Whole Lotta Holes: A Systematic Security Analysis of IOAM

Emilien Wansart
Montefiore Institute
Université de Liège
Liège, Belgium
emilien.wansart@uliege.be

Maxime Goffart
Montefiore Institute
Université de Liège
Liège, Belgium
maxime.goffart@uliege.be

Benoit Donnet
Montefiore Institute
Université de Liège
Liège, Belgium
benoit.donnet@uliege.be

***Abstract*—In Situ Operations, Administration, and Maintenance (IOAM) is an in-band telemetry protocol that enables network devices to embed operational telemetry data directly into the packet headers of user traffic. IOAM is designed to operate within a *Limited Domain*, i.e., an isolated and self-managed network environment protected by strict filtering mechanisms such as firewalls. However, this assumption may not hold in real-world deployments, where misconfigurations can expose the network to a range of security threats—an aspect that has received limited attention from the research community.**

In this paper, we present a systematic security analysis of IOAM within Limited Domain environments, covering deployments such as Internet Service Provider networks and cloud infrastructures. We define a comprehensive threat model and identify concrete attack scenarios, including topology reconnaissance, telemetry poisoning, and denial-of-service attacks. We experimentally demonstrate the feasibility of these attacks in a controlled laboratory environment and discuss their potential impacts. We further propose and evaluate mitigation strategies, culminating in a kernel-level implementation that secures IOAM data through encryption and authentication, thereby providing strong confidentiality and integrity guarantees.

I. Introduction

In recent years, networks have grown significantly in size and complexity due to the development of cloud services, content providers, and an ever-growing user base [1], [2], [3], [4]. Monitoring these networks has become a significant challenge for both operators and equipment vendors. To address these challenges, many network telemetry protocols [5] have been developed, standardized, and deployed. In particular, in-band network telemetry protocols, such as INT [6], In Situ Operations, Administration, and Maintenance (IOAM) [7], *Alternate Marking* [8], and *Active Network Telemetry* [9], enable the collection of telemetry data directly within the data plane by embedding it into packets as they traverse the network (i.e., within packet headers), rather than relying on dedicated probe traffic. This approach reduces measurement overhead and improves the efficiency of data collection. However, despite their ability to handle potentially sensitive operational data, the security aspects of these protocols have received limited attention from the research community.

In-band telemetry approaches inherently rely on three packet-level mechanisms: *encapsulation* (reserving space within packets for telemetry data), *data insertion* (populating that space with measurement data), and *decapsulation* (stripping the telemetry space upon egress). Each of these steps introduces non-negligible risks to the confidentiality and integrity of user traffic. The security implications of such approaches have been studied in the literature. Kong et al. [10] identified several threat vectors specific to INT: a compromised host may trigger a control-plane saturation attack; the absence of enforcement mechanisms leaves the door open to telemetry-evading attacks; queue-occupancy data exposed by INT can be weaponized to orchestrate flooding attacks; and on-path adversaries may intercept INT packets to harvest information for subsequent attacks. Complementing this analysis, Brockners et al. [11] examined the threat landscape of IOAM, highlighting risks such as header and data tampering, injection of forged IOAM headers and data, as well as replay and delay attacks. Despite these efforts, the security of IOAM has not been explored in depth, and potential attack vectors, as well as their corresponding mitigations, have not been systematically studied. Before its widespread deployment, we believe that investigating the security of IOAM and developing a more secure deployment approach are both timely and essential.

In this paper, we systematically explore the security implications of IOAM deployments within misconfigured Limited Domains [12]: network environments that are assumed to be isolated, but where this assumption does not hold in practice. In such settings, the lack of strict boundary enforcement enables a range of threats, including exfiltration of sensitive telemetry information and injection of forged data. Our main contributions are threefold:

- We identify and analyze concrete attack scenarios against IOAM, demonstrating how misconfigured Limited Domains can be exploited to compromise the confidentiality and integrity of telemetry data;
- We evaluate these attacks experimentally, assessing their feasibility and potential impact;
- We propose and assess a cryptographic mechanism that secures IOAM, providing both confidentiality and integrity guarantees for telemetry data.

Our analysis highlights the importance of careful configuration and filtering at the network boundaries to prevent abuse and ensure the integrity and confidentiality of telemetry data.

II. Background

This section focuses on IOAM, first providing a general overview (Sec. II-A), then discussing how telemetry data can be embedded in packets (Sec. II-B).

A. IOAM *Overview*

In Situ Operations, Administration, and Maintenance (IOAM) [7] is a network telemetry protocol that gathers operational and telemetry data directly within the packet as it travels along a path within an IOAM domain. "In-Situ" means that IOAM does not use dedicated telemetry packets but relies on existing data packets. IOAM telemetry can be encapsulated in various protocols, including IPV6 [13] and Network Service Header (NSH) [14]. In this paper, we only consider IPV6 encapsulation, as it is the only encapsulation mode currently publicly available in the Linux kernel [15], [16]. Nevertheless, the security analysis presented in this paper will remain relevant once manufacturers begin commercializing devices that support IOAM in compliance with IETF standards, which has yet to occur.

An IOAM domain, as illustrated in Fig. 1, comprises three categories of nodes. First, *encapsulating* nodes, located at the entry point of the domain (ingress), are in charge of appending the IPV6 Extension Header required to hold the IOAM telemetry (node A in Fig. 1). Second, *transit* nodes insert their telemetry data into the already present IPV6 Extension Header (nodes B and C in Fig. 1). Finally, *decapsulating* nodes, placed at the exit point of the domain (egress), are responsible for stripping the IPV6 Extension Header (node D in Fig. 1).

Nodes insert, update, or remove telemetry data only for packets that belong to their configured *namespace*, as identified by the "Namespace-ID" field in the IOAM header. The namespace mechanism can be used to logically subdivide an IOAM domain and to provide additional context for identifying specific subsets of nodes within the domain.

The IOAM telemetry information consists of multiple IOAM data fields, each of which can be individually enabled or disabled (i.e., via the "Trace-Type" field in the IOAM header) through the configuration of the encapsulating nodes. The IOAM data fields include, among others, hop limit, node ID, IDs of the network interfaces, timestamp, and queue depth.

B. IOAM *Encapsulation Types*

IOAM data are carried in IPV6 Extension Headers using either *inline* mode or *tunnel* mode. When the packet source coincides with the encapsulating node, IOAM data fields may be inserted into a Hop-by-Hop options header, added to the existing IPV6 header (inline mode). Otherwise, since the IPV6 specification [17] forbids in-transit modifications of existing headers, the IOAM data must be carried in an additional IPV6 header. This results in an IPV6-in-IPV6 tunnel across the IOAM domain (tunnel mode). In practice, network operators might ignore this requirement and deploy inline encapsulation even for traffic that only traverses their domain. The reason for doing so is that it reduces packet header overhead (i.e., 40 bytes for the additional IPV6 header) and remains transparent to end-hosts, but it also enables a range of potential attacks. All attacks presented in this paper assume that inline encapsulation is used in the IOAM domain, as they are not applicable to tunnel-based deployments, which provide significantly stronger isolation and robustness. The reason for this is that the outer layer prevents the injection of malicious IOAM data, as it will be carried in the inner header and will not be processed by devices inside the domain. In addition, the outer layer modifies the source IP address, which prevents reconnaissance and spoofing.

IOAM data can be included either as a single Hop-by-Hop option, that every node processes, or as a Destination Option, which is only processed by the destination node [13]. Although IOAM specifies multiple modes of operation [7], [18], this paper focuses on the Pre-allocated Trace Option-Type (PTO), in which encapsulating nodes pre-reserve the space needed in the IPV6 Extension Header. We also consider the IOAM Direct Export (DEX) Option-Type [18], which causes network devices to collect their telemetry data locally and/or export it to external collectors rather than embedding it in the forwarded packets. The PTO is illustrated in Fig. 1, where node A pre-allocates sufficient space for subsequent insertion of IOAM data.

III. Threat Model

An important aspect of the security of IOAM relates to the notion of a *Limited Domain* as defined in RFC 8799 [12]. A Limited Domain can be defined as a bounded network environment under the control of a single administrative entity where devices can rely on consistent local configuration and do not need full Internet-wide discovery or configuration mechanisms. Examples of such Limited Domains include cloud providers and Internet service providers.

The IOAM standards state that it should only be used within Limited Domains and that its data must not leave the Limited Domain boundaries. Filtering methods are expected to be applied to mitigate the risk of data leakage.

A threat model for IOAM is shown in Fig. 2. The depicted scenario considers an external adversary. In this setting, the primary attack surface consists of the boundary routers connecting the Limited Domain (trusted) to the global Internet (untrusted). Telemetry collectors receiving IOAM data via the IOAM DEX mechanism (e.g., over `UDP` or `IPFIX` [19]) may reside inside or outside the IOAM domain, as illustrated by the hatched region in the figure. The *assets* in this model include IOAM-enabled packets, which carry in-band telemetry data, as well as any telemetry exported out-of-band to a collector. The *adversary* is assumed to be an external attacker; however, the model can be extended to insider adversaries or compromised nodes, which are not explicitly shown in the figure. The attacker may operate in a passive manner (eavesdropping on traffic) or in an active manner (injecting or modifying packets). For an external adversary, the *attack surface* is limited to the boundary routers and, when applicable, the external telemetry collectors. In contrast, if an insider adversary or a

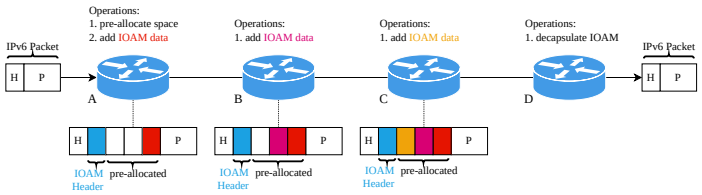

Fig. 1. Generic example of IOAM data insertion. The symbol "H" denotes the IPV6 header, while "P" represents the IPV6 payload.

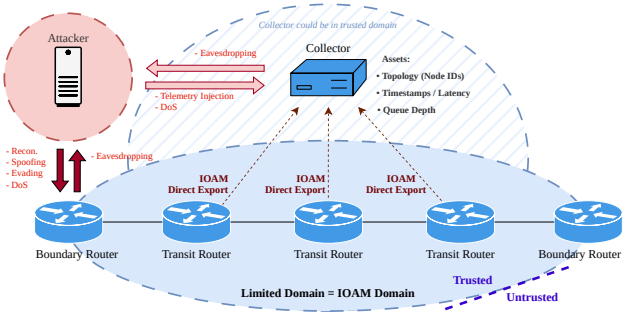

Fig. 2. IOAM threat model (external attacker perspective).

	Encapsulation	In-transit (IOAM operations)	Decapsulation	Collector Export
Confidentiality	—	A1 & A2 Active reconnaissance (ICMP & loopback)	A0 Eaves-dropping	A0 Eaves-dropping
Integrity	A3 Evasion	A4 Poisoning	—	—
Availability	—	A5 & A6 & A7 (Amplified) DoS (PTO & DEX & loopback)	—	—

TABLE I
TAXONOMY OF THE IOAM ATTACKS ANALYZED IN THIS PAPER, BY VIOLATED SECURITY PROPERTY AND BY TARGETED IOAM LIFECYCLE STAGE.

compromised node is considered, the attack surface effectively expands to encompass the entire IOAM domain. Note that in a correctly configured deployment, the telemetry collectors are expected to reside within the Limited Domain, which reduces the exposed attack surface and limits the feasibility of external interception of IOAM data.

IV. IOAM ATTACKS

The primary security concern associated with IOAM comes from its reliance on the concept of Limited Domains. IOAM operations are considered safe only when domain boundaries enforce strict isolation from external networks, as recommended by the IOAM standards [7], [20], [18], [14], [13]. This section examines the security implications that arise when domain-based filtering is absent, misconfigured, or ineffective [21]. The lack of proper filtering at the borders of an autonomous system is a long-standing issue in the context of source-address validation [22], [23], [24]. This facilitates some types of DDoS attacks, particularly reflection and amplification attacks. Therefore, we argue that assuming proper filtering at the border of every autonomous system is an unrealistic assumption.

We analyze all attacks from an external adversary's perspective. Insiders could cause greater damage, but the external setting imposes the most barriers on the attacker and reflects a more realistic threat model.

We organize each presented attack along two dimensions. The first is the *security property* that is violated: confidentiality of the telemetry, integrity of the telemetry view held by the collector(s), or availability of the telemetry inside the IOAM domain. The second is the stage of the IOAM telemetry lifecycle that is abused: the encapsulation stage, the in-transit stage (where each device applies its operation to the IOAM packet depending on its Option-Type and flags), and the decapsulation stage. To these, we add the export stage, which is the optional reporting of the telemetry data to a centralized controller, which would be carried out by the decapsulating node or the DEX-enabled transit nodes (see Sec. II-B). Table I maps the eight attacks onto this grid.

To demonstrate their practical feasibility, all presented attacks (except A0) were replicated in a custom laboratory environment using MSTG [25]. MSTG is a tool that can generate a topology comprising microservices, switches, routers, and firewalls, as a set of IP-connected containers. In the attack figures that follow, each entity (i.e., attacker, router, or collector) corresponds to a `Docker` container. IOAM is deployed using its Linux kernel implementation [15], [16], as there is currently no publicly available RFC-compliant hardware implementation of IOAM.

A. Attack A0: Passive Eavesdropping

Passive eavesdropping arises from an incorrect configuration of the decapsulating node, which fails to remove the IOAM IPV6 Hop-by-Hop Extension Header before forwarding packets outside the IOAM domain. If the domain boundary additionally lacks appropriate egress filtering, IOAM telemetry data may leak to external networks.

An external observer can exploit this information leakage to gain insight into the internal topology and configuration of the IOAM domain, including node identifiers, interface identifiers, timestamps, and other telemetry fields enabled by the operator.

Similarly, IOAM Direct Export (DEX) traffic destined for telemetry collectors may be intercepted when transmitted over untrusted networks or without adequate protection, resulting in the disclosure of the exported telemetry information.

B. Attack A1: ICMP-based Reconnaissance

IOAM can also be exploited for active reconnaissance. Fig. 3 illustrates a scenario in which an attacker sends ICMPv6

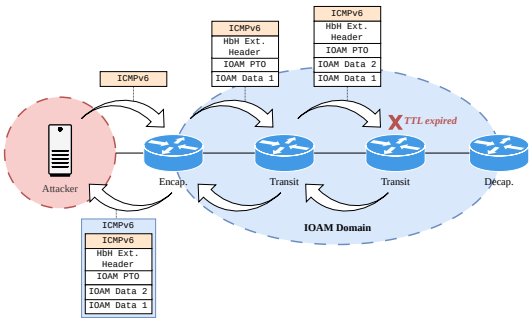

Fig. 3. Reconnaissance of IOAM telemetry data.

`Echo Request` packets with progressively increasing Hop Limit values, mirroring the operation of the `traceroute` tool. When the Hop Limit reaches zero, an intermediate router generates an ICMPv6 `Time Exceeded` message that includes a copy of the original packet without exceeding the minimum IPV6 MTU [26]. As a result, the attacker can retrieve the IOAM Pre-allocated Trace Option-Type (PTO) content, e.g., telemetry data collected from all traversed routers, by parsing the IPV6 Hop-by-Hop option containing the PTO.

This attack is feasible provided that the targeted boundary router (i.e., the encapsulating node) does not filter incoming or outgoing ICMP packets, and that the intermediate routers generate ICMP `Time Exceeded` messages back to the sender.

As with classic port scanning, the extracted telemetry data can subsequently be used to plan and facilitate more targeted attacks. For instance, it allows the attacker to retrieve the IOAM namespace identifier, which represents a subdivision of the IOAM domain. The namespace identifier of the packets must match the one locally configured on the router in order to trigger the processing related to IOAM. As a result, this identifier is required to carry out some attacks, including the denial-of-service attack detailed in Sec. IV-F. Additionally, active reconnaissance aimed at gathering queue-depth measurements of routers in the domain can be used to identify busy nodes. These nodes can then become priority targets for future DoS attacks.

C. Attack A2: Loopback-based Reconnaissance

Another type of active reconnaissance is possible by leveraging the loopback feature of IOAM. When the loopback flag is set, each transit device along the path is instructed to return a truncated copy of the packet to the source, giving it fast access to the telemetry data. The payload of the packets is removed before they are transmitted back to the sender. As this functionality is not currently implemented in the Linux kernel, we develop a custom implementation based on the specification [20].

This attack is similar to attack A1, except that it uses Loopback-enabled packets instead of ICMP. The attacker has to send a single IOAM-enabled packet with an empty PTO with the Loopback flag raised. As a result, the transit nodes on the path will append their telemetry data and send a copy of the packet back to the sender.

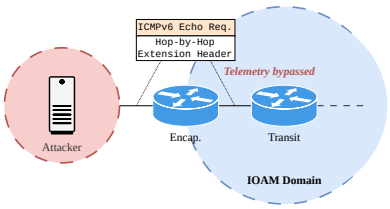

Fig. 4. Telemetry evasion using an empty Hop-by-Hop header.

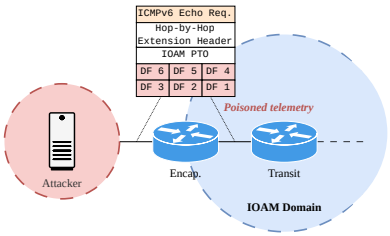

Fig. 5. Telemetry poisoning using pre-crafted IOAM PTO.

This attack is feasible when the boundary router does not filter incoming IPV6 packets carrying IOAM PTOs in Hop-by-Hop options Extension Headers. Recent work has suggested that Hop-by-Hop options may survive on the Internet and cross multiple domains [21], making the attack feasible once a candidate domain has been found. Additionally, the attacker must use the same 16-bit namespace identifier as the intermediate routers; otherwise, the packets will not be processed. Attack A1can be used to gather the namespace ID, or it could also be obtained by brute force.

D. Attack A3: Telemetry Evasion

Fig. 4 illustrates an evasion attack in which an attacker prevents the boundary router (i.e., the encapsulating node) from inserting IOAM telemetry data into the transmitted packets. In the current Linux kernel implementation of IOAM [15], [16], this can be achieved by transmitting packets that already contain a Hop-by-Hop options Extension Header (with any IOAM Option-Type). This occurs because the kernel does not add an IOAM Option-Type to an existing Hop-by-Hop header because doing so is discouraged by the IPV6 specification [17]. As a result, the telemetry collector will not receive any information about the affected traffic, which can lead to blind spots in network monitoring and control.

This attack is feasible when the targeted boundary router does not filter incoming IPV6 packets that contain Hop-by-Hop options Extension Headers.

E. Attack A4: Telemetry Poisoning

Fig. 5 illustrates how an attacker can poison IOAM telemetry data by injecting pre-crafted packets containing a Pre-allocated Trace Option-Type (PTO) with manipulated data fields. Such an attack can disrupt the operations of telemetry collectors that rely on IOAM information for monitoring or control purposes.

In practical deployments, a telemetry collector may perform validation checks to ensure that received IOAM data originate from within the expected domain, for example by inspecting

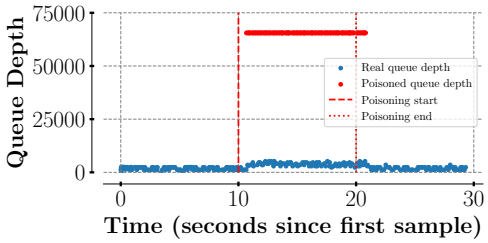

Fig. 6. Queue depth data field poisoning.

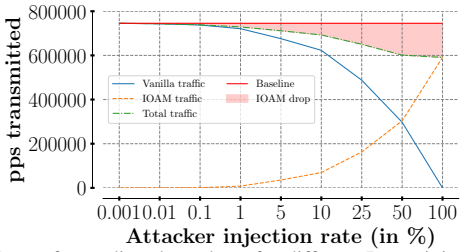

Fig. 7. IOAM forwarding throughput for different IOAM injection rates.

identifying fields such as the namespace ID, node ID, or interface ID. However, these identifiers could be obtained by the attacker via prior reconnaissance as detailed in Sec. IV-B.

This attack is feasible when the targeted boundary router (i.e., the encapsulating node) does not filter incoming IPV6 packets carrying IOAM PTOs in Hop-by-Hop options Extension Headers.

To demonstrate the feasibility of telemetry poisoning, we use MSTG [25] to create a topology consisting of an IOAM domain made of two boundary routers (one encapsulating and the other decapsulating IOAM data) interconnected by a transit router. The decapsulating router uses the `ioam-agent` [27], a telemetry extractor which reports the telemetry data to a collector. During the 30-second experiment, the domain carries a single flow. At time T=10s, the attacker injects pre-crafted packets using the `scapy` [28] library. These poisoned packets contain an IOAM PTO with an artificially high queue depth value of $65,535$ for all nodes along the path. The results, shown in Fig. 6, illustrate the impact of the attack on the telemetry collection for the forwarding node: the forged data appears alongside the legitimate telemetry records, which can lead to incorrect conclusions about network state and potentially trigger inappropriate responses from the telemetry controller. For instance, one can imagine IOAM data being used to efficiently load balance the traffic [29]. In such a situation, the controller may erroneously reroute traffic onto more heavily loaded paths or continually change the routing of traffic, leading to instability in the network.

F. Attack A5: PTO-based Denial-of-Service

In realistic IOAM deployments, the encapsulating nodes do not attach IOAM headers to all incoming packets, as this would lead to significant performance degradation due to the additional processing on the transit devices of the domain [16]. Instead, IOAM should be enabled on a subset of the traffic, according to a parameter called the IOAM *insertion rate*. This parameter is configured on the encapsulating nodes and determines the percentage of the traffic to which an IOAM header is added.

To maximize denial-of-service (DoS) impact, an attacker can inject poisoned IOAM packets carrying an empty Pre-allocated Trace Option-Type (PTO) while enabling all available telemetry data fields, thereby maximizing processing overhead. In addition, by enabling IOAM on all sent packets, the attacker can artificially increase the insertion rate at the encapsulating node, further amplifying the forwarding costs.

This attack is feasible when the targeted boundary router does not filter incoming IPV6 packets carrying IOAM PTOs in Hop-by-Hop options Extension Headers [21]. Additionally, the attacker must use the same 16-bit namespace identifier as the intermediate routers; otherwise, the packets will not be processed. In Sec. IV-B, we describe how to gather the namespace ID using an active reconnaissance technique.

To assess the impact of IOAM-based DoS, we evaluate the performance impact of IOAM on Linux packet forwarding. We use a setup where two devices are connected port-to-port, one serving as a traffic generator, using TREX [30], the other being the Device Under Test[1] (DUT). Fig. 7 illustrates the maximum forwarding throughput in packets per second of the DUT under different attacker injection rates (i.e., the proportion of the attacker's traffic in the domain), with all IOAM data fields enabled. The impact on forwarding capabilities varies significantly, corresponding to a throughput reduction of 3% to 26%. The red area ("IOAM drop" in Fig. 7) represents the throughput loss caused by the additional processing required by IOAM, highlighting the increased impact of IOAM-based DoS compared to regular DoS.

It should be noted that the attacker's injection rate differs from the configured IOAM insertion rate. The insertion rate is the fixed percentage set at the encapsulating nodes and is typically chosen to have only a limited impact on the domain. In contrast, since the attacker enables IOAM on all injected packets, the effective injection rate observed in the network depends on the proportion of the attacker's traffic relative to the legitimate traffic.

G. Attack A6: DEX Amplification DoS

IOAM Direct Exporting (DEX) [31] enables an amplification DoS attack in the case of external reporting, where each DEX-enabled device is configured to report its telemetry data to a fixed telemetry controller address (e.g., via `IPFIX` or `UDP`) when receiving a packet with an IOAM DEX Option-Type. This means that the attacker cannot select an arbitrary victim and is limited to the preconfigured controller(s).

The attack, illustrated in Fig. 8, consists of overwhelming the telemetry controller by transmitting packets that carry an

[1]Specifications: CPU: Xeon 2630 v3 - 8c/16t, RAM: 16GB, NIC: Intel XL710 2x40Gb QSFP+. Single CPU core, using FQ-CoDel.

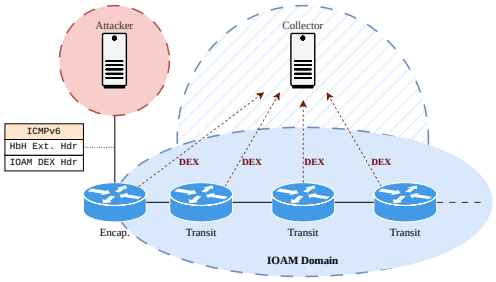

Fig. 8. Amplification attack using IOAM Direct Exporting (DEX).

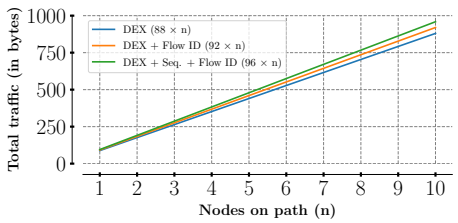

Fig. 9. Total traffic volume received by the controller per packet sent by the attacker, using the DEX amplification DoS attack, depending on the number of routers along the path.

IOAM DEX header with all data fields enabled. This causes all routers along the path to process and generate a telemetry export to the controller, thereby realizing an amplification attack. The amplification factor can be approximated as follows: an attacker sending a packet of size 56 bytes (i.e., 40B IPV6 + 4B Hop-by-Hop + 4B IOAM header + 8B DEX header) can generate n packets, each at least 88 bytes (i.e., 40B IPV6 + 48B IOAM data fields) in size to the controller, where n is the number of devices along the path. Note that 88 bytes represents a lower bound, as telemetry exporters do not realistically export raw data directly over IPV6. In practice, the export is encapsulated in additional protocols (e.g., `IPFIX` over `UDP` or `TCP`), which introduces further overhead and increases the actual amplification factor. Additionally, IOAM DEX supports the usage of two additional fields, being the flow ID and the sequence number of a packet in the given flow. Each of these extensions requires four additional bytes in the source packet. Fig. 9 depicts the total traffic volume received by the controller per packet sent by the attacker, using the DEX amplification DoS attack, depending on the number of routers along the path. As illustrated, the attack impact grows linearly with n, the number of IOAM nodes in the path. A previous study [32], based on large-scale measurements, found that, on average, 65% of the transit paths across an ISP contain at most 5 routers. Subsequent studies [33], [34] revealed that some hops might be hidden due to invisible MPLS tunnels. Thus, the number of traversed nodes inside an ISP (n) might be greater than the five-router upper bound observed for 65% of paths in [32].

In addition to the impact on the controller, the intermediate devices also experience forwarding throughput degradation due to the cost of generating the telemetry exports. For an injection rate of 100%, the forwarding throughput can be reduced by up to 55% [31].

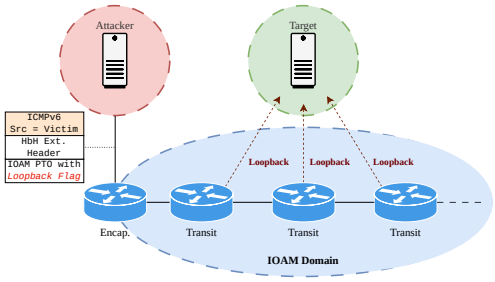

Fig. 10. Amplification attack using the IOAM Loopback flag.

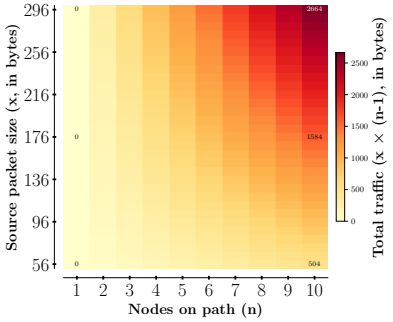

Fig. 11. Total traffic volume received by the target per packet sent by the attacker, using the Loopback amplification DoS attack, depending on the size of the source packet and the number of routers along the path.

H. Attack A7: Loopback Amplification DoS

The loopback feature of IOAM enables another amplification DoS attack, which is illustrated in Fig. 10. It consists of transmitting pre-crafted packets with a spoofed IP source address [23] corresponding to the target, and an IOAM encapsulation with the loopback flag set. Each transit router traversed by the packet generates a copy and sends it back to the spoofed source address, thereby realizing an amplification attack. To maximize the amplification factor, an attacker sending a packet of size $56 \leq x \leq 300$ bytes (i.e., 40B IPV6 + 4B Hop-by-Hop + 4B IOAM header + 8B PTO header + up to 244B for the PTO) can generate $n-1$ packets of the same size to the target, where x is the size of the entire IPV6 header (including the IOAM Hop-by-Hop option) and n is the number of devices along the path.

Fig. 11 depicts the total traffic volume received by the victim per packet sent by the attacker, using the loopback amplification DoS attack, depending on the size of the source packet and the number of routers along the path. As shown, the total traffic volume received by the target can reach ten times the size of the original packet.

V. MITIGATION STRATEGIES

The most effective mitigation against most attacks demonstrated in this paper is the correct enforcement of Limited Domain boundaries through proper traffic filtering (e.g., firewall rules). Specifically, all identified attacks, with the exception of telemetry reconnaissance and evasion, are mitigated when boundary routers of the Limited Domain filter incoming traffic

Attack Type	No HbH Filtering	No HbH + PTO Filtering	Namespace ID Known	ICMP Enabled
ICMP-based Recon.	✗	✗	✗	✓
Loopback-based Recon.	✓	✓	✓	✗
Telemetry Evasion	✓	✗	✗	✗
Telemetry Poisoning	✓	✓	✓?	✗
DoS (PTO-based)	✓	✓	✓	✗
Amplification DoS (Direct Export)	✓	✓	✓	✗
Amplification DoS (Loopback)	✓	✓	✓	✗

TABLE II

FEASIBILITY CONDITIONS FOR IOAM ATTACKS. A CHECK MARK (✓) INDICATES THAT THE CONDITION MUST HOLD FOR THE ATTACK TO SUCCEED. A CHECK MARK FOLLOWED BY A QUESTION MARK (✓?) INDICATES THAT THE CONDITION WILL MOST LIKELY NEED TO BE SATISFIED. A CROSS (✗) INDICATES THAT THE CONDITION IS NOT REQUIRED FOR THE ATTACK TO SUCCEED.

carrying IOAM options in IPV6 Extension Headers. More restrictive filtering policies that drop Hop-by-Hop headers altogether can further strengthen security, particularly against telemetry evasion attacks (see Sec. IV-D).

To mitigate reconnaissance attacks, disabling ICMP `Time Exceeded` messages inside the Limited Domain and/or applying strict outbound ICMP filtering is an effective countermeasure.

Table II summarizes the conditions under which the different IOAM attacks discussed in this paper are feasible, depending on the filtering policies, protocol behavior, and attacker knowledge.

The following list summarizes general IOAM-related mitigation measures, ordered by decreasing effectiveness.

1) Avoid inline IOAM deployment; prefer encapsulation using IP-in-IP tunnels.
2) Filter IPV6 Hop-by-Hop options Extension Headers at boundary routers, in both ingress and egress directions.
3) Disable ICMP `Time Exceeded` inside the IOAM domain to avoid reconnaissance.
4) For IOAM DEX, encrypted channels should be used for communication with external telemetry collectors (e.g., TLS) to avoid data leakage. Enforce rate limiting on telemetry traffic sent to collectors to limit the impact of DoS.
5) Disable IOAM processing on public-facing interfaces.

VI. SECURING IOAM DATA

If boundary enforcement fails, encryption can prevent telemetry leakage, while authentication can prevent the injection of misleading telemetry data. Both protections can be provided by an AEAD scheme [35]. Through encryption, leaked telemetry data cannot be interpreted (e.g., in active reconnaissance attacks, see Sec. IV-B), while authentication prevents adversaries from forging or modifying telemetry information, thereby mitigating poisoning attacks (see Sec. IV-E) as well as IOAM-based DoS attacks relying on crafted data (see Sec. IV-F).

To evaluate this approach, we develop a proof of concept by modifying the existing plaintext IOAM implementation in the Linux kernel [15], [16]. Instead of adding plaintext IOAM data to the IPV6 Hop-by-Hop option at each node, we insert the ciphertext, the cryptographic nonce, and the authentication tag.

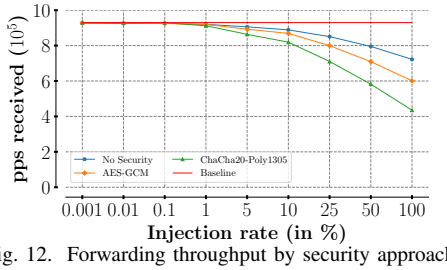

Fig. 12. Forwarding throughput by security approach.

Each node has its own configurable secret key to ensure that the disclosure of the key of a single node does not compromise the entire domain. The key can be rotated at any time using a simple command. Additionally, nonce uniqueness is ensured by maintaining a monotonically increasing nonce for each IOAM namespace on every node, thereby preventing nonce-reuse attacks. The current implementation supports AES-GCM [36] and ChaCha20-Poly1305 [37].

This approach leads to two drawbacks. First, additional processing is required for the AEAD computation. Second, supplementary header space is required for storing the cryptographic nonce (i.e., 12B) and the authentication tag (i.e., 16B) for each node in the domain.

We quantify the computational overhead induced by this approach in Fig. 12, using the same setup as in Sec. IV-F (i.e., port-to-port `TRex`–DUT). Specifically, we measure the forwarding throughput (in packets per second) of a single-core Linux server when using plaintext IOAM ("Vanilla") and when enabling authenticated encryption ("AES" and "ChaCha" in the figure).

For a low injection rate of 1%, which represents a typical deployment scenario, the performance impact is negligible when the plaintext approach is compared with the proposed security solution. However, when IOAM is applied to all packets, which is an unrealistic, worst-case scenario, the performance degradation becomes significant, reaching 16.66% for AES-GCM and 39.58% for ChaCha20-Poly1305.

Additionally, profiling the modified Linux kernel using `perf` confirms that the operations related to the AEAD computation consume substantial CPU time. For this particular measurement, the function `ioam6_sec_aead`, which performs both AEAD encryption and authentication for IOAM, accounts for nearly 10% of the total execution time and represents a

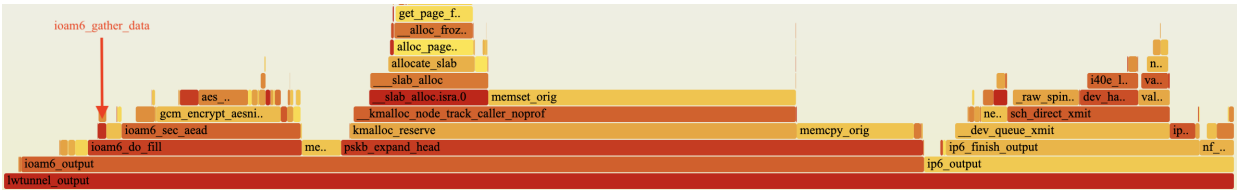

Fig. 13. Flame graph of the functions called by `ioam6_output` in the Linux kernel modified for secure IOAM data. The width of each rectangle is proportional to the execution time among the functions invoked.

significant share of the time spent among the functions invoked by `ioam6_output`, which handles IOAM processing. Yet, memory operations (e.g., socket buffer reallocation or copying the nonce and authentication tag into the packet)—which occur in both the secure and plaintext implementations—account for more execution time than the AEAD operations. These observations are illustrated by the flame graph in Fig. 13, where the width of each rectangle is proportional to the time spent in the corresponding function.

A comprehensive description and evaluation of the kernel-space implementation, including details on nonce management and key configuration, are presented in [38].

VII. Related Work

To the best of our knowledge, this work is the first to systematically analyze the security implications of IOAM deployments.

Among in-band network telemetry protocols other than IOAM — such as INT, AM-PM, and ANT — only INT has been studied from a security perspective. Kong et al. [10] present four attacks against INT and propose a redesign ensuring confidentiality and integrity, later extended with four additional attacks and a refined proposal [39]. Other works enhance INT security through vector homomorphic encryption, enabling machine-learning-based network automation on encrypted data [40], [41], [42], or reduce the resulting computational overhead in switches by switching encryption algorithms at runtime according to the network state [43]. Finally, Zhao et al. [44] store telemetry data in a blockchain to prevent malicious modifications that could harm the network when the data drive automatic reconfiguration.

VIII. Conclusion

While IOAM provides powerful in-band telemetry capabilities that are increasingly relevant for monitoring modern network infrastructures, this paper demonstrates that its mechanisms can be systematically abused when domain boundaries are insufficiently enforced or misconfigured—a threat that has received little attention in the literature prior to this work.

We identified multiple attack vectors, including telemetry poisoning, amplification-based denial-of-service attacks leveraging IOAM Direct Exporting (DEX). Through experimental validation in a controlled laboratory environment, we demonstrated the practical feasibility of these attacks and quantified their impact on both forwarding performance and telemetry infrastructure.

Our findings show that the fundamental barrier against these attacks is the enforcement of the Limited Domain boundary, which, under normal circumstances, filters incoming and outgoing packets carrying IOAM information. However, misconfigurations or insufficient boundary validation can expose the network to significant risks. Furthermore, the absence of encryption and authentication protection for IOAM data fields facilitates forgery and malicious manipulations. To address these vulnerabilities, we designed and evaluated a cryptographic protection mechanism implemented in the Linux kernel, providing confidentiality and integrity guarantees for IOAM data.

Future work includes several promising directions. On the implementation side, a production-ready version of the secure IOAM prototype presented in Sec. VI is a natural next step, including performance optimization at scale and integration with real-world deployment pipelines. Once IETF-compliant hardware implementations become available, the attacks demonstrated in this paper should be evaluated on these new platforms. On the research side, extending the present security analysis to other in-band telemetry protocols—such as INT or *Alternate Marking*—would provide a broader view of the threat landscape. Finally, combining IOAM integrity monitoring with anomaly detection systems could enable real-time identification of telemetry manipulation attempts, opening new avenues for network intrusion detection.

Ensuring that IOAM—and more broadly, in-band telemetry—evolves with security as a first-class design principle will be critical for its safe adoption in production networks.

Acknowledgments

This work has been funded by the CyberExcellence project funded by the Walloon Region of Belgium, under number 211018, and the Feder CyberGalaxia project.

Ethical Considerations

This paper investigates potential security weaknesses in IOAM, including amplification-based DoS attacks. As IOAM is not yet widely deployed, we argue that now is the appropriate time for proactive security analysis, while mitigations can still be incorporated into the standard. Our goal is to identify risks and propose countermeasures before large-scale deployment makes them harder to address.

Reproducibility

All source code, configuration files, and scripts are available at https://github.com/Advanced-Observability/ioam-security.

References

[1] P. Gill, M. Arlitt, Z. Li, and A. Mahant, "The flattening Internet topology: Natural evolution, unsightly barnacles or contrived collapse?" in *Proc. Passive and Active Measurement Conference (PAM)*, April 2008.

[2] A. Dhamdhere and C. Dovrolis, "The Internet is flat: Modeling the transition from a transit hierarchy to a peering mesh," in *Proc. ACM CoNEXT*, December 2010.

[3] H. Zhao and J. Bi, "Characterizing and analysis of the flattening Internet topology," in *Proc. International Symposium on Computers and Communications (ISCC)*, July 2013.

[4] T. Böttger, F. Cuadrado, G. Tyson, I. Castro, and S. Uhlig, "Open Connect everywhere: A glimpse at the Internet ecosystem through the lens of the Netflix CDN," *ACM SIGCOMM Computer Communication Review*, vol. 48, no. 1, January 2018.

[5] M. Yu, "Network telemetry: Towards a top-down approach," *ACM SIGCOMM Computer Communication Review*, vol. 49, no. 1, pp. 11–17, January 2019.

[6] C. Kim, A. Sivaraman, N. Katta, A. Bas, A. Dixit, and L. J. Wobker, "In-Band network telemetry via programmable dataplanes," in *Proc. ACM SIGCOMM Conference Posters and Demos*, August 2015.

[7] F. Brockners, S. Bhandari, and T. Mizrahi, "Data fields for in situ operations, administration, and maintenance (IOAM)," Internet Engineering Task Force, RFC 9197, May 2022.

[8] G. Fioccola, A. Capello, M. Cociglio, L. Castaldelli, M. G. Chen, L. Zheng, G. Mirsky, and T. Mizrahi, "Alternate-marking method for passive and hybrid performance monitoring," Internet Engineering Task Force, RFC 8321, January 2018.

[9] T. Pan, E. Song, Z. Bian, X. Lin, X. Peng, J. Zhang, T. Huang, B. Liu, and Y. Liu, "INT-Path: Towards optimal path planning for in-band network-wide telemetry," in *Proc. IEEE INFOCOM*, April/May 2019.

[10] D. Kong, Z. Zhou, Y. Shen, X. Chen, Q. Cheng, D. Zhang, and C. Wu, "In-band network telemetry manipulation attacks and countermeasures in programmable networks," in *Proc. IEEE/ACM International Symposium on Quality of Service (IWQoS)*, July 2023.

[11] F. Brockners, S. Bhandari, T. Mizrahi, and J. Iurman, "Integrity protection of in situ operations, administration, and maintenance (IOAM) data fields," Internet Engineering Task Force, Internet Draft (Work in Progress) draft-ietf-ippm-ioam-data-integrity-17, March 2026.

[12] B. E. Carpenter and B. Liu, "Limited domains and internet protocols," Internet Engineering Task Force, RFC 8799, July 2020.

[13] S. Bhandari and F. Brockners, "IPv6 options for in situ operations, administration, and maintenance (IOAM)," Internet Engineering Task Force, RFC 9486, September 2023.

[14] F. Brockners and S. Bhandari, "Network service header (NSH) encapsulation for in situ OAM (IOAM) data," Internet Engineering Task Force, RFC 9452, August 2023.

[15] J. Iurman, B. Donnet, and F. Brockners, "Implementation of IPv6 IOAM in Linux kernel," in *Proc. Technical Conference on Linux Networking (Netdev 0x14)*, July 2020.

[16] J. Iurman, E. Wansart, M. Goffart, and B. Donnet, "Mitigating the double-reallocation issue for IPv6 lightweight tunnel encapsulations in the Linux kernel," *arXiv preprint arXiv:2503.14959*, March 2025.

[17] S. Deering and B. Hinden, "Internet protocol, version 6 (IPv6) specification," Internet Engineering Task Force, RFC 8200, July 2017.

[18] H. Song, B. Gafni, F. Brockners, S. Bhandari, and T. Mizrahi, "In situ operations, administration, and maintenance (IOAM) direct exporting," Internet Engineering Task Force, RFC 9326, November 2022.

[19] B. Claise, B. Trammell, and P. Aitken, "Specification of the IP flow information export (IPFIX) protocol for the exchange of flow information," Internet Engineering Task Force, RFC 7011, September 2013.

[20] T. Mizrahi, F. Brockners, S. Bhandari, B. Gafni, and M. Spiegel, "In situ operations, administration, and maintenance (IOAM) loopback and active flags," Internet Engineering Task Force, RFC 9322, November 2022.

[21] J. Iurman and B. Donnet, "The razor's edge: IPv6 extension headers survivability," in *Proc. Passive and Active Measurement Conference (PAM)*, March 2025.

[22] R. Beverly and S. Bauer, "The Spoofer project: inferring the extent of source address filtering on the Internet," in *Proc. USENIX Steps to Reducing Unwanted Traffic on the Internet Workshop (SRUTI)*, July 2005.

[23] R. Beverly, A. Berger, Y. Hyun, and K. Claffy, "Understanding the efficacy of deployed Internet source address validation filtering," in *Proc. ACM Internet Measurement Conference (IMC)*, November 2010.

[24] M. Luckie, R. Beverly, R. Koga, K. Keys, J. A. Kroll, and k. claffy, "Network hygiene, incentives, and regulation: Deployment of source address validation in the Internet," in *Proc. ACM SIGSAC Conference on Computer and Communications Security (CCS)*, November 2019.

[25] E. Wansart, E. Goffart, J. Iurman, and B. Donnet, "MSTG: A flexible and scalable microservices infrastructure generator," in *Proc. IFIP Network Traffic Measurement and Analysis Conference (TMA), Poster Session*, May 2024.

[26] A. Conta, S. Deering, and M. Gupta, "Internet control message protocol (ICMPv6) for the Internet protocol version 6 (IPv6) specification," Internet Engineering Task Force, RFC 4443, March 2006.

[27] J. Iurman, "IOAM agent for Python," 2020, accessed: May 2, 2024. [Online]. Available: https://github.com/Advanced-Observability/ioam-agent-python

[28] SecDev, "Scapy," 2008, [Accessed: April 29, 2024]. [Online]. Available: https://scapy.net/

[29] J. Iurman and B. Donnet, "IPv6 in-situ operations, administration, and maintenance," *Software Impacts*, vol. 6, p. 100036, November 2020.

[30] Cisco, "TRex," 2015, [Last Accessed: April 29, 2024]. [Online]. Available: https://trex-tgn.cisco.com/

[31] M. Goffart, E. Wansart, J. Iurman, and B. Donnet, "Linux kernel support for IOAM direct exporting," in *Proc. Technical Conference on Linux Networking (Netdev 0x19)*, March 2025.

[32] Y. Vanaubel, P. Mérindol, J.-J. Pansiot, and B. Donnet, "MPLS under the microscope: Revealing actual transit path diversity," in *Proc. ACM Internet Measurement Conference (IMC)*, October 2015.

[33] ——, "Through the wormhole: Tracking invisible MPLS tunnels," in *Proc. ACM Internet Measurement Conference (IMC)*, November 2017.

[34] J. Huddleston, M. Luckie, and A. Marder, "Replication: Characterizing MPLS tunnels over Internet paths," in *Proc. ACM Internet Measurement Conference (IMC)*, October 2025.

[35] P. Rogaway, "Authenticated-encryption with associated-data," in *Proc. ACM Conference on Computer and Communications Security (CCS)*, November 2002.

[36] J. A. Salowey, D. McGrew, and A. Choudhury, "AES galois counter mode (GCM) cipher suites for TLS," Internet Engineering Task Force, RFC 5288, August 2008.

[37] Y. Nir and A. Langley, "ChaCha20 and Poly1305 for IETF protocols," Internet Engineering Task Force, RFC 8439, June 2018.

[38] M. Goffart, E. Wansart, and B. Donnet, "Securing IOAM in the Linux kernel: Toward trustworthy in situ network telemetry," in *Proc. Technical Conference on Linux Networking (Netdev 0x1A)*, July 2026.

[39] D. Kong, X. Chen, H. Lin, Z. Zhou, Y. Shen, H. Liu, Q. Cheng, X. Liu, D. Zhang, C. Wu, and M. K. Khan, "Toward security-enhanced in-band network telemetry in programmable networks," *IEEE Transactions on Network and Service Management (TNSM)*, vol. 23, no. 1, pp. 137 – 155, 2026.

[40] X. Pan, S. Tang, and Z. Zhu, "Multilayer network monitoring and data analytics over encrypted telemetry data," in *Proc. IEEE International Conference on Communication Technology (ICCT)*, October 2020.

[41] X. Pan, S. Tang, S. Liu, J. Kong, X. Zhang, D. Hu, J. Qi, and Z. Zhu, "Privacy-preserving multilayer in-band network telemetry and data analytics: For safety, please do not report plaintext data," *Journal of Lightwave Technology*, vol. 38, no. 21, pp. 5855–5866, November 2020.

[42] X. Pan, S. Tang, and Z. Zhu, "Privacy-preserving multilayer in-band network telemetry and data analytics," in *Proc. IEEE/CIC International Conference on Communications in China (ICCC)*, August 2020.

[43] K. Yang, D. Chang, J. Zhang, and L. Niu, "SecPro-INT: Secure programmable in-band network telemetry method," in *Proc. International Conference on Computer and Communications (ICCC)*, December 2024.

[44] Y. Zhao, G. Cheng, and Y. Tang, "SINT: Toward a blockchain-based secure in-band network telemetry architecture," *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 18, pp. 2667–2682, 2023.