Heimdall: Towards Risk-Aware Network Management Outsourcing

Yuejie Wang
Peking University
yuejiewang@pku.edu.cn

Qiutong Men
New York University
qiutong.m@nyu.edu

Yongting Chen
New York University Shanghai
yongting@nyu.edu

Jiajin Liu
New York University Shanghai
jiajinliu@nyu.edu

Gengyu Chen
Carnegie Mellon University
gengyuc@andrew.cmu.edu

Ying Zhang
Meta
zhangying@meta.com

Guyue Liu
Peking University
guyue@pku.edu.cn

Vyas Sekar
Carnegie Mellon University
vsekar@andrew.cmu.edu

***Abstract*—Enterprises are increasingly outsourcing network management (e.g., troubleshooting routing issues) to reduce cost and improve efficiency, either by hiring third-party contractors or by outsourcing to third-party vendors. Unfortunately, recent events have shown that this outsourcing model has become a new source of network incidents in customer networks. In this work, we argue that a *risk-aware* outsourcing approach is needed that enables customers to measure and assess risk transparently and make informed decisions to minimize harm. We first concretely define the notion of risk in the context of outsourced network management and then present an end-to-end framework, called *Heimdall*, which enables enterprises to *assess*, *monitor*, and *respond to* risk. *Heimdall* automatically builds a dependency graph to accurately assess the risk of an outsourced task, and uses a fine-grained reference monitor to monitor and mitigate potential risks during operation. Our expert validation results show that *Heimdall* effectively controls risk for outsourced network operations, resolving 92% of practical issues at the minimal risk level while incurring only a marginal timing overhead of approximately 7%.**

I. Introduction

Enterprises are increasingly outsourcing network management to third-parties, such as contractors and managed service providers (MSPs) (e.g., [15], [19], [17], [16], [18]). The third-party technician (referred to as technician for the rest of our paper) monitors a client's network and provides management functions such as configuring access control rules, troubleshooting routing issues, and monitoring bandwidth usage. The market for network management outsourcing is rapidly expanding (projected to grow to USD 309.4 billion by 2025 [20]) as they reduce operational expenses compared to in-house IT teams.

Although outsourcing management is economically attractive, the current operating model poses significant security risks. In many cases, the technician is granted admin privileges. This power fundamentally violates the zero-trust poilcy [23], posing a risk that can lead to harmful configuration changes that can result in critical security incidents [2], [14], [29] and large-scale network outages [10], [21], [4], [7], [1], [28].

To enforce zero-trust policy for customers without compromising on the economic advantages, what we ideally need is *risk-aware* network management outsourcing, akin to other aspects such as investment risk, supply-chain risk, and liability risk [67]. That is, enterprise customers should be informed and take steps to mitigate associated risks in network management outsourcing workflows before harm occurs. For example, during resolving a routing issue, if the third-party technician issues potentially dangerous operations, we should proactively assess the risk, notify the enterprise, and also provide the necessary tools to assess the situation, and decide whether this should be allowed (with audits) or blocked.

Drawing a parallel to well-known risk management frameworks [67], we argue that outsourced network management needs to include three essential components: (i) *Risk Definition* which clearly defines what constitutes a risk in network management; (ii) *Risk Assessment* to measure and evaluate the risk based on the definition; and (iii) *Risk Monitoring and Response* to continuously monitor the situation and execute a response plan for any identified risk.

To the best of our knowledge, the network management outsourcing problem has been largely ignored in the research literature, and few if any efforts have systematically tried to define, assess, and monitor risks in outsourced network management. [1]

Our goal in this work is to build a practical risk-management framework for outsourced network management workflows. To this end, we design *Heimdall*, a framework to enable customers to make risk-based decisions for outsourced network management. We start by reporting on a survey we conducted with a leading hyperscaler network to understand the types and frequency of outsourced tasks within their operations (§II-B). Building on this valuable information, we focus on risks involving the modification of router configurations. Notably, this category represents a substantial portion of their outsourcing activities and exhibits a high frequency,

Network and Distributed System Security (NDSS) Symposium 2025
24-28 February 2025, San Diego, CA, USA
ISBN 979-8-9894372-8-3
https://dx.doi.org/10.14722/ndss.2025.230043
www.ndss-symposium.org

[1] Our workshop paper [62] highlighted the security risks associated with MSPs but failed to provide a concrete solution. In particular, we did not provide a concrete basis to quantify risk or a practical implementation for risk management.

with potentially tens of thousands of occurrences each day (Table I). We believe that the design principles and foundations underlying *Heimdall* can be applied to other types of tasks (e.g., modifying hardware or software) as well, but defer this for future work.

We address three key challenges in designing *Heimdall*:

1. *Defining risk quantitatively based on network impact on assets:* We introduce a new risk definition for evaluating a technician's operations, specifically by assessing its *network impact* on an organization's assets. This approach draws on the well-established understanding that *assets* are the primary concerns of an enterprise, as they provide the resources necessary for the business to operate and achieve its goals [68], [67], [55], [48]. The only additional information required is the *value and location of each asset* within an organization, which often has been tracked by widely-used asset management software [3], [12], [13]. Compared to prior approaches that rely on the user to provide risk information or historical data [31], [76], [39], [73], [65], our approach significantly simplifies the information and expertise required to assess risk.

2. *Assessing risk using an accurate dependency graph:* To assess risk, we present a novel approach where we formulate the network as a *risk dependency graph* of fine-grained network configuration blocks and assets. This graph captures how modifying each configuration block affects a group of assets, through a chain of dependency edges. Building accurate dependencies is very challenging as network components interact in complex ways, and these interactions are hardly being captured by prior static analysis-based approaches [35], [36]. Our key idea is to track the relations from *configuration blocks* to *forwarding tables* and then analyze how forwarding tables control different assets. Reasoning the generation of forwarding tables enables a more precise dependency model and can be applied across various configuration types.

3. *Monitoring risk via a fine-grained reference monitor:* Instead of using simulation or emulation tools [63], [26], we allow the technician to directly operate on customer networks and use a centralized *reference monitor* [69] to mediate all operations. Before granting any access or executing any operations on customer devices, the reference monitor examines the risk assessment and takes appropriate actions to address risks based on the user-specified policy. Compared to existing router built-in privilege control mechanisms (e.g., Cisco [25]), our centralized reference monitor provides fine-grained control (per operation), and flexibly supports various user-defined risk response policies.

We build an end-to-end system, which to the best of our knowledge is the first prototype designed to mitigate the risks of outsourced network management. To assess the risk control capabilities and overhead of *Heimdall* in practice, we invited 10 network experts from our university's network vendor to participate in the evaluation and resolve 33 practical tickets. This evaluation generated 99 ticket resolution records during 41 hours operation of the system. Our results show that experts using *Heimdall* can resolve 92% issues without exceeding the risk of the root cause block (namely the lowest intended risk level), while incurring a fairly low procedural overhead of 7.4% on average. We also evaluate *Heimdall* using a ticket database including ∼3000 synthetic tickets generated across eight diverse networks. Our results show that *Heimdall* can reduce 80% risk in a large network and 50% risk in a small network for 50% tickets.

Outsourced Tasks	Frequency (/day)	Outsource Reason	Access Type
Troubleshooting	10^4	*Costs*	Virtual
Network turn-up and upgrade	10^3	*Costs*	Physical, Virtual
Hardware repair and replacement	10^3	*Geo-proximity*	Physical
Fiber leasing and maintenance	10^2	*Geo-proximity*	Physical, Virtual

TABLE I: Network outsourcing scenarios in practice based on a hyperscaler network survey

This paper makes the following contributions:

- We are the first to quantitatively define risk for outsourced network operations, providing a guideline for the field (§III)
- We design a novel risk dependency graph that accurately assesses the risk of modifying router configurations. (§IV)
- We develop an end-to-end risk-aware outsourcing prototype and demonstrate its effectiveness in mitigating risk by using it to resolve practical network issues. (§V, §VI)

Ethics: This work does not raise any ethical issues. All network configurations are anonymized. Evaluation results are based on GNS3. No production network is affected.

II. Background and Motivation

In this section, we first provide a background of outsourced network tasks. Then, we motivate risk-aware network management and propose our threat model in the given background.

A. Background on Outsourcing

Outsourcing network management is a common solution for enterprises to reduce cost and improve efficiency. Two typical types of outsourcing cases include: 1) *Hiring third-party contractors*: Large companies hire third-party contractors to do specific network tasks (details in §II-B) to utilize their expertise and save costs; 2) *Outsourcing to third-party vendors*: To reduce cost, small to medium-sized companies outsource their network management to third-party vendors (e.g., MSP) instead of hiring additional employees. For example, the campus network we consulted comprises approximately 400 network devices. Of these, 75%—primarily access switches—are outsourced to third-party vendors for configuration tasks such as VLAN ports and DHCP snooping.

Most outsourced tasks consist of a three-step workflow [8], [11], [22]: (1) Customer creates a new ticket which typically includes information such as the request description, group, impact, urgency, priority, and the assets affected; (2) The ticket is sent to outsourced technician and is then categorized based on its type and priority; and (3) To resolve the ticket, the technician provides services by remotely accessing one or more customers' network devices with *admin* access, which enables the technician to issue both normal and *privileged* commands on these customer devices. After finishing the task, the technician documents their changes and closes the ticket.

B. Survey of Outsourcing Best Practice with a Hyperscaler Network

To gather insights on industry practices for network operations outsourcing, we partnered with a leading hyperscaler network. This network encompasses numerous data centers and a vast backbone network that spans the globe. It covers operations at multiple layers, including IP, optical, and fiber.

During our collaboration, with the data provided by an industry expert, we categorized the types of operations that are outsourced and examined the reasons behind this outsourcing practice. We also analyzed the frequency at which these outsourced operations occur. It is worth mentioning that our study represents a novel effort in exploring this particular aspect of network operations outsourcing. The survey identified four primary types of outsourced network operation tasks summarized in Table I: troubleshooting (*tens of thousands per day*), network turn-up and upgrade (*thousands per day*), hardware repair and replacement (*thousands per day*), and fiber leasing and maintenance (*hundreds per day*). Consider that although this survey is based on historical data from a hyperscaler network, the statistics are likely to be similar for small- to medium-sized enterprises. Given the varying levels of expertise among network administrators, the need for outsourcing network management tasks is expected to be even greater for these smaller organizations.

Network troubleshooting: The largest category is troubleshooting network performance issues regularly. The network has tens of thousands of troubleshooting tickets per day [81]. Most tickets can be solved by well-documented procedures, which are written in Method of Procedures (MOPs) and handed to third-party technician to execute. Only those tricky cases are brought to the full-time employees to leverage their higher privilege and their deeper knowledge to our own network. This largest category reaches a frequency of *tens of thousands per day*.

Regular network turnup and upgrade tasks: To avoid vendor lockin, ensure reliability, and leverage a rich set of device functions, the network uses heterogeneous vendor devices. The network engages a large number of external vendor employees to perform the day-to-day capacity turnup and upgrade tasks to meet their fast-growing demands. The goal is to save cost and leverage vendor employees' domain knowledge and this category occurs *thousands per day*.

Device hardware repair and replacement: This network spans across the globe, encompassing hundreds of Point-of-Presence (POPs) that are situated in rented shared facilities. Given the extensive number of locations, it would be impractical and cost-inefficient to allocate full-time employees at each site. Therefore, it is a common industry practice for hyperscalers to hire local contractors to perform physical network operations such as hardware swapping and device installation. These operations occur *thousands of tasks per day*.

Fiber leasing and maintenance: Fiber infrastructure, a physical network component, is typically deployed and owned by a select few companies specializing in this field, such as Zayo Groups and Lumen Technologies. Medium to large internet companies often opt to lease or purchase long-haul fibers from these third-party vendors. These fiber vendors offer tailored support to their customers, often requiring access to their clients' internal networks. These operational tasks are carried out regularly, with a frequency of *hundreds per day*.

We observe that the first two types of outsourced tasks are driven by domain expertise and costs, while the latter two types are caused by geographical proximity. As a starting point, our focus in this work is configuration troubleshooting which accounts for the majority of outsourcing cases and only requires virtual access. We leave other types of outsourcing cases for future work.

C. A Need for Risk-Aware Outsourcing

Although outsourcing is economically attractive, there are significant security risks. When the third-party technician is granted with *admin privileges*, they can make *any* changes on managed network devices, including potentially harmful or malicious changes, without the customer's knowledge. Some harmful changes may be latent for a long time as shown by recent incidents [2], [24], [14], [29]. In-house network configuration changes that lack the awareness of the impacted surface also lead to major outages in both man-made [10], [21], [4], [7], [1], [28] and automated software made [9], [30] changes. For example, an technician misordered just one BGP filter rule before others, causing CloudFlare, a leading CDN company, to interrupt 50% of requests for about an hour [28].

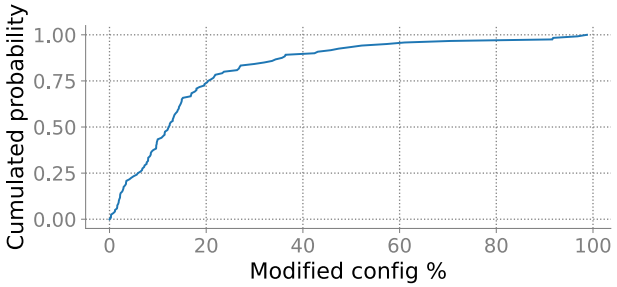

Fig. 1: Most operations only involve a small portion of configuration lines

Opportunity: To further quantify changes described in §II-B, we analyze the amount of configuration modifications involved in these operations. Figure 1 presents the cumulative distribution function (CDF) of configuration changes ratio across backbone devices of the surveyed hyperscaler network over one month. It indicates that most operations (75%) impact a small portion ($\leq$ 20%) of the overall configuration. Prior work also indicates the median change includes only three devices in 75% of the networks operated by a large online service provider [46]. By analyzing the outsourced tasks, we gain insights that the scale and scope of network changes performed by contractors are relatively *small*. This observation helps us develop a framework to quantify their potential risks below.

D. Threat Model

There are three types of principals who may access the network resources: internal or external users, network administrators (denoted by $\mathcal{A}$) and third-party technicians (denoted by $\mathcal{B}$). In *Heimdall*, we assume that users do not participate in network management or attempt to modify network configurations. Administrator $\mathcal{A}$ is always trustworthy, but may not have enough expertise to solve a network task. $\mathcal{A}$ is responsible for the privilege control of technician $\mathcal{B}$, which means $\mathcal{A}$ has complete control of which actions $\mathcal{B}$ is allowed

to take. $\mathcal{B}$ is expected to have enough expertise to solve the task, but also could be the adversary that performs malicious operations. For the rest of the paper, we focus on tasks related to router configuration troubleshooting (such as OSPF, BGP, and filter configurations) due to their high frequency (Table I) and significance [72], [44], [55], [35], [36]. Our primary focus is ensuring the network's correctness, specifically maintaining the reachability between network devices. For other types of tasks that may require modifying hardware and other types of networks such as SDN, and evaluation of other properties such as QoS, we plan to address them in the future.

E. Existing Approaches and Limitations

Existing approaches in related areas such as storage outsourcing [49], [74] or computation outsourcing [47], [78], [61] rely on cryptographic techniques to achieve data privacy, which is not applicable to our scenario of network management outsourcing, since we need the technician to be able to understand the semantics of the portion of network exposed to them. Other studies on the risk of IT outsourcing [43] focus on more general questions such as level of expertise and cost. For privilege control in network management outsourcing, we discuss two strawman solutions and their limitations next, and make a case for a systematic *risk-aware* framework.

Strawman 1: Router built-in privilege management.: Most routers have built-in mechanisms for controlling privileges. For example, Cisco IOS devices allow users to customize a specific set of commands for different privilege levels [5], [25]. Indeed, admin privileges are not always necessary as most network tasks only need small changes on a few devices [80], [66], [46], [70]. By determining the required privileges for a ticket, we can limit the technician's access and minimize potential harm. However, determining the specific commands required for a task and estimating the risk associated with them can be challenging, even for experienced network operators. As a result, most devices typically are configured with an *"all-or-nothing"* approach by either giving the admin privilege or read-only privilege to the user.

Strawman 2: Use a verification tool to validate configuration changes.: Another strawman approach is to leverage verification tools [44], [34], [54] to check whether the technician's changes violate some policies, which may indicate a risk associated with the proposed changes. However, for this approach to work well, a *formal, complete, and correct* specification needs to be provided. In practice, specifications often do not exist or cannot cover all intended policies [55], [38]. Recent tools [38] can automatically synthesize a collection of policies from configurations but still cannot handle any implicit or unforeseen policies. Any unspecified policy poses a risk.

III. Design Overview

In this section, we begin by concretely defining the notion of risk we propose to use. Then, we present the overview of *Heimdall*, a framework that enables enterprises to assess, monitor, and respond to risk. Finally, we discuss three fundamental challenges to realizing this in practice.

A. Quantifying Risk

To get a precise notion of risk, we choose a well-established *quantitative* risk model from the literature [57], [64], which defines risk in terms of (a) *probability of occurrence* and (b) *consequence* of adverse effects, i.e., *Risk = Probability of Occurrence * Consequence*. For a complex system, the risk can be defined based on a set of events E [50] [51], [52], [53], [75], i.e., $Risk(E) = \sum_{e \in E} \mathbb{P}(e) * C(e)$, where an event e's occurrence probability is $\mathbb{P}(e)$ and its consequence is $C(e)$.

To apply this risk definition to our context, we can view ticket resolution as a series of events modifying network configurations. Events that have adverse effects on *enterprise's assets* are the main *threat* to the enterprise [68], [67], [55], [48]. Our definition draws on the established understanding that *assets* are the primary concerns of an enterprise, as they are the basis for business operations and goals [68], [67], [55], [48]. Here we focus on hardware, software, and data assets connected by the network, such as laptops, applications, and digital documents.

Formally, let $Assets$ be all the assets in the customer network, and each asset $S \in Assets$ has a value $S.value$. We use $\mathbb{P}(S|ticket)$ to denote the probability that S can be affected by an event during solving a $ticket$. Then, the risk of the ticket is defined as

$$Risk(ticket) = \sum_{S \in Assets} \mathbb{P}(S|ticket) * S.value \quad (1)$$

Prior work [31], [76], [39], [73], [65] relies on the user to directly provide risk information or historical ticket data to estimate risk. However, users often don't have enough domain knowledge or empirical data to determine risk accurately. Thus, our framework has to compute the risk automatically and the only information required from users is the *value and location of each asset* within an organization. The values of digital assets are typically available through canonical asset management tools [3], [12], [13]. The asset records can reflect criticality in terms of the purchase price, cost of ownership, maintenance and support, downtime impact, replacement cost, etc. The network owners can define a reasonable risk value for each asset based on this information and use it as input. This significantly reduces the expertise needed to quantitatively measure the risk accurately. With this asset-based risk definition, we next show how *Heimdall* manages the risk of outsourced network management tasks.

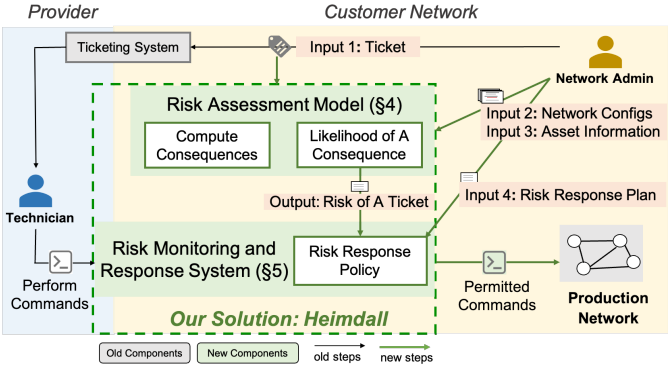

Fig. 2: *Heimdall* overview

B. The Heimdall Workflow

Risk management is a comprehensive process [67], [68] that requires an organization to (i) *assess* risk, and (ii) *monitor and respond* to risk over time. We envision an end-to-end framework called *Heimdall* which manages this entire process to enable customers to make risk-based decisions. As shown in Figure 2, it has two key components: *risk assessment model* and *risk monitoring and response system*.

Component A: Risk Assessment Model (§IV): Risk assessment entails two steps: compute the consequences and the likelihood of each consequence. For the first step, *Heimdall* takes enterprise *assets* information (e.g., location and value) and a snapshot of network *configurations* as inputs and automatically outputs a *risk dependency graph*. This graph captures how modifying each configuration block (defined in §IV-A) affects a group of assets, through a chain of dependency edges. Based on the risk dependency graph, the *consequence* of a configuration change can be determined by evaluating all its related assets.

Challenge 1: Building an accurate risk dependency graph (§IV-A): A naive solution to build dependencies is based on the physical topology. However, this approach could result in a large number of false positives and false negatives, linking assets to non-related configurations or missing critical relations. The inaccurate risk estimation leads to non-effective risk response policies which fail to protect valuable assets.

The next step is to compute the likelihood of each consequence of a specific ticket, which could be affected by the complexity of the ticket, the expertise of the operators and the tools they use (e.g., troubleshooting and monitoring tools).

Challenge 2: Effectively estimating the likelihood of a consequence (§IV-B): The likelihood of a consequence occurring during the ticket resolution may change, even for the same ticket. Various factors may affect its estimates such as the technician's proficiency and the tools they may use. For instance, different technicians may consider the most likely fix differently, thus impacting which configuration they may modify. Similarly, different tools may rank the most likely fix based on diverse algorithms.

Component B: Risk Monitoring and Response System (§V): After assessing the risk associated with a ticket, *Heimdall* continuously monitors and addresses risks per operation. Enterprises can customize their risk response plans to fit their specific needs. For instance, an enterprise may choose to proceed at an *acceptable* risk level and halt operations at an *intolerable* level. *Heimdall* provides an interface for users to create a high-level risk response plan and then combines it with the assessed risk to generate a user-specific *risk response policy* for each ticket.

To enforce a specific risk response policy throughout the ticket resolution process, *Heimdall* continuously assesses technician' operations and closely monitors any potential risks. The technician can perform operations remotely using existing interfaces, such as the command line or GUI. Based on the established risk response policy, only a *limited* set of configurations can be accessed by the operator, with the goal of controlling and minimizing risk. If the technician attempts to access restricted configurations, *Heimdall* will detect this and trigger appropriate actions, as outlined in the risk response plan, such as alerting the customer, etc.

Challenge 3: Correctly and efficiently enforcing response policies (§V-B): With the current routers' built-in mechanisms, it is hard to control which parts of a configuration can be accessed or not (see II-E), thus cannot enforce an effective risk response policy. To correctly control the accessibility of a configuration, a strawman solution could pull the configuration from the router and hide parts of it for the technician to view and modify. Then the risk of changes can be checked to ensure only safe changes are pushed back to the router. This approach allows for correct policy enforcement, but it prevents the technician from using existing interfaces (e.g., commands) and adds noticeable delays to ticket resolution.

IV. Risk Assessment Model

The first component of *Heimdall* is a risk assessment model quantifying the *risk* associated with a network ticket. We first propose assessing the *consequences* of ticket resolution using a risk dependency graph and show our approach to building it with an example network (§IV-A). Then, we discuss the factors affecting the *likelihood* of each consequence and how to compute it in practice (§IV-B).

A. Assessing Consequences

During the ticket resolution process, we assess the consequences of modifying a configuration based on its impact on assets. We focus on endhost assets as the majority of valuable assets (e.g., endhost applications, user data) are located on endhosts, and we discuss how to extend our model for assets in the middle of the network in §VIII. To precisely capture the impact on assets, we use a *configuration block* rather than the entire configuration file as the basic unit of analysis. Similar to prior work [35], [36], [55], a configuration block is a continuous sequence of lines used to configure a specific feature, such as an access-control list or a BGP session.

Example Network: To make our discussion concrete, we use an example network shown in Figure 3 to demonstrate the connections among network devices and configuration blocks. The network has four routers $R1$ to $R4$ connected by physical links. All routers are running the OSPF protocol to connect three internal hosts $h1$ to $h3$, with an asset on each host. $R4$ is also running a BGP protocol to connect to the Internet and is configured with access control rules.

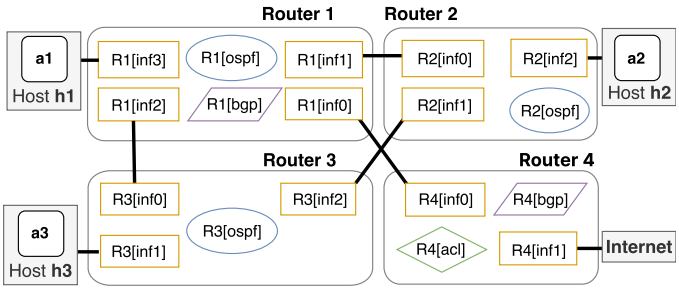

Fig. 3: An example ticket and network

Risk Dependency Graph: We use a *risk dependency graph* (RDG) to represent the relations between configuration blocks and assets. Each block and asset is represented as a node, and

each directed edge shows the flow of dependency from one node to another. Each asset in the network has an RDG with the asset's host node as the root. The RDG's leaves represent all hosts that can reach the root.

Figure 4 shows the RDG of h_3 shown in Figure 3. Edges can be of two types: *inter-block edges* and *block-asset edges*. Inter-block edges capture the relations between configuration blocks. These blocks could be located within a single router or across different routers. For example, an OSPF block ($R3[ospf]$) has edges pointing to each interface block ($R3[inf0]$, $R3[inf1]$, $R3[inf2]$) on the same router whose network is contained in the OSPF network. Block-asset edges show the relations between blocks and endhost assets. For example, an edge points from h_3 to $R3[inf1]$ because asset $a3$ is located on h_3 which is physically connected to the router $R3$ by interface $R3[inf1]$.

A practical solution to building an RDG needs to meet two key requirements: accuracy and scalability. It should be able to accurately infer the relationships and be applied to networks of a reasonable size ($O(100)routers$). To motivate our approach, let us first outline two strawman approaches to building dependency.

Strawman Approaches: The first potential approach is to statically analyze configuration files. For example, prior work [35], [36] uses a parser to analyze configurations and create referential links between blocks based on *tokens*, e.g., an IP address. This method can be directly used to create our inter-block edges. This parsing-based approach can quickly analyze a large number of configuration files, but it also produces many *false positives*, where unrelated blocks are identified as being related (§VII-C).

Another potential approach is to use mutation, by modifying configurations dynamically and evaluating the effect on reachability. For example, we can test whether removing a BGP block will alter the reachability of a specific asset. To ensure accuracy, the approach requires enumerating *all* possible combinations of blocks for every given asset. While this approach works for a few blocks, the computation complexity of the search is exponential, failing to scale as the number of blocks increases.

Our Approach: To achieve both accuracy and scalability, we have two key observations. First, building accurate relations between blocks and assets requires to analyze not only the configuration files but also how the *data plane* is produced by the configurations. The correlation between the configurations and the data plane is complicated, and the ability to precisely capture these relationships will determine the level of accuracy achieved. Second, although inferring the relationships between blocks and assets take time, these relationships tend to be relatively stable. This makes it possible to *pre-compute* these relations and search for specific relations when a ticket arrives. Given a pre-built dependency graph, identifying which assets are associated with a particular block can be cheap to support large networks.

We build relations by *propagating* information from configuration blocks to endhost assets by leveraging *forwarding tables* of the data plane. The key challenge is to track the entire process of forwarding table construction; from configuration blocks, to RIB construction, and finally FIB generation. While prior work (e.g., Batfish [44]) can simulate the process, we need to further *track* how information is propagated across different routers. Inaccurate tracking leads to false positive or false negative relations.

	Type	Description
1	Interface	An $InterfaceBlock$ is related to a route if this interface connects the destination host to the router.
2	Protocol	A $ProtocolBlock$ is related to a route processed by the same routing process.
3	Egress	An $InterfaceBlock$ and a $ProtocolBlock$ are related to the routes it sends to a neighboring router connected by the same interface and routing process.
4	Ingress	An $InterfaceBlock$ and a $ProtocolBlock$ are related to the routes receiving from a neighboring router on the same interface and routing process.
5	ACL	An $ACLBlock$ is related to all routes stored in the same router.

TABLE II: Rules to create dependencies.

As shown in Figure 5, our approach includes two main stages: (1) *configuration blocks→forwarding tables*: the first stage builds *inter-block* relations and associates different types of configuration blocks to the FIB entries; and (2) *hosts→forwarding tables*: the second stage associates hosts to the FIB entries, creating *block-asset* relations.

We focus on three common types of configuration blocks: i) $InterfaceBlock$ that controls a router interface; ii) $ProtocolBlock$ that controls a routing protocol, such as BGP and OSPF; iii) $ACLBlock$ that controls access control rules. These categories cover a majority of the router configuration [44]. For the remaining configuration lines (such as login and time setting), we combine them into a $ManageBlock$. The $ManageBlock$ does not impact endhost assets and is handled separately from other types of blocks (§VI).

The dependency between a block and a route is constructed and updated according to the five rules shown in Table II. We take the route (h_1, h_3) in Figure 5 as an example to explain how each rule is applied. In Stage 1, **Rule 1** is applied to add the interface that directly connects a host to a router, so $R1[inf3]$ is added to the dependency set of $route(dst = h_1)$ in router $R1$, and $R3[inf1]$ is added to the dependency set of $route(dst = h_3)$ in router $R3$. Then, the routes and their dependency sets are passed to the router RIB in RIB initialization phase. From this step, the routers enter the loop phase of updating the data plane until it converges. **Rule 2** is applied when a routing process learns a route from the router RIB. We add the $ProtocolBlock$ of this routing process to the existing dependency of the imported route, e.g., $R3[ospf]$. **Rule 3** is applied when a routing process broadcasts its routes to all its neighbors connected by this protocol and the dependency of an exported route is appended with the egress $InterfaceBlock$ and $ProtocolBlock$ connecting the neighbor, e.g., $R3[ospf]$ and $R3[inf0]$. Upon receiving a route from a neighbor, **Rule 4** adds the $ProtocolBlock$ of the connecting routing protocol and the $InterfaceBlock$ of the ingress interface to the dependency received with the route, e.g., $R1[inf2]$ and $R1[ospf]$, which then will be passed to the router RIB. After the data plane of each router converges,

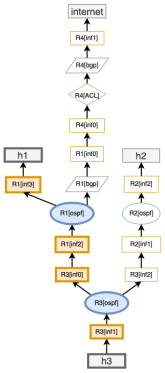

Fig. 4: Risk Dependency Graph of h_3

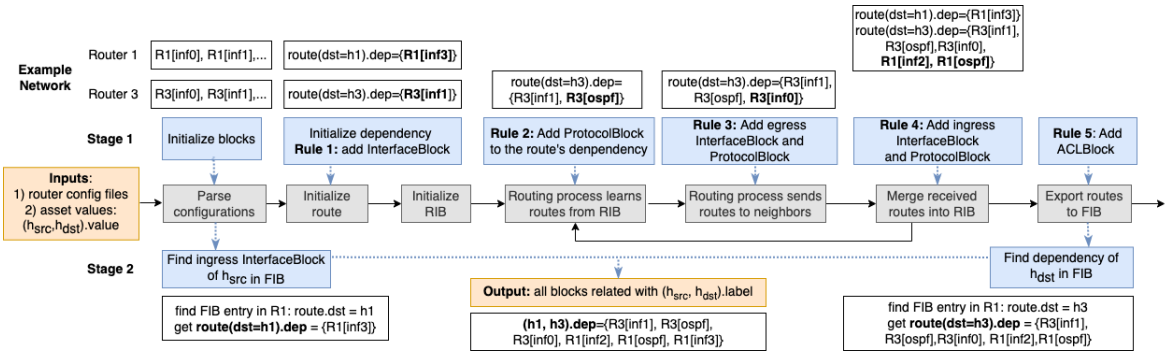

Fig. 5: We use connection (h_1, h_3) shown in Figure 3 as an example to show the process of building risk dependencies. $\rightarrow$ represents the existing logic of constructing FIBs. dashed arrows (- - $\rightarrow$) represent the new logic added by our approach.

the preferred route to each reachable destination is passed to the router FIB, together with the dependency we constructed.

The Stage 2 of our algorithm is to associate a host connection (h_{src}, h_{dst}) with the corresponding route by searching the two related entries of source and destination hosts, e.g., h_1 and h_3, in the FIB entries and extract the dependency attached to the route. As shown in Figure 3, the dependency of the complete route (h_{src}, h_{dst}) is the union of the two dependency sets, e.g., $(h_1, h_3).dep = R1.FIB(dst = h_1).dep \cup R1.FIB(dst = h_3).dep = \{R3[inf1],\ R3[ospf],\ R3[inf0],\ R1[inf2],\ R1[ospf],\ R1[inf3]\}$.

B. Assessing Likelihood of A Consequence

In this subsection, we first examine key factors that influence the *likelihood* of a consequence. Then, we present a model to compute the likelihood. By combining the likelihood of each consequence with the computed consequences, we can evaluate the risk of a particular ticket ($Equation$ 1).

Influencing Factors: Various factors can affect the *likelihood of a consequence* for a given ticket. For example, the complexity of the problem, the level of expertise of technicians, and the tools (such as troubleshooting and monitoring) that are used. We observed that these factors affect the likelihood of consequence in two fundamental ways: (i) by a group of blocks accessed by the technician before fixing the issue, which we call *AccessibleBlocks*, and (ii) by the root cause blocks that can be modified to resolve the ticket, which we call *RootCauseBlocks*. For example, an expert technician may be able to quickly identify the root cause of an issue and access only a few blocks before fixing the problem, while a novice technician may need to access many blocks before finding the root cause. Similarly, different ticket resolution modes (manual, or using automated tools) and different tools could provide different estimates for *AccessibleBlocks* and *RootCauseBlocks* for a ticket. Our goal is to provide a simple model to compute the likelihood of consequence that supports both modes and various existing tools.

Our Approach: Despite the variations in their implementation, we observed that the outputs of existing tools [56], [32], [42], [33] can be abstracted as *a preference order* of network components. A higher order indicates a higher likelihood of the component being the cause of the issue, compared to a lower rank. For example, NetMedic[56] uses historical data to rank the cause of a ticket, and 007[32] uses the path information to compute a weight for related network components and outputs the possible causes ranked by weight. This preference order abstraction also aligns with the manual examination process as a technician would typically focus on the most likely causes to minimize their efforts.

Based on this observation, our model abstracts *AccessibleBlocks* using a $pref()$ function, and $pref(b_1) > pref(b_2)$ indicates that block b_1 is granted earlier than block b_2. This decision can be made either by a tool or a human. For simplicity, we assume a strict preference order, i.e. $pref(b_1) = pref(b_2) \Rightarrow b_1 = b_2$, and we will discuss more details in the next section. Meanwhile, we estimate *RootCauseBlocks* using $\mathbb{P}(c|ticket)$ which is the probability that the root cause block is c for the $ticket$. This value can be provided by the tool, using history stats, or simply assuming it is uniform ($= \frac{1}{total\_blocks}$). We assume each fix only involves one block for now and this can be easily extended to multiple blocks.

With a preference order $pref(b)$ and $\mathbb{P}(c|ticket)$ as two inputs, we can compute the likelihood of a consequence based on conditional probability. Formally, given a $ticket$, we use $\mathbb{P}(S|ticket)$ to denote the probability that an asset S is affected and use $Cause$ to denote all possible root causes of the $ticket$. Then, $\mathbb{P}(S|ticket)$ can be computed as follows:

$$\begin{aligned}\mathbb{P}(S|ticket) &= \sum_{c \in Cause} \mathbb{P}(S|c, ticket) * \mathbb{P}(c|ticket) \\ &= \sum_{c \in Cause} \mathbb{P}(\exists b, pref(b) \geq pref(c) \\ &\qquad \wedge S \in b.assets)) * \mathbb{P}(c|ticket)\end{aligned} \tag{2}$$

Note that the $b.assets$ refers to assets that are related to the block b, which can be obtained by our assessment model (§IV-A). As one asset can be affected by several blocks, to prevent double-counting the same asset, we take the *union* of affected assets in our implementation and discuss other options in §VIII. By combining $Equation$ 1 and $Equation$ 2, we can compute the risk for a given ticket. Next, we discuss how to use the computed risk to create a risk response policy.

V. Risk Monitoring and Response

The second component of *Heimdall* focuses on risk monitoring and response. In this section, we first show how to create a *risk response policy* (§V-A) which outlines appropriate

actions to address each level of risk. Then, we discuss practical challenges to enforcing such a policy in real-time and finally present how *Heimdall* addresses these challenges (§V-B).

A. Risk Response Policy

After assessing risks to an enterprise's assets, the next step is to develop strategies to mitigate or manage them. Different enterprises could create customized risk response plans to fit their specific needs. A risk response plan [67] typically includes a set of guidelines that categorize risks into different levels based on predetermined thresholds. Each level has a specified action to respond to the risk. *Heimdall* provides customers an interface to specify such a high-level risk response plan, and then integrates the ticket's risk information to create a tailored *risk response policy* for each ticket.

Level	L1	L2	L3(Max)
Risk Threshold	30%	60%	100%
Action	None	Alert	Stop

Group	G1	G2	...
Blocks	{R1[inf0], R1[inf1]}	{R1[ospf], R3[ospf]}	{...}

Fig. 6: An example risk response policy

Figure 6 shows an example policy specification. The first part consists of risk thresholds and corresponding actions. The example has three risk levels and states that: $L1$: if the risk falls within 30% of the maximum risk, no action is required, $L2$: if the risk is assessed to be between 30% and 60%, the customer must be alerted and informed, and $L3$: if the risk is larger than 60% and falls in the last level, the operation must be immediately stopped to prevent further risks. If the risk threshold of the last level is not 100%, the default action from the last threshold to the maximum risk is $Stop$. The second part specifies the debugging strategy, i.e. the order of blocks (in groups) to grant access for resolving the ticket. In the example, blocks in the group $G1 = \{R1[inf0], R1[inf1]\}$ will first be considered when trying to resolve the ticket. If the ticket is not solved, the next group $G2$ is then considered, etc. The second part can be pre-computed (e.g., based on the debugger) or dynamically decided while resolving the ticket.

Iterative Access Granting: Dividing the blocks into multiple groups enables *Heimdall* to provide precise risk management. Access to the groups is granted to the provider in *iterations*. For each iteration, the provider is granted access to one group of blocks. If the ticket cannot be resolved in this iteration, the provider can request additional access. The approval of each group will depend on the risk level and associated action. For instance, if the risk level is low, the group may be automatically approved, while if the risk level is medium, manual approval by the customer may be required. If approved, the provider proceeds to the next iteration and this process continues until the ticket is resolved or stopped, e.g., if the risk level is considered to be too high to continue. While this iterative approach provides fine-grained risk control, it might result in a slower ticket resolution process, as the provider might go through multiple iterations to obtain all the necessary information. This is a fundamental trade-off we have to make to mitigate risk and we measure its overhead in §VII-A.

B. Real-Time Risk Monitoring System

Given a risk response policy as input, the risk monitoring and response system needs to provide two key functionalities: (1) *Risk Response Policy Enforcement:* To enforce the policy, the system needs to monitor the provider's operations and associated risk at runtime and take appropriate actions based on the level of risk. (2) *Iterative Granting Workflow:* If the ticket cannot be resolved with the granted blocks, the provider can require additional blocks iteratively through the granting workflow. The second functionality can be realized by extending the current interface, but the first functionality may be difficult to achieve.

The key challenge is to *correctly and efficiently* enforce granted access. First, the system must ensure that the provider cannot access blocks that have not been granted, which leads to an incorrect monitored risk value and failure to respond to potential risks. Second, the system must be able to accurately calculate the risk based on the provider's actions, otherwise, it will not be able to take the correct actions as specified in the policy. Lastly, in order to be practical and facilitate adoption, the system should impose minimal overhead on the provider's operations and be compatible with existing interfaces.

Strawman Approach: To meet these requirements, a naive solution is to employ existing router authorization techniques (e.g., Cisco TACACS+ [6]). This mechanism allows the admin to create a blacklist/whitelist of commands and authorize users' operations based on the list. However, this mechanism may fail in two aspects: (i) it can only check commands at a coarse level, which cannot be mapped to fine-grained configuration blocks (discussed in §II-E); (ii) different tickets may require different levels of access on various devices. This means that during ticket resolution, it may be necessary to reconfigure multiple routers several times. Furthermore, these authorization techniques vary by vendor and may not be available for all devices.

Reference Monitor: Our approach is based on the observation that the provider performs operations remotely, and all actions pass through *a choke-point* before being executed on different devices in the customer network. This choke-point exists in RMM software and is currently used to maintain communication channels with devices. This provides an ideal position to examine each operation and take appropriate actions to address potential risk.

We insert a *reference monitor* [69] at the choke-point to mediate all actions performed by the technician. The reference monitor takes the risk response policy as input and maintains a list of granted blocks, and maintains a state machine inside it to track the current block in a fine-grained way. Upon receiving each command, it either permits or denies the command according to the granted blocks. As our evaluation shows (§VII-C), the latency overhead added by the reference monitor is negligible when compared to the overall time required to resolve the ticket. Meanwhile, our system tracks the risk value of the current ticket and takes actions based on the risk level. For example, if the risk has reached an intolerable level, it will halt the operation and notice the customer.

VI. Implementation

In this section, we discuss our implementation of the risk dependency graph and a proof-of-concept system as a candidate solution to serve as a basis for outsourcing workflows.

Risk Assessment Model: Our risk assessment model and risk response policy are built on top of Batfish [44]. We modified $\sim$ 2000 lines of Java code in Batfish to build the risk dependency for a given snapshot of the network configurations. We first leverage Batfish to parse the configuration files into vendor-neutral objects to create our configuration blocks. *Heimdall* currently identifies each *InterfaceBlock*, *ProtocolBlock* (e.g., BPG, OSPF) and *ACLBlock* as an individual block, and then treats the remaining lines as a *ManageBlock*.

We modify Batfish following the routing logic and dataflow to *update the dependency*. Specifically, we take advantage of the route messages sent between virtual routers when updating their data plane, and build a map between routes and related configuration blocks. We append our dependency to the route messages, and add the ingress/egress protocols and interface blocks accordingly. The main iterations in Fig. 5 mentioned above are performed by `computeDataPlane` process in Batfish until the route information on all routers converges. At this point, the routes in Batfish and their Haimdall dependency follow the `RIB` and `FIB` dataflow in Fig. 5 stage 1, which we later link with the endhosts using the `FIB` in stage 2 to complete our risk dependency model. *Heimdall* can support all configuration formats supported by Batfish, including those from Cisco, Juniper, and Arista.

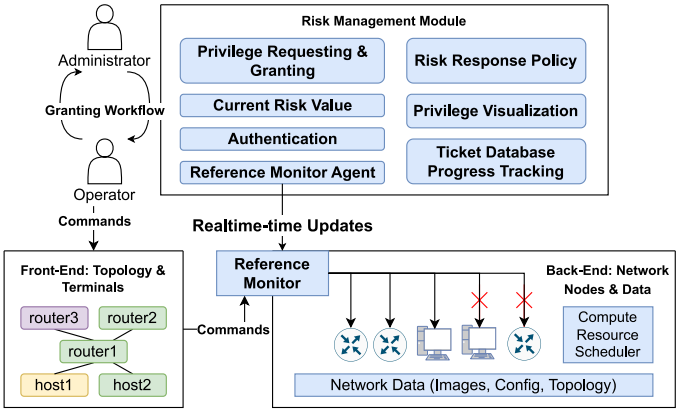

Fig. 7: A proof-of-concept implementation of reference monitor and risk management modules.

Risk Monitoring and Response System: To demonstrate the capabilities of our risk monitoring and response system, we build a proof-of-concept prototype on top of GNS3 [26]. We extend GNS3 to enable risk monitoring and response, and implement the iterative granting workflow. As shown in Fig. 7, we augment the GNS3 GUI to provide the technician with visualized privilege information and enables the operator to require more blocks. We add the *reference monitor* module as a telnet proxy to intercept commands from different sources of the front-end consoles (e.g., router and endhost consoles).

VII. Evaluation

We evaluate *Heimdall* on a collection of networks (e.g., campus, ISP, enterprise) to address the following questions: *(a): Is Heimdall practical to use in solving real-world network problems?* We invited 10 experts, collecting 99 experiment records with a total of 41 hours of testing. *(b): How does Heimdall aid in assessing the risk of different tickets?* We show that *Heimdall* enables risk assessment for different tickets and can reduce up to 75% risk for 50% issues of various types (e.g., BGP, OSPF, ACL) and on different topologies; *(c): The performance of individual components of Heimdall* We show that (i) our dependency model can achieve a high accuracy compared to existing models, and can scale to large-scale networks, and (ii) our reference monitor provides fine-grained control by adding only 4-5ms of latency;

Evaluation Networks Our evaluation networks cover different scales, structures, and configuration types. Expert validation includes three networks with real BGP, OSPF, and ACL configurations: an enterprise network, a university network [44], and a backbone network. We demonstrate the topologies of the three networks graphically in Appendix A. We use an enlarged set of networks and tickets in the simulation for risk assessment, including two more networks from Topology Zoo [58], and three Fat-tree networks (k=4,8,16). Statistic details of the networks are shown in Table III.

A. Expert Validation

We conduct a thorough study of the usability of *Heimdall* through expert validation. In this section, we invite 10 experts (detailed backgrounds listed in Appendix Table V) from network vendors of our university to solve 33 different network tickets (summarized in Appendix Table IV). The tickets include connectivity, packet drop, and quality-of-service(QoS) issues, caused by BGP, OSPF, Interface, or Route-map configuration errors. During the experiments, experts try to resolve the issue by modifying configuration blocks they currently can access. If they are unable to solve the issue, they request privilege elevation to the next level for more block accesses. This procedure is repeated until task completion or exceeding the time limit. We collected 99 experiment records, among which 93 valid experiments completed the task within the time limit.

Time and Risk Breakdown: Figure 8 shows the detailed time breakdown of all tickets tested by the experts. Each bar shows the time breakdown of an individual ticket, each ticket is assigned to multiple participants and here we only show the average values. The average time to solve a task is $\sim$ 550s, with a standard deviation of $\sim$ 389. Figure 9 shows the step-by-step risk and the least possible risk to solve each ticket. Most tickets (26 out of 30) are solved with less than 50% risk, consistent with the risk of root cause blocks. Outliers in time and risk may happen due to some non-optimal technical choices of our expert.

Risk Control: We calculate the eventual risk level upon experiment completion and compare it with that of root cause blocks. We first show the results in Fig. 10, in the order of completion time. Each point refers to an experiment. X-axis indicates their duration, and y-axis shows their final risk level relative to the root cause. In all valid experiments, experts can solve 92% of tasks with no extra privilege granted higher than the risk of the root cause blocks. We observe the correlation between risk level and finishing time: over 25% of experiments

ID	Network	#routers	#hosts	#links	#config lines	#blocks	#tickets	avg hops	max risk
A	Enterprise	10	8	26	1095	101	200	3.3	56
B	University	13	8	25	1652	120	461	4.7	56
C	Backbone	11	9	22	980	110	177	4.1	72
D	Bics	49	98	162	4410	340	469	4.9	9506
E	Columbus	86	68	169	6968	458	770	8.5	4556
F	FatTree04	20	16	48	1544	144	129	4.5	240
G	FatTree08	72	64	320	8448	800	529	4.7	4160
H	FatTree16	272	256	2304	52224	5248	318	4.9	65280

TABLE III: Eight Evaluation networks

finished in more than 16.7min requires higher privilege level than the root cause block. We also group the relative risk level of experiments by the type of tickets in Fig.11. We find that experts can solve 94% and 86% Ethernet interface and BGP tasks respectively, with no extra privilege granted. This is because most interface issues are simple and involve a small number of blocks. Given the complexity of BGP configuration, these tasks shows a higher variety in final risk levels: some experts achieve the target using alternative configuration blocks, resulting in 9.5% tests finished with risk lower than the root cause block. This experiment demonstrates that *Heimdall* can effectively control risk of outsourced network operations without impacting the ticket resolution.

Real World Incidents: To further demonstrate the capability of *Heimdall* in handling complex real world scenarios, we reproduce 3 real-world network outage incident tasks according to their disclosed incident details[7], [30], [10]. In Fig. 10 and 11, red points represent incident tickets. They span from BGP optimizer error, route-map mismatching, to prefix list mistake. Experts diagnose and fix erroneous network behaviors including wrong routing paths, disconnection and packet drops. Fig. 10 shows that incident experiments distribute loosely in the 13-33min interval due to higher complexity, with over 30 common issue experiments lying around. This indicates that *Heimdall* is practical in solving real world incidents, and our evaluation tickets cover a good range of such scenarios.

Expert Ratings: We collect the quantitative ratings from all 10 participating experts regarding the similarity between *Heimdall* and practical consoles and workflows. We show the results in Fig. 12. Overall, the survey from the operators suggested that *Heimdall* is easy to use, as our operating interfaces are not much different from current practice, with a similarity rating of 4.2. On the other hand, the experts' feedback indicates that our new workflow sometimes requires an updated mindset to carefully check more details before asking the customer to access more blocks. We believe this is a necessary trade-off towards reducing risk for network management outsourcing.

B. Effectiveness of Risk Assessment

To evaluate our risk assessment model under more conditions, we create an enlarged *ticket database* with a total of 3053 tickets in addition to the tickets in expert validation. These tickets are generated on 8 networks (Table III) covering 22 different types (Table VI) of tasks described in the Cisco Official Cert Guide [60] and StackExchange [27]. We categorize the issues into 3 problem types based on the root cause: BGP, OSPF, and ACL.

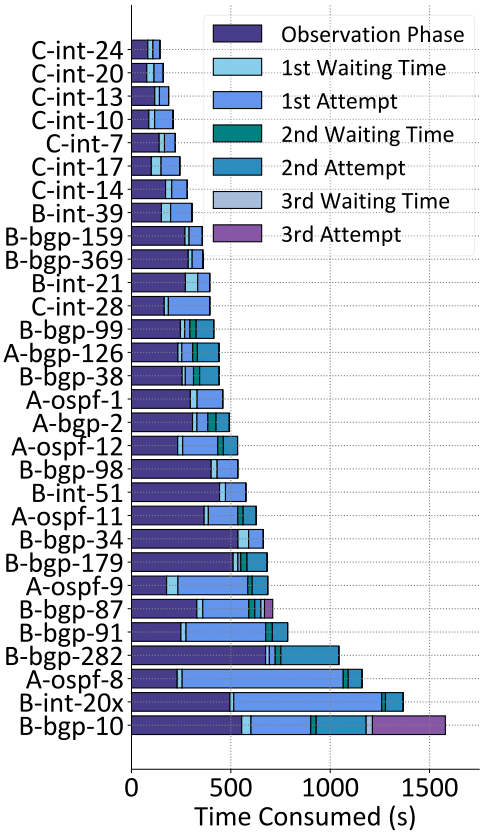

Fig. 8: Average time breakdown of expert validation.

We implemented three debuggers, which take a list of $impacted\_hosts$ and output a rank of different configuration blocks, simulating different patterns a human expert would use to locate and resolve the issue in real-world scenarios. They are *History_debugger*, *Path_debugger*, and *Random_debugger*, which rank blocks based on statistic data of the root_cause of historical tickets, distance to the shortest path between the affected hosts, or using random ordering.

Our metric is the *risk value* associated with resolving a ticket. Based on the order in which blocks are granted and the root cause, we can compute the resulting risk for a given ticket ($Equation$ 2). In our synthetic experiments, we assign one asset to each host with a value of 1. We give a performance overview of the path-based debugger on the 8 evaluation networks in Fig. 13. Throughout the test, we examined six factors that impact risk assessment: (1) block type, (2) debugger type, (3) number of batches, (4) ticket type, (5) network type and size, and (6) number of changes.

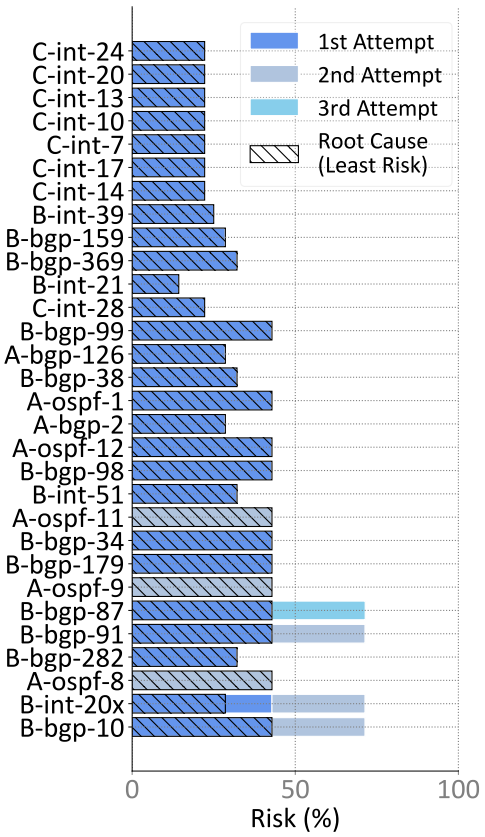

Fig. 9: Average granted risks in expert validation.

Results: In this section, we first report the aggregate results for 8 networks and use network NetC as an example to show how each factor affects risk assessment. The detailed results for other networks can be found in Appendix B.

• ***Block Type:*** Fig. 14 simulates the *worst-case* scenario by modifying $\{1, 2, 4, 8\}$ block(s) with the highest risk. Results show that even for large networks (e.g., D, E, and H), given access to one block may affect up to 50% connections, and 8 blocks may affect over 90% connections[2]. This experiment indicates that a few modifications to configuration blocks by a malicious provider can affect a significant number of assets without risk control.

• ***Debugger Type:*** Fig. 15 shows how different debuggers affect the risk. Here we assign a single block at a time to obtain detailed results. The min line shows the minimal risk and is calculated based on the risk value of the root_cause block. This represents the most ideal strategy that directly identifies and fixes the issue. Results show that to resolve 50% tickets, the $History\_debugger$ incurs 60% less risk than the $Random\_debugger$. This experiment shows that accurately evaluating the probability $\mathbb{P}(c|ticket)$ in $Equation$ 2 can help reduce the probability of affecting unrelated assets while resolving the ticket.

• ***Number of Batches:*** Fig. 16 shows that the risk can be reduced by providing fine-grained risk levels. When the number of batches is small, access to more redundant blocks is granted, increasing the risk of resolving the same ticket. But the effect becomes less significant after 8 or 16 batches.

• ***Risk granularity:*** Fig. 17 shows the minimum required risk for solving issues using router-level versus block-level risk assessment in network C. Given the best level of expertise, say the technician identifies the root cause directly, using router-level still posts higher risk than our block-granularity approach.

• ***Other Factors:*** We show the evaluation results of other factors including *network type and size*, *ticket type*, and *number of changes* in Fig. 18 and Appendix B. We observe that a greater proportion (50%-70%) of tickets are solved within 25% risk for larger networks (e.g., E, H), but the risk level is higher for smaller networks (e.g., A, B). Ticket type also affects the risks, which is consistent with our observation of risk level regarding the complexity of different ticket types in expert validation. We evaluate multiple changes for complicated incident scenarios, as the number of changes increases, the overall risk tends to rise.

C. Evaluation of Individual Components

Next, we evaluate the effectiveness of two key components of *Heimdall*: risk dependency model and reference monitor.

Performance and Accuracy of Risk Dependency Model: Fig. 19 shows the accuracy of our dependency model by comparing it with the strawman approach [35] which statically builds dependencies using predefined tokens (e.g., IP address or AS numbers). To normalize across all networks, the accuracy is divided by the union of blocks identified from both approaches. From the results, we can see false positive cases including wrong dependencies within a router and across routers.

False positives within a router: Many protocol blocks such as OSPF or BGP contain tokens from all interface blocks. If one interface block is linked with the protocol block, it is likely that it will then be linked with all other interface blocks. It creates a false positive because a label implies a route and it is only dependent on at most two interfaces on a router.

False positives between routers: If a false positive interface exists, it will build its dependency to another interface block on another router that is physically linked with the router it resides, as long as they are correctly configured (within the same subnet). Then the newly linked interface will create more false positives according to the single router situation. Thus, any false positive interface block will lead the number total false positive number to grow exponentially.

Time Overheads of *Heimdall*: For the computational overheads of Heimdall, Fig. 20 shows the time of building a risk dependency model for networks of different sizes. *T1* is the time of computing the dependency defined in Fig. 5, and *T2* is the total time which adds the time of outputting the map from a configuration block to the set of affected assets. *T1* is mostly impacted by the complexity of the data plane, and the cost is low (less than 10s) even for large networks such as $NetH$. *T2* is impacted by not only the size of the network (e.g. net$H$) but also the average path length of host connections (e.g. net$E$), which also contributes to the complexity of computing the reverse map. These computations are designed to be performed in advance for a network, so that they will not fall on the technicians when solving a ticket.
For operational overheads, the reference monitor imposes negligible end-to-end overhead on task completion time, compared

[2]Altering one block may not cause disconnection, but can still affect default routing and multipath behavior.

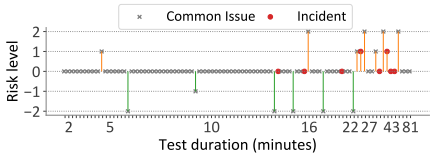

Fig. 10: Relative risk of tickets (ordered by solving time)

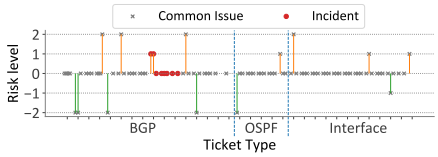

Fig. 11: Relative risk of resolved tickets

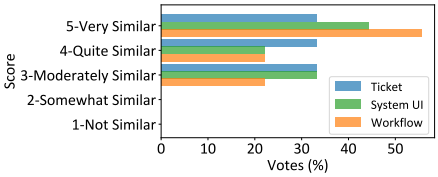

Fig. 12: Survey about similarities between experiments and the real world

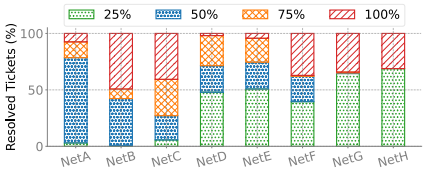

Fig. 13: Average risk of path-based debugger

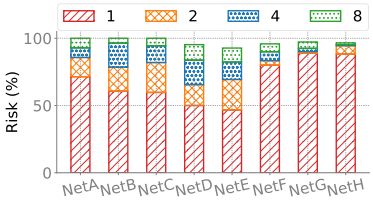

Fig. 14: Risk of malicious modifications on {1,2,4,8} blocks

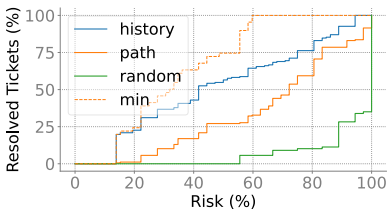

Fig. 15: NetC troubleshooting risks with different debuggers

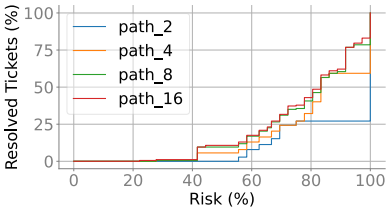

Fig. 16: NetC troubleshooting risks with different batch sizes

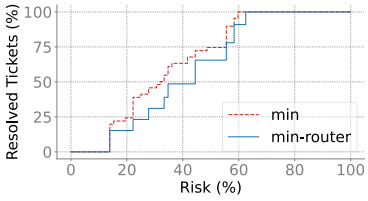

Fig. 17: NetC router-granularity vs. block-granularity risks

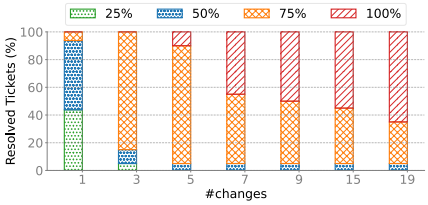

Fig. 18: Risk of multiple changes in NetC

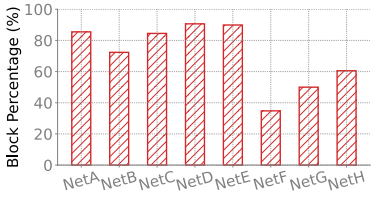

Fig. 19: Accuracy comparison of dependency models.

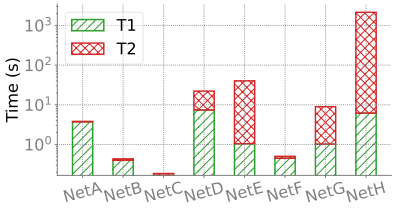

Fig. 20: Time of computing the RDG for each network

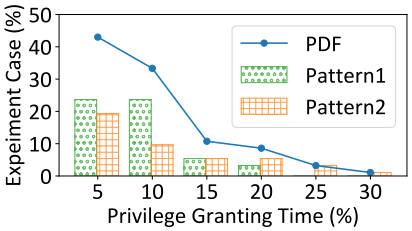

Fig. 21: Risk control procedure overhead

to technicians' operation and administrator's procedural time. It processes commands at the expense of only 4-5ms per command, which is a negligible overhead of less than 0.5% in total time, the main source of overheads should still be the waiting time and consideration time observed in our expert validation. We measure the end-to-end overhead of *Heimdall* by recording the timestamp of each operating stage and privilege approval stage, and analyze the time components for solving each task. First, we notice that the bootstrapping (The waiting time is when experts are blocked and requiring privilege elevation. And we normalize it by calculating the percentage of waiting time in the full experiment time. Fig. 21 shows that over 40% tasks are finished with less than 5% overheads of waiting for privilege and procedural operations, and the average overheads of *Heimdall* is only 7.4%. We also observe that the time component of solving tasks shows two types of operating habits: Some experts spend significantly higher observation time before making exact changes, thus the waiting overhead in all such experiments is less than 20%, and 86% of which is less than 10%. Meanwhile, other experts require privilege elevation more frequently before actually diagnosing the issue, resulting in higher procedural time during the experiment.

VIII. Discussion

Problem scope of Heimdall: In this paper, we focus on the risk of outsourced network management to third parties. The risk management of Heimdall applies to in-house admins as well, but we believe that third-party operators are more likely to perform malicious operations on the network configurations and choose them as the target scenario of Heimdall.

Accuracy of the risk model: The asset value input is the key to the accuracy of configuration block risk assessment. Existing asset management techniques [3], [12], [13] can provide accurate asset records for most cases, except when the assets are rapidly growing and the asset information might be out of date. Other factors that might impact the accuracy of our model include the fidelity of Batfish simulation we adopt in the workflow, and assessing risks with a dynamic data plane after some changes are made to the network.

Finer granularity configurations: It is possible to further divide a configuration block. such as distinguishing different

address-families. A finer granularity may improve the precision of dependencies, but may also negatively impact usability (e.g., expressing and explaining partial BGP access to the technician is much more complicated than a single BGP configuration block). Another possible proposal for finer granularity is to consider each command separately or count the number of operations made by the technician. The problems of this approach are: 1) it is more complicated to associate every potential command with its risk when the result is dynamic, compared with assessing the risk of existing configuration blocks, and 2) multiple commands performed by the technician are logically tightly coupled, so the modifications within the same block should not be accumulated multiple times given the correlation between these commands.

Dealing with multiple root_cause blocks: In our evaluation, we mostly focus on individual tickets each caused by a single misconfigured root_cause block. *Heimdall* can easily adapt to multiple root_cause blocks. If a ticket is caused by multiple blocks, the actual risk is decided by the union of the risk of all root_cause blocks.

Limitations of Heimdall: Our current risk dependency model focuses on router configurations and assumes endhosts to be bare-metal servers. However, if the network involves complex components such as middleboxes, virtual machines/containers, or service mesh, the model requires refinement to ensure high accuracy. Also, as Heimdall currently relies on user input to control the frequency of user intervention, Heimdall could become more user-friendly by incorporating a better GUI or user interaction scheme. In addition, our current risk assessment and monitoring model only controls the write access. It is possible to control read access, but this might impact usability, and measuring the potential threat from reading configurations is not straightforward. Lastly, our risk assessment model treats each asset independently, and we take the union of affected assets to avoid double-counting the same asset. However, this approach could still lead to overcounting of assets that are correlated. We think that this part can be further improved by identifying the dependencies between assets.

IX. Related Work

Network Verification: Various verification tools exist [44], [34]. In the context of network management outsourcing, these tools may be used to verify that some user-specified properties are not compromised by the changes made by the third party. Our solution takes a different perspective, namely we focus on the importance of assets and proactively estimate the maximum impact of each configuration block regarding the value of network assets. Our approach does not require the users to specify any network behaviors or properties. Instead, we ask the user to provide asset information, which requires much lower level of expertise than using verification tools.

Risk-aware Network Management: Risk-aware approaches have been employed for capacity-planning [31], [76] and traffic engineering [39], [73], [65]. Various aspects of risk have been considered, such as link failures [76], [39] impact on customer traffic during network changes [31], and revenue falls below an acceptable level [65]. Our definition of risk considers the impact on an organization's assets. Unlike prior methods [31], [76], [39], [73], [65] that aim to maximize network utilization or availability, our goal is to identify configurations that can be safely modified to resolve tickets.

Quantitative Risk: In *Heimdall* we use a general quantitative risk model commonly used by existing works (e.g., [37]), namely the product of probability and impact. There have been other more complex quantitative models such as attack graphs [45] used in risk assessment, but our work focuses on the dependency between configuration blocks and assets. Considering our granularity and asset definition, we choose the model that can best utilize the computed dependency.

Network Debugging: Many debugging approaches exist to locate network failures. [79], [40], [33], [56] first construct a fault inference graph based on network tracing data and then locate the most possible root cause from the graph. [32], [71] infer link or node failures by tracing data or measuring the packet-delivering success probability on paths. [42], [59] locate the root cause of failed links by relying on the network topology and additional traffic information. These approaches are orthogonal and complementary to ours, and as shown in our evaluation, a good troubleshooting tool can be used with *Heimdall* to effectively reduce risk.

Network Dependency Model: Many approaches have been proposed to find dependencies of network components. Some discover dependencies between services, applications, and switches based on the time correlation of messages [41] or cause analysis of service failures [79]. [33], [56] obtain dependencies among processes, nodes, and links, from performance observations or history data of component states. [35], [36] can build fine-grained dependencies between configuration blocks by parsing configuration files. NetCov [77] builds dependencies between configurations and data plane elements to compute test coverage, but it currently cannot support routing protocols other than BGP (e.g., OSPF).

X. Conclusions

While outsourcing network management is an attractive option for enterprises to reduce operational expenses, it is also posing significant security risks and turning into sources of more incidents. In this paper, we present *Heimdall*, the first *risk-aware* framework that enables customers to assess, monitor, and respond to the risk of outsourced network tasks. Our pilot study and evaluations show that *Heimdall* is a feasible tool for solving real-world issues and can significantly reduce potential risk. We have focused on addressing configuration issues for legacy networks, yet there remain many other types of issues (e.g., software bugs) and networks (e.g., SDN networks) to explore. We see *Heimdall* as the first step towards defining and managing risk for outsourced network management, and hope our initial results can lead to a broader discussion on understanding outsourced network management.

Acknowledgment

The authors would like to thank the anonymous reviewers and shepherd for the valuable feedback and suggestions. This work was supported by Peking University startup funding and NYU Shanghai startup funding. Guyue Liu is the corresponding author.

REFERENCES

[1] "Analysis of CenturyLink/Level(3) outage," https://blog.cloudflare.com/analysis-of-todays-centurylink-level-3-outage/, (Accessed on 2022/10/30).

[2] "Apt10 targeted norwegian msp and us companies in sustained campaign," https://www.recordedfuture.com/apt10-cyberespionage-campaign/.

[3] "Asset Tracking Software, Asset Tracker, Hardware & Software Inventory Management - ManageEngine AssetExplorer," https://www.manageengine.com/products/asset-explorer/track-it-assets.html, (Accessed on 01/13/2023).

[4] "Azure status history — Microsoft Azure," https://azure.status.microsoft/en-us/status/history/, (Accessed on 2023/06/04).

[5] "Cisco Privilege Levels," https://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst3850/software/release/3se/security/configuration_guide/b_sec_3se_3850_cg/b_sec_3se_3850_cg_chapter_011.pdf.

[6] "Cisco website," https://www.cisco.com/.

[7] "Cloudflare outage on June 21, 2022," https://blog.cloudflare.com/cloudflare-outage-on-june-21-2022/, (Accessed on 2023/06/04).

[8] "Download incident management handbook as PDF for free," https://www.manageengine.com/products/service-desk-msp/free-incident-management-handbook.html, (Accessed on 01/09/2023).

[9] "Google Cloud Platform experienced networking unavailability in zone europe-west2-a," https://status.cloud.google.com/incidents/qRpmq9arsrCV7xWdfftF, (Accessed on 2023/06/04).

[10] "Google Compute Engine Incident #16019," https://status.cloud.google.com/incident/compute/16019, (Accessed on 2023/06/04).

[11] "How remote access plus works? — manageengine," https://www.manageengine.com/remote-desktop-management/help/architecture.html.

[12] "IT Asset Management Software — xAssets," https://www.xassets.com/it-asset-management-software, (Accessed on 01/13/2023).

[13] "IT Asset Management Tool — ITAM — Freshservice," https://www.freshworks.com/freshservice/lp/it-asset-management/, (Accessed on 01/13/2023).

[14] "Kaseya weaponized to deliver sodinokibi ransomware : msp," https://www.reddit.com/r/msp/comments/c2wls0/kaseya_weaponized_to_deliver_sodinokibi_ransomware/ern2i9b/?utm_source=share&utm_medium=web2x.

[15] "Managed network services - aviat networks," https://aviatnetworks.com/services/aviatcare/managed-network-services/.

[16] "Managed network services — centurylink," https://www.centurylink.com/business/managed-services/managed-network-services.html.

[17] "Managed network services : Fujitsu global," https://www.fujitsu.com/global/services/infrastructure/network/managed-network/.

[18] "Managed network services — ibm," https://www.ibm.com/services/network/managed.

[19] "Managed network services — verizon business," https://www.verizon.com/business/products/managed-network-services/.

[20] "Managed services market size to reach usd 309.4 billion by 2025 - valuates reports," https://www.prnewswire.com/news-releases/managed-services-market-size-to-reach-usd-\309-4-billion-by-2025---valuates-reports\-301049291.html.

[21] "More details about the October 4 outage — Engineering at Meta," https://engineering.fb.com/2021/10/05/networking-traffic/outage-details/, (Accessed on 2022/10/30).

[22] "N-central architecture," https://success.solarwindsmsp.com/kb/solarwinds_n-central/N-central-architecture.

[23] "NIST: Zero-trust architecture," https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-207.pdf.

[24] "Operation cloud hopper," https://www.pwc.co.uk/cyber-security/pdf/cloud-hopper-report-final-v4.pdf.

[25] "Passwords and privilege levels - hardening cisco routers [book]," https://www.oreilly.com/library/view/hardening-cisco-routers/0596001665/ch04.html, (Accessed on 01/09/2023).

[26] "The software that empowers network professionals," https://www.gns3.com/.

[27] "Stackexchange website," https://stackexchange.com.

[28] "Today's Outage Post Mortem — Cloudflare," https://blog.cloudflare.com/todays-outage-post-mortem-82515/, (Accessed on 2023/06/04).

[29] "Validating the solarwinds n-central "dumpster diver" vulnerability," https://blog.huntresslabs.com/validating-the-solarwinds-n-central-dumpster\-diver-vulnerability-5e3a045982e5.

[30] "Verizon and a BGP Optimizer Knocked Large Parts of the Internet Offline," https://news.ycombinator.com/item?id=20267790, (Accessed on 2023/06/04).

[31] O. Alipourfard, J. Gao, J. Koenig, C. Harshaw, A. Vahdat, and M. Yu, "Risk based planning of network changes in evolving data centers," in *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, 2019, pp. 414–429.

[32] B. Arzani, S. Ciraci, L. Chamon, Y. Zhu, H. H. Liu, J. Padhye, B. T. Loo, and G. Outhred, "007: Democratically finding the cause of packet drops," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. Renton, WA: USENIX Association, Apr. 2018, pp. 419–435. [Online]. Available: https://www.usenix.org/conference/nsdi18/presentation/arzani

[33] P. Bahl, R. Chandra, A. Greenberg, S. Kandula, D. A. Maltz, and M. Zhang, "Towards highly reliable enterprise network services via inference of multi-level dependencies," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 4, p. 13–24, aug 2007. [Online]. Available: https://doi.org/10.1145/1282427.1282383

[34] R. Beckett, A. Gupta, R. Mahajan, and D. Walker, "A general approach to network configuration verification," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, 2017, pp. 155–168.

[35] T. Benson, A. Akella, and D. A. Maltz, "Unraveling the complexity of network management." in *NSDI*, 2009, pp. 335–348.

[36] T. Benson, A. Akella, and A. Shaikh, "Demystifying configuration challenges and trade-offs in network-based isp services," in *Proceedings of the ACM SIGCOMM 2011 Conference*, 2011, pp. 302–313.

[37] D. Bilar, "Quantitative risk analysis of computer networks," Ph.D. dissertation, 2003, copyright - Database copyright ProQuest LLC; ProQuest does not claim copyright in the individual underlying works; Last updated - 2023-03-02. [Online]. Available: http://proxy.library.nyu.edu/login?qurl=https%3A%2F%2Fwww.proquest.com%2Fdissertations-theses%2Fquantitative-risk-analysis-computer-networks%2Fdocview%2F305345300%2Fse-2%3Faccountid%3D12768

[38] R. Birkner, D. Drachsler-Cohen, L. Vanbever, and M. Vechev, "Config2Spec: Mining network specifications from network configurations," in *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. Santa Clara, CA: USENIX Association, Feb. 2020, pp. 969–984. [Online]. Available: https://www.usenix.org/conference/nsdi20/presentation/birkner

[39] J. Bogle, N. Bhatia, M. Ghobadi, I. Menache, N. Bjørner, A. Valadarsky, and M. Schapira, "Teavar: striking the right utilization-availability balance in wan traffic engineering," in *Proceedings of the ACM Special Interest Group on Data Communication*, 2019, pp. 29–43.

[40] A. Chen, Y. Wu, A. Haeberlen, W. Zhou, and B. T. Loo, "The good, the bad, and the differences: Better network diagnostics with differential provenance," ser. SIGCOMM '16. New York, NY, USA: Association for Computing Machinery, 2016. [Online]. Available: https://doi.org/10.1145/2934872.2934910

[41] X. Chen, M. Zhang, Z. M. Mao, and P. Bahl, "Automating network application dependency discovery: Experiences, limitations, and new solutions." in *OSDI*, vol. 8, 2008, pp. 117–130.

[42] A. Dhamdhere, R. Teixeira, C. Dovrolis, and C. Diot, "Netdiagnoser: Troubleshooting network unreachabilities using end-to-end probes and routing data," in *Proceedings of the 2007 ACM CoNEXT Conference*, ser. CoNEXT '07. New York, NY, USA: Association for Computing Machinery, 2007. [Online]. Available: https://doi.org/10.1145/1364654.1364677

[43] M. J. Earl, "The risks of outsourcing it," *MIT Sloan Management Review*, 1996.

[44] A. Fogel, S. Fung, L. Pedrosa, M. Walraed-Sullivan, R. Govindan, R. Mahajan, and T. Millstein, "A general approach to network con-

figuration analysis," in *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'15. USA: USENIX Association, 2015, p. 469–483.

[45] M. Frigault and L. Wang, "Measuring network security using bayesian network-based attack graphs," in *2008 32nd Annual IEEE International Computer Software and Applications Conference*, 2008, pp. 698–703.

[46] A. Gember-Jacobson, W. Wu, X. Li, A. Akella, and R. Mahajan, "Management plane analytics," in *Proceedings of the 2015 Internet Measurement Conference*, 2015, pp. 395–408.

[47] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive verifiable computing: Outsourcing computation to untrusted workers," in *Proceedings of the 30th Annual Conference on Advances in Cryptology*, ser. CRYPTO'10. Berlin, Heidelberg: Springer-Verlag, 2010, p. 465–482.

[48] R. Goel, A. Kumar, and J. Haddow, "Prism: a strategic decision framework for cybersecurity risk assessment," *Information & Computer Security*, vol. 28, no. 4, pp. 591–625, 2020.

[49] H. Hacigümüş, B. Iyer, C. Li, and S. Mehrotra, "Executing sql over encrypted data in the database-service-provider model," in *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '02. New York, NY, USA: Association for Computing Machinery, 2002, p. 216–227. [Online]. Available: https://doi.org/10.1145/564691.564717

[50] Y. Y. Haimes, "Total risk management," *Risk analysis*, vol. 11, no. 2, pp. 169–171, 1991.

[51] ——, *Risk modeling, assessment, and management*. John Wiley & Sons, 2005.

[52] ——, "On the definition of vulnerabilities in measuring risks to infrastructures," *Risk Analysis: An International Journal*, vol. 26, no. 2, pp. 293–296, 2006.

[53] ——, "On the complex definition of risk: A systems-based approach," *Risk Analysis: An International Journal*, vol. 29, no. 12, pp. 1647–1654, 2009.

[54] K. Jayaraman, N. Bjørner, J. Padhye, A. Agrawal, A. Bhargava, P.-A. C. Bissonnette, S. Foster, A. Helwer, M. Kasten, I. Lee *et al.*, "Validating datacenters at scale," in *Proceedings of the ACM Special Interest Group on Data Communication*, 2019, pp. 200–213.

[55] S. K. R. Kakarla, A. Tang, R. Beckett, K. Jayaraman, T. Millstein, Y. Tamir, and G. Varghese, "Finding network misconfigurations by automatic template inference," in *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, 2020, pp. 999–1013.

[56] S. Kandula, R. Mahajan, P. Verkaik, S. Agarwal, J. Padhye, and P. Bahl, "Detailed diagnosis in enterprise networks," in *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, ser. SIGCOMM '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 243–254. [Online]. Available: https://doi.org/10.1145/1592568.1592597

[57] S. Kaplan and B. J. Garrick, "On the quantitative definition of risk," *Risk analysis*, vol. 1, no. 1, pp. 11–27, 1981.

[58] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, 2011.

[59] R. R. Kompella, J. Yates, A. Greenberg, and A. C. Snoeren, "Ip fault localization via risk modeling," in *Proceedings of the 2nd Conference on Symposium on Networked Systems Design and Implementation - Volume 2*, ser. NSDI'05. USA: USENIX Association, 2005, p. 57–70.

[60] R. Lacoste and B. Edgeworth, *CCNP Enterprise Advanced Routing ENARSI 300-410 official CERT Guide*. Cisco Press, 2020.

[61] X. Lei, X. Liao, T. Huang, H. Li, and C. Hu, "Outsourcing large matrix inversion computation to a public cloud," *IEEE Transactions on Cloud Computing*, vol. 1, no. 1, pp. 1–1, 2013.

[62] G. Liu, A. Li, C. Canel, and V. Sekar, "Watching the watchmen: Least privilege for managed network services," in *Proceedings of the 20th ACM Workshop on Hot Topics in Networks*, 2021, pp. 147–154.

[63] H. H. Liu, Y. Zhu, J. Padhye, J. Cao, S. Tallapragada, N. P. Lopes, A. Rybalchenko, G. Lu, and L. Yuan, "Crystalnet: Faithfully emulating large production networks," in *Proceedings of the 26th Symposium on Operating Systems Principles*, 2017, pp. 599–613.

[64] W. W. Lowrance and J. Klerer, "Of acceptable risk: Science and the determination of safety." *Journal of The Electrochemical Society*, vol. 123, no. 11, p. 373C, nov 1976.

[65] D. Mitra and Q. Wang, "Stochastic traffic engineering for demand uncertainty and risk-aware network revenue management," *IEEE/ACM Transactions on networking*, vol. 13, no. 2, pp. 221–233, 2005.

[66] D. Plonka and A. J. Tack, "An analysis of network configuration artifacts." in *LISA*, 2009, pp. 79–91.

[67] S. M. Radack *et al.*, "Managing information security risk: Organization, mission, and information system view," 2011. [Online]. Available: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=908207

[68] R. K. Rainer Jr, C. A. Snyder, and H. H. Carr, "Risk analysis for information technology," *Journal of Management information systems*, vol. 8, no. 1, pp. 129–147, 1991.

[69] F. B. Schneider, "Least privilege and more [computer security]," *IEEE Security & Privacy*, vol. 1, no. 5, pp. 55–59, 2003.

[70] Y.-W. E. Sung, X. Tie, S. H. Wong, and H. Zeng, "Robotron: Top-down network management at facebook scale," in *Proceedings of the 2016 ACM SIGCOMM Conference*, 2016, pp. 426–439.

[71] C. Tan, Z. Jin, C. Guo, T. Zhang, H. Wu, K. Deng, D. Bi, and D. Xiang, "Netbouncer: Active device and link failure localization in data center networks," ser. NSDI'19. USA: USENIX Association, 2019, p. 599–613.

[72] A. Tang, S. K. R. Kakarla, R. Beckett, E. Zhai, M. Brown, T. Millstein, Y. Tamir, and G. Varghese, "Campion: debugging router configuration differences," in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, 2021, pp. 748–761.

[73] B. Vidalenc, L. Ciavaglia, L. Noirie, and E. Renault, "Dynamic risk-aware routing for ospf networks," in *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*. IEEE, 2013, pp. 226–234.

[74] W. Wang, Z. Li, R. Owens, and B. Bhargava, "Secure and efficient access to outsourced data," in *Proceedings of the 2009 ACM Workshop on Cloud Computing Security*, ser. CCSW '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 55–66. [Online]. Available: https://doi.org/10.1145/1655008.1655016

[75] D. W. Woods and R. Böhme, "Sok: Quantifying cyber risk," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 211–228.

[76] Y. Xia, Y. Zhang, Z. Zhong, G. Yan, C. Lim, S. S. Ahuja, S. Bali, A. Nikolaidis, K. Ghobadi, and M. Ghobadi, "A social network under social distancing: risk-driven backbone management during covid-19 and beyond," in *18th USENIX Symposium on Networked Systems Design and Implementation*. USENIX Association, 2021, pp. 217–231.

[77] X. Xu, W. Deng, R. Beckett, R. Mahajan, and D. Walker, "Test coverage for network configurations, https://arxiv.org/abs/2209.12870," 2022.

[78] J. Yuan and S. Yu, "Efficient privacy-preserving biometric identification in cloud computing," in *2013 Proceedings IEEE INFOCOM*, 2013, pp. 2652–2660.

[79] E. Zhai, A. Chen, R. Piskac, M. Balakrishnan, B. Tian, B. Song, and H. Zhang, "Check before you change: Preventing correlated failures in service updates," in *Proceedings of the 17th Usenix Conference on Networked Systems Design and Implementation*, ser. NSDI'20. USA: USENIX Association, 2020, p. 575–590.

[80] P. Zhang, A. Gember-Jacobson, Y. Zuo, Y. Huang, X. Liu, and H. Li, "Differential network analysis," in *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, 2022, pp. 601–615.

[81] Y. Zhang, N. Hu, C. Verge, and S. O'Brien, "Cross-layer diagnosis of optical backbone failures," ser. IMC '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 419–432. [Online]. Available: https://doi.org/10.1145/3517745.3561447

APPENDIX

A. Experiment Networks

As described in section VII, our evaluation experiments are conducted on three different types of networks with different topologies and configurations.

Figure 22 presents the enterprise network with 8 hosts and 10 routers. It consists of 1 BGP AS and 1 OSPF area.

Ticket ID	Root Cause Block	Matching Type
C-int-24	r5[GigabitEthernet3/0]	Interface:22
C-int-20	r4[GigabitEthernet2/0]	Interface:22
C-int-13	r2[GigabitEthernet2/0]	Interface:22
C-int-10	r1[GigabitEthernet2/0]	Interface:22
C-int-7	r11[GigabitEthernet3/0]	Interface:22
C-int-17	r3[GigabitEthernet2/0]	Interface:22
C-int-14	r2[GigabitEthernet3/0]	Interface:22
B-int-39	as2dist2[GigabitEthernet6/0]	Interface:22
B-bgp-159	as2dist1[bgp2]	BGP:1
B-bgp-369	as1border1[bgp1]	BGP:1
B-int-21	as2dept1[GigabitEthernet1/0]	Interface:22
C-int-28	r6[GigabitEthernet3/0]	Interface:22
B-bgp-99	as3border2[bgp3]	BGP:8
A-bgp-126	isp2[bgp102]	BGP:10
B-bgp-38	as3border1[bgp3]	BGP:10
A-ospf-1	r1[ospf10]	OSPF:13
A-bgp-2	isp4[bgp104]	BGP:5
A-ospf-12	r2[ospf10]	OSPF:15
B-bgp-98	as3border2[bgp3]	BGP:8
B-int-51	as2border2[GigabitEthernet0/0]	Interface:22
A-ospf-11	r2[ospf10]	OSPF:15
B-bgp-34	as1border2[bgp1]	BGP:1
B-bgp-179	as2dist2[bgp2]	BGP:8
A-ospf-9	r2[ospf10]	OSPF:12
B-bgp-87	as3border2[bgp3]	BGP:5
B-bgp-91	as3border2[bgp3]	BGP:8
B-bgp-282	as2border1[bgp2]	BGP:8
A-ospf-8	r2[ospf10]	OSPF:13
B-int-20x	as2dept1[GigabitEthernet0/0]	Interface:22
B-bgp-10	as1border2[bgp1]	BGP:8
A-inc-23	r9[bgp103]	Incident:23
C-int-24	r9[bgp900]	Incident:24
C-int-25	r10[bgp1000]	Incident:25

TABLE IV: Information of tickets used in expert validation, including root cause blocks, and matching problem type in VI. Tickets are sorted by total time consumed for test case samples.

Figure 23 presents the university network topology including 8 hosts and 13 routers. This network includes 3 BGP ASes and 1 OSPF area. Finally, Figure 24 presents the backbone network topology including 9 hosts and 11 routers. The backbone network consists of 2 BGP ASes and 2 OSPF areas.

B. Evaluation

Issue types:: Table IV shows the issues and their corresponding root cause types. Table VI presents the real-world issues described in Section VII. We categorize each issue into 3 problem types based on the root cause: BGP, OSPF, and ACL. The description explains how to resolve the ticket and what effect it may have.

Test Results:: In addition to the test results from Section VII-B, we present test results of the other 7 test networks shown in Table III. Figure 25 to Figure 31 each shows the results of one test network, the figures from left to right are the results of: 1) the risk CDF of using different debuggers 2) the risk range of different task types using the Path_debugger 3) the risk CDF of grouping the blocks into different numbers of batches 4) the risk of using router-granularity or block-granularity privilege control.

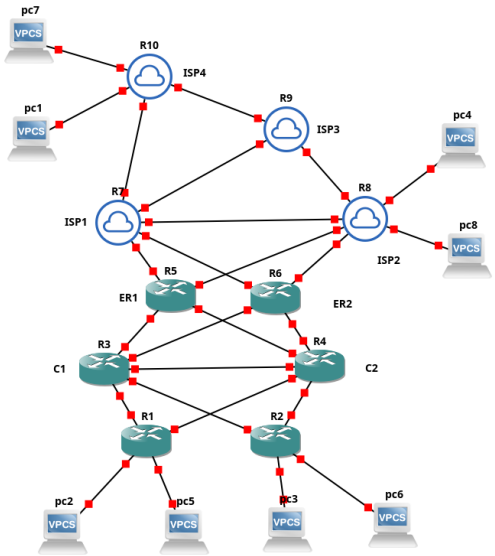

Fig. 22: Network A: Enterprise network topology

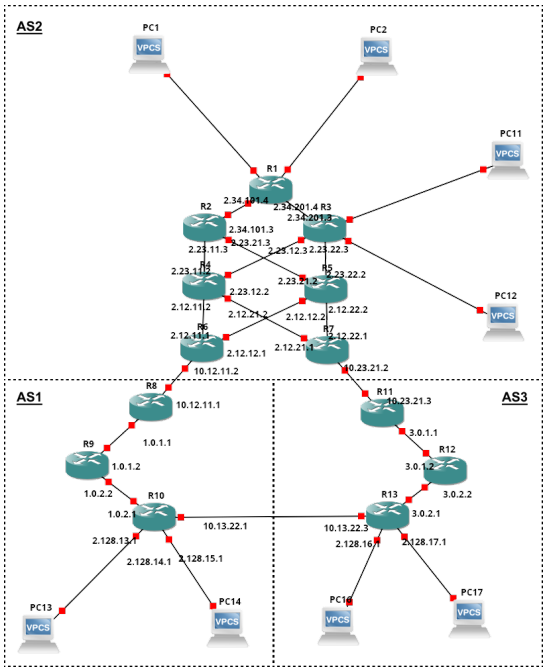

Fig. 23: Network B: University network topology

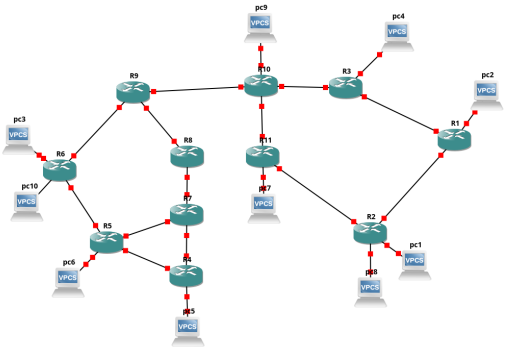

Fig. 24: Network C: Backbone network topology

Expert ID	Working Experience			Certificates	Overall Expertise Level
	Configuration	Years	Troubleshooting Frequency		
1	BGP, OSPF, ACL	3 Months - 1 Year	Weekly	-	Medium
2	BGP, OSPF, ACL	> 3 Years	Weekly	CCNP	High
3	BGP, OSPF, ACL	1 - 3 Years	Daily	CCNP	Medium
4	OSPF, ACL	1 - 3 Years	Weekly	CCNA	Medium
5	ACL	3 Months - 1 Year	Monthly	HCIE	Low
6	OSPF, ACL	> 3 Years	Daily	-	Medium
7	BGP, OSPF, ACL	> 3 Years	Monthly	CCIE	High
8	OSPF, ACL	1 - 3 Years	Monthly	CCNP	Medium
9	BGP, OSPF, ACL	> 3 Years	Daily	CCIE	High
10	BGP, OSPF, ACL	> 3 Years	Weekly	CCNA, HCIE	High

TABLE V: Expertise information of experts

Problem Type	Test No.	Root Cause	Description
BGP	1	AS misconfiguration	Add/Modify the BGP AS of a remote network and advertise correct routes to neighbor routers.
	2	IP prefix-list misconfiguration	Modify IP prefix list to permit/deny specific traffic.
	3	Local-preference misconfiguration	Modify the local-preference of neighbor routers to set traffic to flow in a specific path.
	4	Missing route redistribution	Add redistribution settings to advertise routes obtained from all routing protocols.
	5	Network IP misconfiguration	Modify BGP neighbor IPs and advertise correct routes to neighbor routers.
	6	Missing connected route configuration	Add static/connected route settings for advertising routes not related to routing protocols.
	7	Route map weight misconfiguration	Modify the BGP weight to change specific traffic routes
	8	Network IP missing	Delete/Modify BGP neighbor network IP addresses to manage networks in an AS.
	9	Route redistribution misconfiguration	Add/Modify/Delete BGP redistribution subnets.
	10	Router id misconfiguration	Modify the unique identifier of a BGP router in an AS.
OSPF	11	Remote area misconfiguration	Modify area numbers related to each interface.
	12	Neighbor network misconfiguration	Modify neighbor network IPs and advertise correct routes to neighbor routers.
	13	Process ID misconfiguration	Modify the OSPF process ID to correctly engage the routing protocol with interfaces and redistribute routes.
	14	Missing area configuration	Adding a new OSPF area to an OSPF router by adding new subnets to an area.
	15	Neighbor network missing	Delete/Modify OSPF neighbor network IP addresses to manage networks in an area.
	16	Redistribution missing	Delete redistribute BGP subnets in an OSPF area.
	17	Redistribution misconfiguration	Modify the AS number to redistribute in an OSPF area.
	18	OSPF cost misconfiguration	Modify cost values to change the interface on OSPF routers.
ACL	19	ACL misconfiguration	Add/Modify ACL to permit/deny specific traffic.
	20	ACL missing	Delete an ACL rule in the access list to permit/deny access.
	21	ACL group misconfiguration	Modify ACL group name in interface blocks to permit/deny specific traffic.
Interface	22	Interface misconfiguration	Modify interface IP address on routers.
Incident	23	Address-family misconfiguration	BGP optimizer error of advertising more specific network prefixes in BGP address-family
	24	Route-map misconfiguration	Route-map mismatch caused by advertising wrong network prefix in BGP route-map
	25	BGP preference misconfiguration	Prefix list denial mistake of configuring higher preference for a filter that rejects all prefixes

TABLE VI: Type of issues in real world

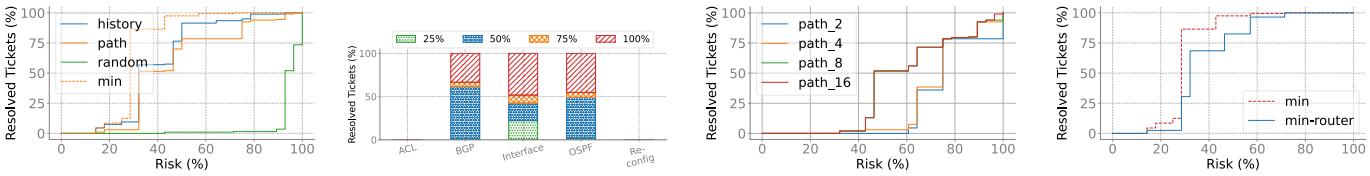

Fig. 25: NetA test results

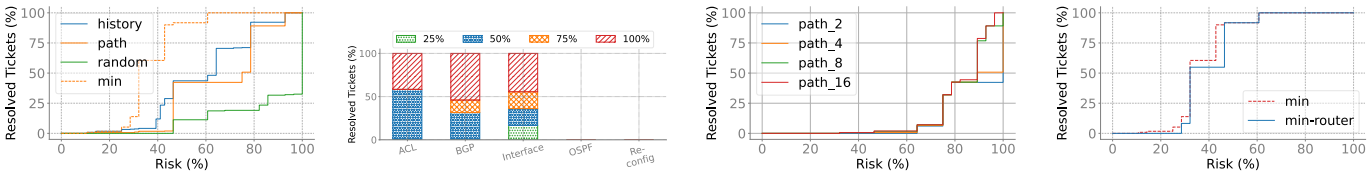

Fig. 26: NetB test results

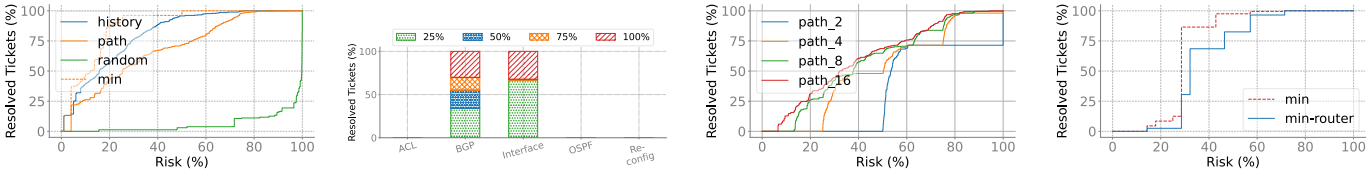

Fig. 27: NetD test results

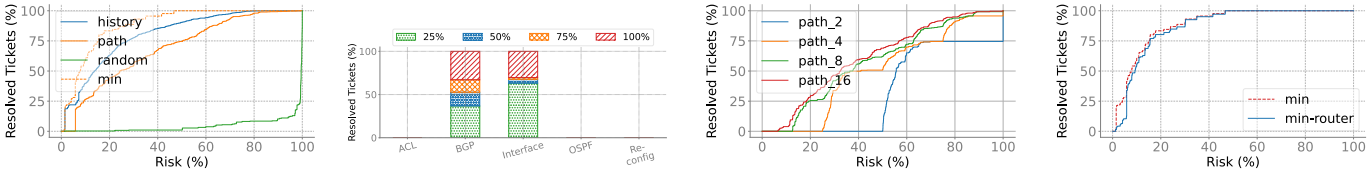

Fig. 28: NetE test results

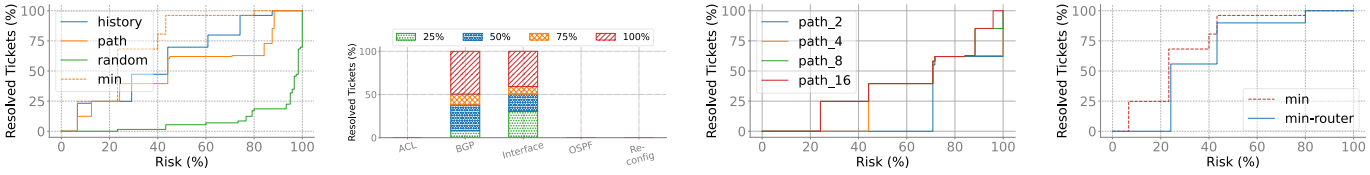

Fig. 29: NetF test results

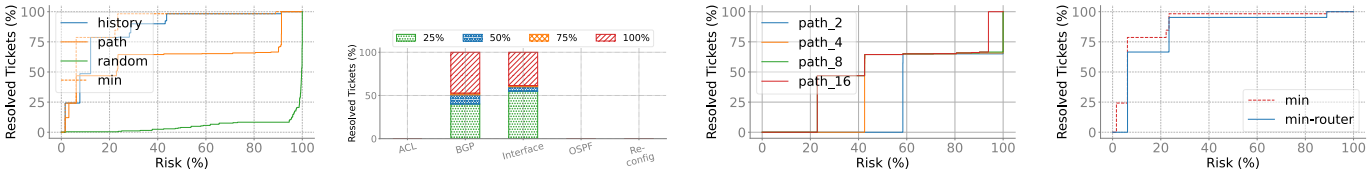

Fig. 30: NetG test results

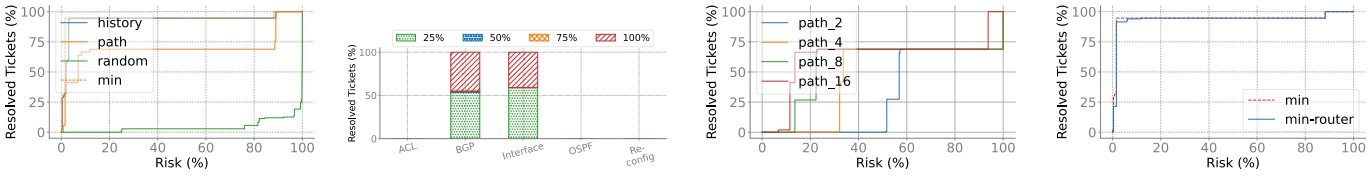

Fig. 31: NetH test results