Utilizing Semantic Policies for Secure BGP Route Dissemination

Sethuram Balaji Kodeswaran, Palanivel Kodeswaran, Anupam Joshi
Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County, USA
(kodeswar,palanik1,joshi)@cs.umbc.edu

Filip Perich
Shared Spectrum Company
Vienna, VA 22182, USA
fperich@sharedspectrum.com

***Abstract*— Policies in BGP are expressed as routing configurations that determine how route information is shared among neighbors to control traffic flows across networks. This process is limited in its expressibility, time consuming and error prone which can lead to configurations where policies are violated or there are unintended consequences that are difficult to detect and resolve. In this paper, we propose an alternate mechanism for policy based networking that relies on using additional semantic information associated with routes expressed in an OWL ontology. Policies are expressed using SWRL to provide fine-grained control where the routers can reason over their routes and determine how they need to be exchanged. In this paper, we focus on security related BGP policies and show how our framework can be used in implementing them. Additional contextual information such as affiliations and route restrictions are incorporated and polices specified which can be reasoned over to infer the correct configurations that need to be applied which is easy to deploy, manage and verify for consistency.**

I. Introduction

Border Gateway Protocol (BGP) was originally designed as a simple path vector protocol to share routing information between autonomous systems (AS) which has today, become the de-facto inter-domain 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 error-prone and difficult to manage. In addition, routes are expressed as IP prefixes with no additional meta data describing the network represented, any sharing restrictions, underlying network technology etc. This limits the flexibility with which policies can be specified. For example, a policy such as *"Share 128.121.0.0/18 only with tier 1 partners"* or one such as *"Share restricted routes only with trusted peers"* are difficult to implement without extensive knowledge apriori to setup the router configurations. Furthermore, implementing these configuration changes requires highly skilled personnel and for scenarios such as emergency response and army battle field operations, this is not suitable as minimizing time and complexity of deployment is essential.

To solve this issue, we propose an alternate model to achieve policy based routing that can provide fine grained policy specification to automate network configuration and ease network management. The model relies on two key components; namely a tagging mechanism that allows routes to convey higher level semantic information that can be used in conjunction with information about the participating BGP peers and a framework for specifying rules in an easy to use, formal model that can be checked for consistency. In our model, ASes encode routes that they originate with descriptions conveying semantics such as what this route represents, who this route can be shared with, traffic type limitations etc using RDF/OWL. This description is encoded as a special optional and transitive path attribute in BGP so that it can traverse routers that are not setup to use this meta-data. Our motivation for using OWL[10] (specifically, OWL-DL), besides being a W3C standard, is mainly its capabilities for expressing formal semantics, defining 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 labeling schemes that raise interoperability issues.

Utilizing the framework, BGP speakers can run a reasoning engine that can reason over the RDF descriptions for the various routes and invoke rules depending on the correct set of actions that need to be enforced. Our framework utilizes SWRL[6] 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 this framework, we can now realize highly configurable route exchanges to influence how routes and thereby traffic within an internetwork needs to be controlled in a manner that is easier to manage and machine understandable.

In this paper, we focus on typical import and export policies used by BGP routers to determine what routes to accept from peer advertisements and what routes to advertise to peers. We show how our architecture can be used to provide fine

This work was supported in part by DARPA under contract W31P4Q-06-C-0395

grained levels of control that is simple to implement and easy to verify for correctness. We have developed a network ontology to be used to describe BGP protocol packets with attributes to describe the route meta-data and example policies to fine tune the BGP decision process.We have also developed a simulation toolkit in NS2 to implement aspects of our proposed architecture allowing us to simulate various scenarios and how policies can be expressed to offer desired behavior. The rest of the paper is organized as follows. Section II describes our proposed architecture, section III describes our extensions to BGP, section IV and V describe the use case considered and our simulation toolkit, section VI presents some of the related work in this area with our conclusions in section VII.

II. Proposed Architecture

Policy based networks employ mechanisms that allow network operators to specify at a high level, rules defining how packet flows are handled within a network, how network resources are allocated, define access control restrictions and levels of service. All these policies are then enforced by configuring the network devices with the requisite primitives so that the desired actions are performed on the data streams. We have previously proposed an architecture for policy based networks which we envision to be managed as a multi-tier system [9], [8] with hierarchical policy enforcement with the highest level of the hierarchy being the central NOC for an ISP and the lowest level being 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 being managed. Built into the 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. Through this checking, the overall consistency of the system can be maintained. The validation actions themselves are expressed through policies set forth typically by an enterprise network administrator.

In this work, we have adapted this framework to handle BGP interactions and use it to specify routing policies. Figure 1 illustrates the various components of our proposed architecture as applied to a generic ISP/enterprise comprised of several ASes managed by different administrators (this could be due to differences in geographical locations, administrative demarcation etc.). In this work, we limit our discussion to how we view the various components of our general architecture working to drive the BGP decision process. More details on the architecture itself is available in [9], [8].

The *Enterprise Policy Data Store* (EPDS) is a central repository of all of polices that govern the ISP. 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). Each AS Policy Repository (PR), as part of its initialization, will contact the EPDS to obtain the set of policies that are relevant to this AS

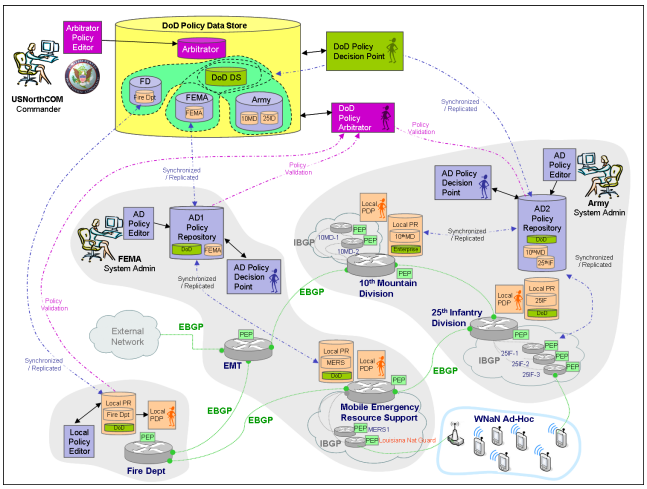

Fig. 1. Proposed Architecture

(and any that are ISP/enterprise wide). The EPDS is constantly synchronized with the PRs in the system.

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 and feasibility check. The EPA uses the policies stored within the EPDS to perform these validation checks. In this manner, the EPA ensures that any policy entered into the system conforms to certain global system constraints.

The *Policy Repository* (PR) is a data store for a collection of policies. Typically this will be an AS specific PR (AS-PR) or arbitrary collection (of routers) specific PR (referred to as local-PR). The AS-PR contains all policies specific to the AS and any policies that are ISP wide. 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. Polices can be added, deleted or modified from a PR through a Policy Editor. Any such changes are first forwarded to the EPA for validation. If they are consistent, then the EPA applies these changes into the EPDS. This addition will then propagate to the PR chain so that all the PRs in the hierarchy are updated.

The *Policy Decision Point* (PDP) is the entity that is responsible for reasoning over the BGP route exchanges and 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. 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 PEPs Configuration Conformance Enforcer). In addition, 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. In the case of our BGP example, the PDP can be a local process running on the router itself or can be a remote server process. In the later case, this is similar to the model used by IRV[5] for delegating route authentication from BGP nodes.

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

- Requesting and storing its configuration from the local-PDP that is responsible for this device
- Delegating 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
- Reporting errors and status updates to the local-PDP

The *Network Ontology* (NetOnto) is the OWL ontology that we define to mark up the routes being exchanged. 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. OWL has axiomatic and model-theoretic semantics, which allows for verification of knowledge expressed in OWL constructs. 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. The ontologies can be extended for capturing declaratively 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. The route descriptions are carried in the BGP updates as optional transitive attributes either as directly embedded in bit efficient format, contain a URL to the description or use UUIDs that imply a certain well known description. A PEP extracts this description and adds to it, any extra contextual information including aspects such as peer identity, 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 (in this work, as we are dealing with import/export policies, the PDP filters appropriately the routes that are exchanged).

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. 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 acknowledgment of whether or not the requested change is allowed or rejected. In addition, the PE also offers views of the managed network such as topology views, status of network devices and links etc.

Utilizing this framework, BGP routers in a network can be finely controlled using policies expressed at a higher abstract level. At each router, BGP protocol messages (and in this case, BGP Update messages) are inspected to see if they carry semantic description attributes. These updates are then offloaded to a separate forwarding path. The semantic description is extracted, any additional contextual detail available to the PEP is added to complete the RDF description and sent to the PDP for reasoning. We use RDF's abbreviated XML encoding format for this purpose. The PDP runs a reasoner that takes the RDF 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 NLRIs to realize the necessary policies.

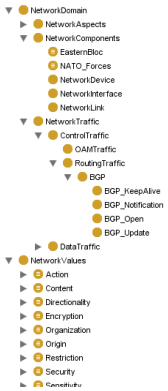

Fig. 2. Network Ontology

III. BGP Extensions

To apply the above 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 [7] 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.

IV. 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.

V. Simulation Toolkit

We used the ns-BGP [4] 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. For the network ontology, we used Protege as the editor for specifying our ontology. Jess was used as the reasoning engine. The choice of Jess was mainly motivated by its easy integration with Protege. Other reasoning engines can be used as a replacement if desired.

To begin, we defined an ontology to use for our BGP example. The ontology is available online at [1]. 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, ?routeradvertising) ∧
dest(?adv, ?peer) ∧
owner(?routeradvertising, ?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. 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 ShareWithFriendly 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. Figure 3 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.

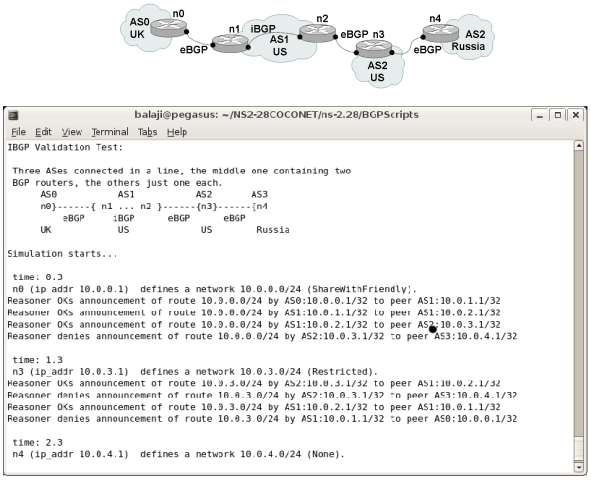

Fig. 3. Simulation Output

VI. Related Work

There has been significant research on securing BGP. SBGP[7] proposes a comprehensive architecture for sercuring BGP using public key certificates. SBGP uses a pair of PKIs, one for address authentication and the other for route validation. SoBGP[2] 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 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. IRV[5] provides route validation through Interdomain Route Validation (IRV) servers running on the ASes. On receiving an UPDATE message, based on the local policy, the receiver can query the remote IRV server for veracity. The IRV servers in turn can enforce their local policies while responding to queries, thus providing control of data to the AS. RPSL[3] 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.

VII. Conclusion

In this paper, we present our architecture for implementing a framework for secure BGP route exchanges controlled by policies that can take external semantic information into consideration for the decision process. The framework relies on reasoning over contextual descriptions of the routes and the identity of the routers to invoke the right set of actions as defined in the policies. A simulation framework and an example ontology in OWL is also presented with its application to an use case example with policies specified in SWRL. We are currently extending our ontology and looking into mechanisms for policy aggregation, conflict resolution, validation and prioritization.

References

[1] http://www.cs.umbc.edu/ kodeswar/ontologies/NetworkOnto.owl.

[2] Secure Origin BGP (SoBGP) Certificates. Internet Research Task Force, June 2003. (draft-weis-sobgp-certificates-00.txt).

[3] C. Alaettinoglu, C. Villamizar, E. Gerich, D. Kessens, D. Meyer, T. Bates, D. Karrenberg, and M. Terpstra. Routing Policy Specification Language (RPSL). Internet Engineering Task Force: RFC 2622, June 1999.

[4] T. Feng, R. Ballantyne, and L. Trajkovic. Implementation of bgp in a network simulator. In *Proc. Applied Telecommunications Symposium, ATS'04*, pages 149–154, April 2004.

[5] G. Goodell, W. Aiello, T. Griffin, J. Ioannidis, P. McDaniel, and A. Rubin. Working around bgp: An incremental approach to improving security and accuracy in interdomain routing. In *Proc. ISOC Network and Distributed System Security (NDSS '03)*, San Diego, CA, February 2003.

[6] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. Swrl: A semantic web rule language combining owl and ruleml. Technical report, W3C Member submission 21 may 2004, 2004.

[7] S. Kent, C. Lynn, and K. Seo. Secure border gateway protocol(s-bgp). *IEEE Journal on Selected Areas in Communication*, 18:582–592, 2000.

[8] S. B. Kodeswaran and A. Joshi. Content and context aware networking using semantic tagging. In *ICDEW '06: Proceedings of the 22nd International Conference on Data Engineering Workshops (ICDEW'06)*, page 77, Washington, DC, USA, 2006. IEEE Computer Society.

[9] S. B. Kodeswaran, O. Ratsimor, A. Joshi, and F. Perich. Utilizing Semantic Tags for Policy Based Networking. In *Globecom 2007 (accepted for publication)*, November 2007.

[10] D. L. McGuinness and F. van Harmelen. Owl web ontology language overview. Technical report, W3C Recommendation 10 February 2004, 2004.