Authorizing Network Control at Software Defined Internet Exchange Points

Arpit Gupta*, Nick Feamster*, Laurent Vanbever†
*Princeton University †ETH, Zurich
http://sdx.cs.princeton.edu/

Abstract

Software Defined Internet Exchange Points (SDXes) increase the flexibility of interdomain traffic delivery on the Internet. Yet, an SDX inherently requires multiple participants to have access to a single, shared physical switch, which creates the need for an authorization mechanism to mediate this access. In this paper, we introduce a logic and mechanism called *FLANC* (A Formal Logic for Authorizing Network Control), which authorizes each participant to control forwarding actions on a shared switch and also allows participants to delegate forwarding actions to other participants at the switch (*e.g.*, a trusted third party). FLANC extends "says" and "speaks for" logic that have been previously designed for operating system objects to handle expressions involving network traffic flows. We describe FLANC, explain how participants can use it to express authorization policies for realistic interdomain routing settings, and demonstrate that it is efficient enough to operate in operational settings.

Categories and Subject Descriptors: C.2.1 [Computer-Communication Networks] *Network Architecture and Design*: Network Communications

General Terms: Algorithms, Design, Experimentation

Keywords: software defined networking (SDN); Internet exchange point (IXP); BGP

1 Introduction

Software Defined Internet Exchange Points (SDXes) [12, 13] improve the flexibility and function of interdomain routing on the Internet, but the SDX inherently requires multiple participating autonomous systems to access a shared physical switch at the Internet Exchange Point (IXP). The SDX needs a mechanism to ensure that each participant can install only flow table entries for traffic flows that it is authorized to control. For example, no SDX participant should be allowed install a forwarding table entry that does not correspond to a route that is advertised in BGP. SDX participants should also be able to conditionally delegate authorization to local or remote participant. Consider a third-party service which can protect a network against Denial-of-Service (DoS) attack traffic such those offered Verisign [30], Radware [24], and Arbor Networks [5]). In this scenario, the victim's network would like to delegate to a third party the authority to perform set of forwarding/redirection *actions* over the subset of its traffic (*e.g.*, only HTTP traffic), *only* when DoS attack is detected.

Unfortunately, existing SDXes do not have a mechanism that allows participants to express or enforce this type of control. In fact, even *conventional* networks do not provide adequate mechanisms for rich authorization policies. For example, when a network is subject to a DoS attack and wants to route its traffic through a third party, it must allow that third party to temporarily "hijack" the delegated IP prefix, effectively exploiting BGP's existing security weaknesses to become a man-in-the-middle for traffic that is destined to the victim's IP prefix. Even proposed mechanisms such as RPKI have no mechanism for fine-grained delegation of network control [14]. SDX needs a more expressive authorization framework to ensure that only authorized parties can make such modifications to the flow table, only when the delegation conditions are satisfied.

This paper presents a FLANC(Formal Logic for Authorizing Network Control), general logic that can determine whether any particular action on traffic flows is authorized; we also implement FLANC and integrate it with our existing SDX implementation. Central to FLANC's design is a trusted *reference monitor* that can determine whether a principal (*i.e.*, any entity that is trying to perform an action) is authorized to perform actions on a *network flowspace* (*i.e.*, groups of traffic flows that have header values in common). This reference monitor is analogous to the ones enforcing authorization for certain operating system actions [31]. The reference monitor takes both authorization policies (*e.g.*, one network principal's statement that a different principal is authorized to perform certain actions) and credentials as input (*e.g.*, information about who owns an IP prefix) and attempts to prove that the principal who issues a request is authorized to perform the requested actions on the data plane.

Realizing FLANC required us to extend existing authorization logics to support expressions that concern network flowspace. Although existing authorization logics provide a useful foundation, they typically concern operations on discrete operating system objects (*e.g.*, files, processes) and thus do not apply to network flowspace, which has more complicated relationships, such as subset relationships. Our work extends authorization logics so that they can be applied to network control. In a network setting, an object is a set of data packets that can be represented as a subset of flowspace; a request to perform an operation is an attempt to read (*i.e.*, see a particular set of packets) or write (*i.e.*, affect the packets' properties or how they are forwarded); and a principal is a network control policy that is implemented either as device configuration or, in the case of SDN, a control program. Although FLANC is general and can apply to any SDN control environment where multiple participants or control applications share access to physical switch

*SOSR'16,*March 14–15, 2016, Santa Clara, CA.

http://dx.doi.org/10.1145/2890955.2890956

NAL [26]	$A ::= [[v : S]]$ $S ::= A$ says $S \mid S \wedge S \mid S \vee S \mid (\forall v : S) \mid (\exists v : S) \mid S_F$
FLANC	$S_F ::= A$ owns $F \mid T$ $F ::= * \mid id \mid F \cup F \mid F \cap F \mid F \subseteq F \mid$ $T ::= F \rightsquigarrow F \mid T \cup T \mid T \cap T \mid T \subseteq T \mid T \wedge \chi$

Table 1: *FLANC Syntax. FLANC extends the NAL grammar with S_F, terms that allow for statements about flowspace.*

resources, we focus in this short paper on FLANC's application to SDX. We have implemented FLANC and released it to the public as an extension to SDX [15]. Our evaluation shows that that FLANC is expressive and flexible enough to capture authorization policies that participants might express at SDX, and that it introduces only minimal overhead to existing flow modification operations: our implementation of the FLANC reference monitor requires less than 100 microseconds to authorize flow table modifications.

The rest of the paper is organized as follows. Section 2 presents the background information on existing authorization logic, relates FLANC to existing authorization logics, such as NAL [26], and then presents the FLANC authorization logic. Section 3 discusses how we apply FLANC to SDX, Section 4 evaluates FLANC in a setting that emulates an operational IXP, Section 5 discusses related work, and Section 6 summarizes and outlines future research directions.

2 Authorizing Network Control

We provide a brief background on existing authorization logics that we adopt for FLANC in Section 2.1. We then introduce the FLANC authorization logic in Section 2.2, illustrating how we extend the existing works with new axioms and inference rules.

2.1 Background

In conventional authorization frameworks [1, 2, 3, 4, 9, 18, 19, 20, 31], a guard acting as a reference monitor checks whether a *principal* can perform a certain *request* (*e.g.*, read, write) on one or more *objects* (*e.g.*, file, process). A request to access a resource or obtain service is accompanied by credentials. To enforce a given policy, a *reference monitor* uses *credentials* in conjunction with the request to derive a *formula* representing the authorization policy. If it succeeds in deriving this formula, it grants the request; otherwise, it denies the request.

An authorization logic enables principals to express their beliefs using *says* statements. For example, the formula *A says S* is interpreted to mean that $S \subseteq \mathscr{W}(A)$, where $\mathscr{W}(A)$ represents the set of beliefs that A holds. The $\rightarrow$ "speaks for" relation is a partial order that obeys many of the same laws as implication. Thus, $B \rightarrow A$ represents *B speaks for A*, meaning that if B makes a statement, then that is logically equivalent to A having made the same statement (*i.e.*, $\mathscr{W}(B) \subseteq \mathscr{W}(A)$).

Nexus Authorization Logic (NAL) [26] introduced two new concepts beyond other conventional authorization logic (*e.g.*, Abadi's CDD [1]), which we also use in FLANC: *state predicates*, which enable reasoning about various state parameters; and *restricted delegation*, which allows delegation of subset of beliefs against the delegation of complete set of beliefs proposed by principal-centric authorization frameworks.

Table 1 summarizes the syntax of NAL, which we build on in our design of FLANC. The table also shows how we extend the NAL syntax to operate on network flowspace; the next section describes both these extensions to the logic, as well as the additional axioms and inference rules we use to reason about authorization for network control.

2.2 FLANC Authorization Logic

Conventional authorization logics are typically applied in systems where (1) resources are discrete: an object might be a file, for instance, and (2) set of allowable actions typically involve read and write operations. Applying authorization logic to networks requires grappling with two significant additional challenges:

1. A network authorization logic must permit actions over *sets of objects*. Specifically, the logic must provide for predicates that operate on sets (and subsets) of packets, expressed in terms of *flowspace*.
2. A network authorization logic must also permit *sets of actions* on the flowspace and deny others; concisely expressing these permissions is challenging.

In the context of network control, a *resource* is a set of traffic flows that lie within some *flowspace* (*e.g.*, all packets with a certain set of destination IP addresses), and the set of allowable *actions* involve operations that not only include forwarding the packet on particular output ports, but also potentially rewriting the packet into a different part of flowspace. If, as in previous work [11], we can think of each packet as a set of key-value pairs representing a "located packet" (*i.e.*, a packet and its location in the network), then each possible action, including forwarding the packet, can be thought of as simply a *transformation* on the packet's metadata. We can thus view the set of allowable actions as simply a set of allowable transformations on the packet.

The network authorization logic's transformations are inspired by the notion of transfer functions in Header Space Analysis (HSA) [17], but there are some subtle differences. Transfer functions defined in HSA express one-to-one mappings between the packets in input flowspace to the packets in the transformed flowspace (or multiple packets in the case of multicast). In contrast, the network authorization logic specifies a range of points (*i.e.*, a region of flowspace) instead of a specific point to which the input packet can be transformed. The set of all the mappings transforming the packets in the input flowspace to the region specified in the transformed space constitute the set of allowed transformations.

Although many of the same principles of existing authorization logics may apply (*i.e.*, those from Section 2.1 and RT [21]), these logics must be extended with additional axioms/inference rules so that they can be applied over network flowspace. In this section, we consider how the principles from Section 2.1, including ownership and delegation, apply to network flowspace. We also consider how additional operations, such as the ability for a control program to rewrite a packet, require us to extend existing authorization logics.

Axioms and Inference Rules. FLANC has axioms that allow a reference monitor to reason about (1) *ownership*, or which principals own a particular resource (*i.e.*, packets traversing flowspace) (Section 2.2.1); (2) *allowed actions*, which allow a principal to express which actions a control program can perform on packets in some portion of flowspace (Section 2.2.2); and (3) *delegation*, the process by which one principal can give another permissions to operate on packets in some portion of flowspace (Section 2.2.3).

2.2.1 Ownership

Ownership relates the principals in the network with the objects that they own. In the context of network control, objects are regions of flowspace (which in turn define subsets of traffic). The owner of an object is authorized to speak on behalf of that object, and an ownership relationship is established with a (signed) statement from a root of trust. This root of trust can be centralized as in case

of RPKI or can be bitcoin-like decentralized public ledgers similar to Namecoin [22]. The RPKI is a CA infrastructure that establishes prefix ownership relationships. Decentralized flowspace resource ownership works are still in nascent stage, thus FLANC currently utilizes centralized CAs to establish resource ownership. In the authorization logic, a CA makes a signed statement establishing that some principal owns some portion of flowspace (*e.g.*, an AS might own an IP prefix). To express these notions of ownership, we introduce a new "owns" connective in FLANC, as shown in Table 1:

Axiom 1 (Ownership)**.** *R says (A owns F) ⇒ A owns F*

A signed statement from a root of trust, R, saying that A owns F establishes that A owns F. If the reference monitor trusts R, then the reference monitor believes that A owns F and can thus perform some set of actions (or transformations) on F. Note that there can be two owners (sender and receiver) for any data traffic.

2.2.2 Actions

In contrast to existing authorization logics, which allow a reference monitor to reason about sets of actions on system resources (*e.g.*, read, write), the network logic must allow reasoning about a much richer set of actions on network flowspace. These actions include:

- **Forwarding** the packet out a particular port (keeping in mind that forwarding to some ports may be allowed, while forwarding to others may be disallowed).
- **Dropping** the packet.
- **Modifying** fields in the packet's header. As with forwarding, certain modifications may be allowed, whereas others may be disallowed.
- **Reading** the packet, or aggregated statistics about a group of packets.

All of these actions can be represented as modifications on a virtual packet header that represents a located packet. For example, using the parlance from previous work [11], we can represent forwarding as simply modifying a virtual packet header representing the packet's `outport` value. We can also express read actions as modifications that change the packet's `outport` to a port that is performing a read action.

FLANC allows a resource owner to define the set of allowable actions for a part of flowspace.

Axiom 2 (Actions)**.** *A owns $F_A \Rightarrow T_A \subseteq \mathscr{W}(A)$, where $T_A : F_A \rightsquigarrow F_A^{'}$.*

where T_A is a set of *transformations* for some set of packets in the network flowspace F_A to some other part of flowspace, $F_A^{'}$. In other words, if A owns flowspace F_A (as might be established by the ownership axiom), then its belief set can have some set of actions, T_A, which maps packets from the part of flowspace that it owns into some other flowspace, $F_A^{'}$.

As we discussed earlier, there can be two owners for any data traffic. We need to carefully consider $F_A^{'}$ for these two type of owners. For example, should the receiver be able to rewrite the source IP of the packets (which would effectively amount to spoofing the source IP address of the packets) arbitrarily? Should the sender be able to rewrite destination IP addresses or ports arbitrarily? To address this problem we allow the receivers to express packet transformations on where the packet is *destined*, but it should not be able to perform modifications that modify information about the packet's provenance (*i.e.*, where the packet is coming from) and vice versa.

$\dfrac{C \xrightarrow{v:T_A} A, C \xrightarrow{v:T_B} B}{C \xrightarrow{v:(T_A \cup T_B)} (A,B)}$	(Aggregate)
$\dfrac{C \xrightarrow{v:T_B} B, B \xrightarrow{v:T_A} A}{C \xrightarrow{v:(T_B \cap T_A)} A}$	(Transitive)
$\dfrac{C \xrightarrow{v:(T_A \wedge \chi_A)} A, C \xrightarrow{v:(T_B \wedge \chi_B)} B}{C \xrightarrow{v:(T_A \wedge \chi_A \cup T_B \wedge \chi_B)} (A,B)}$	(Conditional Aggregate)
$\dfrac{C \xrightarrow{v:(T_B \wedge \chi_B)} B, B \xrightarrow{v:(T_A \wedge \chi_A)} A}{C \xrightarrow{v:(T_B \wedge \chi_B \cap T_A \wedge \chi_A)} A}$	(Conditional Transitive)

Table 2: *Inference rules for FLANC.*

2.2.3 Delegation

A resource owner can allow other principals to perform actions on flowspace that it owns by *delegating* those actions to them. A owner of some network flowspace might delegate authorization to perform a certain set of actions on flowspace.

Axiom 3 (Restricted Delegation)**.** *A says ($B \xrightarrow{v:T} A$) ⇒ ($B \xrightarrow{v:T} A$), where $T \subseteq \mathscr{W}(A)$.*

Here B speaks for set of transformation actions (T). Note that the axiom also stipulates that A must be authorized to say T (*i.e.*, perform actions T on flowspace F) in order to delegate to B.

Delegating actions on flowspace Authorization to perform an action T on F might come from multiple delegations, in which case the authorized action is the *union* of multiple delegated actions, summarized as the following inference rule: A says $(C \xrightarrow{v:T_A} A)) \wedge (B$ says $(C \xrightarrow{v:T_B} B)) \Rightarrow C \xrightarrow{v:(T_A \cup T_B)} (A,B)$. In other words, C is authorized to perform an action on some portion of flowspace, if and only if the requested action is the subset of union of the actions that are delegated to C.

Because no principal can delegate authorization for actions over flowspace that it is not authorized to perform, any delegation from one principal to another must also be the *intersection* of that delegation statement and actions for which it has the authorization. The inference rule corresponding to this restriction can be summarized as: A says $(B \xrightarrow{v:T_A} A) \Rightarrow B \xrightarrow{v:(T_A \cap T^*)} A$, where T^* is the set of transformation actions that A is authorized to perform. That is, A can only delegate a set of actions that are a subset of what A is authorized to perform.

2.2.4 Conditional Delegation

In some cases, a principal may want to either allow actions or delegate conditionally, to specify that a principal should only be allowed to perform certain actions when certain conditions are true.

One practical example of conditional delegation might be revocation, which can simply be expressed as a time-bounded condition. Another example might be the DoS mitigation example we discussed in section 1. In this example, a principal might want to say that the third party is allowed to perform some particular action on flowspace that it owns such as modifying the `outport` or `dstip`

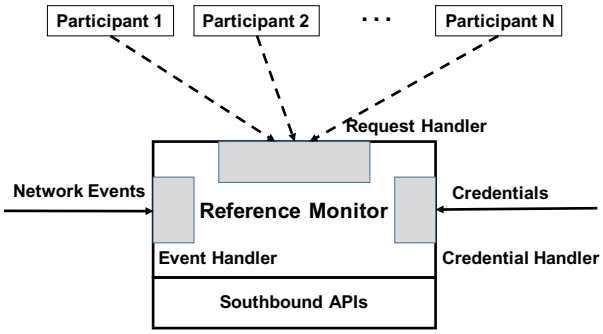

Figure 1: *The FLANC reference monitor has three components:* Event Handler, Request Handler, *and* Credential Handler.

of the packet if and only if some condition is true (*e.g.*, an intrusion detection system has indicated that a denial of service attack is underway). This requires a way for the reference monitor to verify that the conditions have been satisfied. We assume that there is some way for the reference monitor to obtain such statements, such as via signed statements from a trusted monitoring appliance similar to DOTS [25].

To allow for these expressions, we introduce the notion of *conditional actions* and *conditional delegations*. For conditional delegations we have, A says $(B \xrightarrow{v:(T \wedge \chi)} A) \Rightarrow (B \xrightarrow{v:(T \wedge \chi)} A)$, where χ is a boolean expression that must be true for the delegation to be valid. In other words, A allows B to act on behalf of some flowspace only when all of the conditions in χ are true. The reference monitor can use these axioms to derive inference rules for conditional delegations, as shown in Table 2. When a principal expresses a conditional delegation in terms of such a set of conditions, a reference monitor will re-evaluate any authorization decision that depends on a condition that changes.

3 Applying FLANC to SDX

We now apply FLANC to authorization problems that arise at an Internet Exchange Point (IXP) and integrated FLANC with our public implementation of the SDX controller [12, 13, 15]. We first describe how FLANC can check the authorization of flow table modifications from local participants who own the flow space that they are trying to control. We then describe how FLANC's delegation mechanisms can be used to authorize third-party traffic control, as is necessary in certain applications that rely on third-party services (*e.g.*, DDoS defense).

3.1 FLANC Design and Operation at SDX

SDX allows participants to specify fine-grained interdomain routing policies; participants may be local to the IXP or remote; our existing implementation composes policies from both remote and local participants [12, 13]. Our recently released "industrial-grade" SDX controller (iSDX) makes it possible to realize this functionality at the world's largest IXPs [12, 15]. iSDX partitions the control plane across IXP participants (both local and remote). In this design, each participant's controller sends flow table modifications as JSON messages to the *fabric manager*. FLANC's reference monitor checks the authorization of each of these attempted modifications.

The reference monitor has three components: (1) an event handler, (2) a request handler, and (3) a credential handler, as shown in Figure 1. The *event handler* processes incoming network events. These events are anything that might cause authorization decisions to change; they could include BGP updates, intrusion alerts, or information about traffic loads. The *credential handler* receives the list of credentials from resource owners and third parties for use in authorization proofs. Finally, the *request handler* (1) translates the principal's requested actions into a set of flowspace transformations (T_{req}) and (2) determines whether principal is authorized to perform T_{req}.

The SDX controller must ensure that participants are authorized to perform the requested action. Simple examples of this include modifications by local participants for inbound traffic control, but more complex scenarios involving third parties can also arise. For example, suppose that a network such as Princeton (P) subscribes to Verisign's (V) DoS protection service. The local IPS box at Princeton identifies a fraction of attack traffic with low confidence locally and requests redirection of this traffic through a remote scrubber. Conventionally, this redirection occurs at the level of an IP prefix. SDX allows P to push flow rules at the upstream IXPs to (1) block the attack traffic it identified locally with higher confidence, (2) redirect the potential attack traffic to V's network at finer level of granularity. The upstream IXP that receives such a request for redirection must ensure that P is allowed to delegate control for the prefix that it is attempting to block or redirect. The fabric manager at the IXP uses FLANC's reference monitor to perform this check.

3.2 Checking Authorization

Suppose the local IPS box at P identifies that the HTTP service over `dstip=128.112.136.35` is potentially under attack. P's controller at the upstream IXP sends the following flow-mod request to the fabric manager:

$\texttt{dPort=80} \wedge \texttt{dIp=128.112.136.35} \rightarrow \texttt{fwd}(V)$

The fabric manager translates this request into a flow table modification request and sends it to the reference monitor (RM). Given the request, the reference monitor must determine: (1) who the request speaks for (which can be determined by successive application of delegation axioms, as in Section 2.2.3); (2) whether the principal is authorized to perform the requested actions.

Associating the request with a principal. The first part of the proof involves attributing the messages received over a control channel to a particular principal. For this part we can borrow heavily from previous works on delegating authorization to control channels [31], whereby a reference monitor can establish that messages received over a control channel in fact speak for a particular principal. In our authorization framework, we apply the delegation axioms as in Section 2.2.3 to establish that messages sent over the secure control channel C_n represent the controller c acting on behalf of principal n.

Determining whether the request is authorized. In this example, P makes claim to perform set of actions (T_{req}): "P says $T_{req} \subseteq \mathscr{W}(P)$". The goal is to derive reference monitor's authorization policy P_{RM}: "RM says $T_{req} \subseteq \mathscr{W}(P)$". The reference monitor grants the request if and only if P_{RM} can be derived from P's request, and other credentials available—using the axioms/inference rules discussed in section 2.2. RM begins the derivation with the ownership credentials which CAs provide to RM. These credentials represent the formula, CA says "P owns F_P". The reference monitor says that CA speaks for it: "RM says (CA $\rightarrow$ RM)"; it combines these two formulas to infer that, "RM says (P owns F_P)". Now, the reference monitor will follow the principles that define the actions a resource owner can perform to define $T_P : F_P \rightsquigarrow F_P'$. Applying the

Goal	Formulas	Analysis
RM says $T_{req} \subseteq \mathscr{W}(P)$	*CA says "P owns* F_P*"* *RM says* $(CA \rightarrow RM)$	$T_P := F_P \rightsquigarrow F'_P$ $T_{req} \subseteq T_P$

Table 3: *An example of deriving formulas using FLANC.*

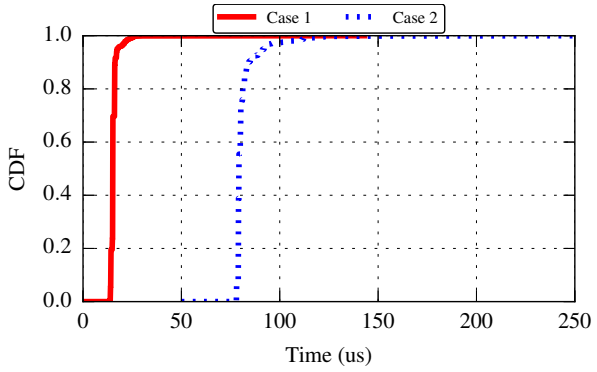

Figure 2: *FLANC's performance authorizing remote and third party requests at SDX.*

Action axiom, the reference monitor can derive:

$$(RM \text{ says } (T_P \subseteq \mathscr{W}(P)))$$

Now this can be combined with the analysis over T_{req} as:

$$(RM \text{ says } (T_P \subseteq \mathscr{W}(P))) \wedge (T_{req} \subseteq T_P)$$

to derive P_{RM}: RM says $(T_{req} \subseteq \mathscr{W}(P))$. Table 3, shows how the *RM* at SDX will use the available credentials to derive whether P is authorized to redirect the attacked traffic to the Verisign.

Authorizing Third Party Requests at SDX. In the previous example, Princeton can delegate the task to steer the attack traffic to Verisign to a third party (TP) service using existing protocols like DOTS [25]. In such a case SDX needs to determine whether the flow-mod requests made by TP is authorized or not. The reference monitor will use the authorization policy provided by P to determine the authorization of such third party requests.

4 Evaluation

We now demonstrate that using FLANC to authorize network control requests entails minimal performance overhead, thus demonstrating that it can be deployed in operational settings. We specifically focus on performance of FLANC integrated with SDX authorizing remote and third party requests.

Experiment Setup. All machines for the evaluation use a 64-bit 8-core 3.6 GHz processor. We use the `syslog` from the IPS box deployed in Princeton University. The data has the log of all the attack events between $21-27^{th}$ October, 2015. During this period we identified 159,231 high confidence and 393,660 low confidence attack events.

Demonstrating FLANC's Scalability Performance. In this experiment we consider two cases: (1) P requests to block high-confidence attacks and steer the low-confidence attacks to V; and (2) P delegates this task to a third party. This provides a more realistic scenario where requests are made directly by the remote IXP participants and third party service providers—delegated to perform the network control task by the resource owners. In this experiment, P expresses its authorization policy for delegation as T_{TP}: $\{dstip = PRFX_P, dstport = * - \{80,443\}\} \rightsquigarrow \{port = V\}$—allowing steering of all the traffic destined to P except the Web (HTTP and HTTPS) traffic. It sends this policy to the reference monitor at the upstream SDX, where it is received and processed by the *event handler* module. Figure 2 shows the cumulative distribution of time taken by the reference monitor to perform the authorization task for the two cases. It takes longer to authorize third party requests as the reference monitor has to perform an additional intersection operation, $T_P \cap T_{TP}$ compared to the operations shown in Table 3 for authorizing remote requests. In both the cases the time required for authorization is a small fraction of the time required to process an incoming flow-mod request by the SDX controller—demonstrating that the performance overhead is minimal.

5 Related Work

Lampson *et al.* [18] developed the original framework to reason about credential-based authorization. Abadi *et al.* formalized a logic based on this framework [3] and applied it in the Taos operating system [31]. Appel *et al.* [4] developed a high-order logic where the principals (not the reference monitor) generated the proofs of authorization. NAL [26] augmented the authorization logic proposed in CDD [1, 2] by adding modes for restricted and conditional delegations—enabling delegation of subset of credentials, and delegations dependent of system state respectively. FLANC adopts many of the concepts from NAL, CDD, and Abadi's earlier work, extending the framework to operate on network flowspace. Netquery [29] used NAL for developing a trusted knowledge plane for federated networks such as the Internet. Netquery focuses on establishing trust for the information that participating networks share about network topology, configuration, and performance. In contrast FLANC attempts to check the authorization of actions rather than establishing trust between the participants. Similarly the problems addressed by several other trust management systems [8, 16, 21, 32] are also orthogonal to the ones presented in this paper.

Flowvisor [27, 28] delegates control to different entities by isolating distinct portions of flow space; in contrast, authorization policies are often more fine-grained and need not require complete isolation. PANE [10] allows a single administrator to delegate control to the end-users or applications. FLANC considers multiple resource owners delegating their control to other networks. Bailey *et al.* [6] proposed to drop traffic for unauthorized paths (with invalid RPKI state) at SDX directly. While similar in spirit to FLANC, their solution is restricted to BGP and RPKI. Baldin *et al.* [7] showed how to delegate resources such as bandwidth and flow table rule space in multi-domain, multi-controller networks, which is orthogonal to the problem of authorizing ownership and delegation of network control. Fortnox [23] enforces security policies by prioritizing them over the conflicting policies from the network control applications. Fortnox's mechanisms for conflict resolution may be useful in FLANC.

6 Conclusion and Future Work

To support scenarios where multiple parties need control over flow table entries in a shared switch, such as at an IXP, we introduced FLANC, an authorization logic for network control; developed mechanisms for implementing this logic; and demonstrated the expressiveness of FLANC by applying it to authorization problems that arise in IXPs. FLANC extends existing authorization logics to network control by incorporating actions and axioms concerning

network flowspace. We integrated FLANC with SDX and showed that it can supoort realistic authorization policies while introducing only minimal overhead to flow table modification.

Our work thus far on FLANC creates many exciting possibilities for future work concerning both network applications and the logic itself. On the formal side, possible follow-on work might involve proving the soundness of FLANC's proof construction technique. Second, richer set of terms and connectives in the logic may ultimately allow operators to express authorization policies that concern aggregate statistics, operations on packet contents, and better reasoning about authorization when packets in flowspace are subject to a sequence of operations. On the more practical side, we believe that FLANC's logic and authorization framework may very well apply to other domains, from multi-tenant data centers to the Internet of Things. Finally, as FLANC is incrementally deployable and can be incorporated into software defined networks as well as conventional networks, exploring how to integrate FLANC with both existing and "green field" networks offers many exciting opportunities.

Acknowledgments

We thank Jennifer Rexford, Marco Canini, Rüdiger Birkner, Robert MacDavid, Bryan Larish, Chris Tengi, David Walker, Nate Foster, David Jorm, Paul Gampe, and the anonymous reviewers for the feedback and comments. This research was supported by National Science Foundation Awards CNS-1539920.

References

[1] M. Abadi. Access Control in a Core Calculus of Dependency. *Electronic Notes in Theoretical Computer Science*, 172:5–31, 2007. (Cited on pages 2 and 5.)

[2] M. Abadi. Variations in Access Control Logic. In *Deontic Logic in Computer Science*, pages 96–109. Springer, 2008. (Cited on pages 2 and 5.)

[3] M. Abadi, M. Burrows, B. Lampson, and G. Plotkin. A Calculus for Access Control in Distributed Systems. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 15(4):706–734, 1993. (Cited on pages 2 and 5.)

[4] A. W. Appel and E. W. Felten. Proof-carrying authentication. In *Proceedings of the 6th ACM Conference on Computer and Communications Security*, pages 52–62. ACM, 1999. (Cited on pages 2 and 5.)

[5] Arbor Networks' DDoS Protection. *arbornetworks.com/ddos-attacks*. (Cited on page 1.)

[6] J. Bailey, D. Pemberton, A. Linton, C. Pelsser, and R. Bush. Enforcing rpki-based routing policy on the data plane at an internet exchange. In *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, HotSDN '14, pages 211–212, New York, NY, USA, 2014. ACM. (Cited on page 5.)

[7] I. Baldin, S. Huang, and R. Gopidi. A Resource Delegation Framework for Software Defined Networks. In *HotSDN '14*, pages 49–54, Chicago, IL, August 2014. ACM. (Cited on page 5.)

[8] M. Blaze, J. Feigenbaum, and A. D. Keromytis. Keynote: Trust management for public-key infrastructures. In *Security Protocols*, pages 59–63. Springer, 1999. (Cited on page 5.)

[9] H. DeYoung, D. Garg, and F. Pfenning. An authorization logic with explicit time. In *Computer Security Foundations Symposium, 2008. CSF'08. IEEE 21st*, pages 133–145. IEEE, 2008. (Cited on page 2.)

[10] A. D. Ferguson, A. Guha, C. Liang, R. Fonseca, and S. Krishnamurthi. Participatory networking: An api for application control of sdns. In *ACM SIGCOMM Computer Communication Review*, volume 43, pages 327–338. ACM, 2013. (Cited on page 5.)

[11] N. Foster, A. Guha, M. Reitblatt, A. Story, M. J. Freedman, N. P. Katta, C. Monsanto, J. Reich, J. Rexford, C. Schlesinger, A. Story, and D. Walker. Languages for Software-Defined Networks. *IEEE Communications Magazine*, 51(2):128–134, 2013. (Cited on pages 2 and 3.)

[12] A. Gupta, R. MacDavid, R. Birkner, M. Canini, , N. Feamster, J. Rexford, and L. Vanbever. An industrial-scale software defined internet exchange point. In *USENIX NSDI*, Santa Clara, CA, 2016. (Not cited.)

[13] A. Gupta, L. Vanbever, M. Shahbaz, S. P. Donovan, B. Schlinker, N. Feamster, J. Rexford, S. Shenker, R. Clark, and E. Katz-Bassett. SDX: A Software Defined Internet Exchange. In *ACM SIGCOMM*, pages 579–580, Chicago, IL, 2014. ACM. (Cited on pages 1 and 4.)

[14] G. Huston and R. Bush. Securing BGP. *Internet Protocol Journal*, 14(2), June 2011. (Cited on page 1.)

[15] iSDX Github Repository. *https://github.com/sdn-ixp/iSDX*. (Cited on pages 2 and 4.)

[16] T. Jim. Sd3: A trust management system with certified evaluation. In *Security and Privacy, 2001. S&P 2001. Proceedings. 2001 IEEE Symposium on*, pages 106–115. IEEE, 2001. (Cited on page 5.)

[17] P. Kazemian, G. Varghese, and N. McKeown. Header Space Analysis: Static Checking for Networks. In *USENIX Conference on Networked Systems Design and Implementation (NSDI)*, San Jose, CA, May 2012. (Cited on page 2.)

[18] B. Lampson, M. Abadi, M. Burrows, and E. Wobber. Authentication in distributed systems: Theory and practice. *ACM Transactions on Computer Systems (TOCS)*, 10(4):265–310, 1992. (Cited on pages 2 and 5.)

[19] N. Li, J. Feigenbaum, and B. N. Grosof. A logic-based knowledge representation for authorization with delegation (extended abstract). In *Proceedings of the 1999 IEEE Computer Security Foundations Workshop*, pages 162–174. IEEE Computer Society Press, June 1999. (Cited on page 2.)

[20] N. Li, B. N. Grosof, and J. Feigenbaum. Delegation Logic: A logic-based approach to distributed authorization. *ACM Transaction on Information and System Security (TISSEC)*, February 2003. To appear. (Cited on page 2.)

[21] N. Li, J. C. Mitchell, and W. H. Winsborough. Design of a role-based trust-management framework. In *Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on*, pages 114–130. IEEE, 2002. (Cited on pages 2 and 5.)

[22] Namecoin. *http://namecoin.info*. (Cited on page 3.)

[23] P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson, and G. Gu. A Security Enforcement Kernel for OpenFlow Networks. In *ACM SIGCOMM Workshop on Hot Topics in Software Defined Networks (HotSDN)*, pages 121–126. ACM, 2012. (Cited on page 5.)

[24] RadWare DefensePro. *http://www.radware.com/Products/DefensePro/*. (Cited on page 1.)

[25] T. Reddy, P. Patil, M. Geller, D. Wing, S. Rao, and M. Boucadair. Information model for ddos open threat signaling (dots), 2015. (Cited on pages 4 and 5.)

[26] F. B. Schneider, K. Walsh, and E. G. Sirer. Nexus Authorization Logic (NAL): Design Rationale and Applications. *ACM Transactions on Information and System Security (TISSEC)*, 14(1):8, 2011. (Cited on pages 2 and 5.)

[27] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar. Flowvisor: A network virtualization layer. *OpenFlow Switch Consortium, Tech. Rep*, 2009. (Cited on page 5.)

[28] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. M. Parulkar. Can the production network be the testbed? In *USENIX OSDI*, 2010. (Cited on page 5.)

[29] A. Shieh, E. G. Sirer, and F. B. Schneider. Netquery: A knowledge plane for reasoning about network properties. In *ACM SIGCOMM Computer Communication Review*, volume 41, pages 278–289. ACM, 2011. (Cited on page 5.)

[30] Verisign DDoS Protection. *verisigninc.com/website-availability/ddos-protection/index.xhtml*. (Cited on page 1.)

[31] E. Wobber, M. Abadi, M. Burrows, and B. Lampson. Authentication in the Taos Operating System. *ACM Transactions on Computer Systems (TOCS)*, 12(1):3–32, 1994. (Cited on pages 1, 2, 4 and 5.)

[32] W. Zhou, Y. Mao, B. T. Loo, and M. Abadi. Unified declarative platform for secure netwoked information systems. In *Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on*, pages 150–161. IEEE, 2009. (Cited on page 5.)