consequently, their steps in merging policies can be formally verified using a theorem-proving model. In order to combine multiple policies, the language depends on closed-world assumption reasoning. In this case, the system assumes that all rules are to be evaluated only by the knowledge contained within a knowledge base. This allows a reasoning engine to yield a solution in a finite time.

By utilizing semantic technologies to drive our framework, we can now realize dynamic reconfiguration of knowledge as new facts can be inferred through the policies specified. Current relational technologies and those based on static schema are dependent on pre-existent knowledge and do not offer this flexibility. There are a plethora of tools available to drive the ontology specification, verification, reasoning engine, etc. that can be incorporated to build such a system that can be deployed on a large scale.

5. Enforcing security policies in the proposed framework

5.1. Securing enterprise networks

The use case we consider in this work is that of a secure enterprise that wants to enforce prioritization on types of content that can flow across the links comprising its network [9]. We assume the enterprise has profiled its network and assessed security credentials to all the links and routers. As an example, a link that is fully within the premises of the enterprise (physically secure) is assessed as a "safe" link, one that is a VPN running over an external service providers network may be assessed as "potentially unsafe" while a wireless RF hop may be assessed as "unsafe." The enterprise applications are "smart" and can encode content level tags into the data packets that carry semantic information about the content as well as the application/user/device. For this example, as we are interested in the security semantics, applications additionally provide information about the sensitivity of the content (such as secret, top secret or normal), type of content and the security credentials of the context within which they run. For such an enterprise, the following policies would be appropriate:

- Only "Safe" links can be used to carry "TopSecret" data
- All data over "Open" links need to be encrypted
- Restrict multimedia flows in the network to max of 75% link capacity
- Allow admin traffic preferential service over network backups
- Allow user access to data only if user clearance is high enough

5.1.1. Simulation toolkit

We used NS2 to simulate such an enterprise. The network topology considered was a random network with links classified with a "security" tag that defined their safety levels. We assume the nodes belong to a single Autonomous Domain (AD) and run a link state routing protocol. We modified the standard FTP/CBR applications to allow for the specification of semantic descriptions into the packet streams. For the Network Ontology, we used Protege as the editor for specifying our ontology. We use Jess [10] as the reasoning engine in our framework although any other reasoning engine could be used as desired.

To begin, we defined an ontology to represent our enterprise. The ontology is available online at [11]. In our implementation, our ontology also contains special instances of classes representing the various actions that a PEP should take such as dropping data, encrypting data etc. These special instances also contain the low level primitive commands that need to be invoked to realize the necessary behavior. In our case, these commands are expressed as a snippet of Tcl code that can be evaluated by NS2. For example, a policy such as *All unencrypted secret data over "open" links need to be encrypted* can be expressed logically in SWRL as:

```
DataTraffic(?d) ∧
datasensitivity(?d,?sensitivity) ∧
Secret(?sensitivity) ∧
encryptionstatus(?d,?encryptstatus) ∧
UnEncrypted(?encryptstatus) ∧
nextHop(?d,?nexthop) ∧
securityLevel(?nexthop,?securitylevel) ∧
Open(?securitylevel) ∧
EncryptData(?action)
  → inferredAction(?d,?action)
```

The *EncryptData* instance has the following Tcl command encoded in it indicating the device understandable actions that need to be taken.

```
set clsfr[ get-classifier $interimRouterId]
$ns at [$ns now] "$clsfr install-interceptor
     encryptdata $flowid $srcId $sport
     $destId $dport $qdelay $overhead"
```

Using this methodology, we can now define the various actions that a Policy Enforcement Point (PEP) could take and assign to each of these actions, the corresponding primitive commands (Tcl snippets). The Policy Decision Point (PDP) was implemented as a Java process that received OWL streams from a client PEP (a network router within NS2), invoke the reasoner and send back the Tcl commands

depending on the actions that needed to be invoked. The PEP (NS2) would then execute the commands received from the PDP.

5.2. Secure routing

In this section, we show how security can be incorporated into the BGP routing process using our framework [12].

5.2.1. BGP extensions

Border Gateway Protocol (BGP) was originally designed as a simple path vector protocol to share routing information between autonomous systems (ASs) which has today, become the de-facto interdomain routing protocol enabling the Internet. Autonomous systems (ISPs, enterprises etc) use policies to define how the routes are to be shared and among which peers. These policies can be driven by various factors such as commercial peering agreements, security considerations, load balancing requirements etc. These policies are then implemented in the network routers as configuration parameters to control the protocol behavior. One of the main challenges frequently faced is ensuring that network configuration settings are applied consistently throughout the network so that the correct actions are taken by the network devices both within an autonomous system and across boundaries. However, this is often a manual process that is error-prone and difficult to manage.

To apply our framework to provide BGP route dissemination that takes into account the security credentials and external relationships, we needed to make two modifications to the protocol. The first modification is aimed at establishing identity of the BGP peers in a secure and verifiable manner. For this purpose, we assume the BGP session establishment process is extended to include the sharing of signed credentials to validate the identity of the BGP peers and their affiliations. Prior work such as S-BGP [13] have shown that this is feasible using a public key infrastructure and signed certificates. This modification is necessary as it is important for a BGP router to establish the identity of its peer so that the routes learned from and advertised to this peer can be handled correctly. The second modification is to include with the route advertisement in the BGP update messages, an additional optional and transitive attribute that conveys semantic meta-data about that NLRI. The intent here is for the originating AS to provide this information to allow other nodes to handle this route appropriately. The interim routers are allowed to add to this description as necessary (keeping the original intact) in a manner that is secure and cannot be repudiated. In this work, we are concerned about the import/export policies in use in the BGP decision process. The modifications allow our framework to, for each route that is being advertised to or learned from, contact a PDP, the PDP will reason over the semantic information provided for that route and the policies that need to be enforced, and will communicate to the BGP node whether or not, that route can be shared or accepted.

5.2.2. Use case

The use case we consider in this paper is that of a secure version of BGP where there are constrains on route exchanges between BGP peers. As with the real Internet, BGP nodes are owned by different agencies that have different affiliations. During the initial session establishment, nodes exchange their identity information to indicate the agencies to which they belong. These agencies or organizations have external socio-economic, political or financial relationships that will influence the BGP nodes in their exchanges. Routes advertised by each AS is tagged with additional semantic information to describe aspects such as its confidentiality, sharing restrictions etc. For such a use case, the following policies would be appropriate:

- Routes marked as “ShareWithFriendly” can only be exchanged between routers that belong to organizations that have a collaborative relationship.
- Routes marked as “Restricted” can only be shared between nodes that belong to the same parent organization (even if they are different divisions of that organization).
- Routes marked to be used only for data backup traffic are installed only during non-peak hours.
- Allow a route to be used only for data traffic that has a specified or higher clearance level.

5.2.3. Simulation toolkit

We used the ns-BGP [14] extension to NS2 to implement our framework. The network topology considered is a linear network with nodes grouped into various ASes. Each node is initialized with credentials that specify what organization the node belongs to. We modified the BGP session establishment process to allow the exchange of these credentials so that the BGP nodes can establish the identity and affiliation of the peers that they are interacting with. We added an additional optional transitive attribute to the BGP update protocol messages intended to convey additional semantic information about the route.

To begin, we defined an ontology to use for our BGP example. The ontology is available online at [11]. We modeled the various BGP protocol messages and constructs. Since we are dealing with import/export policies, we modeled special instances of classes representing the various actions that a BGP router (PEP) should take such as whether a route should be advertised or not, whether a route should be accepted or not etc. These special instances contain the low level primitive commands that need to be invoked to realize the necessary behavior. In our case, we implemented handlers in the NS2 implementation to handle the response coming back from the reasoner to determine whether a route should be included in an advertisement or whether a route that was received, should be accepted (these commands are expressed as snippets of Tcl code that are evaluated by NS2). For example, a policy such as All routes are shareable with a peer as long as the peer and the originating router are owned by the same organization can be expressed in SWRL as:

```
BGP_Update(?adv) ∧
interimRouter(?adv, ?routeadvertising) ∧
dest(?adv, ?peer) ∧
owner(?routeadvertising, ?org) ∧
owner(?peer, ?org) ∧
AllowRouteAdvertisement(?allow)
  → inferredAction(?adv, ?allow)
```

The *AllowRouteAdvertisement* instance has the following Tcl command encoded in it indicating the device understandable actions that need to be taken.

```
set Response "OK"
```

In this case, if the reasoner asserts this rule, the corresponding Tcl command will be sent back as the reasoner's response. Using this methodology, we can now define any arbitrary action that a PEP could take and assign to each of these actions, the corresponding primitive commands (Tcl snippets) to be executed. The PDP (reasoner) was implemented as a Java process that received RDF streams from a client PEP (a BGP agent within NS2), invoke the reasoner and send back the Tcl commands depending on the actions that needed to be invoked. The Protege IDE served the role of a Policy Editor.

Using this framework, we implemented our typical use case scenario focusing on the import/export policies for BGP. For our example, we consider a network of four autonomous domains with five BGP routers as shown in (Fig. 5). The Autonomous Domain AS0 belongs to UK forces. The Autonomous Domains AS1 and AS2 belong to two organizations within the US military. Finally, the last Autonomous Domain AS3 belongs to Russian military. During the initial BGP session establishment, the identity of each of the peers is

established. This indicates the organization that the router belongs (US_{Milcom}, UK_{Milcom}, $Russian_{Milcom}$ etc) which is tracked in the "owner" property of the network devices. Some of these organizations have external relationships (such as NATO to which US_{Milcom} and UK_{Milcom} belong). Such external relationships are modeled through OWL restrictions on properties. For example, a device that is part of NATO is modeled as a one where there is a necessary and sufficient constraint that the owner is either an instance of US_{Milcom}, UK_{Milcom} or $France_{Milcom}$. Each router that originates a route includes a description that at the least, indicates the sharing restrictions for that route. In the current version, we have values such as None (which is similar to the "internet" community attribute in BGP), Restricted and ShareWith-Friendly as examples. The intention here is that a route marked as "ShareWithFriendly" can only be shared with a peer who can be considered friendly. For example, if we considered forces within NATO to be friendly's, a SWRL policy to permit the routes marked as "ShareWithFriendly" to be exchanged could be written as:

```
BGP_Update (?adv) ∧
interimRouter(?adv, ?routeadvertising) ∧
dest(?adv, ?peer) ∧
NATO_Forces (?routeadvertising) ∧
NATO_Forces (?peer) ∧
routeRestriction(?adv, ?restriction) ∧
ShareWithFriendly(?restriction) ∧
AllowRouteAdvertisement(?allow)
  → inferredAction(?adv, ?allow)
```

Once the simulation starts, each router advertises its routes with its peers in order to compute its routing table. The simulation proceeds until all routes are computed and the routers settle on their tables. Note that when two routers belonging to UK_{Milcom} and US_{Milcom} (AS0 and AS1) are in a BGP session and while none of the routers have explicitly been identified as belonging to NATO, the reasoner can deduce this relationship and allow route exchanges between them. Similarly the reasoner can deduce that the route exchange cannot be allowed between AS2 and AS3 as they do not have an explicit relationship that permits this.

Fig. 6 is a snapshot of the system with the nodes contacting the reasoner to determine if routes can be exchanged and the responses received.

In this manner, we can now setup arbitrary relationships between routers and can specify policies through higher level rule based mechanisms to implement fine grained control over the protocol. This example can be easily extended to scenarios where the relationships are short lived and arbitrary such as in emergency response scenarios (where organizations may temporarily want to share information for providing quick response), application need driven (such as for supporting live event feeds) etc. by extending on the ontology and defining the desired policies.

5.3. Mapping enterprise level trust relations into the data forwarding plane

Our architecture allows exercising fine grain control over both the routing and data forwarding planes. The previous section showed how enterprise level security policies could be mapped into appropriate routing policies. In this section, we focus on mapping prior enterprise level trust relations into appropriate policies at the data forwarding plane. One of the goals of our architecture is to allow operators to declaratively specify how to handle network traffic. For example, operators should be able to specify that sensitive data be sent along a secure path, with the network automatically finding such a path. A secure path would ideally contain a sequence of optical links or encrypted data over open links. We could augment this definition of a secure path to factor in the trust relations between the link owner and data source. For example, in an enterprise architecture we could require that sensitive information be encrypted at the network layer only if the next hop link is not owned by the organization. This rule could be expressed in SWRL as:

```
   DataTraffic (?d) ∧
datasensitivity (?d,?sensitivity) ∧
Secret (?sensitivity) ∧
src (?d, ?src) ∧
owner (?scr,?org)
nextHop (?d,?next) ∧
zEnd (?next,?end) ∧
owner (?next,?org) ∧
owner(?next,?org2) ∧
notEqual(?org,?org2) ∧
encryptionstatus(?d,?encryptstatus) ∧
UnEncrypted (?encryptstatus) ∧
securityLevel (?next,?securitylevel) ∧
Open (?securitylevel) ∧
EncryptData (?action) → inferredAction (?d,?action)
```

So in this manner, we can allow operators to write high level policies that take into account both the security semantics of data as well as existing trust relationships among enterprise entities.

6. Related work

Policy based networks and approaches have been the focus of extensive research in recent years. Quality of service oriented initiatives such as Intserv [15] and Diffserv [16] rely on policies to drive flow classification, admission control, resource reservations etc. However, the policies used are limited in their expressibility and restricted to traffic forwarding semantics with little support for features such as content adaptation, specialized routing etc. In this respect, the active networks [6,7] took the approach of allowing a more generic per packet handling semantic with the packets determining what the router should do with them, which differs from our approach in which the router (using its specified policies) controls how the packet is handled and not the other way.

There has been significant research on securing BGP. SBGP [13] proposes a comprehensive architecture for securing BGP using public key certificates. SBGP uses a pair of PKIs, one for address authentication and the other for route validation. SoBGP [17] provides more flexibility compared to SBGP. In addition to the above PKIs, a third type of certificate is used which provides routing policy and local topology. When a route is received, it is compared for consistency

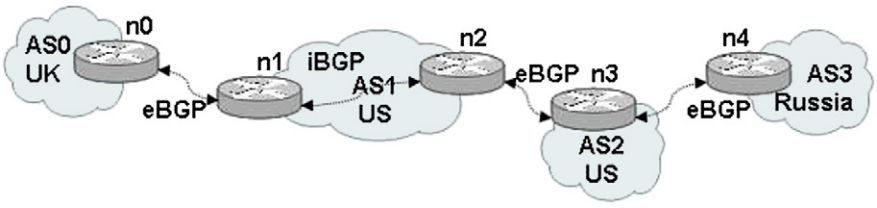


Fig. 5. Topology.

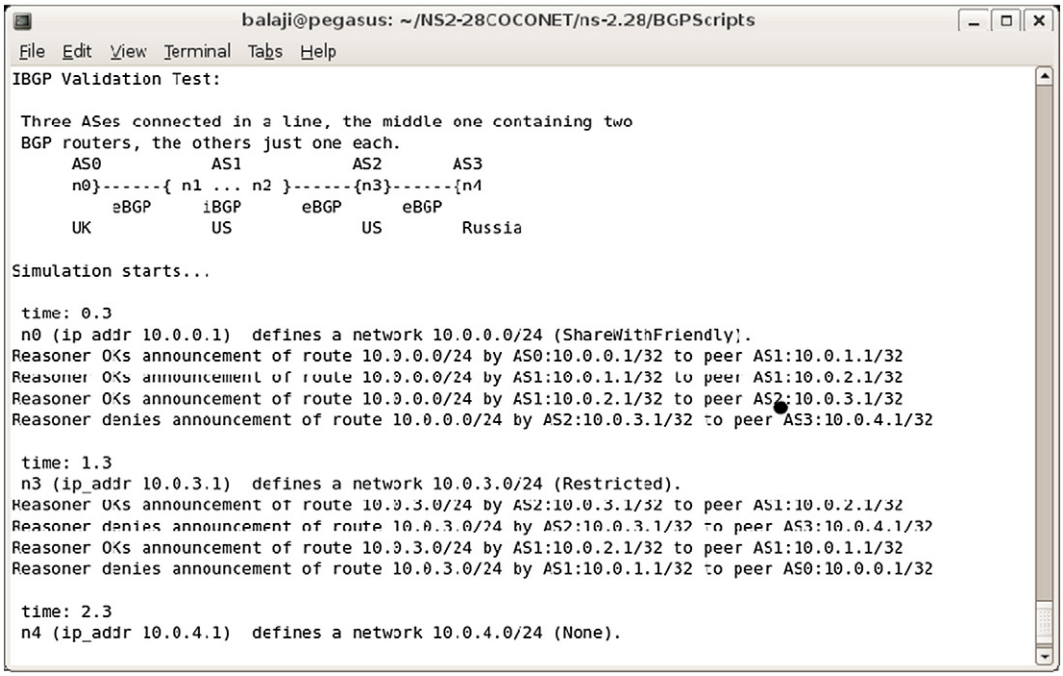


Fig. 6. Simulation output.

with the topology database and dropped if found to be inconsistent. The architecture is more flexible as there are no fixed structures of authority and ASes can decide on their own for accepting routing announcements and policies. RPSL [18] is an object oriented language for specifying routing policies from which router configurations can be automatically generated. RPSL generated router configurations can aid in preventing internet router misconfigurations but it does not support inference and is limited in expressibility.

KaOS [19] was one of the early languages to use ontologies for policy specification. The core policy ontology had support for modeling actors, action classes, groups, places and related attributes. KaOS also had support for policy conflict detection and harmonization. Uszok et al. [20] describe a later version of KaOS with its three layer policy management hierarchy designed at ease of use for both users and application developers. Kagal et al. describe Rei, an ontology based policy language developed for pervasive computing environments in [21]. Rei is based on deontic concepts of rights, prohibitions, obligations and dispensations having explicit support for speech acts. Nejdl et al. propose an ontology based policy language for enforcing security policies in [22]. Tonti et al. compare various semantic web based policy languages in [23]. Similarly, Carminati et al. propose using semantic web based languages for enforcing user privacy in social networks based on trust relationships in [24]. There is a large body of work related to creating ontologies for reasoning about security such as those in [25–28].

There have been recent efforts in using the semantic web for security applications. The authors in [29] propose using a combination of conventional security mechanisms and the ability to reason about security at a semantic level for enforcing security in autonomous systems. Also, they describe a set of requirements that need to be supported for implementing a semantic firewall. [30] proposes using context as the first principal for policy specifications governing access control in pervasive environments. Their approach stems from the fact that traditional subject/role based policies wouldn't work in pervasive environments due to the ad hoc mode of collaborations, where the roles and identities of the entities involved is not known ahead of the actual collaboration. They also propose using semantic languages for policy specification to aid in policy reasoning, conflict resolution and policy adaptation.

7. Conclusion

In this work, we have described an extensible security framework that is based on policies. Our policies are specified in semantic web languages which make them amenable to interoperability, conflict resolution and reasoning. We described our policy based network built on top of semantically tagged packets. In our framework, applications semantically tag packets with meta-data describing the contents being carried. We then showed how our framework can be used for securing enterprise networks and BGP routing.

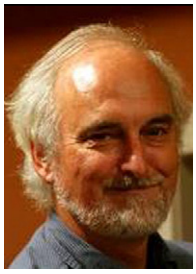
References

[1] R. Yavatkar, D. Pendarakis, R. Guerin, "RFC 2753: A Framework for Policy-based Admission Control," Jan. 2000. [Online]. Available: http://www.faqs.org/rfcs/rfc2753.html.
[2] P. Kawadia, V. Kumar, A cautionary perspective on cross-layer design, Wireless Communications, IEEE [see also IEEE Personal Communications] 12 (1) (Feb. 2005) 3–11.
[3] A. Ramachandran, Y. Mundada, M.B. Tariq, N. Feamster, "Securing Enterprise Networks Using Traffic Tainting," (2009).
[4] D.L. McGuinness, F. van Harmelen, "Owl web ontology language overview," W3C Recommendation 10 February 2004, Tech. Rep., 2004. [Online]. Available: http://www.w3.org/TR/owl-features/.
[5] I. Horrocks, P.F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean, "Swrl: A semantic web rule language combining owl and ruleml," W3C Member submission 21 may 2004, Tech. Rep., 2004. [Online]. Available: http://www.w3.org/Submission/SWRL/.
[6] D. Wetherall, J. Guttag, D. Tennenhouse, "Ants: A toolkit for building and dynamically deploying network protocols," 1998. [Online]. Available: citeseer.ist.psu.edu/wetherall98ants.html.
[7] D.S. Alexander, W.A. Arbaugh, M. Hicks, P. Kakkar, A. Keromytis, J.T. Moore, C.A. Gunter, S.M. Nettles, J.M. Smith, The SwitchWare active network architecture, IEEE Network Magazine 12 (3) (1998) 29–36 Special issue on Active and Controllable Networks. Available: http://www.cis.upenn.edu/switchware/papers/switchware.ps.
[8] A. Westerinen, J. Schnizlein, J. Strassner, M. Scherling, B. Quinn, S. Herzog, A. Huynh, M. Carlson, J. Perry, S. Waldbusser, "RFC 3178: Terminology for Policy-Based Management," November 2001. [Online]. Available: http://www.faqs.org/rfcs/rfc3178.html.
[9] P. Kodeswaran, S. Kodeswaran, A. Joshi, T. Finin, Enforcing Security in Semantics Driven Policy Based Networks, Proceedings of the 24th International Conference on Data Engineering Workshops, Secure Semantic Web, 2008, pp. 490–497, Available: http://ebiquity.umbc.edu/_file_directory_/papers/401.pdf.
[10] "Jess." [Online]. Available: http://www.jessrules.com/jess/index.shtml/.
[11] "http://www.cs.umbc.edu/kodeswar/ontologies/NetworkOnto.owl." [Online]. Available: http://www.cs.umbc.edu/kodeswar/ontologies/NetworkOnto.owl.

[12] P. Kodeswaran, S.B. Kodeswaran, A. Joshi, F. Perich, Utilizing semantic policies for managing BGP route dissemination, IEEE INFOCOM 2008 — IEEE Conference on Computer Communications Workshops, IEEE, 2008, pp. 1–4, Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4544611.
[13] S. Kent, C. Lynn, K. Seo, Secure border gateway protocol(s-bgp), IEEE Journal on Selected Areas in Communication 18 (2000) 582–592.
[14] T. Feng, R. Ballantyne, L. Trajkovic, Implementation of bgp in a network simulator, Proc. Applied Telecommunications Symposium, ATS'04, April 2004, pp. 149–154.
[15] R. Braden, D. Clark, S. Shenker, "RFC 1633: Integrated Services in the Internet Architecture: An Overview," June 1994. [Online]. Available: http://www.faqs.org/rfcs/rfc1633.html.
[16] S. Blake, D.L. Black, M.A. Carlson, E. Davies, Z. Wang, W. Weiss, "An Architecture for Differentiated Services," December 1998, Status: INFORMATIONAL.
[17] "Secure Origin BGP (SoBGP) Certificates. Internet Research Task Force, June 2003 (draft-weis-sobgp-certificates-00.txt)."
[18] C. Alaettinoglu, C. Villamizar, E. Gerich, D. Kessens, D. Meyer, T. Bates, D. Karrenberg, M. Terpstra, "Routing Policy Specification Language (RPSL)," Internet Engineering Task Force: RFC 2622, June 1999.
[19] A. Uszok, J. Bradshaw, R. Jeffers, N. Suri, P. Hayes, M. Breedy, L. Bunch, M. Johnson, S. Kulkarni, J. Lott, KAoS Policy and Domain Services: Toward a Description-logic Approach to Policy Representation, Deconfliction, and Enforcement, Proceedings POLICY 2003. IEEE 4th International Workshop on Policies for Distributed Systems and Networks, IEEE Comput. Soc, 2003, pp. 93–96, Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1206963.
[20] A. Uszok, J.M. Bradshaw, J. Lott, M. Breedy, L. Bunch, P. Feltovich, M. Johnson, H. Jung, New developments in ontology-based policy management: increasing the practicality and comprehensiveness of KAoS, POLICY, 2008, Available: http://portal.acm.org/citation.cfm?id=1445726.
[21] L. Kagal, T. Finin, A. Joshi, A policy language for a pervasive computing environment, IEEE 4th International Workshop on Policies …, 2003, Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.12.7910&rep=rep1&type=pdf.
[22] W. Nejdl, D. Olmedilla, M. Winslett, C.C. Zhang, "Ontology-based policy specification and management," in *In 2nd European Semantic Web Conference (ESWC)*, Springer, 2005, pp. 290–302.
[23] G. Tonti, J. Bradshaw, R. Jeffers, R. Montanari, N. Suri, "Semantic Web languages for policy representation and reasoning: A comparison of …," Springer. [Online]. Available: http://www.springerlink.com/index/BJHY2D977GF4E3QW.pdf.
[24] B. Carminati, E. Ferrari, R. Heatherly, M. Kantarcioglu, B. Thuraisingham, A Semantic Web Based Framework for Social Network Access Control, Symposium on Access Control Models and Technologies, 2009, Available: http://portal.acm.org/citation.cfm?id=1542207.1542237.
[25] C. Blanco, J. Lasheras, R. Valencia-García, E. Fernández-Medina, A. Toval, M. Piattini, A systematic review and comparison of security ontologies, ARES, 2008, Available: http://portal.acm.org/citation.cfm?id=1371887.
[26] "OWL Security Ontologies in OWL." [Online]. Available: http://www.csl.sri.com/users/denker/owl-sec/ontologies/.
[27] A.S. Peter, P.S., L.V. Ekert, An Ontology for Network Security Attacks, Proceedings of the 2nd Asian Applied Computing Conference (AACC04), LNCS 3285, 2004, pp. 317–323, Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.61.6065.
[28] A. Ekelhart, S. Fenz, M. Klemen, E. Weippl, Security ontologies: improving quantitative risk analysis, HICSS, 2007, Available: http://portal.acm.org/citation.cfm?id=1255933.
[29] R. Ashri, T. Payne, D. Marvin, M. Surridge, S. Taylor, Towards a Semantic Web Security Infrastructure, Semantic Web Services 2004 Spring Symposium Series, 2004, ["lib/utils:month_9040" not defined]. Available: http://eprints.ecs.soton.ac.uk/9040/.
[30] A. Toninelli, R. Montanari, L. Kagal, O. Lassila, A Semantic Context-aware access Control Framework for Secure Collaborations in Pervasive Computing Environments, ISWC'05: Proceedings of the 5th International Semantic Web Conference, 2005.

Palanivel Kodeswaran is a Ph.D candidate in Computer Science at the University of Maryland, Baltimore County (UMBC). His research interests are in the broad areas of systems and networking. He is currently focusing on applying the declarative paradigm for managing application and network adaptations using semantic policies. He is also interested in developing practical privacy solutions. He obtained his Bachelors degree from the College of Engineering Guindy, Anna University and Masters degree from UMBC in 2005 and 2008 respectively, all in computer science.

Anupam Joshi is a Professor of Computer Science and Electrical Engineering at UMBC. He obtained a B. Tech degree in Electrical Engineering from IIT Delhi in 1989, and a Masters and Ph.D. in Computer Science from Purdue University in 1991 and 1993 respectively. His research interests are in the broad area of networked computing and intelligent systems. His primary focus has been on data management for mobile computing systems in general, and most recently on data management and security in pervasive computing and sensor environments. He is also interested in Semantic Web and Data/Web Mining, where he has worked on personalizing the web space using a combination of agents and soft computing. His other interests include networked HPCC. He has served as guest editor for special issues of IEEE Personal Communications, Communications of the ACM, and served as an Associate Editor of IEEE Transactions of Fuzzy Systems from 99 to 03.



Tim Finin is a Professor of Computer Science and Electrical Engineering at the University of Maryland, Baltimore County. He has over 30 years of experience in applications of Artificial Intelligence to problems in information systems and intelligent interfaces. His current research is focused on the Semantic Web, human language technology and on analyzing and extracting information from online social media systems. He holds degrees from MIT and the University of Illinois and has also held positions at Unisys, the University of Pennsylvania, and the MIT AI Laboratory. He has chaired many major conferences, served as a board member of the Computing Research Association and been a AAAI councilor. He is currently an editor-in-chief of the Elsevier Journal of Web Semantics.

Sethuram Balaji Kodeswaran obtained his Ph.D and Masters degree in Computer Science from UMBC in 2008 and 1999 respectively. His research interests are in mobile and wireless networks as well as content based networking.