Toward Flexible Auditing for In-Network Functionality

Anonymous Author(s)

1 RESEARCH PROBLEM & MOTIVATION

Networks today increasingly support in-network functionality via network function virtualization (NFV) or similar technologies. Such approaches enable a wide range of functionality to be deployed on behalf of end systems, such as offloading Tor services [5], enforcing network usage policies on encrypted traffic [6], or new functionality in 5G [3]. An important open problem with such approaches is *auditing*. Namely, such services rely on third-party network providers to faithfully deploy and run their functionality as intended, but often have little to no insight as to whether providers do so. To address this problem, prior work provides point solutions such as verifiable routing with per-packet overhead [1], or audits of security practices [4]; however, these approaches are not *flexible*—they are limited to auditing a small set of functionality and do not allow tradeoffs between *auditing coverage* and *overhead*. In this paper, we propose NFAudit, which allows auditing of deployed NFs with a flexible approach where a wide range of important properties can be audited with configurable, low overhead. Our key insight is that the design of simple, composable, and flexible auditing primitives, combined with limited trust (in the form of secure enclaves) can permit a wide range of auditing functionality and configurable—and often low—cost.

2 BACKGROUND & THREAT MODEL

Background: Prior work identified the problem of verifying whether deployed in-network services, policies, and configurations are operating correctly. This includes verified routing [1] and secure logging for detecting policy violations [4]. A key limitation of such prior work is that they require new per-packet fields and processing, increasing bandwidth and CPU overhead. In addition, these solutions do not generalize to auditing a wide range of properties that in-network functionality may want to guarantee.

Threat model: The various parties in our threat model are illustrated in Figure 1. We adopt a threat model similar to the one for SafeBricks [2], as we trust the *customer*, and we do not trust the *provider* and other *NFs* running in the network. The *customer* also trust the NF deployed by the customer in a secure enclave, which consists of code, rules, and/or configurations supplied by an *NF vendor*. Building on this prior threat model, NFAudit includes a controller and an append-only log that is a trusted third party. NFAudit also include agents that are deployed at various points in the system. Agents are trusted by the entities that deploy them, and we assume no collusion with adversarial parties.

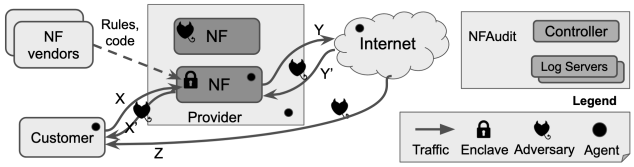

Figure 1: *Different parties that we consider in the deployment environment of NFs. Communications between NFAudit components use secure channels (not shown).*

In our threat model, the adversary is the *provider*, whose goal is to not faithfully provide in-network services specified by the *customer* and/or *NF vendor*.[1] The attacker can modify the software stack anywhere in the provider network (outside the enclave) to inject, modify, reorder, drop, or duplicate packets. These attacks may be transparent or stealthy. The goal of our auditing approach is to detect such attacks in a reliable and flexible way, both in terms of which NF deployment properties can be audited and at what cost in terms of overhead. Further, our approach places evidence of such violations in an append-only log, to assist subsequent investigations of the violations by all parties. To enable this, we propose our system, NFAudit, that serves as an independent observer to gather and publish evidence of NF deployment violations.

3 APPROACH

Goals: The high-level goal of our approach is to enable real-time audits of deployed NFs, to detect provider misbehavior. To make the approach flexible and practical, we include the following subgoals. *First*, we seek to enable reliable audits with limited support from NF providers. We thus assume only that secure enclaves are available to our system for establishing trust in the provider. *Second*, we aim to support a wide range of auditable properties that can be specified by customers and NF vendors. To support such flexibility, we develop composable auditing primitives that rely on a limited set of trusted parties. *Third*, we seek to support explicit trade-offs between auditing fidelity and overhead. To this end, we design our system to support probabilistic audits that can detect violations with high probability and at substantially lower cost than solutions that guarantee to always detect violations.

NFAudit Architecture: NFAudit achieves the high-level goal of real-time auditing by deploying auditing agents at key locations along network paths to be audited. These agents can generate active measurements for auditing (e.g., end-to-end latency measurements) or can passively monitor traffic flowing through the provider network (e.g., for verifying that

[1] A related adversarial model pits the *customer* and/or *NF vendor* against the *provider*. While important, this problem is not our focus.

deployed NFs are traversed by customer traffic). To establish trust for customer-issued audits, we rely on secure enclaves in the provider that can attest to the fidelity of code, data, and computation for agents (and their corresponding NFs) in those enclaves. When an auditing violation occurs, it is essential that the correct auditing data is made available to all parties involved so they can conduct post-hoc resolution. To support this, the agents use secure connections to transmit their auditing data for storage in a distributed append-only log (e.g., a ledger) hosted by an independent third party.

Auditing Primitives: NFAudit supports a wide range of NF audits via composable auditing primitives. They allow customers and NF providers to specify audits as a combination of these common building blocks for many auditable properties in NF deployments. These primitives can address the following auditable properties (and non-exhaustive examples): ① **Packet traversal**: Does a packet travel from node A to B (or along some path P)? ② **NF performance**: Is packet processing time below the agreed latency? ③ **Policy compliance**: Are policy rules such as "ensure packets sent by A never reach B" enforced? ④ **Network performance**: What are the latency, packet loss, bandwidth along path P?

Fidelity/Cost Trade-offs: Prior work ensures high-fidelity auditing by instrumenting every packet that traverses a provider. In NFAudit, we not only support such per-packet audits, but also allow auditing users to reduce this cost at the expense of auditing coverage. We use probabilistic audits, where measurement of auditing properties is performed on one of the packets with probability r (typically random). Assuming that the adversary cannot predict when the audit will occur, such audits place limits on how often the adversary can violate audited guarantees without detection.

4 TRAVERSAL AUDITING EXAMPLE

To make our approach concrete, we now focus on traversal auditing as an example. In this scenario, referring to Figure 1, we assume that the adversary manipulates (at least some of) the packet contents before they entering the NF (X), or after leaving the NF (i.e., along paths {X', Y', Y} or {X', Z}). Our goal is to detect this with high probability and low cost.

Auditing with primitives: We use a primitive that collects per-packet payload hashes at each agent along the path. NFAudit then detects violations of *traversal without modification* by comparing the payload hashes collected by any pair of agents

Evaluation of tradeoffs: We now demonstrate the tradeoffs between auditing coverage and cost, when compared to approaches that use per-packet auditing. For this analysis, we must specify the rate of packets traversing the system. In the case of 40 Gbps link, there will be 22 Mpps if the packet size is 64 B, or 2.75 Mpps if the average packet size is 500 B. We denote the fraction of traffic that the adversary will modify is p and the sampling rate of NFAudit to be r. The probability of detecting such an attack *within one second*: $Pr = 1-(1-p)^{m*r}$. Note that we assume simple random sampling over all packets, though we could adopt more sophisticated sampling methods according to different adversarial models and auditing goals.

System	Setup	2.75 Mpps		22 Mpps	
		Overhead	Pr_{evade}	Overhead	Pr_{evade}
AuditBox	$p=0.001, r=1$ [2]	66.0 MB/s, 2,750 kops/s	0	528.0 MB/s, 22,000.0 kops/s	0
	$p=0.0001, r=1$	"	0	"	0
VRP (OPT)	$p=0.001, r=1$	231.0 MB/s, 5,500 kops/s	0	1,848.0 MB/s, 44,000.0 kops/s	0
	$p=0.0001, r=1$	"	0	"	0
NFAudit	$p=0.001, r=0.001$	0 **KB/s**, 5.5 kops/s	0.0638	0 **KB/s**, 44.0 kops/s	2.76E-10
	$p=0.0001, r=0.01$	0 **KB/s**, 55.0 kops/s	0.0639	0 **KB/s**, 440.0 kops/s	2.79E-10

Table 1: ***Auditing overhead and coverage comparison of AuditBox [4], OPT [1], and NFAudit to detect an attack. "Operation" means MAC/GMAC of the packet payload or pseudo-random function (only for VRP). The MB/s denotes the size of required headers or trailers.***

Importantly, the probability of evading detection is vanishingly small even for low auditing sampling rates. For example, the attacker will evade detection for one second of time with a probability of 2.76E-10 given a packet rate of 22 Mpps, an auditing sample rate of 1/100 packets ($r=0.01$) and a stealthy adversary that manipulates only 1/10,000 packets ($p=0.0001$). Even with a lower packet rate of 2.75 Mpps, the likelihood of evasion for one second is only 0.0638, and this becomes exponentially smaller with additional monitoring time (1.139E-12 with 10 seconds).

Table 1 compares the auditing overhead and coverage of recent approaches and NFAudit. To simplify the setup we do not consider the impact of hops as VRP (OPT [1]) will perform the operations for every hop. The main takeaway is that NFAudit can provide extremely high fidelity (up to nine 9's of coverage) at three orders of magnitude less overhead.

5 CONCLUSION

We proposed a flexible approach to NF monitoring that can achieve flexible auditing goals with configurable cost, and demonstrated its advantages using audits of packet-traversal guarantees. We are building a prototype of NFAudit that uses Intel SGX for a secure enclave, and developing, implementing, and evaluating proposed auditing primitives. Key future work entails building more auditing use cases and evaluating cost/benefit trade-offs for alternative implementation choices.

Ethics: This work does not raise any ethical issues.

REFERENCES

[1] T. H.-J. Kim et al. Lightweight source authentication and path validation. (SIGCOMM '14).

[2] R. Poddar et al. SafeBricks: Shielding Network Functions in the Cloud. (NSDI '18).

[3] 5G Network Slicing - Make Network Slicing easy. Retrieved September 14, 2022 from https://www.ericsson.com/en/network-slicing.

[4] G. Liu et al. Don't Yank My Chain: Auditable NF Service Chaining. (NSDI '21).

[5] M. Reininger et al. Bento: safely bringing network function virtualization to Tor. (SIGCOMM '21).

[6] P. Grubbs et al. Zero-Knowledge Middleboxes. (USENIX Security '22).