DUELMIPs: Optimizing SDN Functionality and Security

Timothy Curry
University of Connecticut, Storrs, CT, USA

Gabriel De Pace
University of Rhode Island, Kingston, RI, USA

Benjamin Fuller
University of Connecticut, Storrs, CT, USA

Laurent Michel
University of Connecticut, Storrs, CT, USA

Yan (Lindsay) Sun
University of Rhode Island, Kingston, RI, USA

Abstract

Software defined networks (SDNs) define a programmable network fabric that can be reconfigured to respect global networks properties. Securing against adversaries who try to exploit the network is an objective that conflicts with providing functionality. This paper proposes a two-stage mixed-integer programming framework. The first stage automates routing decisions for the flows to be carried by the network while maximizing readability and ease of use for network engineers. The second stage is meant to quickly respond to security breaches to automatically decide on network counter-measures to block the detected adversary. Both stages are computationally challenging and the security stage leverages large neighborhood search to quickly deliver effective response strategies. The approach is evaluated on synthetic networks of various sizes and shown to be effective for both its functional and security objectives.

2012 ACM Subject Classification Theory of computation → Network optimization; Networks → Network security; Security and privacy → Trust frameworks

Keywords and phrases Network security, mixed integer programming, large neighborhood search

Digital Object Identifier 10.4230/LIPIcs.CP.2022.17

Funding This work is supported by ONR Grant N00014-19-1-2327. B. F. is supported by NSF Awards #1849904, 2141033. L.M. and T. C. are partially supported by the Synchrony Chair in Cybersecurity. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. government.

1 Introduction

Software-Defined Networks (SDN) create a programmable network framework, potentially allowing organizations to simultaneously account for functionality, usability, and trust. Optimization techniques have been used to improve an SDN configuration *along a single dimension*: *Functionality* [2,17,24,36], *Usability* [21,28,32] or *Trust* [6,35]. Little work [10,19] integrates these goals because of the antagonistic objectives of the three dimensions. For instance, a fully permissive functionality policy based on shortest path optimizes routing but reduces trust as any adversary has unrestricted access through the network. For those readers unfamiliar with network attacks, Example 1 presents a toy network and attack.

This work produces optimization models that simultaneously consider routing, usability, and trust of the proposed SDN configuration. We consider two use cases:

Planning. When a new application is being stood up and network engineers want a configuration. To explain configurations, it is crucial to use models with clear objectives that prove solution quality. One is comfortable with a moderate computation time.

Rapid Response. Adapting when a security event occurs. Attackers rely on the sluggishness of defense responses to pivot within an organization [12]. In this setting, one needs to prepare a response in minutes. Automated responses are used to allow a security analyst to craft a permanent response.

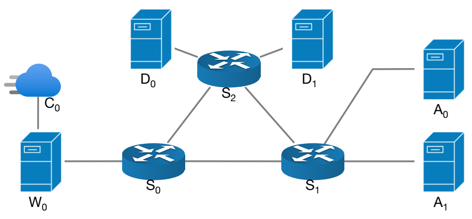
This work introduces DuelMIPs a *reactive* two-stage online mathematical programming model in which a mixed-integer programming (MIP) method incorporates conflicting requirements of routing, usability, and trust. The goal is, given a set of required flows and disallowed flows, to find a configuration that either routes or blocks flows while minimizing (1) the complexity of the resulting configuration, (2) the residual trust reduction if some device in the network is possibly compromised. The **Routing Model** is computationally heavy and runs first to produce a functional and usable configuration. The routing model, which one executes whenever functional requirements have substantively changed, produces routing decisions for all SDN devices in the network. It synthesizes small configurations that are easy to understand and minimizes conflicts and defects that hurt interpretability [8, 34]. The production of an optimized configuration is followed by a deployment of the decisions onto the SDN devices.

Network appliances detect security events [20, 25]. The **Trust Model** is run in response to such a detection. This new information affects trust levels of the targeted devices and related devices, allowing the Trust model to decide on actions to mitigate the detected threat. Depending on which device was targeted, the network flows, and the loss of trust, the second stage decides on traffic restrictions to minimize the global trust reduction. It proposes blocking rules meant to hamper lateral movements within the network.

This paper offers the following contributions:

- It adapts multi-commodity flows, common-place in supply chains, to the specifics of routing in data networks. It handles wildcards in routing rules and the production of SDN programs (rule sets) at SDN devices.
- It articulates a trust model that is suitable for rapid response to security events. This model is a physics-based approach using an elastic string networks analogy to capture inter-node trust relations and mitigate the impact on trust with counter-measures.

▶ **Example 1.** Consider the network topology and routing policy presented in Figure 1. This is a simplified example from our evaluation topology described in Section 5. The three parts of Figure 1 are the inputs for the DuelMIPs's routing model. Figure 1a shows the physical connection between the SDN routers and end hosts. The goal of the **Routing Model** is to install rulesets on the routers such that the flows in the required (Figure 1b) and restricted tables (Figure 1c) will be serviced and blocked, respectively. In both cases, flows are described by a triple of source (W_0), destination (A_0), and application-level protocol (`SQL`). There are multiple paths that could be chosen for each flow (the set of possible paths is defined by the topology) and *many* different valid rulesets to satisfy each pathing choice. Specifically, SDN routers route a packet according to the most specific (and first) rule that applies to that packet, describing which physical *switch port* the packet should be sent on (or not forwarded). This is known as the longest prefix match rule. We slightly abuse notation in Table 1 by listing the device connected through the *switch port*.



(a) Network topology.

Required
$W_0 \xleftrightarrow{\texttt{HTTP}} A_0$
$W_0 \xleftrightarrow{\texttt{HTTP}} A_1$
$A_0 \xleftrightarrow{\texttt{SQL}} D_0$
$A_1 \xleftrightarrow{\texttt{SQL}} D_1$

(b) Required flows.

Restricted
$W_0 \xleftrightarrow{\texttt{HTTP}} D_0$
$W_0 \xleftrightarrow{\texttt{HTTP}} D_1$
$W_0 \xleftrightarrow{\texttt{SQL}} D_0$
$W_0 \xleftrightarrow{\texttt{SQL}} D_1$

(c) Restricted flows.

Figure 1 Network topology, routing policy, and inputs for DuelMIPs's routing model.

Table 1 Rulesets for SDN routers that satisfy the requirements presented in Figure 1. These rules are part of the output of DuelMIPs in this example. Note the red, bold rule at the top of S_2's ruleset is part of the trust model output, while all other rules come from the routing model.

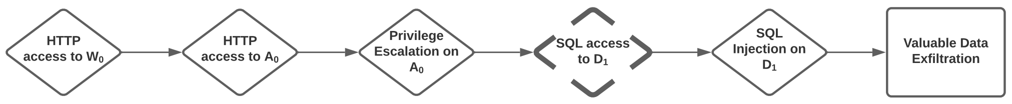
S_0 Ruleset		S_1 Ruleset		S_2 Ruleset	
Match	Action	Match	Action	Match	Action
$(W_0, D_0, *)$	DROP	$(*, W_0, *)$	SEND(S_0)	**$(D_1, A_0, \texttt{SQL})$**	**DROP**
$(W_0, D_1, *)$	DROP	$(*, A_0, *)$	SEND(A_0)	$(*, A_0, *)$	SEND(S_1)
$(*, W_0, *)$	SEND(W_0)	$(*, A_1, *)$	SEND(A_1)	$(*, A_1, *)$	SEND(S_1)
$(*, A_0, *)$	SEND(S_1)	$(*, D_0, *)$	SEND(S_2)	$(*, D_0, *)$	SEND(D_0)
$(*, A_1, *)$	SEND(S_1)	$(*, D_1, *)$	SEND(S_2)	$(*, D_1, *)$	SEND(D_1)

The routing model in DuelMIPs can generate rules with match fields containing up to two wildcards. (As rules are described by a triple, three wildcards would mean that the same rule is applied universally which is not appropriate for a router.) Furthermore, the model prefers to use rules with more wildcards (which match any value), as it allows for more compact and readable rulesets. This is in line with ruleset configurations designed by human network engineers [28]. Producing rules with wildcards is important as network engineers will only deploy rulesets that they can understand and modify. An example configuration that could be produced by DuelMIPs is in Table 1.

One downside of using wildcard-based rules is that they allow *incidental* flows to be serviced within the network. Consider the solution to the given input represented by Table 1 (minus the bolded, red rule in S_2's table). This is the output of DuelMIPs's routing model. The use of wildcards makes the ruleset readable. However, it routes flows that are not included in the network policy. A packet of type $A_0 \xrightarrow{\texttt{SQL}} D_1$ will be routed $A_0 \longrightarrow S_1 \longrightarrow S_2 \longrightarrow D_1$ because of S_1's fifth rule and S_2's fifth rule. An analogous route exists for the return traffic $D_1 \xrightarrow{\texttt{SQL}} A_0$. Wildcard usage is cruical for ruleset compactness, utility, and readability, but it can permit incidental internal flows that enable network attacks.

The **Trust Model** reduces the trust loss due to incidental flows in cases when an end host is marked as suspicious. Suppose a network appliance alerts DuelMIPs that W_0 has received suspicious traffic from an external client, C_0. Modern network attacks often require multiple steps to reach their goal [22], called exploit chains. The suspicious traffic at W_0 is due to an attacker who is attempting to leverage the exploit chain (see Figure 2). The trust model is run with W_0 as the focus of distrust. DuelMIPs observes the incidental `SQL` flow between D_1 and A_0. Since A_0 is directly connected to W_0, there is a *transitive* trust loss induced on D_1. DuelMIPs reduces this loss by blocking incidental flows with firewalls (if the improvement surpasses the complexity cost of adding the firewalls). In this case, a drop rule is added to S_2's routing table, shown in red. This rule does not block any required flows.

Figure 2 Exploit chain using in a pivot attack on the example network. Here the only link that DuelMIPs could affect via a routing configuration is shown with a dashed outline.



Organization. The rest of this work is organized as follows: Section 2 provides an overview of the model goals, Section 3 presents the models, Section 4 the assessment metrics, Section 5 our experimental setup, Section 6 the results, and Section 7 concludes.

2 DuelMIPs objectives

This section outlines the routing (Sec 2.1) and trust (Sec 2.2) objectives. Section 3 describes how they give rise to mathematical programming models. In the following, a network is formed from set of *devices* $D = H \cup R$ that are either hosts H or SDN routers R. Assume that $|D| = n$. Devices are connected by network arcs (wires) from $A \subseteq D \times D$. Giving rise to a graph $\mathcal{G}(D, A)$. A data flow $f = \langle s, d, t \rangle$ indicates the requirement to transport data packets for an application-level protocol t from a source device s (typically a host) to a destination device d (again, typically a host). A protocol t is from $\mathcal{P} = \{\texttt{SSL}, \texttt{HTTP}, \texttt{LDAP}, \texttt{MYSQL}, \texttt{SMTP}, \cdots\}$ with $|\mathcal{P}| = p$. Let $\mathcal{F} = D^2 \times \mathcal{P}$ be the universe of all possible flows and $|\mathcal{F}| = n^2 \cdot p$.

2.1 Routing Objective

The network configuration problem is given a set of required $F^+ \subseteq \mathcal{F}$ and disallowed $F^- \subseteq \mathcal{F}$ flows. The objective is to find a set of rules, for each router, which routes all flows in F^+ and ensures that all flows in F^- are blocked. We consider routing rules of the type $\langle \texttt{s}, \texttt{d}, \texttt{t} \rangle \Rightarrow$ `action` in which $s, d \in D \cup \{*\}$, $t \in \mathcal{P} \cup \{*\}$ and the wildcard `*` means that any value in the corresponding set is permissible. Headers of the packet being routed [26] are against s, d, and t to select the applicable rule from the routing table. The `action` could be dropping the packet, or forwarding to a specific neighboring router g, i.e., `action` $\in \{\texttt{DROP}, \texttt{SEND}(g)\}$. For example, $\langle *, \texttt{192.168.1.10}, \texttt{HTTP} \rangle \Rightarrow \texttt{DROP}$ states that any inbound `HTTP` traffic meant for `192.168.1.10` should be dropped. Observe how a route for a flow f is implemented by routing rules in network devices along the path. For instance, in Figure 1, the flow $W_0 \overset{\texttt{HTTP}}{\longleftrightarrow} A_1$ uses the fourth wildcard routing rule in S_0 and the third in S_1 to reach its destination A_1.

When configuring a network one must determine 1) the paths for all flows in F^+ and 2) the table rules to be installed in routers. Specifically, the routing objective is the following:

1. Ensure that all flows in F^+ have a single path through the network (known as single path routing used in intranet routing [23] and Internet routing [27]),
2. Ensure that all flows in F^- are blocked,
3. Ensure for each r on the path of $f \in F^+ \cup F^-$ there is *at most one wildcard rule* that matches f. We do not require that $f \in F^-$ are blocked at the first possible device so this constraint is applied to both F^+ and F^-,
4. Minimize the total length of routes for required flows, and
5. Minimize the complexity of SDN rules that are deployed on each device.

Property 3 improves usability of the resulting SDN rules. See Section 4 which builds on [5,33]. This objective ensures that routes are set for all required flows that avoid routing loops and minimize latency (shortest paths). This should be achieved with a small and simple rule set that is *usable* by network engineers, i.e., the rules must be natural and easily explainable. While wildcards reduce the complexity of a rule set, they *also* authorize *incidental flows* that were not explicitly required nor explicitly forbidden, this set is denoted by I. It is this set of *incidental* flows that is the focus of the trust model. We use F_r to denote the set of all routable flows, mathematically $F_r = F^+ \cup I$. We use $F_r(a,b)$ to restrict to all flows between hosts a and b with analogous notation for F^+, F^- and I.

2.2 Trust Objective

The trust model minimizes the impact of a *detected* security event on the *overall trust* in the network. It is used in the setting when an attacker is trying to pivot to collect important *privileges* [22], usually resulting in compromising, i.e., gaining unfettered access to some critical component such as a customer database. For instance, one could compromise the Gmail password of an HR representative and try this password in the HR system, learn biographic information about a system administrator and use it to carry out a spear phishing campaign. In many modern exploit chains, the attacker relies on *similarity* between systems they are compromising to carry out such a pivot and gain new privileges.

This section describes basic trust modeling concepts [1,14,18,37]. A trust model is defined over an undirected graph $\mathcal{G}'(H, F_r)$ as above H represents network end devices or hosts, while F_r is the set of flows that can be routed in the network. Note this graph $\mathcal{G}'$ is different than $\mathcal{G}(D, A)$ considered in the routing model. We consider two time values `Init` and `Final`, however, all notation and modeling can naturally be extended to a sequence of events over time.

Trust. Let the trust of a host $h \in H$, be $\texttt{Trust}(h) \in \mathbb{R}^+$. Let $\texttt{Trust}^{\texttt{Init}}(h)$ and $\texttt{Trust}^{\texttt{Final}}(h)$ refer to the initial and final trust values for a host reflecting its hardening [29].

Similarities. *Incidental* flows increase connectivity between hosts and offer additional pathways to reach other hosts. This paper codifies similarity as the fraction of *incidental flows* over *total flows* between two hosts. That is, given a universe of possible flows $F_r(a,b)$ between two distinct hosts a and b, a set of required flows $F^+(a,b)$ between them and $I(a,b)$ as the set of incidental flows between a and b, one can define

$$s_{a,b} \overset{def}{=} |F^+(a,b) \cup I(a,b)|/|F_r(a,b)| \in [0,1] \tag{1}$$

as the similarity between hosts a and b [16,40]. It is affected by paths in $\mathcal{G}'$ connecting a to b and it affects how the trust of a device changes as a result of security events at nearby connected devices.

Event. Let $e(h_i)$ denote a security event affecting host $h_i \in H$. The event $e(h_i)$ induces a *direct trust reduction* defined as $\texttt{TrustRed}(h_i) = \texttt{Trust}^{\texttt{Init}}(h_i) - \texttt{Trust}^{\texttt{Final}}(h_i) \geq 0$.

Event Set. An event set $\mathcal{E}$ conveys the simultaneous occurrence of multiple events.

Trust models propagate trust reductions that one experiences as a result of an event set $\mathcal{E}$. A trust reduction at host h_i *affects* all hosts indirectly connected to h_i. This propagation of *indirect trust reductions* is captured with an elastic string model which will be described in Section 3.2. The objective is to choose traffic blocking measures that minimize the total trust reduction.

3 Optimization Models

The optimization framework consists of two models, a **Routing Model** and a **Trust Model**. The Routing model produces network configurations that meet functional requirements and are usable by network engineers (we describe metrics in Section 4). The Trust model minimizes trust reduction of a presented network compromise.

The coupling between the two models is achieved through *similarity variables* defined in Section 2.2. These variables are derived from the routing model solution and can be altered due to trust model decisions. An increased similarity between two nodes can either increase or reduce trust levels. Simpler, shorter and more usable configurations often leverage routing wildcards that adversely impact node similarities and prompt the creation of more complex defensive measures. This tension between desirable and secure configurations is the heart of the problem hardness.

3.1 Routing Model

The routing model decides on the content of all routing tables to carry required flow, block restricted flows while minimizing routing cost and maximizing the usability of the rule sets to be installed on SDN devices. The input is the graph $\mathcal{G}(D, A)$, required flows F^+, and disallowed flows F^-. The mathematical formulation is a multi-commodity network flow. Required flows originate from a super-source node, flow into their actual source node (the host from which the flow originates) and must reach a super-sink via the actual destination node. An unusual structure in the model results from wildcards, a feature not normally seen in multi-commodity flows. There are, for each network device, Boolean variables to convey forwarding or dropping actions for each possible packet header that condition the routing rule (the $r^s_{i,j}$ and w^s_i variables below).

Parameters

F^+, F^- – set of required (resp. restricted) flows that must be routed (resp. blocked).
$F = F^+ \cup F^-$ – set of all flows in network routing policy.
S – set of possible SDN routing table headers in network.
S_f – set of all SDN routing table headers that flow f matches.
$S^*_f \subset S_f$ – set of SDN routing table headers with wildcards that flow f matches.
R, H, $D = H \cup R$ – set of routers, hosts and all devices in the network.
$A \subseteq D \times D$ – set of arcs (links) in the network.
σ,τ – abstract flow network super source and super sink.
$\delta^-(i)$ and $\delta^+(i)$ – set of predecessors, resp. successors of device $i \in D$.
$\mathsf{src}(f)$ and $\mathsf{dst}(f)$ – the source and destination of flow f
$\mathsf{cost}(s)$ – the cost (in terms of complexity) of using header s for a routing rule.
$\mathcal{M}$ – big-M constant.

Decision Variables

$c^f_{i,j} \in \{0, 1\}$ – Link (i, j) *carries* flow f.
$p^f_{i,j} \in \{0, 1\}$ – Link (i, j) is *permitted* (via chosen SDN rules) to carry flow f.
$a^f_i \in \{0, 1\}$ – Flow f *reaches* device $i \in D$.
$r^s_{i,j} \in \{0, 1\}$ – Device i *forwards* flows matching header $s \in S$ to device j.
$w^s_i \in \{0, 1\}$ – Device i *firewalls* flows matching header $s \in S$.

Constraints

The first constraints model network flows (injection 2 and conservation 3,4).

$$\forall f \in F \ : \ c^f_{\sigma,f.src} = 1 \tag{2}$$
$$\forall j \in R \ : \ \sum_{i\in\delta^-(j)} c^f_{i,j} = \sum_{k\in\delta^+(j)} c^f_{j,k} + \sum_{s\in S_f} w^s_j \tag{3}$$
$$\forall j \in H \ : \ \sum_{i\in\delta^-(j)} c^f_{i,j} = \sum_{k\in\delta^+(j)} c^f_{j,k} \tag{4}$$
$$\forall f \in F \ : \ a^f_\sigma = 1 \tag{5}$$
$$\forall f \in F \ : \ p^f_{\sigma,\mathsf{src}(f)} = 1 \tag{6}$$
$$\forall f \in F \ , \forall j \in \delta^+(\sigma) \text{ where } i \neq \mathsf{src}(f) \ : p^f_{\sigma,i} = 0$$

Equations 5 - 10 ensure that flows are well formed. Namely, they properly spawn from the source, arrive at the sink, are carried when required (F^+) and blocked when forbidden (F^-).

$$\forall f \in F \ : \ p^f_{\mathsf{dst}(f),\tau} = 1 \tag{7}$$
$$\forall f \in F \ , \forall i \in \delta^-(\tau) \text{ where } i \neq \mathsf{dst}(f) \ : p^f_{i,\tau} = 0$$
$$\forall f \in F \ , \ \forall i \in \delta^+(\mathsf{src}(f)) \ : p^f_{\mathsf{src}(f),i} = 1 \tag{8}$$
$$\forall f \in F^+ \ : c^f_{\mathsf{dst}(f),\tau} = 1 \tag{9}$$
$$\forall f \in F^- \ : c^f_{\mathsf{dst}(f),\tau} = 0 \tag{10}$$

The remaining constraints focus on the rule implementations on network devices.

$$\forall (i,j) \in A \ , \ \forall f \in F \ : \ p^f_{i,j} = \bigvee_{s\in S_f} r^s_{i,j} \tag{11}$$
$$\forall (i,j) \in A \ , \ \forall f \in F \ : \ c^f_{i,j} \leq p^f_{i,j} \tag{12}$$
$$\forall j \in D \ , \ \forall f \in F \ : \ a^f_j = \bigvee_{i\in\delta^-(j)} (a^f_i \wedge (p^f_{i,j})) \tag{13}$$
$$\forall (i,j) \in A \ , \ \forall f \in F \ : \ \sum_{s\in S_f} r^s_{i,j} \geq c^f_{i,j} \tag{14}$$
$$\forall j \in R \ , \ \forall f \in F \ : \ \sum_{k\in\delta^+(j)} \sum_{s\in S^*_f} r^s_{j,k} + \sum_{s\in S^*_f} w^s_j \leq 1 + \mathcal{M}(1 - a^f_j) \tag{15}$$

Equation 11 states that a flow is permitted to travel over a link if there is a rule installed on a device that allows it to do so. Equation 12 ensures that a required flow will only travel over a link if that link permits it. Equation 13 enforces that a flow is able to reach a device j only if the flow can reach a direct predecessor and the predecessor has a rule installed that permits the flow to travel over the direct link to j.

Equation 14 requires that a proper forwarding rule is installed if a link is set to carry a flow. Equation 15 states that if a flow reaches a router, then there is at most one wildcard rule covering that flow on the router. This constraint is needed to ensure clear switch behavior (see Section 4). Equation 15 is vacuously true for a flow that does not reach the router.

Routing Objective

The objective function for the routing model is

$$\min \quad \alpha_0 \cdot \sum_{f \in F} \sum_{(i,j) \in A} c^f_{i,j} + \alpha_1 \cdot \sum_{s \in S} \left(\sum_{(i,j) \in A} r^s_{i,j} \cdot \mathsf{cost}(s) + \sum_{i \in R} w^s_i \cdot \mathsf{cost}(s) \right)$$

It minimizes the routing costs (number of links utilized) and the complexity of all rule sets.

3.1.1 Incidental Flow Extraction

Finally, it is desirable to compute the set of incidental flows I introduced by using rules with wildcards. We identify such flows using any complete graph traversal algorithm over the topology and rulesets to identify flows that were not in F^+ nor F^- but are routable.

3.2 Trust Model

The trust model receives information regarding the trustworthiness of network hosts (H), as well as a list of all deliverable flows, i.e., $F_r = F^+ \cup I$. The purpose of the model is to reason about the connections induced by incidental flows (I) that are potentially damaging and should be blocked via additional rules.

This model is inspired by physics. Each host $h \in H$ is modeled as a metal ball resting at a specific height in one-dimensional space. The host's initial trust M_h is its *mass* while the height of a ball represents $\texttt{TrustRed}(h)$. The model considers a similarity graph $\mathcal{G}'(H, C)$ in which the edge set $C \subseteq F_r$ may have a connection for each flow in F_r. The edge set C is defined as

$$C = \{(h_1, h_2) | (h_1, h_2) \in F_r \wedge s_{h_1,h_2} > 0\} \tag{16}$$

Namely, an edge exists if there is a flow between the two hosts and the two hosts have a non-zero similarity as defined in Equation 1. Note that similarity is over $\mathcal{G}'(H, C)$ while routing considered $\mathcal{G}(D, A)$. In the physics analogy, each edge in C is an *elastic string whose tautness captures the similarity*. Indeed, highly similar nodes that are connected are more likely to see their trust move in unison if one is affected by a security event whereas highly dissimilar nodes would not affect each other if one was compromised. Tautness is modeled as the maximal length that the elastic string can have. At the onset, there are no trust reductions anywhere, i.e., all balls lie on a flat surface at height 0.

Without loss of generality, assume a single security event $e(h)$. The event $e(h)$ causes $\texttt{TrustRed}(h) > 0$ by picking up host h to a prescribed height equal to the trust reduction directly incurred. That is, *height* $= \texttt{TrustRed}(h)$. Raising host h to some height impacts connected neighbors. Those hosts are lifted (a trust reduction) because of their respective similarities to h. The behaviors of the elastic strings impact the magnitude of the collateral lifts. In the physics analogy, connected devices are lifted as a function of their mass and connections to neighbors. Massive hosts (highly trusted) are not easily lifted leading to a dissipation of the upward force through the stretching of the elastic links as well as tensile forces. There are multiple ways to model similarity:

- alter the initial length of strings,
- characterize the elasticity constant behind each string, or
- bound the maximal extension length of a string.

which have subtle computational implications. Each approach leads to quantitatively different yet qualitatively similar results. We chose to encode similarity as the maximum extension length of a string. This proved most suitable for linear optimization.

Theoretical trust models [9] exist to address how trust can propagate [31]. Here, we focus on how *distrust* propagates, i.e., direct and indirect trust loss. Distrust propagation is different and is usually application dependent. Guha et al. [15] review the process for developing a trust model for an application. The elastic string model, which is novel, elegantly captures both direct and indirect (i.e., transitive) trust loss and is intuitive. It reflects the nature of modern attackers for two reasons: (1) it chains together exploits across the network and build up capabilities, (2) it capture pivoting abilities through similarities between nodes.

Parameters

H – set of nodes in the trust network. Same as set of hosts in physical network.
M_h – mass of node $h \in H$.
C – similarity connections. Each connection $(h_1, h_2) \in C$ connects h_1 to h_2. See Eq 16.
$C(h)$ – set of connections touching host $h \in H$.
L_c – maximum length of string for connection $c \in C$.
K_c – elastic constant for connection $c \in C$.
$\mathcal{E} \subseteq H$ - set of hosts experiencing trust loss due to potential network compromise.
$\mathcal{E}(h) \subset \mathbb{R}^+$ - trust reduction experienced by host $h \in \mathcal{E}$.
F^+ – the set of required flows to be delivered based on the routing policy.
I – the set of incidental flows induced by the network configuration.
$F^+(h_1, h_2) \subseteq F^+$ – the set of flows in F^+ involving h_1 and h_2.
$I(h_1, h_2) \subseteq I$ – the set of flows in I involving h_1 and h_2.

Decision Variables

$tl_h \in \mathbb{R}_{\geq 0}$ – trust loss for host $h \in H$.
$f_{c,h} \in \mathbb{R}$ – elastic force experienced by host $h \in H$ due to connection $c \in C$.
$t_{c,h} \in \mathbb{R}$ – tensile force experienced by host $h \in H$ due to connection $c \in C$.
$b_h \in \mathbb{R}_{\geq 0}$ – supporting force from ground experienced by host $h \in H$.
$l_h \in \mathbb{R}_{\geq 0}$ – lifting force experienced by host $h \in H$.
$s_{h_1,h_2} \in \mathbb{R}_{\geq 0}$ – flow similarity between hosts $h_1, h_2 \in H$.
$w_c \in \mathbb{R}_{\geq 0}$ – effective maximum length of connection $c \in C$.
$z_f \in \{0, 1\}$ – binary variable to install rule to block an incidental flow $f \in I$.

Constraints

The first set of constraints deal with network events which compromise the trust of hosts, causing them to be lifted

$$\forall h \in \mathcal{E} \ : \ tl_h \geq \mathcal{E}(h) \ , \qquad \forall h \in H \setminus \mathcal{E} \ : l_h = 0 \qquad (17, 18)$$

Equation 17 lower bounds the trust loss for the hosts involved in the event to be at least the loss specified in the event. These hosts will then experience a lifting force necessary to stay at height tl_h and be in equilibrium described by Equation 29. Equation 18 states that hosts that are not involved in the event do not experience a lifting force.

$$\forall h \in H \ : b_h \leq M_h \ , \qquad b_h > 0 \iff tl_h = 0 \qquad (19, 20)$$

Equation 19 bounds the supporting force exerted by the ground on a host to be no greater than the weight of the host. Equation 20 dictates that the supporting force on a host is present if and only if the host is resting on the ground (did not incur a trust loss).

$$\forall (h_1,h_2) \in C \ : w_{(h_1,h_2)} = L_{(h_1,h_2)} - s_{h_1,h_2} \tag{21}$$

$$\forall (h_1,h_2) \in C \ : s_{h_1,h_2} = |F^+(h_1,h_2)| \ + \sum_{f \in I(h_1,h_2)} (1 - z_f) \tag{22}$$

Equation 21 gives the effective length of a string to be used in the force calculations. Higher similarity between two hosts entails a shorter maximal distance between them. Equation 22 calculates the flow similarity between two hosts. By design, $s_{h_1,h_2} \leq L_c$ for all $(h_1,h_2) \in C$.

$$\forall c \in C \ : f_{c,c_1} = K_c \cdot (tl_{c_2} - tl_{c_1}) \ , \qquad f_{c,c_2} = K_c \cdot (tl_{c_1} - tl_{c_2}) \tag{23, 24}$$

$$\forall c \in C \ : |tl_{c_1} - tl_{c_2}| \leq w_c \ , \qquad t_{c,c_1} + t_{c,c_2} = 0 \tag{25, 26}$$

Equations 23 and 24 determine the elastic forces a host experiences due to connections in which it is involved. This calculation is based directly on Hooke's Law. Equation 25 ensures that the distance between the hosts does not exceed the maximum length of the connection. Equation 26 states that the tensile forces experienced by the hosts sum to zero, i.e., the forces are equal in magnitude and opposite in direction.

$$\forall c \in C \ : t_{c,c_1} < 0 \iff tl_{c_1} > tl_{c_2} \ , \quad t_{c,c_1} \neq 0 \iff w_c - |tl_{c_1} - tl_{c_2}| = 0 \tag{27, 28}$$

Equation 27 states that a host can only experience negative tension from a connection if it is higher than the other host in the connection. Furthermore, Equation 28 prevents a host from experiencing any tension from a connection if the string is not at its maximum length. The last set of constraints require the solution to be an equilibrium state for the string network.

$$\forall h \in H \ : \left(\sum_{c \in C(h)} f_{c,h} + t_{c,h} \right) + b_h + l_h - M_h = 0 \tag{29}$$

Equation 29 stipulates that the upward and downward forces experienced by each host must balance. This integrates lifting, ground, tensile, gravitational, and connection forces. To summarize, equations 17-29 are based on laws of classical mechanics and the desire to capture indirect impacts of events described at the beginning of this subsection.

Trust Objective

The trust model minimizes a linear combination of the heights of the hosts and the number of needed additional firewall rules. This mitigates trust reduction across network without adding excessive complexity to the current routing configuration. This objective function is expressed below (β_0, β_1, and γ_f are parameters).

$$\min \beta_0 \sum_{h \in H} tl_h + \beta_1 \sum_{f \in I} \gamma_f \cdot z_f \tag{30}$$

Solving the Trust Model

A direct resolution of the trust model cannot complete to optimality in the small amount of time (minutes) one would have to react to a security incident.

With LNS [30] as a meta-strategy, one can achieve high-quality solutions in seconds to minutes at worst. The incomplete search can then proceed as shown in Algorithm 1 with t being a parameter denoting the chunk size for the LNS.

Algorithm 1 Solve_TrustModel(M over $G(V, E), e(h) \in \mathcal{E}, t$).

```
$S^* = \text{solve}_{LNS}\ MIP(M(e(h)) \cup \{z_f = 0 | f \in F_r\})$
while $more\ time$ do
    $Pick\ X$ such that $|X| = t$ and $X$ is a random subset of $I$
    $S = \text{solve}_{LNS}\ MIP(M(e(h)) \cup \{z_f = S^*(z_f) | f \in F_r \setminus X\})$
    if $S < S^* \wedge \text{feasible}(S)$ then
        $S^* = S$
    end if
end while
```

The algorithm starts by fixing *all* z variables to 0, i.e., it reports on the trust reduction one would experience when taking no actions whatsoever. $M(e(h))$ denotes the MIP model for the considered event. Then it proceeds in waves of LNS searches, each focusing on a random subset X (of fixed cardinality t) of edges that coincide with incidental flows. Each LNS searches fixes all other edges ($E \setminus X$) to their values in the incumbent solution ($S^*(z_f)$) and optimizes over the z_f for all $f \in X$. Any feasible improving solution S in adopted as the new incumbent. For our testing, we set t to be 50.

3.3 Integrated Approach

It is natural to consider an integrated approach with a single model that encompasses both routing and security requirements. Such an model can be produced with a few changes. First, the routing model can no longer limit its attention to the flows in $F = F^+ \cup F^-$. Indeed, the trust part of the model must decide whether to block incidental flows whose existence is implied by the value chosen for the routing variables. Thus, one must consider F as the entire universe of flows, namely $F = \mathcal{F}$. An immediate consequence is a significantly larger set of decision variables for routing. Second, incidental flows are readily encoded in the a^f_τ variables. Third, the trust model must restrict its decisions to z_f variables for flows in $\mathcal{F} \setminus (F^+ \cup F^-)$ since required flows cannot be blocked (and restricted flows are already blocked). Finally, the objective function can be a lexicographic ordering of the objectives from routing and trust (i.e., a weighted sum).

We implemented such a model and the resulting approach was not tractable. On the introductory toy example, it produces the same solution as the 2-stage approach. Looking ahead we evaluate our two stage model on pod-4 and pod-6 fat tree topologies found in data center networks (see Section 5). On the smallest pod-4 instances of the empirical evaluation, the integrated model had a duality gap in excess of 5% after 8 hours of computation time. The last solution offered the same routing as the 2-stage model, while the loss of trust was twice as big (the value of the trust objective that is being minimized). Worse, the model size alone is prohibitive, causing the solver pre-processing phase to take more than 10 minutes. The Achilles' heel of the approach is the quadratic size of $\mathcal{F}$. Finally, note that an LNS approach here requires an incumbent solution that the integrated model does not find early in the execution (the first solution on pod-4 is found after one hour of runtime).

Another tempting avenue is column-generation [11]. The on-demand creation of routes with the addition of an indicator variable to choose such a route seems direct. Yet, every route must create several variables for the incidental flows introduced by itself *but also because of the interplay with other selected routes.* A column generation must therefore produce a bundle of several columns (for the route and the incidental flows it creates by itself) as well as examine all the existing routes to add another set of columns for the incidental flows that arise from mixing routes. While this is doable and possibly competitive, we believe the complexity of the implementation would exceed the two-stage approach.

4 Assessment

Optimization Assessment. To assess our optimization model we report on the following standard metrics: 1) solve time, 2) optimality gap, 3) model size.

Functional Assessment Our motivating application for DuelMIPs is in data center networks where links are uniform and high bandwidth [3]. The following quality metrics are used:

Normalized Path Length. For each f_{h_1,h_2} the path length divided by the shortest network path between h_1 and h_2.

Normalized Number of Rules. The total number of rules written to SDN devices throughout the network is normalized by the number of required rules.[1] The desire for compact rulesets is driven by two factors: 1) SDN devices are limited in their TCAM memory and 2) human interpretability.

SDN Rules. The total number of rules assigned to the SDN devices.

Wildcard Rules. The number of rules that are not fully specified. Wildcard rules are more complex than fully specified rules and thus should be given additional weight when judging the complexity of a configuration.

Usability Assessment. The literature on usability of network rules [4, 5, 33, 38, 39] considers the following factors to be important: structural complexity of the ruleset, size of the ruleset, misconfigurations, conflicts and the presence of comments to aid in the intention of the rules.

All rules have the form of $(\texttt{src}, \texttt{dest}, \texttt{protocol}) \rightarrow \textit{action}$ in which *action* can be `drop` or `send(port)`. Therefore, the structural complexity of each rule largely depends on the number of wildcards. Recall that wildcards can reduce the number of rules needed to specify a routing policy, yet multiple occurrences on the same device can induce conflicts that weaken clarity. The number of rules and wildcard rules is reported in Functional Assessment.

We focus on the number of *conflicts* in the ruleset, using the classification due to Al-Shaer and Hamed [5]. Before describing conflict types recall that SDN devices match the most specific rule, if two rules are the same specificity, the first listed rule is applied. Conflicts indicate the complexity for human understanding.

Shadowing. A generic rule R_B appears before a specific rule R_A, hiding R_A.

Generalization. A generic rule R_A appears after a specific rule R_B.

Correlation. Two rules match some packets in common.

Redundancy. Rule R_A generalizes R_B and both feature the same action.

Irrelevance. Rule R_A matches no packets or all packets are handled by earlier rules.

Trust Assessment. To assess the Trust Model, we consider two versions. The first we call Initial, denoted as $\aleph$, where no extraneous flows are blocked. This corresponds to a model in which all z_f are fixed to 0. The second version is the actual solution, denoted as F. The event node in each scenario is lifted to height 8 (see Section 5). We split our evaluation into two components: 1) directly examining the solution quality and 2) evaluating the solution with respect to stopping potential attacks. Before these metrics, we introduce required notation.

Notation. Denote heights buckets of $[0, 2], (2, 4], (4, 6], (6, 8]$, as vLow, Low, Med, High respectively. Consider the multigraph $G_{\mathsf{m}}(H, E)$, where edges in E represent a permitted flow between two end hosts in H. Define $G_{\mathsf{m},F}$ as G_{m} when $I = \emptyset$.

[1] Let $\texttt{RequiredRules}(f) = 1$ if $f \in F^-$ and $\texttt{RequiredRules}(f) = \texttt{ShortestPath}(f) - 1$ otherwise. Then we normalize the number of rules by $\sum_{f \in F} \texttt{RequiredRules}(f)$.

Table 2 Network Roles. For intrapod flows, each server in a pod communicates with a chosen *manager* server within the pod via `HTTP`. Communicate pattern is created randomly with the required type and number of flows.

Pod #	Role	Each server communicates with
1	Web server	2-3 application servers via HTTP **NO** HTTP to database or authentication
2	Application Server	1-2 content servers in each pod via RPC protocol 1-2 databases via SQL protocol 1-2 authentication servers via LDAP
3, 4	Content Server	Application
5	Database Server	Application
6	Authentication Server	Application

Node Heights Δ. The difference between configurations $\aleph$ and F of $|\mathsf{vLow}|, |\mathsf{Low}|, |\mathsf{Med}|, |\mathsf{High}|$.

Potential Attacks – Attack Paths and Min Cut. Let $G_{\mathsf{m},T}$ be the subgraph of G derived from DuelMIPs$'s$ solution and let n denote the number of edges in I removed. Let $G_{\mathsf{m},R}$ be a subgraph of G where n random edges in I are removed. We consider two metrics:

1. How many attack paths are removed by DuelMIPs's defensive actions. Consider all paths up to some maximum length in G_m which i) contain at least one edge from I and ii) that connect a web server (an attack source a) to a database or authentication server (attack target t), denoted as $P_{a,t}(G_\mathsf{m})$. The number of removed attack paths is measured by $\mathsf{Gsize}(G_{\mathsf{m},T}) := P_{a,t}(G_{\mathsf{m},T})/P_{a,t}(G_\mathsf{m})$ (defined analogously for $G_{\mathsf{m},R}$).
2. The variety of attacks available to the attacker. Measured by the relative min-cut

$$\mathsf{resMin}(G_{\mathsf{m},T}, a, t) := \frac{\mathsf{minCut}(G_{\mathsf{m},T}, a, t) - \mathsf{minCut}(G_{\mathsf{m},F}, a, t)}{\mathsf{minCut}(G_\mathsf{m}, a, t) - \mathsf{minCut}(G_{\mathsf{m},F}, a, t)}$$

   with $\mathsf{resMin}(G_{\mathsf{m},R,a,t})$ defined analogously. Define $\mathsf{resMin}(G_{\mathsf{m},T})$ as the expected value of $\mathsf{resMin}(G_{\mathsf{m},T}, a, t)$ over all webservers a and database or authentication servers t with $\mathsf{resMin}(G_{\mathsf{m},R})$ defined analogously.

5 Experimental Setup

We created synthetic instances that model traffic that would be observed in a moderately sized data center. The instances utilize the Fat-Tree [3] topology commonly seen in data centers due to its large aggregate bandwidth, high fault tolerance, and robust scalability. We use six pods, allowing for 54 endpoint devices to be present with 46 SDN devices. We consider these endpoint devices to be servers in our experiments.

Network Topology, F^+, and F^-. We assign roles to the servers in each of the six pods. The required and restricted communication patterns are based around these roles, this is found in Table 2. All evaluation considers three variants: no wildcards, at most one wildcard per rule, and at most two wildcards. No incidental flows are possible when no wildcards are used, i.e. $I = \emptyset$. The parameters in the objective are set as $\alpha_0 = 10$ and $\alpha_1 = 1$.

The routing model evaluation is based on 15 instances. Callbacks were used to terminate an optimization if the optimality gap was within 5% and the runtime had exceeded 3 hours.

Trust Inputs. Each network device was assigned a mass based on role. Maximum mass for web, application, content, database, and authentication devices is $1, 2, 3, 3, 4$ respectively. Mass is drawn uniformly from the interval $[\frac{m}{2}, ..., m]$. All event nodes are lifted to a height

Table 3 Optimization and functional assessments for routing model solutions when varying the number of wildcards in the rule header. The 15 instances were each evaluated with all three settings.

	0 wildcards			$\leq$ 1 wildcards			$\leq$ 2 wildcards		
Optimization	μ	min	max	μ	min	max	μ	min	max
Solve Time (s)	51	48	54	8.1K	910	12.6K	11.3K	10.8K	13.2K
Optimality Gap (%)	0	0	0	0	0	0	2	2	3
Millions of Variables	6.08	6.08	6.09	7.65	7.65	7.66	7.71	7.70	7.71
Millions of Constraints	1.75	1.73	1.76	2.33	2.30	2.35	2.90	2.87	2.93
Functional	μ	min	max	μ	min	max	μ	min	max
Norm Path Length	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.3
Norm # of Rules	1.0	1.0	1.0	.47	.46	.49	.51	.41	.60
# of Wildcard Rules	0	0	0	606	590	627	577	460	800
Total Rules	1340	1310	1360	627	612	647	677	546	803

of 8 (maximum height of 10). A height of 10 would reflect a node certainly compromised and which should be islanded outright. In all instances, the elasticity constants for strings were set to 1 to simplify the analysis of the outputs. The maximum length for any connection was set to 8 to allow nodes to rest on the ground if appropriate defensive actions are taken.

Trust Objective Parameters. In the trust model objective, we set $\beta_0 = 10$, the penalty associated to the trust loss (height) of a node, $\beta_1 = 1$ for simplicity, and each firewall rule added to an SDN device induced a penalty cost (γ_f) based on the protocol being blocked. Specifically, for `HTTP` and `RPC` firewalls, the penalty was set to 1. For `SQL` traffic, the penalty was 2. Finally, for `LDAP`, the penalty was 3. Adding firewalls that filter packets using critical protocols would be more likely to impact overall function.

Trust Optimization. The generated network configurations were given to the trust model for analysis. Only the solutions from the one wildcard and two wildcard methodologies were considered. That is, $I = \emptyset$ if no wildcards are used. The LNS strategy presented in Algorithm 1 was used with a time limit of five minutes.

6 Results

All evaluations were conducted on a Linux machine equipped with an Intel Xeon CPU E5-2620 2.00 GHz and 64GB of RAM. The MIP solver used was Gurobi Optimizer 9.1.

Routing. Here we ask if DuelMIPs's routing stage: 1) produces solutions are output quickly, 2) generates good configurations, and 3) produces rulesets that are lighter or less complex than fully-specific rulesets. The routing model assessment is in Table 3.

Increasing wildcard usage significantly affects the model size and solving time. Yet, even with up to 2 wildcards, this remains acceptable for use in network planning.

With no wildcards, all rules used the shortest path, and all restricted flows were blocked at the first available SDN device. When wildcards are allowed, the number of rules decreases. Interestingly, adopting 2 wildcards per rule does not deliver any further rule set compression In some instances, we even observed an increase in the total number of rules when compared to 1 wildcard. The ruleset size increases were due to Equation 15 (fully specified rules had to be added alongside some 2 wildcard rules to prevent its violation), but were outweighed by the improvements to the usability of the rulesets. Indeed, recall that ruleset size is not the sole indicator of usability. A larger ruleset comprised of simple-to-understand two wildcard rules could be considered more usable than a smaller ruleset that contains mainly one wildcard rules.

A Python implementation of the state machine from [5] is used to detect conflicts. DuelMIPs includes the constraint that each required flow matches at most one wildcard rule. Two wildcard rules are correlated due to Equation 15. Thus, DuelMIPs can output non-wildcard rules first, followed by wildcard rules without any risk of correlations. DuelMIPs may exhibit correlation and generalization, it should never exhibit Irrelevance, Shadowing, or Redundancy. This was confirmed in all experiments. The 1 wildcard instances displayed an average of 21 generalizations and 10 correlations and 2 wildcard instances displayed an average of 23 generalizations and 26 correlations. This slight increase is due to more general rules that require (potentially multiple) specific rules to handle exceptions.

Trust. The Trust model takes a routing solution, a set of incidental flows, and an event node referred to as a potentially compromised host. It must produce recommendations for reducing collateral damage. The Trust Model runs in at most five minutes which is appropriate. In many instances, the LNS uses several hundred iterations to get those results. Table 4 shows results, averaging over the 54 potential security events, for each of the 15 routing instances that use up to 1 or 2 wildcards, respectively.

The optimizer drastically affects the height of nodes. For the routing solutions using at most 1 wildcard, we see most hosts shift from High to Medium or Low to vLow. The result is more dramatic with 2 wildcards with hosts shifting from High to vLow. This is achieved by blocking only 1% of incidental flows in the 1 wildcard case, and, on average, 3% of the incidental flows in the 2 wildcard case. No required flows are blocked.
Potential Attacks The trust model recommendations reduce an adversary's ability to pivot in the network. In Table 5, we consider the trust model's output when a web server is the event node. In both wildcard settings, the incidental flows that DuelMIPs decides to firewall meaningfully reduce an attacker's potential attack paths (see Section 4 for the metric definition). Here the maximum path length is five nodes, which accounts for up to 5 pivots (most attacks use only a handful [7]) The two wildcard setting allows an attacker more freedom to pivot in the network. It is important that the trust model's recommendations improve the security metrics more. This is confirmed. In either setting, DuelMIPs's choices are superior to removing random edges. This indicates that the trust model appropriately considers the network structure when choosing which flows to block.

7 Conclusion

This work presents DuelMIPs, a two stage optimization approach. The first model, which focuses on routing, is an IP that creates SDN rules for a moderate size data center network in at most a few hours. The Routing Model outputs are functional and usable. By using wildcards intelligently, the routing model is able to compress rule sets at the cost of allowing some extraneous flows to be routed through the network.

The second model is a MIP that focuses on Trust. It is run in response to an identified compromise and considers residual trust loss due to the extraneous flows created by the Routing Model. It quickly creates recommendations that prevent an attacker from spreading to other similar nodes. To retain tractability it utilizes LNS.

This work focused on the complex task of balancing SDN rulesets for functionality, usability, and trust. The Routing and Trust models are coupled through the set of extraneous flows. As mentioned in Section 3.2, other notions of similarity can be used to model other

Table 4 The average number of hosts within each trust reduction classification for each 1 and 2 wildcard routing configuration given to the trust model. Note that each cell is averaged over all 54 possible singleton trust reduction events. For each level in $\mathsf{Level}_\triangle = \mathsf{Level}_F - \mathsf{Level}_\aleph$.

	1 wildcard				2 wildcard			
Instance	$\mathsf{High}_\triangle$	$\mathsf{Med}_\triangle$	$\mathsf{Low}_\triangle$	$\mathsf{vLow}_\triangle$	$\mathsf{High}_\triangle$	$\mathsf{Med}_\triangle$	$\mathsf{Low}_\triangle$	$\mathsf{vLow}_\triangle$
1	-8	6	-1	3	-14	-21	-2	37
2	-8	7	-5	5	-23	-17	9	30
3	-8	7	-3	4	-44	5	9	30
4	-8	7	-2	3	-20	-19	0	38
5	-8	7	-2	3	-27	-1	-3	31
6	-7	6	-1	1	-19	-17	0	36
7	-8	7	-5	5	-27	-13	3	36
8	-9	9	-15	15	-13	-20	-3	35
9	-7	7	-3	3	-36	21	9	6
10	-7	7	-7	7	-14	-23	-1	38
11	-8	8	-6	6	-11	-25	3	33
12	-8	9	-3	2	-23	-10	-3	30
13	-8	6	-1	3	-28	-12	5	35
14	-8	7	-1	2	-20	-24	10	35
15	-8	7	-4	5	-28	-3	-3	34

Table 5 Improvement of subgraph size and residual min-cut for DUELMIPS and random recommendations. Attack source is web server and target is database or authentication server.

	$\leq$ 1 wildcards				$\leq$ 2 wildcards			
Metric	μ	σ^2	min	max	μ	σ^2	min	max
$\mathsf{Gsize}(G_{\mathsf{m},T})$	0.95	< 0.01	0.84	1.0	0.79	0.01	0.56	1.0
$\mathsf{Gsize}(G_{\mathsf{m},R})$	0.97	< 0.01	0.94	0.99	0.97	0.01	0.88	1.0
$\mathsf{resMin}(G_{\mathsf{m},T})$	0.95	< 0.01	0.86	1.0	0.31	0.02	0.15	0.69
$\mathsf{resMin}(G_{\mathsf{m},R})$	0.99	< 0.01	0.93	1.0	0.95	< 0.01	0.91	0.99

network security best practices. An important piece of future work is understanding the impact of multiple event nodes on the Trust Model solve time. We see two potential applications: 1) supporting a response to an attacker that has compromised multiple nodes and 2) allowing the Trust Model to be useful in the Planning scenario as well. One could try to preemptively deploy firewall rules that are likely to prevent collateral damage over a variety of scenarios [13].

References

1 Alfarez Abdul-Rahman and Stephen Hailes. A distributed trust model. In *Proceedings of the 1997 workshop on new security paradigms*, pages 48–60, 1998.

2 Sugam Agarwal, Murali Kodialam, and TV Lakshman. Traffic engineering in software defined networks. In *2013 Proceedings IEEE INFOCOM*, pages 2211–2219. IEEE, 2013.

3 Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. A scalable, commodity data center network architecture. In *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*, SIGCOMM '08, pages 63–74, New York, NY, USA, 2008. ACM. doi:10.1145/1402958.1402967.

4 Saeed Al-Haj and Ehab Al-Shaer. Measuring firewall security. In *2011 4th Symposium on Configuration Analytics and Automation (SAFECONFIG)*, pages 1–4. IEEE, 2011.

5 Ehab S Al-Shaer and Hazem H Hamed. Firewall policy advisor for anomaly discovery and rule editing. In *International Symposium on Integrated Network Management*, pages 17–30. Springer, 2003.

6 Rashid Amin, Nadir Shah, Babar Shah, and Omar Alfandi. Auto-configuration of acl policy in case of topology change in hybrid sdn. *IEEE Access*, 4:9437–9450, 2016.

7 Giovanni Apruzzese, Fabio Pierazzi, Michele Colajanni, and Mirco Marchetti. Detection and Threat Prioritization of Pivoting Attacks in Large Networks. *IEEE Transactions on Emerging Topics in Computing*, 8(2):404–415, April 2020. `doi:10.1109/TETC.2017.2764885`.

8 Randall J Boyle and Raymond R Panko. *Corporate computer security.* Pearson, 2015.

9 Jin-Hee Cho, Kevin Chan, and Sibel Adali. A survey on trust modeling. *ACM Computing Surveys (CSUR)*, 48(2):1–40, 2015.

10 Timothy Curry, Devon Callahan, Benjamin Fuller, and Laurent Michel. DOCSDN: Dynamic and optimal configuration of software-defined networks. In *Australasian Conference on Information Security and Privacy*, pages 456–474. Springer, 2019.

11 George B. Dantzig and Philip Wolfe. Decomposition principle for linear programs. *Oper. Res.*, 8(1):101–111, February 1960. `doi:10.1287/opre.8.1.101`.

12 Rup Kumar Deka, Kausthav Pratim Kalita, Dhruba K Bhattacharya, and Jugal K Kalita. Network defense: Approaches, methods and techniques. *Journal of Network and Computer Applications*, 57:71–84, 2015.

13 Ron S Dembo. Scenario optimization. *Annals of Operations Research*, 30(1):63–80, 1991.

14 Diego Gambetta et al. Can we trust trust. *Trust: Making and breaking cooperative relations*, 13:213–237, 2000.

15 Ramanthan Guha, Ravi Kumar, Prabhakar Raghavan, and Andrew Tomkins. Propagation of trust and distrust. In *Proceedings of the 13th international conference on World Wide Web*, pages 403–412, 2004.

16 Guibing Guo, Jie Zhang, and Neil Yorke-Smith. Leveraging multiviews of trust and similarity to enhance clustering-based recommender systems. *Knowledge-Based Systems*, 74:14–27, 2015.

17 Jun He and Wei Song. Achieving near-optimal traffic engineering in hybrid software defined networks. In *2015 IFIP Networking Conference (IFIP Networking)*, pages 1–9. IEEE, 2015.

18 Lance J Hoffman, Kim Lawson-Jenkins, and Jeremy Blum. Trust beyond security: an expanded trust model. *Communications of the ACM*, 49(7):94–101, 2006.

19 MHR H.R. Khouzani, Zhengliang Liu, and Pasquale Malacaria. Scalable min-max multi-objective cyber-security optimisation over probabilistic attack graphs. *European Journal of Operational Research*, 278(3):894–903, 2019. `doi:10.1016/j.ejor.2019.04.035`.

20 Ansam Khraisat, Iqbal Gondal, Peter Vamplew, and Joarder Kamruzzaman. Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, 2(1):1–22, 2019.

21 Ahmed Khurshid, Xuan Zou, Wenxuan Zhou, Matthew Caesar, and P Brighten Godfrey. Veriflow: Verifying network-wide invariants in real time. In *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, pages 15–27, 2013.

22 Man Yue Mo. One day short of a full chain: Real world exploit chains explained, March 2021. URL: `https://github.blog/2021-03-24-real-world-exploit-chains-explained/`.

23 John Moy. Rfc2328: Ospf version 2, 1998.

24 Xuan-Nam Nguyen, Damien Saucez, Chadi Barakat, and Thierry Turletti. Officer: A general optimization framework for openflow rule allocation and endpoint policy enforcement. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 478–486. IEEE, 2015.

25 Ahmed Patel, Qais Qassim, and Christopher Wills. A survey of intrusion detection and prevention systems. *Information Management & Computer Security*, 2010.

26 Jon Postel. Internet protocol—darpa internet program protocol specification, rfc 791, 1981.

27 Yakov Rekhter, Tony Li, Susan Hares, et al. A border gateway protocol 4 (bgp-4), 1994.

28 Myriana Rifai, Nicolas Huin, Christelle Caillouet, Frédéric Giroire, D Lopez-Pacheco, Joanna Moulierac, and Guillaume Urvoy-Keller. Too many sdn rules? compress them with minnie. In *2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–7. IEEE, 2015.

29 Karen Scarfone, Wayne Jansen, Miles Tracy, et al. Guide to general server security. *NIST Special Publication*, 800(123), 2008.

30 P. Shaw. Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. In *Proceedings of Fourth International Conference on the Principles and Practice of Constraint Programming (CP'98)*, pages 417–431. Springer Verlag, October 1998.

31 Yan Lindsay Sun, Wei Yu, Zhu Han, and KJ Ray Liu. Information theoretic framework of trust modeling and evaluation for ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 24(2):305–317, 2006.

32 Stefano Vissicchio, Laurent Vanbever, Luca Cittadini, Geoffrey G Xie, and Olivier Bonaventure. Safe update of hybrid sdn networks. *IEEE/ACM Transactions on Networking*, 25(3):1649–1662, 2017.

33 Artem Voronkov, Leonardo A Martucci, and Stefan Lindskog. Measuring the usability of firewall rule sets. *IEEE Access*, 8:27106–27121, 2020.

34 John Wack, Ken Cutler, and Jamie Pole. Guidelines on firewalls and firewall policy. Technical report, BOOZ-ALLEN AND HAMILTON INC MCLEAN VA, 2002.

35 Lei Wang, Qing Li, Yong Jiang, and Jianping Wu. Towards mitigating link flooding attack via incremental sdn deployment. In *2016 IEEE Symposium on Computers and Communication (ISCC)*, pages 397–402. IEEE, 2016.

36 Wen Wang, Wenbo He, and Jinshu Su. Enhancing the effectiveness of traffic engineering in hybrid sdn. In *2017 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2017.

37 Yonghong Wang and Munindar P Singh. Formal trust model for multiagent systems. In *IJCAI*, volume 7, pages 1551–1556, 2007.

38 Tina Wong. On the usability of firewall configuration. In *Symposium on usable privacy and security*, 2008.

39 Avishai Wool. A quantitative study of firewall configuration errors. *Computer*, 37(6):62–67, 2004.

40 Cai-Nicolas Ziegler and Georg Lausen. Analyzing correlation between trust and user similarity in online communities. In *International Conference on Trust Management*, pages 251–265. Springer, 2004.