# Watching the watchmen: Least privilege for managed network services

Guyue Liu
New York University Shanghai

Ao Li
Carnegie Mellon University

Christopher Canel
Carnegie Mellon University

Vyas Sekar
Carnegie Mellon University

## ABSTRACT

Many enterprises outsource network management (e.g., troubleshooting failures, monitoring performance) to third-party managed service providers (MSPs) to reduce cost. Unfortunately, recent incidents show that MSPs themselves have become an attractive launchpad to gain access to customer networks. In this work, we argue that such incidents arise due to a violation of *the least privilege principle*. We revisit the MSP outsourcing problem through this least-privilege view, identify key challenges in realizing this framework, and present initial ideas toward this goal. In particular, we propose providing the MSP provider an isolated "digital twin" environment to resolve problems and prevent providers from directly accessing the customer production network. Changes are verified before importing them into the production network, ensuring there are no privilege violations. Our preliminary experiments show that our approach can resolve practical problems (e.g., misconfigurations) and is effective in reducing the attack surfaces for MSP customers.

**ACM Reference Format:**
Guyue Liu, Ao Li, Christopher Canel, and Vyas Sekar. 2021. Watching the watchmen: Least privilege for managed network services. In *Proceedings of The 20th ACM Workshop on Hot Topics in Networks (HotNets'21).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3484266.3487380

## 1 INTRODUCTION

Small- to medium-sized enterprises (e.g., banks, hospitals, stores) are outsourcing their network management to third-party managed service providers (MSPs) (e.g., IBM, Fujitsu, CenturyLink [13–17]). MSPs remotely monitor a client's network and provide various management functions; e.g., configuring firewall rules, troubleshooting Wi-Fi issues, and monitoring bandwidth usage. Compared to in-house IT teams, MSPs offer reduced operational expenses and the market is large (e.g., USD 178.5 billion in 2019 and projected to grow to USD 309.4 billion by 2025 [18]).

As the MSP model gains traction, MSPs are themselves becoming attractive attack targets, especially as launchpads to attack their customers [2, 11, 31]. Compromising a single large MSP could provide attackers with access to hundreds to thousands of customer networks. In particular, small MSP providers have fewer resources and employees, so they can be attacked with less effort. This threat is not hypothetical and many high-profile incidents have already been reported; e.g., the APT10 threat group's Operation Cloud Hopper campaign [25] targeted hundreds of MSPs worldwide [2].

Analyzing these incidents, the key reason behind these attacks is that MSPs have unrestricted access to clients' networks. Such access is indeed necessary for some operations; e.g., installing software or configuring firewall rules. However, it also opens doors for an adversary to abuse these privileges for subversive purposes; e.g., stealing intellectual property, gathering information for commercial advantage [2, 24, 31], or installing ransomware [11].

In this work, we revisit this problem from first principles—through classical concepts of *least privilege* [45] and *access control* [40]. Ideally, we want to confine the MSP's privileges to only the necessary capabilities specific to a task, verify that this occurred, and audit the actions of the MSP technicians. Thus, even if some misbehavior does occur, its impact on the client can be limited and/or detected.

To this end, we propose *Heimdall*, a new architecture to enable least-privilege when using MSPs. The high-level idea in Heimdall is to create a sandboxed "digital twin " environment for technicians to work on that mimics the production network. This prevents a technician from directly accessing the production network. After the technicians make changes, they are verified so that legitimate changes are applied to the production network and violations are intercepted.


*HotNets'21, November 10-12, 2021, Virtual Event, UK*

ACM ISBN 978-1-4503-7020-2…$15.00
https://doi.org/10.1145/3484266.3487380

Realizing this idea in practice, however, is challenging. First, any system needs to add a minimal burden for MSP customers and technicians and be *compatible* with existing workflows. Second, the system needs to be *efficient*. Checking privilege violations should not affect normal operations. Finally, the system needs to be *trustworthy*, giving MSP customers confidence that it can detect privilege violations correctly, even in the presence of attacks.

To address these challenges, we present the initial design and implementation of Heimdall, which consists of three key components. First, we design a simple yet expressive language for MSP customers to specify their policies on privilege levels for various network resources (e.g., network interface). Second, we propose a *twin network*, an emulated network environment that mimics the production network but is isolated to restrict malicious behavior, for the technician to resolve problems. A naive approach to create a twin network could use existing emulation tools [23, 28, 41] to clone the entire production network. However, this may expose sensitive data and consume significant resources. We propose a *task-driven approach* to create a minimalist twin network relevant to the given task, so that the technician can only access a minimal view of the production network. Finally, we introduce a *policy enforcer* that sits between the twin network and the production network to mediate accesses and eliminate policy violations. To guard the enforcer against attacks, we run it inside a trusted execution environment (e.g., Intel SGX).

Given increasing network complexity, we believe it is necessary to enable enterprises to outsource network management while minimizing the attack surface created by outsourcing. Our preliminary prototype and experiments show that our approach resolves practical problems (e.g., misconfigurations) and reduces attack surfaces for MSP customers.

## 2 BACKGROUND AND MOTIVATION

In this section, we provide background information on the structure of managed network services. Then, we highlight attacks that have exploited vulnerabilities in these systems.

### 2.1 MSP Services and Workflow

To reduce operational costs, enterprises are outsourcing network services to managed service providers (MSPs) [13–17]. Typical services include incident management (e.g., switch or wireless access point failure), change management (e.g., modify routing policies or firewall rules), and performance management (e.g., monitor bandwidth use).

To provide services remotely, MSPs rely on Remote Management and Monitoring (RMM) tools [10, 12, 19, 21, 27]. A typical RMM tool [7, 20] is based on a client-server architecture: one central server collaborates with one to many agents, which run on the enterprise network's devices (Figure 1). The RMM server is responsible for authenticating users and authorizing access to the agents. These agents have *root privileges*

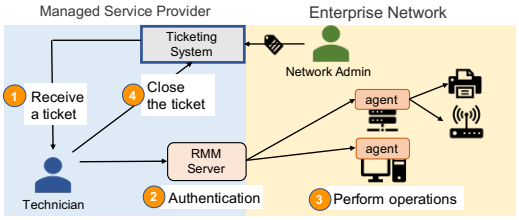



**Figure 1: Current MSP workflow: a central RMM server controls agents that have root access on network devices.**

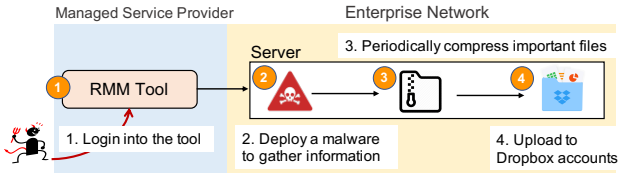



**Figure 2: MSPs are a launchpad to steal enterprise data.**

to perform commands directly on the endpoints that can run them (e.g., servers, desktops). For endpoints that cannot run agents (e.g., printers), a remote agent probes the endpoints.

Figure 1 shows a typical workflow with four steps [7, 20]: (1) A technician receives a *ticket* (created by the network administrator or by a monitoring system) from the ticketing system. For example, a ticket could be: a web service running on server *H* cannot receive packets; (2) After receiving the ticket, a technician logs into the *RMM server* (via a web console), which authenticates their identity; (3) Once authenticated, the technician has full control over network devices. They may start the debugging process at the effected host, for example, by bringing a network interface up/down. If they suspect that the issue is not associated with the host, but is actually caused by intermediate switches or middleboxes, then they can examine and modify configurations on these network devices as well. For example, the technician might check whether a firewall is blocking the port used by the web service. Since the RMM agents have root access, the technician can issue both normal and privileged commands; and (4) After finishing the task, the technician documents their changes and closes the ticket.

### 2.2 Motivating Incidents

Although outsourcing is economically attractive, recent incidents suggest that it creates new types of security threats for enterprises as MSPs themselves increasingly become attack targets. In many ways, these incidents are ironic in that the exact processes meant to secure enterprise networks have become vectors for attack. Consider two example incidents.

**Example 1 – Data breach:** One of the most well-known real-world MSP attacks was launched by the hacker group APT10 [2, 25]. As Figure 2 depicts, the attackers infiltrated MSP customers' servers via the RMM software and deployed

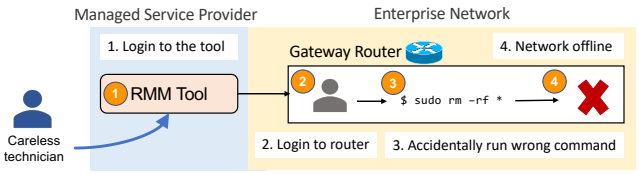



**Figure 3: Security incidents are often accidents.**

malware that gathered proprietary data (e.g., credentials, intellectual property), compressed it, and exfiltrated it to Dropbox [5] accounts. The attack lasted over six months and compromised law firms, banks, and manufacturers.

**Example 2 – Network outage:** While the remote management model exposes risk to purposeful attacks, unchecked access to customer infrastructure leads to *accidental* damage as well. In a hypothetical scenario shown in Figure 3, a technician is supposed to configure a gateway router to create a separate subnet for newly-installed servers. Unfortunately, the technician carelessly executes an erroneous command, causing a large network outage that impacts the customer's business for several hours. While human errors are inevitable, we envision a design that enables enterprises to detect them in advance and limit the scope of their damage.

## 3 HEIMDALL OVERVIEW

Looking at these incidents, the attacks are possible because there is a violation of *least privilege* [45, 46]. To quote from Saltzer and Schroeder, the principle states that: *"Every program and every user of the system should operate using the least set of privileges necessary to complete the job."* The existing MSP architecture is at odds with this principle. Once an MSP technician is authenticated, they have almost *unrestricted* access to all hosts, routers, and middleboxes on the enterprise's network. Since it is unlikely that resolving a ticket requires root privileges on the entire network, this access model clearly violates the principle of least privilege.

Thus, a natural question that motivates our work is *can we rethink the MSP workflow to enable, enforce, and validate adherence to the principle of least-privilege?* Cast in this light, it is clear that existing solutions are inadequate. For instance, some vendors are upgrading their products to provide stronger authentication (e.g., multi-factor authentication, password management) [8, 26, 30], but these mechanisms are too coarse-grained and do not address the original issue of an unnecessarily large attack surface. A rogue technician or an attacker that passes the authentication (e.g., by phishing credentials) can still cause the above example incidents. Instead, this problem requires a fine-grained authorization mechanism that controls what operations an MSP technician is trusted to perform on a *particular* device.

As a first step toward securing MSP customers, we envision a new architecture for MSPs, known as Heimdall, that

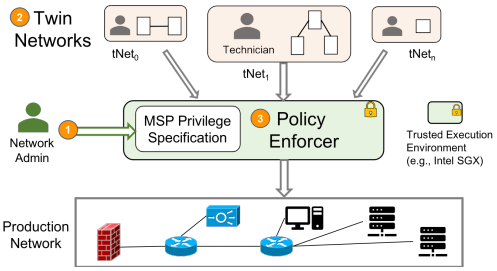


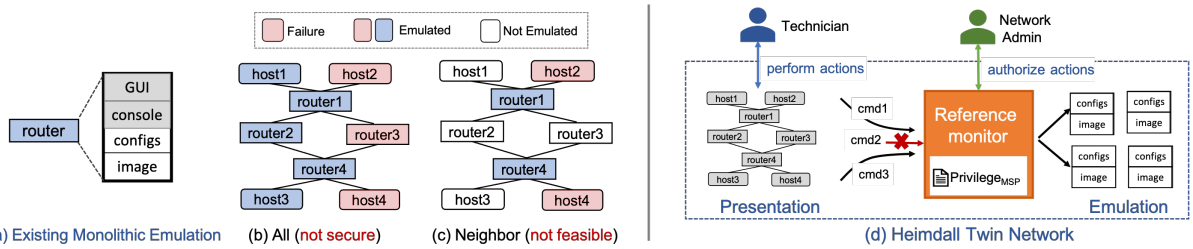

**Figure 4: Heimdall overview: (1) An admin specifies a $Privilege_{msp}$ using our expressive language; (2) A technician works on the isolated twin network; (3) Changes to the twin network propagate to the production network and are verified by the enforcer.**

achieves least privilege by default. Figure 4 shows the three-step Heimdall workflow: (1) An admin uses our expressive domain-specific language to specify a $Privilege_{msp}$ that defines a technician's privileges; (2) For each ticket, a *twin network* — which mimics the production network but is *isolated* to prevent malicious behavior — is created for the technician to modify offline; (3) After the technician resolves the ticket inside the twin network, the *policy enforcer* analyzes any changes before importing them into the production network, to avoid privilege violations. Given this approach to ensuring least-privilege for MSPs, three challenges arise.

**Challenge 1 – Specify fine-grained priviledges efficiently:** While it is trivial to provide a technician either all privileges or none, crafting a *fine-grained* $Privilege_{msp}$ is more difficult. A naive approach is to specify a set of privileges for each network component, but this is tedious and error-prone considering the large number of network components, the diversity of network services (e.g., monitoring, upgrading, troubleshooting), and the dynamic nature of a network (e.g., privileges may need to evolve over the life cycle of a ticket).

**Challenge 2 – Emulate a network securely and faithfully:** Existing emulation tools (e.g, CrystalNet [41], GNS3 [28]) that clone all network elements can expose sensitive data (e.g., an IPSec key). An alternative is to emulate a subset of the network while hiding unrelated components, but differentiating between relevant and irrelevant network elements is nontrivial and has important consequences: missing a relevant element could yield a different failure scenario, while including an irrelevant element increases the attack surface.

**Challenge 3 – Trustworthiness:** For an enterprise to trust Heimdall, the policy enforcer must guarantee that the least-privilege properties are satisfied even in the presence of attacks. Heimdall itself must be secure so that it does not become another launchpad for attacks. The system must audit [40] users' actions and provide tamper-resistant *audit trails*, an important tool in ensuring network policy compliance, that can be reviewed later to analyze a technician's




(a) Existing Monolithic Emulation (b) All (not secure) (c) Neighbor (not feasible) (d) Heimdall Twin Network

**Figure 5: (a) Using monolithic emulators cannot achieve both security and feasibility. E.g., given a ticket reporting that host2 cannot communicate with host4, two options exist: (b) emulate all elements of the production network, or (c) emulate the neighbors of the affected elements (hosts 2 and 4). (d) Our twin network decouples traditional monolithic emulation into presentation and emulation layers, and inserts a reference monitor in between to mediate all actions.**

network modifications. In practice, it is also challenging to import changes into the production network (e.g., updating routers in the wrong order can result in inconsistent behavior).

We also have several practical requirements. An expensive or high-overhead solution would nullify the original motivation for outsourcing. Thus, Heimdall should be low-overhead and require little additional hardware. We also require a solution that does not disrupt network operations or add heavy burdens for network administrators or MSP technicians.

# 4 PROOF-OF-CONCEPT DESIGN

In this section, we elaborate on these design challenges with respect to the privilege specification, twin network, and policy enforcer, and discuss initial ideas to address them.

## 4.1 Privilege Specification

It is nontrivial to define privileges for each network object (e.g., router, switch, interface) given the large number of objects and diverse possible actions. Therefore, a simple yet expressive domain specific language (DSL) is needed. The goal of this DSL is to help an admin create a *fine-grained* privilege specification, known as a $Privilege_{msp}$, efficiently. The $Privilege_{msp}$ is a set of predicates that each correspond to a specific technician action and evaluate to `true`, indicating that the technician is allowed to perform the action, or `false`, if the action is prohibited. For example, {`allow(ip,` $r_1$`)`} specifies that a technician can modify router $r_1$'s IP address but cannot perform other actions (e.g., change its password).

In our current work, Heimdall includes a convenient front-end interface, based on JSON, that builds on the specification DSL from Batfish [37]. We extend Batfish to take privileges for different network resources as inputs as well as provide a framework for translating network policies into our DSL. Thus, the admin can specify both privileges and network policies using the same interface.

## 4.2 Twin Network

To prevent an MSP technician from directly accessing the enterprise production network, in Heimdall, technicians resolve problems on an emulated twin network.

**Strawman solutions:** A natural starting point in building a twin network is to leverage existing network emulation tools, such as CrystalNet [41] and GNS3 [28]. These tools can faithfully emulate the behavior of complex networks. However, they are built atop a *monolithic* model that makes it difficult to achieve both security and feasibility. As shown in Figure 5 (a), an emulated node (e.g., a router or an endhost) often consists of four parts: a GUI, a console, configurations, and an image. These parts are *tightly coupled*, meaning that the user can either emulate an entire node or nothing at all. As a result, this often leads to an *all-or-nothing* solution.

To see this more concretely, consider a ticket describing that host2 cannot communicate with host4 (Figure 5). To resolve this ticket, there are two strawman approaches for using existing emulators. The first (Figure 5 (b)) is to clone the entire production network into the twin network. This enables the technician to resolve the problem, but compromises the least-privilege principle by exposing all nodes to the technician. The second approach (Figure 5 (c)) is to only clone elements that neighbor hosts 2 and 4. This achieves better security by hiding most network elements, but makes it *infeasible* for the technician to solve the problem, as the misconfigured router3 is not accessible. While there are more clever approaches to find relevant elements to emulate (e.g., all elements on the path between hosts 2 and 4), it is challenging to find a *minimal* set that faithfully reproduces the issue. A natural question is *how can we design a twin network that achieves both security and feasibility?*

**Proposed approach:** We observe that an emulated node can be separated into presentation components (GUI and console) and emulation components (configurations and image). The presentation components enable the technician to understand the problem and thus must be accessible. On the other hand, the emulation components contain information that is unrelated to the problem and thus should be restricted. Based on this observation, we propose to *decouple presentation from emulation* and *use a reference monitor to mediate* the communication between the presentation and emulation layers.

Our twin network design shown in Figure 5 (d) consists of three key components: (1) a **presentation layer** that presents the topology to a technician and provides interfaces for them

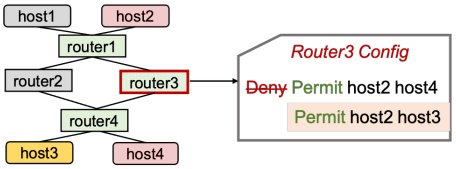



**Figure 6: Benign and malicious actions appear similar.**

to perform actions to resolve a ticket; (2) an **emulation layer** that runs software and configurations to emulate components of the production network, such as routers, switches, and endhosts; and (3) a **reference monitor** that resides in the middle, mediating each request sent from the presentation layer to the emulation layer and ensuring that the $Privilege_{msp}$ is not violated. This design enables a technician to resolve many practical problems while enforcing $Privilege_{msp}$. We discuss the limitations of the twin network in §7.

## 4.3 Policy Enforcer

The twin network enables a technician to resolve a ticket while guaranteeing that their actions do not violate the $Privilege_{msp}$. The policy enforcer performs this verification and applies changes made inside the twin network to the production network. However, this raises practical issues.

Following from the example in §4.2, consider Figure 6, where a technician determines that the reason why host2 cannot communicate with host4 is that an access control list rule on router3 is misconfigured. The technician fixes the problem by changing the relevant rule from `Deny` to `Permit`. However, the technician may maliciously modify another rule that allows host2 to reach sensitive host3, an action that violates a constraint in network policies. It is difficult to detect this malicious action by just restricting allowed *commands*, as the technician uses similar legitimate commands to fix the issue.

**Strawman solutions:** To detect a malicious action, one strawman approach is to continuously verify the network policies in the twin network (i.e., after the technician performs every action). This approach has two drawbacks. First, verifying the policy is time-consuming (e.g., 25 seconds to check 175 constraints) and can significantly slow down a technician's work. Second, it may generate many false alerts, as it is not always possible to ensure that an action is consistent with the network policies (e.g., rebooting a router may temporarily violate reachability). In addition to verified and correctly-applied changes, we must also provide the enterprise with assurance that an MSP's behavior complies with the $Privilege_{msp}$ and network policies. Moreover, in the event that some violations escape the $Privilege_{msp}$, we need forensic audit trails to help identify issues retroactively.

**Proposed approach:** Our proposed policy enforcer has three key modules: (1) a *verifier* that checks the output of the twin network against network policies; (2) a *scheduler* that orders changes and pushes them to the production network; and (3) *auditing* that builds audit trails to be reviewed later. To secure these modules and achieve trustworthiness, our current work leverages Intel SGX enclaves [34] to run the policy enforcer, which provides strong security guarantees (e.g., data integrity) with a small trusted computing base.

# 5 PRELIMINARY EXPERIMENTS

**Setup:** We evaluated two example networks with real configurations: an enterprise network and a university network [37], and Table 1 shows details of each network. We reproduce different types of real-world issues described on StackExchange [29], such as an OSPF issue [9], an ISP reconfiguration [3], and a VLAN issue [1]. We use config2spec [32] to generate network policies from configuration files.

| Network | #routers | #hosts | #links | #policies | lines of configs |
|---|---|---|---|---|---|
| Enterprise | 9 | 9 | 22 | 21 | 1394 |
| University | 13 | 17 | 92 | 175 | 2146 |

**Table 1: Evaluation networks.**

**Pilot study on usability:** We conduct a pilot study using Heimdall to solve real issues for MSP customers, and measure how long it takes for a technician to resolve an issue. We test with three different practical issues each on the enterprise and university networks. To mimic an experienced MSP technician who is comfortable with their tools, one of our authors acts as the technician. We measure the time from receiving a ticket to fixing the issue.

We compare Heimdall against the current approach where a technician is given direct access to the full production network. Under this approach, the debugging process has three steps: (1) the technician connects to the network, (2) performs operations, and (3) saves the necessary changes. On top of these, Heimdall requires three extra steps, namely to generate a $Privilege_{msp}$, set up a twin network, and verify and schedule changes. Our goal is to measure the time overhead of these new steps rather than the expertise of the technician, so to create a "level playing field", the technician performs a prepared list of commands to fix each issue.

Figure 7 shows the results for the enterprise network (we omit the university results due to their similarity). Compared to the current approach, Heimdall adds 28 seconds of latency overhead on average, 15s for a simple issue (ISP reconfiguration), and 42s for a complex issue (VLAN troubleshooting). From the time breakdown for each step, we see that the most time is spent performing operations to resolve the issue.

**Attack surface/feasibility trade-offs:** There is a fundamental trade-off between security (a small attack surface) and debugging feasibility (access to relevant network devices). We measure this trade-off for Heimdall versus the alternatives. We compare with two baseline solutions: (1) *All*, which gives the technician access to all nodes, and (2) *Neighbor*, which gives access to affected nodes and their neighbors only. These represent the two extreme solutions shown in Figure 5.

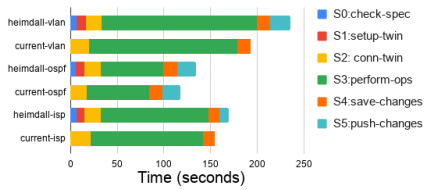



**Figure 7: Time to solve three real issues (vlan, ospf, isp) on an enterprise network.**

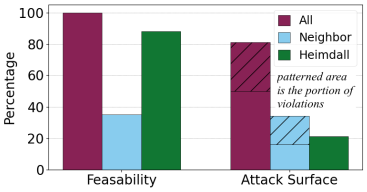



**Figure 8: Feasibility and attack surface for an enterprise network.**

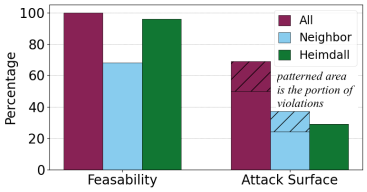



**Figure 9: Feasibility and attack surface for a university network.**

We evaluate two metrics: (1) feasibility (whether the technician can solve the issue) and (2) attack surface (how much of the customer network is exposed). There is no accepted metric to measure attack surface, so we propose a weighted combination of exposed surface and actual violations:

$$\text{Attack\_Surface}\ (\%) = \left(\frac{\Sigma_{n\in\text{nodes}} C_n}{\Sigma_{n\in\text{nodes}} A_n}\cdot 0.5 + \frac{VP}{P}\cdot 0.5\right)\cdot 100$$

where $C_n$ and $A_n$ are allowed and available commands on node $n$, respectively, $VP$ is the number of violated policies, and $P$ is the number of provided policies.

First, we create an issue by bringing down each interface. Then, for each technique, we check whether the technician can access the root cause node (feasibility). Finally, we search all possible commands on accessible nodes, measure potential policy violations, and compute the attack surface. Figures 8 and 9 show that compared to the two baseline solutions, Heimdall significantly reduces the attack surface by up to 39% and 40% for the enterprise and university networks, respectively. Meanwhile, Heimdall achieves feasibility close to that of fully open privileges. This is the "best of both worlds": a small attack surface with only a minor feasibility decrease.

## 6 RELATED WORK

**Least privilege:** Least privilege is a classical security principle [45, 46] to defend against many common attacks (e.g., privilege escalation [44]). Prior work has applied least privilege to host applications [43], mobile applications [36], and operating systems [39]. Our novelty is to revisit it for a new use case: third-party managed network services.

**Network verification:** We extend network verification tools [37] to take MSP privilege specifications as input, as well as leverage their capabilities to verify network policies. These tools alone cannot address our problems.

**Network isolation:** Traditional network isolation techniques (e.g., VLANs, virtual private networks) slice networking resources on top of the production network. In contrast, our goal is to prevent direct access to the production network.

**Network simulation/emulation:** There exists a wealth of network simulation and emulation tools [22, 23, 38, 41, 42, 47] and testbeds [4, 6, 33, 35]. Their focus is on how to faithfully and efficiently emulate a production network, so most tools try to mimic *every* network component, breaking least-privilege and leaking sensitive information.

## 7 DISCUSSION

**Limitations of the twin network:** The inherent shortcomings of emulation mean that the twin network is not a panacea. Fortunately, prior work [41] has shown that emulated networks are effective for debugging misconfigurations and software bugs. For other issues, an *emergency mode* in which the reference monitor bypasses the twin network and sends commands directly to the production network via the policy enforcer could be necessary. How to determine whether a problem requires this emergence mode is an open question.

**Privilege escalation:** The technician's privileges may need to evolve over time, likely escalating from more to less restrictive, as they address an issue. For example, to diagnose a point-to-point connectivity issue, a technician may first be given access to routing policies, but may later require permission to modify firewall rules if the issue is not related to routing. Supporting escalations safely is a necessity [44]. An open question is how to differentiate valid escalations from malicious attempts to subvert the least-privilege model.

**User experience:** Heimdall expects administrators to specify privileges for different tasks. While our DSL aims to provide a simple interface, this still requires understanding intended network behavior. How should resources and privileges be presented and translated into easy-to-understand behavior?

**Beyond legacy networks:** We have focused on addressing configuration issues for legacy networks, but there are many other types of issues (e.g., software bugs) and networks (e.g., SDN) to explore still. We see Heimdall as the first step towards securing MSP customer networks, and a future goal is to optimize our architecture to enable more domains.

**Acknowledgements:** We thank the reviewers for their valuable comments and insightful feedback, and Aayush Agarwal, Ali Hasan Adnan Khaja, and Tharasa Vudi for their help with an earlier version of Heimdall. This work was supported in part by the CONIX Research Center, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA, ERDF Project AIDA (POCI-01-0247-FEDER-045907), in part by the NSF/VMware Partnership on Software Defined Infrastructure as a Foundation for Clean-Slate Computing Security (SDI-CSCS) program under Award No. CNS-1700521.

**Ethics:** This work does not raise any ethical issues.

# REFERENCES


[1] [n. d.]. Access port config. https://networkengineering.stackexchange.com/questions/45650/access-port-config. ([n. d.]).

[2] [n. d.]. APT10 Targeted Norwegian MSP and US Companies in Sustained Campaign. https://www.recordedfuture.com/apt10-cyberespionage-campaign/. ([n. d.]).

[3] [n. d.]. Changing configuration on Cisco router. https://networkengineering.stackexchange.com/questions/58663/changing-configuration-on-cisco-router. ([n. d.]).

[4] [n. d.]. CloudLab website. https://www.cloudlab.us/. ([n. d.]).

[5] [n. d.]. Dropbox Official Site - Work Better, Safer, Together. https://www.dropbox.com. ([n. d.]).

[6] [n. d.]. Emulab.Net - Emulab. https://www.emulab.net/portal/frontpage.php. ([n. d.]).

[7] [n. d.]. How Remote Access Plus works? | ManageEngine. https://www.manageengine.com/remote-desktop-management/help/architecture.html. ([n. d.]).

[8] [n. d.]. How to Implement the Principle of Least Privilege in your MSP. https://myki.com/blog/implement-the-principle-of-least-privilege\-in-your-msp/. ([n. d.]).

[9] [n. d.]. I can't ping the other router using OSPF protocol, Why? https://networkengineering.stackexchange.com/questions/29016/i-cant-ping-the-other-router-using-ospf\-protocol-why. ([n. d.]).

[10] [n. d.]. Kaseya Products Overview - Learn About Our IT Solutions | Kaseya. https://www.kaseya.com/products/. ([n. d.]).

[11] [n. d.]. 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. ([n. d.]).

[12] [n. d.]. LogMeIn Remote Access | Secure Remote Desktop Software. https://www.logmein.com/. ([n. d.]).

[13] [n. d.]. Managed network services - Aviat Networks. https://aviatnetworks.com/services/aviatcare/managed-network-services/. ([n. d.]).

[14] [n. d.]. Managed Network Services | CenturyLink. https://www.centurylink.com/business/managed-services/managed-network-services.html. ([n. d.]).

[15] [n. d.]. Managed Network Services : Fujitsu Global. https://www.fujitsu.com/global/services/infrastructure/network/managed-network/. ([n. d.]).

[16] [n. d.]. Managed Network Services | IBM. https://www.ibm.com/services/network/managed. ([n. d.]).

[17] [n. d.]. Managed Network Services | Verizon Business. https://www.verizon.com/business/products/managed-network-services/. ([n. d.]).

[18] [n. d.]. 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. ([n. d.]).

[19] [n. d.]. ManageEngine - IT Operations and Service Management Software. https://www.manageengine.com/. ([n. d.]).

[20] [n. d.]. N-central architecture. https://success.solarwindsmsp.com/kb/solarwinds_n-central/N-central-architecture. ([n. d.]).

[21] [n. d.]. N-central On-Premises Remote Monitoring Software | SolarWinds MSP. https://www.solarwindsmsp.com/products/n-central. ([n. d.]).

[22] [n. d.]. ns-2 network simulator. https://www.isi.edu/nsnam/ns/. ([n. d.]).

[23] [n. d.]. ns-3 network simulator. https://www.nsnam.org/. ([n. d.]).

[24] [n. d.]. NVD - CVE-2020-7984. https://nvd.nist.gov/vuln/detail/CVE-2020-7984. ([n. d.]).

[25] [n. d.]. Operation Cloud Hopper. https://www.pwc.co.uk/cyber-security/pdf/cloud-hopper-report-final-v4.pdf. ([n. d.]).

[26] [n. d.]. Password and Documentation Management for MSPs and ITSPs | Passportal MSP. https://www.passportalmsp.com/. ([n. d.]).

[27] [n. d.]. Remote Support & Access Software | ConnectWise Control. https://www.connectwise.com/software/control. ([n. d.]).

[28] [n. d.]. The software that empowers network professionals. https://www.gns3.com/. ([n. d.]).

[29] [n. d.]. StackExchange Website. https://stackexchange.com. ([n. d.]).

[30] [n. d.]. Top 3 Risks of Not Having a Privileged Access Management System. https://www.passportalmsp.com/blog/top-3-risks-of-not-having-a-privileged-access\-management-system. ([n. d.]).

[31] [n. d.]. Validating the SolarWinds N-central "Dumpster Diver" Vulnerability. https://blog.huntresslabs.com/validating-the-solarwinds-n-central-dumpster\-diver-vulnerability-5e3a045982e5. ([n. d.]).

[32] Rudiger Birkner, Dana Drachsler-Cohen, Laurent Vanbever, and Martin Vechev. 2020. Config2Spec: Mining Network Specifications from Network Configurations. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. Santa Clara, CA, 969–984.

[33] Xu Chen, Z Morley Mao, and Jacobus Van Der Merwe. 2009. ShadowNet: a platform for rapid and safe network evolution. In *Proceedings of the 2009 conference on USENIX Annual technical conference*. 3–3.

[34] Victor Costan and Srinivas Devadas. [n. d.]. Intel SGX Explained. Cryptology ePrint Archive, Report 2016/086 (2016). https://datatracker.ietf.org/doc/draft-brockners-proof-of-transit/. ([n. d.]).

[35] Dmitry Duplyakin, Robert Ricci, Aleksander Maricq, Gary Wong, Jonathon Duerig, Eric Eide, Leigh Stoller, Mike Hibler, David Johnson, Kirk Webb, et al. 2019. The design and operation of CloudLab. In *2019 USENIX Annual Technical Conference*.

[36] Adrienne Porter Felt, Kate Greenwood, and David Wagner. 2011. The effectiveness of application permissions. In *Proceedings of the 2nd USENIX conference on Web application development*. 7–7.

[37] Ari Fogel, Stanley Fung, Luis Pedrosa, Meg Walraed-Sullivan, Ramesh Govindan, Ratul Mahajan, and Todd Millstein. 2015. A General Approach to Network Configuration Analysis. In *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation (NSDI'15)*. USENIX Association, USA, 469–483.

[38] Nikhil Handigol, Brandon Heller, Vimalkumar Jeyakumar, Bob Lantz, and Nick McKeown. 2012. Reproducible Network Experiments Using Container-based Emulation. In *ACM CoNEXT*.

[39] Maxwell N Krohn, Petros Efstathopoulos, Cliff Frey, M Frans Kaashoek, Eddie Kohler, David Mazieres, Robert Tappan Morris, Michelle Osborne, Steve VanDeBogart, and David Ziegler. 2005. Make Least Privilege a Right (Not a Privilege). In *HotOS*.

[40] Butler W Lampson. 2004. Computer security in the real world. *Computer* 37, 6 (2004), 37–46.

[41] Hongqiang Harry Liu, Yibo Zhu, Jitu Padhye, Jiaxin Cao, Sri Tallapragada, Nuno P Lopes, Andrey Rybalchenko, Guohan Lu, and Lihua Yuan. 2017. Crystalnet: Faithfully emulating large production networks. In *Proceedings of the 26th Symposium on Operating Systems Principles*. 599–613.

[42] Amy Ousterhout, Jonathan Perry, Hari Balakrishnan, and Petr Lapukhov. 2017. Flexplane: An Experimentation Platform for Resource Management in Datacenters. In *USENIX NSDI*.

[43] Niels Provos. 2003. Improving Host Security with System Call Policies. In *USENIX Security Symposium*. 257–272.

[44] Niels Provos, Markus Friedl, and Peter Honeyman. 2003. Preventing Privilege Escalation. In *USENIX Security Symposium*.

[45] Jerome H Saltzer and Michael D Schroeder. 1975. The protection of information in computer systems. *Proc. IEEE* 63, 9 (1975), 1278–1308.
[46] Fred B Schneider. 2003. Least privilege and more [computer security]. *IEEE Security & Privacy* 1, 5 (2003), 55–59.
[47] Amin Vahdat, Ken Yocum, Kevin Walsh, Priya Mahadevan, Dejan Kostić, Jeff Chase, and David Becker. 2002. Scalability and accuracy in a large-scale network emulator. 36, SI (2002), 271–284.