StateShield: Real-Time Defenses Against Information Leakage Over Connectionless Protocols

Haibin Li, Qi Li, *Senior Member, IEEE*, Yingying Su, Xuewei Feng, Chuanpu Fu, *Graduate Student Member, IEEE*, and Ke Xu, *Fellow, IEEE*

***Abstract*— Connectionless protocols such as ICMP and UDP are manipulated to construct novel information leakage channels by which attackers can disrupt TCP connections or leak secret information. Existing solutions have mainly focused on repairing vulnerable protocols through OS patches, which are OS-specific and slow to deploy. Other traditional defenses either cannot cover these attacks or are prone to incur unintended dropping of legitimate packets due to the heavily manipulated IP spoofing technique in these attacks. In this paper, we present StateShield, an in-network, real-time defense against state-of-the-art information leakage attacks over connectionless protocols. StateShield can detect and defend against various information leakage attacks without incurring unintended dropping of legitimate traffic, even when attackers heavily spoof the IP addresses of legitimate clients. To achieve that, we propose three indicators that can cover major attack vectors of connectionless information leakage channels and are effective for detecting more than ten attack variants. We design the architecture of StateShield based on programmable switches, with efficient data structures for monitoring and on-demand defense components in the data plane. We develop two novel defense components to mitigate UDP and ICMP-based information leakage channels automatically while achieving minimal unintended dropping of legitimate packets. Our extensive experiments show that StateShield can effectively mitigate more than ten attack variants in real time without hurting the services over legitimate connectionless packets, and the defense provided by StateShield is robust under high-intensive background traffic over connectionless protocols.**

***Index Terms*— Defenses, information leakage, connectionless protocols, side channel.**

Received 18 October 2023; revised 10 June 2024 and 8 December 2024; accepted 11 February 2025; approved by IEEE TRANSACTIONS ON NETWORKING Editor Y. Liu. This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFB3102301, in part by the National Science Foundation for Distinguished Young Scholars of China under Grant 62425201, in part by the Science Fund for Creative Research Groups of the National Natural Science Foundation of China under Grant 62221003, in part by the Key Program of the National Natural Science Foundation of China under Grant 62132011, and in part by the National Natural Science Foundation of China under Grant U22B2031. *(Corresponding author: Ke Xu.)*

Haibin Li is with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China, and also with the College of Information and Communication, National University of Defense Technology, Changsha 410073, China (e-mail: lihb15@tsinghua.org.cn).

Qi Li is with the Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing 100190, China (e-mail: qli01@tsinghua.edu.cn).

Yingying Su is with the Zhongguancun Laboratory, Beijing 100094, China (e-mail: suyy@mail.zgclab.edu.cn).

Xuewei Feng, Chuanpu Fu, and Ke Xu are with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: fengxw06@126.com; fcp20@mails.tsinghua.edu.cn; xuke@tsinghua.edu.cn).

This article has supplementary downloadable material available at https://doi.org/10.1109/TON.2025.3543734, provided by the authors.

Digital Object Identifier 10.1109/TON.2025.3543734

I. INTRODUCTION

IN RECENT years, connectionless protocols such as UDP and ICMP, have repeatedly emerged as new attack vectors for constructing novel information leakage attacks [1], [2], [3], [4]. Unlike TCP, connectionless protocols have no state machine to maintain the packets' state, making it difficult to verify the legitimacy of connectionless packets, especially for the legitimacy of valid and acceptable connectionless packets, such as ICMP echo requests (*i.e.*, ping), DNS requests and QUIC initial packets over UDP. End hosts that receive valid connectionless packets, *e.g.*, ping, will likely send back responses correspondingly. Such properties of connectionless protocols allow attackers to easily lure targets to send out packets, leading to a change in the state of shared resources in targets (*e.g.*, IPID counter increasing or ICMP rate limit triggered). By sending packet sequences to probe the target and observing the resulting state changes in shared resources within the targets (as depicted in Figure 1), attackers can exploit information leakage channels (ILCs [1]) to extract leaked information. This allows them to craft sophisticated and stealthy attacks that are highly challenging to mitigate.

ILCs over connectionless protocols can cause severe harm to network security, such as compromising communication security, breaking network isolation, leaking confidential or private data, etc. For example, Klein [1] uncover a series of covert channels over connectionless UDP and ICMP that can result in leakage of confidential information via a firewall, exfiltration via a DMZ host, exfiltration via co-resident containers, etc., affecting various OSes such as Linux, Windows, MacOS, and OpenBSD, etc. For another example, Feng et al. [2] uncover a series of network side channels over ICMP that enable attackers to infer elements of TCP connections (*e.g.*, port number, TCP sequence number, etc.) for TCP hijacking. Those attacks are demonstrated to be effective across multiple OSes in the real world, posing severe threats to the security of protocols and systems.

Connectionless ILCs present several distinct characteristics and challenges for defenses, setting them apart from traditional attacks. Firstly, unlike disruptive and volumetric DDoS attacks, connectionless ILC attacks are highly stealthy with low bandwidth consumption and relatively low packet transmission rate, making traditional threshold-based filtering less effective in mitigating them. Secondly, ILC attackers may employ a multitude of spoofed source IP addresses to probe the target, making it extremely challenging to trace the origins

[1]We refer to covert channel (CC) and side channel (SC) as information leakage channels (ILCs).

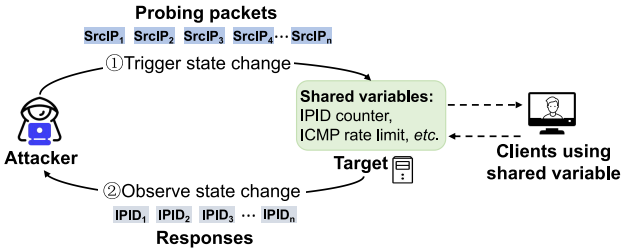


Fig. 1. Connectionless ILC attacks.

of the attacks. Thirdly, valid connectionless packets play a crucial role in constructing the channel, resulting in difficulty differentiating between attack packets and legitimate connectionless packets. This further complicates the defense process, making it challenging to identify and filter out malicious traffic from legitimate traffic accurately.

Existing solutions are inadequate for effectively defending against ILC attacks. The defacto solutions to ILCs proposed by researchers have mainly focused on patching vulnerable protocols through OS updates (e.g., [1] [2]). However, vulnerability patching depends on OS vendors' efforts and is slow to take effect. Other existing detection and defense methods available for addressing this problem also have their limitations. They either do not cover ILCs or are susceptible to the unintended dropping of legitimate packets. As for detection, anomalous traffic detection methods such as [5], [6] only classify the traffic as normal or abnormal without identifying the accurate types of attacks. This lack of granularity hinders the implementation of fine-grained defense mechanisms against ILCs. Regarding defense, traditional approaches based on filtering, rate limiting, or other dropping strategies are unsuitable for ILCs due to their stealthy nature and the heavy utilization of IP spoofing in ILCs. Attackers can even spoof the source IPs of legitimate clients to trigger filters, leading to the unintended dropping of legitimate packets. Consequently, this can cause service degradation for specific legitimate clients (*e.g.*, the disruption of ping services). Recent defenses utilizing programmable switches [7], [8], [9], [10], [11] do not cover connectionless ILCs.

In this paper, we aim to mitigate state-of-the-art connectionless ILCs and defend against the major attack vectors that can be repeatedly manipulated for ILC attacks. Given the limitations of the existing solutions, we argue that an ideal defense against state-of-the-art ILCs should be real-time, OS-agnostic, and burden-free for end hosts, while minimizing unintended disruption to legitimate packets. To this end, we propose StateShield, an in-network, real-time defense against information leakage attacks over connectionless protocols based on programmable switches. Unlike vulnerability patching solutions, StateShield aims to actively monitor ILC attack events in target hosts efficiently and provide real-time protection to the hosts under attack. Furthermore, unlike the classic defense ideology that works by detecting the attack source, filtering malicious packets, or whitelisting benign traffic, StateShield works by identifying the ILC-related state of the target hosts in real-time and thwarting the attacks by obfuscating the manipulated state of the target host to destroy the leakage channel. The protection provided by StateShield is OS-agnostic and burden-free for end hosts, without requirements of modifications on end hosts, and with minimal side effects for legitimate packets.

To make such defense realistic, we propose the following designs:

(1) **Small set of representative indicators for monitoring various ILC variants.** We systematically investigate the potential ILC attack vectors based on connectionless protocols and identify three indicators that can simultaneously cover various ILC attack variants, which enables a good balance between attack coverage and monitoring overhead.

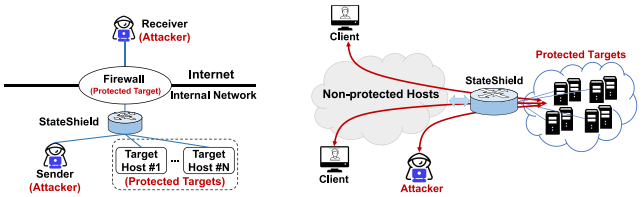
(2) **On-demand and real-time defense based on accurate identification of victim target.** At the core of the StateShield architecture, we design an ILC Event Monitor with efficient data structures for detecting ILC attack victims, and state-obfuscation-based defense components that are activated on demand to protect these targets under attack. The precise identification of ILC victims allows the protected subnet to have real-time awareness of the ILC attack situation, while the on-demand defense mode minimizes the potential interference of defense with normal traffic. Both detection and defense are designed to run in the data plane in real time.

(3) **Mitigating attacks by obfuscating the state of target hosts while maintaining connectionless services.** We develop two novel defense components, *i.e.*, Dynamic Address Mapper (DAM) and ICMP Reply Agent (IRA), which can mitigate major connectionless ILCs based on state obfuscation without dropping packets. Specifically, DAM is designed to mitigate UDP-based ILCs by dynamically mapping the large source address space of the UDP probing packets into a smaller but uncertain space on the fly, which prevents attackers from triggering and observing predictable state changes. DAM also ensures the continuity of service for potential legitimate UDP requests that may be mixed with the attack traffic by forwarding the risk-eliminated packets to the target. For ICMP-based ILCs, IRA mitigates the attacks by temporarily segregating ILC victims from ICMP echo request probing packets. IRA also maintains the service of ping for potential legitimate clients by switch-generated reply messages on behalf of the target under attack. Both DAM and IRA protect target hosts by disrupting the attacker's expected state changes and confusing the attacker through state obfuscation.

We implement StateShield in the Barefoot Tofino2 switch using P4 language. Our extensive experiments on a series of state-of-the-art ILC attacks show that StateShield can effectively detect and mitigate the ILC attack without causing unintended dropping of legitimate packets. The defense of StateShield is real-time, robust, and incurs neglectable latency.

Contributions. Our main contributions are the following:

- Based on the empirical study of attack vectors manipulated by state-of-the-art connectionless ILCs, we propose three indicators effective for detecting more than ten variants of connectionless ILC attacks. By monitoring these indicators, StateShield can simultaneously cover various ILC attacks across multiple OSes with a moderate monitoring scope (*i.e.*, overhead), providing a basis for triggering fine-grained mitigations.
- We propose StateShield, a novel in-network state-obfuscation-based defense architecture based on programmable switches, enabling switches to protect hosts



(a) Covert Channel Scenario. (b) Side Channel Scenario.

Fig. 2. Threat model.

from ILC attacks in real time. In StateShield, the ILC Event Monitor module monitors the state of hosts, and the state-obfuscation-based defense components thwart the detected attacks on-demand simultaneously. The defense of StateShield is OS-agnostic and end-host-transparent.

- We design two novel defense components (*i.e.*, DAM and IRA), which can effectively mitigate ILCs. These defense components work by obfuscating the state of the manipulated shared resources in the target host, which can destroy the channels constructed by attackers without dropping packets, thus preserving the services for potential legitimate connectionless packets.
- We develop a StateShield prototype using the Tofino2 switch, making it the first defense system capable of mitigating state-of-the-art connectionless ILC attacks. Through extensive evaluation on over ten ILCs, StateShield demonstrates high AUC and F1 scores for detection, while effectively and robustly defending against attacks without adverse impact on legitimate traffic.

II. Background and Motivation

In this section, we introduce connectionless ILCs and the limitations of traditional defenses.

A. Emerging Threats of Connectionless ILCs

Connectionless ILCs of our problem scope include network covert channels and side channels over ICMP and UDP.

Connectionless Protocols. Unlike connection-oriented TCP [12], both ICMP [13] and UDP [14] are connectionless protocols. Attackers can easily craft ICMP and UDP packets to probe any targets for malicious purposes. Since connectionless packets have no states to maintain the connection context, it is very difficult for stateful firewalls to determine the legitimacy of connectionless packets, especially valid and acceptable requests such as ping over ICMP, DNS request or QUIC initial packet over UDP, etc.

Connectionless Information Leakage Channels. In our context, *information leakage channels (ILCs)* refer to network covert channels and side channels that attackers can leverage to leak unauthorized secret information by manipulating network protocols. In particular, we focus on connectionless ILCs, *i.e.*, ILCs over connectionless protocols (*e.g.*, [1], [2], [3], [4]), which are novel threats that distinguish themselves from traditional ILCs. Unlike traditional ILCs (*e.g.*, network covert timing and storage channels [15], [16], [17], [18]) that are extensively studied [11], [19], [20], [21], connectionless ILCs are new types of attacks that existing solutions cannot address them adequately (see Section II-B). Typical examples are listed in Table I.

The procedures of connectionless ILC attacks are illustrated in Figure 1. Typically, connectionless ILCs are constructed with bursty or continuous packet sequences over connectionless protocols, crafted to manipulate the state of shared resources (*e.g.*, variables such as IPID counter, ICMP rate limit, etc.) in specific targets. Note that the target may also use these shared variables to maintain the state of its sessions with third-party clients, and the attacker's goal is to infer the communication state between the target and third-party clients. In general, the attacker leverages two basic operations to construct the ILCs. First, the attacker sends sequences of probing packets (with source IPs spoofed at the attacker's discretion) over UDP or ICMP to trigger specific state changes of the shared resources in the targets. Next, the attacker sends one or several packets to the target and then observes the state of the shared variables in the target by examining the received responses from the target. Note that in this operation, the attacker may use another receiver host controlled by the attacker as illustrated in the covert channel case in Figure 2(a), or alternatively, use its real IP as illustrated in the side channel case in Figure 2(b). By repeating and flexibly combining these two basic operations, attackers can arbitrarily trigger state changes and observe state changes of the shared variables in the target so as to infer the communication state between the target and third-party clients. Based on that, the attacker can establish a stable channel that enables the leakage of information related to the communication state between the target and third-party clients. Note that each packet in ILC-related packet sequences may appear no different from a normal connectionless packet, but collectively, these packets can build up into attacks with unexpected and severe consequences.

State-of-the-art connectionless ILCs have been shown to be feasible in reality with severe consequences. For example:

- *Covert Channels over UDP and ICMP*. Klein [1] exploits the vulnerabilities of the connectionless IPID generation algorithms in mainstream OSes and proposes a series of novel ILC attacks over UDP and ICMP (*e.g.*, LE, LF, WE, *etc.*, as shown in Table I). The attacker controls a sender and a receiver to cooperatively send ICMP or UDP probing packets to the target host, and the information is leaked through the manipulated IPID counter states. Those attacks can lead to severe consequences with wide-ranging impacts, *e.g.*, leaking private and confidential data, penetrating firewalls and breaking the isolation status between networks, etc., posing threats with serious and unexpected security consequences across multiple popular OSes including Linux, Windows, OpenBSD, *etc.*
- *Side Channels over ICMP.* Feng et al. [2] proposes a series of IPID side-channel attacks (VCD, CD, SI, AI in Table I), which can be manipulated to infer elements of TCP connections, enabling off-path attackers to launch TCP hijacking against arbitrary Linux servers. The author demonstrates that those attacks can be used to detect and tear down an SSH connection and manipulate web applications or BGP routing tables, causing serious consequences.

B. Limitations of Traditional Defenses

We investigate the limitations of traditional defense in the scenario of connectionless ILCs and find that existing defenses are inadequate for defending against connectionless ILCs.

TABLE I
ILC ATTACKS COVERED BY STATESHIELD PROTOTYPE

Attack	Target OS	Manipulated Protocol	Manipulated Attack Vectors	Channel Type	Attack Scenarios (Attack Variants)
Linux Exfiltration (LE) [1]	Linux	UDP/IPv4	-Spoofed UDP probing -Sender & receiver in sync	CC	Exfiltration through Linux Targets via DMZ, Container(variants: LE-1,LE-2,LE-3,LE-4 [1])
Linux Firewall Piercing (LF) [1]	Linux	ICMP/IPv4	-Spoofed ICMP echo probing -Sender & receiver in sync	CC	Exfiltration through Linux Firewall (variants:LF-1, LF-2 [1])
Linux Alias Resolution (LA) [1]	Linux	UDP/IPv4	-Spoofed UDP probing -Sender & receiver in sync	CC	Linux Host Alias Resolution (variants:LA-1a,LA-1b,LA-1c [1])
Windows Exfiltration (WE) [1]	Windows	UDP/IPv4	-Spoofed UDP probing -Sender & receiver in sync	CC	Exfiltration through Windows Targets vis DMZ(variants:WE-1,WE-2 [1])
Windows Alias Resolution (WA) [1]	Windows	UDP/IPv4	-Spoofed UDP probing -Sender & receiver in sync	CC	Windows Host Alias Resolution (variants: WA-1a, WA-1b [1])
OpenBSD Firewall Piercing (OF) [1]	OpenBSD	ICMP/IPv4	-ICMP echo probing -Sender & receiver in sync	CC	Exfiltration through OpenBSD Firewall (variants: OF-1,OF-2,OF-3 [1])
Victim Client Detection (VCD) [2]	Linux	ICMP/IPv4	-Forged ICMP Frag Needed -Spoofed ICMP echo probing	SC	Leakage of Victim IP for Hijacking [2]
Connection Detection (CD) [2]	Linux	ICMP/IPv4 TCP/IPv4	-ICMP echo probing -TCP SYN/ACKs probing	SC	Leakage of Connection Port for Hijacking [2]
Sequence Inference (SI) [2]	Linux	ICMP/IPv4 TCP/IPv4	-ICMP echo probing -TCP RST packets probing	SC	Leakage of SEQ Number for Hijacking [2]
ACK Inference (AI) [2]	Linux	ICMP/IPv4 TCP/IPv4	-ICMP echo probing -TCP ACK packets probing	SC	Leakage of ACK Number for Hijacking [2]

Vulnerability patching. Vulnerability patching, i.e., repairing vulnerable protocols at OS kernel level, is a natural solution proposed by researchers to address side channels [2], [3], [4] and covert channels [1] vulnerabilities. However, vulnerability patching is OS-specific without generality, and its effectiveness highly depends on the cooperation of OS vendors. As a delayed remedy, the time for the vulnerability patching to take effect is usually long. Even worse, some vendors did not take any actions for a long time after being notified of the vulnerability [1]. Thus, vulnerability patching is not enough to mitigate ILCs adequately.

One typical example is IPID randomization at OS kernel level. Although RFC6274 [22] proposed IPID randomization as a possible security improvement in 2011, a large-scale measurement by Salutari [23] 7 years later showed that only 2% of hosts use random IPID, which indicates that IPID randomization is not widely adopted by OS vendors and the real-world adoption rate of randomized IPID remains low and grows very slowly. In addition, IPID randomization risks disrupting the intended protocol semantics of the IPID field, which can lead to a higher rate of malformed or dropped packets due to possible IPID collisions according to [1].

Filtering-based defenses. Filtering-based defenses (*e.g.*, IP blocklisting, filtering ICMP traffic, etc.) are frequently used in defenses against network attacks, *e.g.*, traditional scanning [24] and DDoS [7], [8]. By tracking the behavior of source IPs by methods such as Bro [24], malicious source IPs will be flagged and added to the IP blocklist for filtering. However, filtering-based defenses are not suitable for connectionless ILCs, because connectionless ILCs are often launched over ICMP and UDP packets with spoofed source IP addresses, and the attack source is difficult to trace. In ILC attacks, the intensity of traffic sent from each spoofed source IP can be very low (e.g., one or several packets per spoofed IP), which makes it fall far below the threshold for adding those IPs to the blocklist. Worse still, since attackers can spoof arbitrary source IP addresses (including the IP addresses of legitimate clients), attackers may easily induce filtering-based defenses to filter traffic originating from legitimate clients by spoofing legitimate clients' packets for probing.

Drop-on-detection defenses based on anomaly detection. Existing ML-based anomaly detection methods [5], [6] primarily focus on detecting attacks and typically only propose dropping packets or flows marked as anomalous as a defense strategy [5]. This approach can lead to unintended dropping of legitimate packets, particularly in the context of ILC attacks, where attacker can spoof the active legitimate client's IP to probe the target. Despite existing ML-based detection techniques being capable of identifying anomalous traffic with spoofed source addresses, they overlook a critical aspect in novel ILC scenarios: attackers deliberately spoofing the addresses of active legitimate users to send probing packets to the target host, thereby inducing the defense system to trigger filtering mechanisms against legitimate users. For instance, legitimate packets may be mistakenly dropped when they share the same flow as spoofed attack packets. Our analysis in Appendix A demonstrates this issue.

III. PROBLEM STATEMENT

In this section, we introduce threat model, attack coverage, deployment scenarios, and defense Challenges.

Threat Model. Our threat model is generally consistent with the threat models of both [1] and [2]. We consider both side channel and covert channel scenarios as shown in Figure 2. We focus on connectionless ILC attacks, which involve excessive probing over connectionless ICMP and UDP against victim destinations (i.e., the targets). The goals of the adversary can be diversified, *e.g.*, probing the state of shared variables of the target (*e.g.*, IPID counter, ICMP rate limit, etc.) to construct various covert-channel or side-channel attacks listed in Table I. The adversary's goals can also be triggering specific states to launch other potential attacks with excessive ICMP or UDP-based probing. Our threat model takes into account powerful adversaries with the following capabilities: (1) The attacker may have the capability to spoof arbitrary source IP addresses. (2) The attacker may have compromised some hosts in the internal network of the protected targets.

We assume that StateShield has access to ICMP and UDP traffic data collected by operators to learn the normal behavior of ICMP and UDP, from which StateShield can

learn appropriate thresholds for attack detection. We assume the programmable switch that runs StateShield cannot be compromised by the adversary.

Attack coverage. StateShield's primary focus is to enable real-time mitigation against connectionless ILCs while incurring minimal unintended dropping of potential legitimate packets. As an initial step, our prototype can mitigate more than 10 state-of-the-art ILCs (as listed in Table I) which manipulate the attack vectors of excessive ICMP or UDP probing.

Note that the following attacks are not in our scope: (1) Attacks that do not rely on attack vectors over ICMP and UDP; (2) Attacks that can be conducted with only a single or very few numbers of packets, which present no anomalous patterns at all. (3) Attacks that are volumetric, which fall into the category of DDoS.

Deployment Scenario. We envision StateShield to be deployed on the border switch of an internal network or demilitarized zone (DMZ). The StateShield switch is connected to protected targets such as hosts, firewalls, virtual machines, and containers. Typical deployment scenarios are shown in Figure 2. Note that the operator can specify all subnet machines or only add those leakage-sensitive machines as StateShield-protected targets, *e.g.*, servers hosting sensitive services or confidential data. Since firewalls are already known as victims of information leakage channel attacks [1], they are also included as the protected targets of StateShield.

Challenges. Challenge I: The Stealthy Nature of ILCs. ILCs are carried out with low bandwidth consumption and low rates, which are very stealthy and differ significantly from volumetric attacks (*e.g.*, DDoS). The stealthy nature of ILCs makes them challenging to detect.

Challenge II: Indeterminate Legitimacy of Connectionless Packets. Connectionless ILCs are difficult to defend because they use valid ICMP and UDP packets for probing, making it nearly impossible to determine packet legitimacy.

Challenge III: Unidentifiable Attack Source and Risk of Unintended Dropping of Legitimate Packets. Due to the heavy utilization of IP spoofing, attackers conceal their locations highly covertly, making defenses targeted at attack sources ineffective. Even worse, ILC attackers can spoof legitimate users' packets to probe the target, increasing the risk of the defense mistakenly dropping packets from legitimate clients.

IV. StateShield Design Overview

A. Design Principles

Core Idea. Our core idea is to design a defense system deployed on programmable switches, enabling them to monitor potential ILC attacks and launch real-time protection measures, thus protecting multiple hosts simultaneously and achieving end-host-transparent defense. Taking into account the limitations of existing solutions and the challenges posed by connectionless ILCs, we present StateShield, a novel defense based on efficient monitoring of ILC attack events and on-demand state-obfuscation-based defenses without dropping packets. Upon detecting ILC attack events, StateShield obfuscates the states corresponding to the ILC packet sequences, thereby destroying the attacker's channel. Unlike OS vulnerability patching solutions, this state-obfuscation-based defense confuses the attacker by making the target's state changes uncertain, blurry, and unpredictable, thus preventing the attacker from obtaining expected information.

To design defenses tailored for ILCs, we envision the following key properties: (1) The defense should minimize the unintended dropping of legitimate packets. This ensures that security operators do not have to worry about potential negative consequences, such as dropping legitimate traffic, even in the presence of false positives. (2) The defense should actively maintain normal services for legitimate packets during the defense process. We argue that defenses preserving service continuity (i.e., service-preserving) are particularly suitable for ILC scenarios. ILCs are stealthy, non-volumetric, and can be launched easily by attackers at low cost. In such cases, high-cost defenses that indiscriminately discard traffic, like those used to mitigate DDoS attacks, are not optimal. Instead, defenses that can effectively mitigate attacks while remaining friendly to legitimate traffic are more desirable.

Intuition and Observations for State-obfuscation-based Defenses. State-obfuscation-based defense originates from this observation: the success of an ILC attack depends on triggering specific state changes of the shared variable in the target by sending ILC probing packet sequences. By designing defenses that can obfuscate the states of the shared variable in an unexpected way, we can eliminate the risk factors in the ILC packets, confusing the attacker and preventing them from inferring meaningful information (*i.e.,* breaking the conditions for constructing ILCs).

In addition, we observe that packets of connectionless ILCs can be classified into two categories: packets with data-exchange purposes (*e.g.*, DNS or QUIC requests over UDP) and packets with diagnostic or notification purposes (*e.g.*, ICMP echo requests). Since UDP and ICMP are very different in protocol semantics and network functions, our intuition is to design fine-grained defense components for each of them respectively, so as to fulfill the following missions: (1) Obfuscating the state of the channel in order to thwart the attacks. (2) Ensuring that legitimate request packets mixed with the ILC traffic receive the deserved responses so as to minimize the potential side effects of defense.

B. StateShield Architecture

StateShield consists of four modules that collaborate to achieve real-time detection and mitigation, shown in Figure 3. It has two logical steps: detection and mitigation.

Detection. We design Threshold Selection Module in the control plane and ILC Event Monitor in the data plane for detection.

- *Threshold Selection.* To boost detection accuracy, we design the Threshold Selection Module in the control plane for learning suitable thresholds for online detection. In particular, we use unsupervised models to learn the normal ICMP and UDP behavior in an offline training mode, through which the thresholds for judging abnormal states can be carefully selected (see details in Section V-C).
- *ILC Event Monitor.* To identify ILCs, we design the ILC Event Monitor in the data plane for detecting packet sequences with ILCs from the inbound traffic. It constantly monitors the indicators that can track the ILC-related states of protected targets in real time. To adapt to programmable switches with limited resources, we use indicators that can cover multiple attack variants for monitoring (see

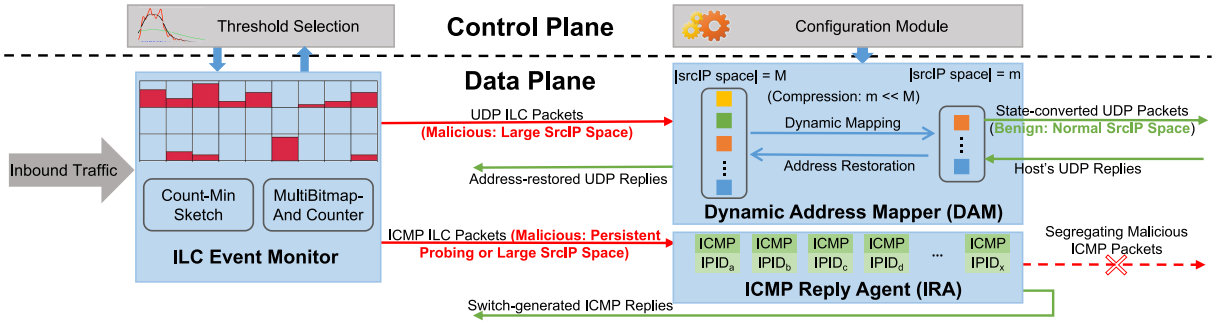


Fig. 3. StateShield architecture.

TABLE II
OVERVIEW OF STATESHIELD DEFENSE SYSTEM

Indicators for Attack Monitoring	Corresponding State of Packet Sequences to Target	Corresponding Attack Vector	Detectors in ILC Event Monitor	Defense Components based on State Obfuscation	Covered Attack Variants
Distinct UDP SrcIPs per DstIP	UDP SrcIP Storm	Excessive UDP Probing with Numerous Spoofed SrcIPs	MultiBitmap-AND Counter (new)	DAM (new)	LE (LE-1,LE-2,LE-3,LE-4) [1], LA (LA-1a, LA-1b,LA-1c) [1], WE (WE-1, WE-2) [1], WA (WA-1a, WA-1b) [1], etc.
Distinct ICMP SrcIPs per DstIP	ICMP SrcIP Storm	Excessive ICMP Probing with Numerous Spoofed SrcIPs	MultiBitmap-AND Counter (new)	IRA (new)	LF (LF-1, LF-2) [1], VCD [2], etc.
Number of ICMP Packets per DstIP	Persistent ICMP Probing	Excessive ICMP Probing with One or Few SrcIPs	Count-Min Sketch	IRA (new)	OF (OF-1, OF-2, OF-3) [1], CD [2], SI [2], AI [2], etc.

details in Section V-A) and efficient data structures as our detectors (see details in Section V-B). Once an ILC attack is detected, ILC Event Monitor will trigger the defense modules to immediately activate the protection measures for the target under attack.

Mitigation. We design two defense components based on state obfuscation in the data plane and a Configuration Module in the control plane for mitigation.

- *Configuration Module.* The Configuration Module is mainly responsible for updating configurations (e.g., hash functions used in the data plane), and dynamically enabling or disabling defense components in real-time and on-demand based on the current state of attack occurrence and termination. For example, once ILC Event Monitor detects a risky state appearing in a target (*i.e.*, any of the indicators are higher than the threshold), it will send a *digest* to the control plane to activate corresponding defense components to protect the target host. Upon receiving the signal, Configuration Module adds the IP of the target host under attack into the host list with activated defense, so that the connectionless packet sequences with ILCs destined to the target will be processed by defense components first before being forwarded to that target. Once the attack is no longer detected, the Configuration Module can effortlessly remove the activated entry for that host. Subsequently, the programmable switch will terminate the activated defense measures applied on the connectionless packets sent to the target and only function as a traffic forwarder for that target, making the defense on-demand.
- *Defense Components based on State Obfuscation.* We design two defense components based on state obfuscation, *i.e.*, *Dynamic Address Mapper* (DAM) and *ICMP Reply Agent* (IRA), as illustrated in Figure 3. The defense components are responsible for providing on-demand and real-time defense against ILCs for targets under attack. Specifically, for packet sequences with ILCs that interact with the target for data exchange (*e.g.*, DNS or QUIC requests over UPD), we use *Dynamic Address Mapper* to eliminate the risk state of packet sequences before forwarding the packet sequences to the target. For packet sequences that interact with the target for diagnostic or notification functions (*e.g.*, ICMP echo requests), we use *ICMP Reply Agent* to segregate the target from the persistent probing while generating informational responses by the switch. Both components are designed to maintain responses for potential legitimate connectionless requests as much as possible. We will introduce details of the defense components in Section VI.

Based on the proposed architecture, we develop a defense system with its overview shown in Table II, which is the first defense system that follows the proposed design goals and architecture. In particular, we propose indicators with high coverage that can cover various ILC attacks and design efficient detectors for monitoring in the data plane (*i.e.*, Section V). Then, we develop two novel on-demand defense components that do not incur unintended dropping (*i.e.,* Section VI). In the next two sections, we will present the details of our key designs.

V. EFFICIENT MONITORING OF ILC EVENTS

In this section, we introduce StateShild's components for efficiently monitoring ILC attacks. We first introduce the indicators we propose for monitoring, then introduce detectors in the data plane, with a focus on the MultiBitmap-AND Counter newly designed for the counting task in our scenario. Finally, we introduce the threshold selection module in the control plane.

A. Attack-Sensitive Indicators With High Coverage

Due to the limited resources of the switches, we carefully select attack-sensitive indicators that can cover multiple ILC variants so as to detect attacks as extensively as possible with moderate monitoring overhead. To achieve that, we investigate the possible attack vectors for constructing connectionless ILCs (see the fourth column of Table I). Specifically, we focus

on the shared attack vectors that are repeatedly used in multiple attacks, from which we identify indicators that are representative and effective for detecting a wide range of attacks, as shown in Table II. These attack-sensitive indicators will deviate significantly from their values under normal traffic conditions when an ILC attack occurs. Therefore, detecting the abnormal states of these indicators can help uncover potential ILC attacks behind the indicator anomalies. We propose three indicators with high coverage of connectionless ILCs as follows:

- **Distinct UDP SrcIPs per DstIP.**[2] This indicator is used to monitor the shared attack vector *Excessive UDP Probing with Numerous Spoofed SrcIPs* that leads to the abnormal state of *UDP SrcIP Storm* of packet sequences. The attack vector is repeatedly used in ILCs such as Linux Exfiltration (LE-1 to LE-4), Linux Alias Resolution (LA-1a to LA-1c), Windows Exfiltration (WE-1, WE-2), Windows Alias Resolution (WA-1a, WA-1b) described in [1], etc.
- **Distinct ICMP SrcIPs per DstIP.** This indicator is used to monitor the shared attack vector *Excessive ICMP Probing with Numerous Spoofed SrcIPs* that leads to the abnormal state of *ICMP SrcIP Storm* in packet sequences. The attack vector is repeatedly used in ILCs such as Linux Firewall Piercing attacks (LF-1 and LF-2 described in [1]), and Victim Client Detection (VCD) attack described in [2], etc.
- **Number of ICMP Packets per DstIP.** This indicator is used to monitor the shared attack vector *Excessive ICMP probing with One or Few SrcIPs* that leads to the abnormal state of *Persistent ICMP Probing*. The attack vector is repeatedly used in ILCs such as OpenBSD Firewall Piercing attacks (OF-1, OF-2, OF-3 described in [1]), and side channel variants including CD, Sequence Inference (SI), and ACK Inference (AI) described in [2], etc.

Note that the proposed indicators are derived from ILC attack vectors and highly correlated with ILC attack behavior. By monitoring each of the attack-sensitive indicators and identifying anomalies in them, StateShield can unveil potential ILC attacks against protected targets and trigger the appropriate defenses accordingly.

B. Efficient Detectors in the Data Plane

To achieve efficient monitoring in the data plane, we use efficient probabilistic data structures as detectors of ILC Event Monitor: the Count-Min sketch [25] for counting the *Number of ICMP packets per DstIP*, and a newly designed MultiBitmap-AND Counter for counting the *Number of Distinct SrcIPs per DstIP* for ICMP and UDP.

MultiBitmap-AND Counter. Inspired by *Linear-time Probabilistic Counting* [26] and *Direct Bitmap* [27], we design a new probabilistic data structure *i.e.*, *MultiBitmap-AND Counter*, to count the number of unique UDP SrcIPs and ICMP SrcIPs sent to the targets, which is well-suited to our scenarios and feasible to implement in programmable switches. The data structure of MultiBitmap-AND Counter is shown in Figure 4. It consists of r rows (corresponding to r hash functions) and w buckets for each row, and each bucket is a bitmap with n bits

[2]*DstIP* denotes the destination IP, *SrcIP* denotes the source IP, and *IP Storm* is used to describe a sudden surge in the quantity of IP addresses.

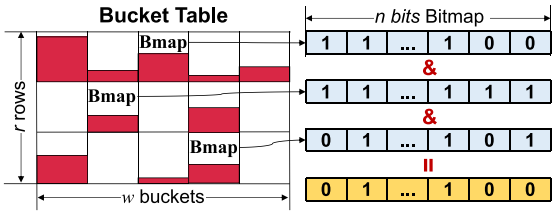


Fig. 4. MultiBitmap-AND counter.

that records the distinct SrcIPs that reach the bucket. For each incoming packet, we first map the destination IP to one bucket of each row through the row hash functions, then calculate the bit position by the source IP through the bitmap hash function, and set the corresponding bits of r bitmaps to 1. Note that for all bitmaps, we use the identical bitmap hash function. For the result of a specified destination IP, MultiBitmap-AND Counter requires performing a bitwise AND operation among its r bitmaps to obtain the final bitmap. As illustrated in Figure 4, when a certain source IP has been recorded in all r bitmaps, the final bitmap's corresponding bit will be set to 1. Note that the bitwise AND operation can reduce counting errors caused by multiple destination IP sharing the same bitmap.

Finally, we derive the estimated number of distinct SrcIPs reaching the destination IP (target host) based on the formula $\hat{S} = n \ln(n/Z)$ [26], where n represents the bitmap size, and Z represents the number of bits that are set to 0 in the final bitmap. We present the theoretical analysis in Appendix B in the Supplementary Material.

Algorithm of MultiBitmap-AND Counter. For an incoming packet, Algorithm 1 shows how the MultiBitmap-AND Counter completes the update and returns the number of unique source IPs sent to the target. For r different bitmaps, we use the same HASH function ($HASH$) to locate the bit index (Idx) occupied by the packet's source IP and then set the corresponding bits to 1 to complete the counter update (line 12∼19). Since it is difficult to implement bitwise AND operation in the data plane, we use a MIN operation on the r bitmaps to get a course-grained estimation of the bitwise AND count value (line 17∼18, note that the count value obtained by the MIN operation is greater than that obtained by bitwise AND operation). If the estimated value is larger than the detection threshold, StateShield will send a Digest (line 21∼22) to the control plane for triggering a more precise bitwise AND operation on the r bitmaps (line 23∼28) and returning the final result(line 29∼30). Since our defense strategy also needs to be triggered through the Configuration Module in the control plane, this computing mode which involves prejudging in the data plane and reconfirmation in the control plane, does not affect the real-time property of the defense triggering.

C. Threshold Selection in the Control Plane

Establishing appropriate thresholds that can effectively distinguish between the normal and abnormal states of attack-sensitive indicators is crucial for accurately identifying ILC attack events. In our problem domain, the most notable characteristic of connectionless ILC attacks is the abnormal deviation in our three proposed monitoring indicators, closely associated with excessive ICMP and UDP probing. To address this, we employ K-Means as the unsupervised

Algorithm 1 Operations on MultiBitmap-AND Counter

Pkt ← ICMP or UDP packets sent to a target host.
r ← Number of hash functions $HASH\_f_i$ in MultiBitmap-AND Counter, $i \in (1, r)$.
w ← Output size of $HASH\_f$ in MultiBitmap-AND Counter.
$BitMap_{ij}$ ← Bitmaps for MultiBitmap-AND Counter, $i \in (1, r)$, $j \in (1, w)$.
n ← The Bitmap size.
$HASH$ ← Hash function for Bitmaps.
$BMapCount_{ij}$ ← The number of bit 1 of each Bitmap, $i \in (1, r)$, $j \in (1, w)$.
Function `UpdateCounter(`*Pkt*`)`**:**
    //data plane
    $MinValue = n$
    $digest = [Pkt.proto, Pkt.dstIP]$
    for $i \in (1, r)$ **do**
        $j = HASH\_f_i.get(Pkt.proto, Pkt.dstIP)\ \%\ w$
        $Idx = HASH.get(Pkt.srcIP)\ \%\ n$
        if $BitMap_{ij}[Idx] == 0$ **then**
            $BMapCount_{ij}$ += 1
            if $MinValue > BMapCount_{ij}$ **then**
                $MinValue = BMapCount_{ij}$
        $BitMap_{ij}[Idx] = 1$
        $digest.append(j)$
    if $MinValue > threshold$ **then**
        $sendDigest(digest)$
    //control plane
    if $getDigest()$ **then**
        $temp\_BitMap = 2^n - 1$
        for $i \in (1, r)$ **do**
            $j = digest[i + 2]$
            $temp\_BitMap = temp\_BitMap\ \&\ BitMap_{ij}$
    $Z = countBitZero(temp\_BitMap)$
    return $n \ln(n/Z)$

learning technique to learn the normal states of the selected monitoring indicators from benign traffic. This method is chosen due to its simplicity, computational efficiency, interpretability, and proven effectiveness in various anomaly detection applications [5]. The clusters formed by K-Means are straightforward to interpret: each cluster center represents a typical pattern of benign traffic, and the distance from data points to these centers provides a clear metric for identifying anomalies. This interpretability is crucial for setting understandable and justifiable threshold values. The K-Means model is trained using traffic data representing benign ICMP and UDP behavior. It learns cluster centers and assigns data points to clusters based on these benign traffic samples. This information is then used to establish appropriate anomaly detection thresholds for each of the proposed indicators, helping to detect potential attack events behind abnormal states.

To evaluate our detection method's performance across common scenarios, we employ benign traffic data collected from the MAWI dataset [28] as the training dataset for our experiments. This choice is due to the extensive benign traffic available from MAWI, which makes it well-suited for capturing the general patterns of normal ICMP and UDP behavior so as to establish reasonable thresholds for anomaly detection. Note that in real-world deployments, the training dataset can be substituted with actual subnet traffic to establish domain-suited thresholds, particularly when the protected subnet's ICMP and UDP behavior exhibits distinctive patterns. Besides, regular threshold updates can be performed through retraining to accommodate changes in traffic patterns.

VI. Defense Components in the Data Plane

In this section, we introduce our two defense components in the data plane.

A. DAM for Mitigating UDP-Based ILCs

We design Dynamic Address Mapper (DAM), which can mitigate UDP-based ILCs that manipulate the attack vector of *Excessive UDP Probing with Numerous Spoofed SrcIPs*, covering attack variants such as LE, LA, WE, WA (as listed in Table I), etc. Note that the success of those attacks depends on the continuous IPID consumption caused by the large space of SrcIPs over UDP probing packets.

DAM includes a series of operations on the switch to eliminate the risk state of these ILC packet sequences before forwarding them to the target, destroying the ILCs while ensuring data exchange for UDP requests from potential clients. Specifically, DAM constructs a dynamic bidirectional IP transform table for address compression and restoration. It dynamically maps the large source address space of attacker-crafted incoming UDP packets into a smaller and uncertain source address space before forwarding them to the target. This process significantly reduces the extensive source address space created by the attacker, thereby avoiding the aggressive consumption of the target's IPIDs. By doing so, DAM obfuscates the state of the shared IPID counter in the target, preventing the attacker from constructing a stable channel.

Design Challenges. Given the limited resources of programmable switches, it is challenging to design a defense component that achieves the following purposes at the same time: (1) Compressing the large and unknown SrcIPs space dynamically on the fly to break the condition for attack success; (2) Ensuring restorable mapping of IP for the target-responded packets to persistently maintaining the communication of potential legitimate UDP packets. (3) Ensuring the unpredictability of IP mapping to fully thwart the ILC and avoid potential exploit of DAM itself.

Ensuring the unpredictability of IP mapping. Even though the address compression thwarts the SrcIP-Storm-based ILC attacks, there is a risk that a strong adversary might try to exploit DAM itself for other unknown side channels, given that the address space after compression is smaller and IP collision are more likely to happen. To mitigate this risk, we propose two mechanisms to make the IP mapping relationship unpredictable: random update of hash functions

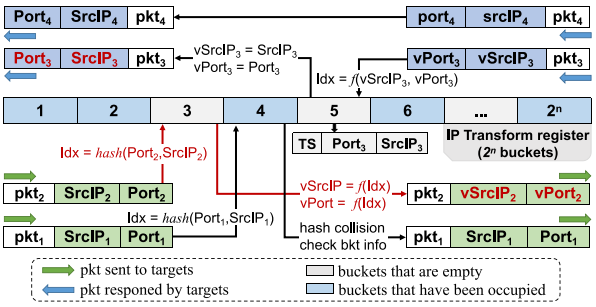


Fig. 5. Dynamic address mapper.

used for IP mapping and random switching of virtual IP pools for address mapping periodically. With these two mechanisms, each spoofed source IP address crafted by attackers will be mapped to an uncertain virtual IP in DAM, which introduces an extra layer of uncertainty and unpredictability in IPID generation.[3] With such an uncertain IP mapping relationship, it will be difficult for attackers to exploit DAM for other potential side channels. Note that the random update of hash functions and the virtual IP pools used in the data plane can be configured through Configuration Module in the control plane.

Design Details of DAM in programmable switch. To fit into the resource-constrained switch data plane, we design DAM as a ***dynamic and efficient*** probabilistic data structure through registers. Figure 5 shows how DAM compresses the unknown and large SrcIPs space of incoming UDP probing packets into a small *virtual* IP address space and how it restores the original SrcIPs from the virtual IPs in UDP responses from the target. Note that the *virtual* IP address space can be specified as arbitrary private IP address spaces that have no overlap with the address space of both the protected subnet and possible incoming packets. The table consists of 2^n buckets, with each bucket storing the original source IP, source port, and current timestamp of incoming packets that are mapped to it. The index of the bucket implicitly contains the virtual source IP and virtual source port pair. An index is a binary number consisting of n bits, where the high k bits specify the virtual IP and the low l bits specify the virtual port. Specifically, the high $32-k$ bits of the 32-bit virtual IP can take a fixed number, while the low k bits are specified by the high k bits of the index. Similarly, the high $16-l$ bits of the 16-bit virtual port can take a fixed number, while the low l bits are specified by the low l bits of the index. Thus, the compressed source IP space of the incoming packets will be 2^k. To address the problem of bucket exhaustion due to the limited resource of the data plane, we set a time limit T_{expire} for each packet to occupy a bucket.

For incoming packets (pkt_1 and pkt_2), a hash function is used to map their source IPs and source ports to a numeric space of $1{\sim}2^n$ to locate the bucket (Idx) that can store their original information. If the bucket specified by the Idx is empty (for example, pkt_2 is mapped to bucket 3, as shown in Figure 5), we will record the current timestamp, packet source IP, and source port to the bucket. Then, the source IP and port of the original packet are replaced with a unique virtual source IP and port pair generated by the bucket index. When hash collisions occur and the specified bucket is already occupied (as with pkt_1 mapped to bucket 4.), further check of the bucket content is required to determine whether the packet's address can be transformed: (1) When ($TS\_current$ - TS_bkt) $> T_{expire}$ which indicates that the bucket occupancy has timed out, the new incoming packet can overwrite the old one; (2) when the source IP and port pair in the bucket are the same as those in the incoming packet, the incoming packet can still use the virtual IP and port pair specified by the index; (3) when an incoming packet hits a currently occupied and not expired bucket, it will be allowed to pass through unmodified (i.e., normal forwarding).

For packets from the protected target (pkt_3 and pkt_4), we determine whether the response packets need to be translated back by checking whether the destination IPs belong to the virtual IP space. For pkt_4, which has not undergone IP translation, it will be forwarded directly in a normal manner. For pkt_3, we calculate the index of the bucket that holds the original destination IP and port based on the last k bits of the virtual destination IP and the last l bits of the virtual port. Then we replace the destination IP and port with the IP and port stored in the bucket and forward the packet based on the restored address accordingly.

With these designs and appropriate parameter setting for T_{expire}, DAM can achieve the robust IP space transformation and restoration in a way that effectively mitigates corresponding ILCs even with the constraints of limited data plane resources, as demonstrated in Sections VII-C and VII-E.

Algorithm of DAM. Algorithm 2 shows the detailed operations of the DAM. We use three different registers to store timestamps, source IP addresses, and ports, respectively. Each of these registers has 2^n buckets, which means that 2^n different source IP and port pairs can be stored.

For an incoming packet (line 10∼12), we calculate the mapping position (Idx) of the packet source IP and port pair in the IP Transform Register and then use the index to generate the virtual IP and virtual port for the packet (line 17∼19). If the buckets specified by the index are empty or occupied by the previous hashed packet for a time value larger than T_{expire}, we will update the timestamp and record the packet source IP and port pair to corresponding registers (line 21∼24). And the source IP and port of the current packet will be replaced with virtual IP and virtual port (line 25∼26). For other conditions, the current packet can use the virtual IP and virtual port when the previous hashed packet is itself.

For a packet from the protected host, if the packet has undergone address translation, we will extract the index of the IP Transform Register (line 36∼37) according to the packet destination IP and port and read the source IP and port stored in the buckets specified by the index (line 38∼39). Then we replace the virtual IP and virtual port of the packet with the original source IP and port (line 40∼41).

B. IRA for Mitigating ICMP-Based ILCs

We design ICMP Reply Agent (IRA), which can mitigate ICMP-based ILCs that manipulate the attack vector of *Excessive ICMP Probing* with ICMP echo request messages, covering attack variants such as LF, VCD, OF, CD, SI, AI (as listed in Table I), etc.

[3]As shown in Appendix C in the Supplementary Material, our case study on Linux shows that when hash function and virtual IP pool change, the mapping relationship between a source IP address crafted by the attacker and its corresponding Linux IPID counters becomes unpredictable.

Algorithm 2 Operations of DAM

```
Pkt ← UDP packets sent to or originated from a target host.
IP_Transform_Reg_TS ← IP Transform Register for Timestamp, 2^n buckets.
IP_Transform_Reg_IP ← IP Transform Register for Src IP, 2^n buckets.
IP_Transform_Reg_Port ← IP transform Register for Src Port, 2^n buckets.
Fix_IP ← The fix IP prefix of the virtual IP.
Port_offset ← A fixed value added to each virtual port.
k & l, n = k + l ← k bits are used for virtual IPs (2^k), and l bits are used for virtual ports (2^l).
HASH ← Hash function for three IP_Transform_Regs.
T_expire ← The time that each bucket can be continuously occupied.
if Pkt.dstIP == target.IP then
    INTransform(Pkt, IP_Transform_Reg_TS, IP_Transform_Reg_IP, IP_Transform_Reg_Port)
if Pkt.srcIP == target.IP then
    OUTTransform(Pkt, IP_Transform_Reg_TS, IP_Transform_Reg_IP, IP_Transform_Reg_Port)
Function INTransform(Pkt, IP_Transform_Regs):
    Idx = (HASH.get(Pkt.srcIP, Pkt.srcPort))[(n − 1) : 0]
    virtual_IP = Idx[(n − 1) : l]
    virtual_Port = Idx[(l − 1) : 0]
    reg_TS = IP_Transform_Reg_TS[Idx]
    if reg_TS == NULL or reg_TS < (TS_now − T_expire) then
        IP_Transform_Reg_TS[Idx] = TS_now
        IP_Ttransform_Reg_IP[Idx] = Pkt.srcIP
        IP_Ttransform_Reg_Port[Idx] = Pkt.srcPort
        Pkt.srcIP = Fix_IP + virtual_IP
        Pkt.srcPort = Port_offset + virtual_Port
    else
        reg_srcIP = IP_Ttransform_Reg[Idx]
        reg_srcPort = IP_Ttransform_Reg_Port[Idx]
        if reg_srcIP == Pkt.srcIP and reg_srcPort == Pkt.srcPort then
            Pkt.srcIP = Fix_IP + virtual_IP
            Pkt.srcPort = Port_offset + virtual_Port
            IP_Transform_Reg_TS[Idx] = TS_now
Function OUTTransform(Pkt, IP_Ttransform_Regs):
    if Pkt.dstIP[31 : k] == Fix_IP then
        Idx[n − 1 : l] = Pkt.dstIP[k − 1 : 0]
        Idx[l − 1 : 0] = Pkt.dstPort[l − 1 : 0]
        reg_srcIP = IP_Transform_Reg_IP[Idx]
        reg_srcPort = IP_Transform_Reg_Port[Idx]
        Pkt.dstIP = reg_srcIP
        Pkt.dstPort = reg_srcPort
```

IRA mitigates the ICMP-based ILCs by segregating excessive ICMP echo request packets sent to probed targets and preserves the ping service by providing ICMP echo reply messages on behalf of the targets. This segregation ensures that the shared resources (*i.e.,*, IPID counters) of the probed target are not affected by ICMP probing, thereby thwarting the ILCs. With IRA, the ping requests of potential legitimate clients during an attack are guaranteed to receive the deserved ICMP echo replies. Since the function of ICMP echo request is to inform the reachability of the target, it is feasible for the switch connected to the target to temporarily generate the ICMP echo reply on behalf of the target when it's under attack. The IPID of switch-generated ICMP echo reply messages can be specified according to RFC 6864 [29] to ensure consistent and reliable handling of the IPID field. Note that IRA only generates the ICMP echo replies on-demand for ICMP echo requests detected as ILCs, and does not involve packets of other protocols. Like DAM, IRA remains inactive when no attacks are detected, operating on-demand to protect a specific target during an ongoing attack based on real-time identification of ILC victims.

VII. Evaluation

In this section, we evaluate StateShield extensively to answer the following questions:

- Can StateShield detect ILC attacks with good accuracy?
- How well does StateShield mitigate various ILC attacks?
- Does StateShield preserves the service of legitimate connectionless packets when activating defense?
- How robust is StateShield's defense when varying the intensity of background traffic or StateShield's parameters?
- Can StateShild mitigate the attacks in real-time? Does StateShield incur extra latency of switching?

A. Experiments Overview

In this section, we introduce the experiment overview.

Evaluation outline. We evaluate StateShield extensively in the following aspects: detection accuracy, mitigation effectiveness, unintended dropping of legitimate packets, defense robustness, and time overhead. We list all attacks we evaluated in Table III. As is shown in Table III, our extensive experiments demonstrate that StateShield can effectively mitigate more than 10 types of ILCs, and achieves damage-free property when mitigating the attacks. StateShield is robust in mitigation effectiveness under various intensities of background traffic. Besides, StateShield reacts to attacks in real-time, and its defense incurs neglectable latency.

Prototype Implementation. We implement StateShield prototype on Barefoot Tofino2, with switch functions by P4-16 and controller by Python. In *MultiBitmap-AND Counter*, r is set to 3, w is 1024 and each bitmap has 1024 bits. DAM is implemented as 2^n buckets, with n regularly set as 16 (*i.e.*, 65,536 pairs of virtual IPs and virtual ports for mapping). We use the high 7 bits of the bucket index to specify virtual IPs and the low 9 bits to specify virtual ports.

Testbed. We conducted attack and defense experiments in a realistic campus network testbed, comprising an Intel Tofino2 switch connected to three servers for experiments and additional machines generating real-world background traffic. The first server (denoted as S_a) acted as both the attacker and the response collector for spoofed packets. The second server (denoted as S_t) was the protected target. To analyze the responses of S_t to packets with spoofed source addresses, we programmed the switch to forward S_t's responses for spoofed packets with non-existing source addresses to S_a for experimental analysis. The third server (denoted as S_l) generated legitimate requests to S_t and collected responses from it. We ran various real services (e.g., HTTP, SSH, DNS) on the target server to verify the effectiveness of StateShield's defense in different real service scenarios.

TABLE III
EVALUATED ATTACKS, DEFENSE EFFECTIVENESS METRICS, AND STATESHIELD'S OVERALL PERFORMANCE

Evaluated Attacks	Sender's behavior towards target (with parameters of original papers)	Attacker's expected state change in target (Necessary condition for attack success)	Mitigation Effectiveness Metric	Mitigation Effectiveness	Service-Preserving for Legal Packets
LE [1]	Go through L addresses, each spoofs 2M-1 QUIC packets over UDP (e.g., $L = 14000, M = 6$)	For each spoofed address, IPID counter increase >= M	For most of spoofed addresses, IPID counter increase < M	✓	✓
LF [1]	Go through L addresses, each spoofs 2M-1 ICMP echo packets	For each spoofed address, IPID counter increase >= M	IPID state will not be affected by ICMP echo packets	✓	✓
LA [1]	Go through L addresses, each spoofs 2M-1 UDP packets	For each spoofed address, IPID counter increase >= M	For most of spoofed addresses, IPID counter increase < M	✓	✓
WE [1]	Send QUIC/UDP packets with 10000 spoofed addresses in a second	IPID PathSet growth rate reaches 10000/s (purge triggered)	IPID PathSet growth rate << 10000/s	✓	✓
WA [1]	Send QUIC/UDP packets with 10000 spoofed addresses in a second	IPID PathSet growth rate reaches 10000/s (purge triggered)	IPID PathSet growth rate << 10000/s	✓	✓
OF [1]	Send 32768 ICMP echo packets	32768 random IPID values generated	IPID state will not be affected by ICMP echo packets	✓	✓
VCC [2]	Constantly send ICMP echos with different spoofed addresses until collision	Each probed IPID counter will increase accordingly	IPID counter will not be affected by ICMP probing packets	✓	✓
CD [2]	Constantly send ICMP echos with attacker's real IP	The probed IPID counter will increase accordingly	IPID counter will not be affected by ICMP probing packets	✓	✓
SI [2]	Constantly send ICMP echos with attacker's real IP	The probed IPID counter will increase accordingly	IPID counter will not be affected by ICMP probing packets	✓	✓
AI [2]	Constantly send ICMP echos with attacker's real IP	The probed IPID counter will increase accordingly	IPID counter will not be affected by ICMP probing packets	✓	✓

Real-world ILC Attack Launching. We used Python and the Scapy library to construct and send packets, launching realistic ILC attacks. To ensure these attacks were effective, we employed several techniques: utilizing multithreading techniques to accelerate the probing process, allowing rapid scanning of potential target ports and sequence number ranges (Side Channel) and quickly consuming the target host's IPID space (Covert Channel); precisely controlling packet sending intervals to ensure stable and observable signal transmission (e.g., in LE attack); and varying packet attributes to enhance stealthiness and avoid detection by network security mechanisms, etc. For each reproduced attack, the packet sending rates and packet sequence scales were implemented following the descriptions in the original attack papers [1], [2]. To fully evaluate the effectiveness of StateShield, our experiments covered the attack variants with the strongest Sender in [1].

Real-World datasets for Evaluation. To evaluate StateShield's effectiveness in real-world scenarios, we used *tcpreplay* tool to replay real-world internet or data center traces as background traffic. This included traffic from the MAWI [28] internet backbone and two campus data centers [30]. Note that ILC attacks launched within high-quality real-world internet or data center background traffic closely mirror the coexistence of attack and normal traffic in actual network environments, making the evaluation more realistic.

Data Collection. We used the tcpdump tool on both attacker and target server to capture all inbound and outbound traffic, assisting in analyzing and evaluating the effectiveness of the attacks and defenses.

B. Evaluating Detection Accuracy

We validate that StateShield can identify various ILCs among high-speed traffic from massive benign users. This enables the defense module to effectively mitigate such threats.

Baselines in Detection Evaluation. We establish eight state-of-the-art detection methods in the literature. These methods extract various features from flow [31], [34], packets [6], [35], Sketch [8], and frequency domain [5], [33], while applying ML [32], [36] or fixed rules [8] for detection. We deploy the open-source methods without modification and implement the closed-source systems which rely on specific devices, with the hyper-parameters in their original papers.

Metrics. We mainly evaluate the accuracy by measuring AUC and F1-score, as the metrics are widely used in related studies [6], [32], [34], [35], [36], [37], [38]. Note that, these metrics are directly related to other metrics (*i.e.*, TPR, FPR, Precision, and Recall) which allows us to eliminate the bias on metrics [39]. To enable fair comparison, we evaluate the metrics at the packet level, i.e., calculating the accuracy metrics by comparing the ground-truth label and the assigned label for each packet.

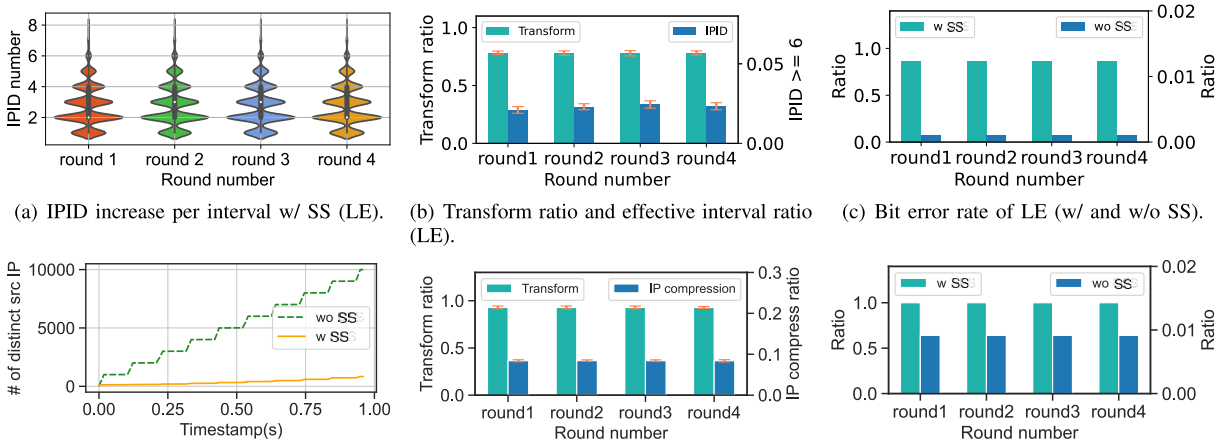
Results. As shown in Table IV, StateShield outperforms eight baselines by detecting ten ILCs with 0.952 average AUC and 0.895 average F_1. In particular, StateShield achieves the best performance (highlighted in green[4]) in AUC for detecting attacks such as LE, LF, LA, WE/WA, OF, VCC, CD, and AI. For F1, StateShield achieves the best results in detecting LF, LA, WE/WA, OF and VCC. Despite StateShield not achieving the best performance in a few attack variants (e.g., in CD attacks, where the low attack rate closely mimics normal traffic patterns leading to a decrease in F_1 value), it still achieves the best results in most ILC attack variants, demonstrating its excellent capability in detecting ILC attacks. One important reason for the overall good detection performance of StateShield is the utilization of the proposed attack-sensitive indicators and the appropriate thresholds obtained by the Threshold Selection Module.

Remark. This comparison with classical works is not meant to emphasize the superiority of our approach over theirs but rather to verify the effectiveness of StateShield's special design for the detection of ILCs, and also deepen the understanding of the challenges in the context of ILC detection. For example, some ILCs may evade detection due to their stealthy nature, especially for methods that are designed for volumetric attacks (*e.g.*, Jaqen, an excellent solution for ISP-centric DDoS defense). Note that we fully recognize the good performance of these baselines in their specific scenarios.

[4]We highlight the best in green, worst in red, and values too small with "-" in Table IV.

TABLE IV
DETECTION ACCURACY OF STATESHIELD AND BASELINES ON 10 ILC ATTACKS

Evaluated Attacks	Metrics	CICFlowMeter [31]	Jaqen [8]	FlowLens [32]	Whisper [33]	Kitsune [6]	NetBeacon [34]	nPrintML [35]	N3IC [36]	StateShield
LE	AUC	0.790	0.975	0.881	0.971	0.644	0.998	0.999	0.871	0.999
	F_1	0.407	0.700	0.498	0.468	0.499	0.962	0.999	0.488	0.998
LF	AUC	0.997	0.997	0.940	0.033	0.815	0.997	0.996	0.642	0.999
	F_1	0.893	0.864	0.474	-	0.500	0.995	0.846	0.567	0.999
LA	AUC	0.751	0.964	0.723	0.992	-	0.974	0.998	0.834	0.999
	F_1	0.348	0.563	0.321	0.417	-	0.595	0.933	0.420	0.997
WE/WA	AUC	0.947	0.970	0.500	0.991	0.728	0.995	0.998	0.984	0.999
	F_1	0.500	0.540	0.499	0.403	-	0.707	0.853	0.576	0.993
OF	AUC	0.634	0.636	0.635	0.913	0.486	0.637	0.713	0.637	0.942
	F_1	0.678	0.703	0.686	0.520	0.501	0.705	0.793	0.705	0.918
VCC	AUC	0.637	0.636	0.794	0.681	0.529	0.637	0.810	0.637	0.941
	F_1	0.705	0.703	0.696	0.541	0.507	0.705	0.795	0.705	0.806
CD	AUC	0.988	0.997	0.994	0.998	0.682	0.985	0.999	0.643	0.999
	F_1	0.524	0.857	0.612	0.449	0.500	0.960	0.893	0.721	0.511
SI	AUC	0.997	0.626	0.610	0.978	0.673	0.625	0.575	0.626	0.687
	F_1	0.978	0.697	0.591	0.617	0.705	0.692	0.624	0.693	0.849
AI	AUC	0.498	0.997	0.916	0.997	0.984	0.971	0.999	0.979	0.999
	F_1	-	0.746	0.800	0.853	0.928	0.531	0.999	0.555	0.985
Average	AUC	0.804	0.866	0.777	0.839	0.693	0.869	0.899	0.761	0.952
	F_1	0.629	0.708	0.576	0.534	0.607	0.762	0.859	0.603	0.895



(a) IPID increase per interval w/ SS (LE).
(b) Transform ratio and effective interval ratio (LE).
(c) Bit error rate of LE (w/ and w/o SS).
(d) Unique Src IPs to S_t (WE).
(e) Transform ratio and IP compression ratio (WE).
(f) Bit Error Rate of WE (w/ and w/o SS).

Fig. 6. Evaluation of effectiveness.

C. Evaluating Mitigation Effectiveness

We evaluate StateShield's mitigation effectiveness on attacks listed in Table III. Our experiments demonstrate that StateShield can effectively mitigate these attacks.

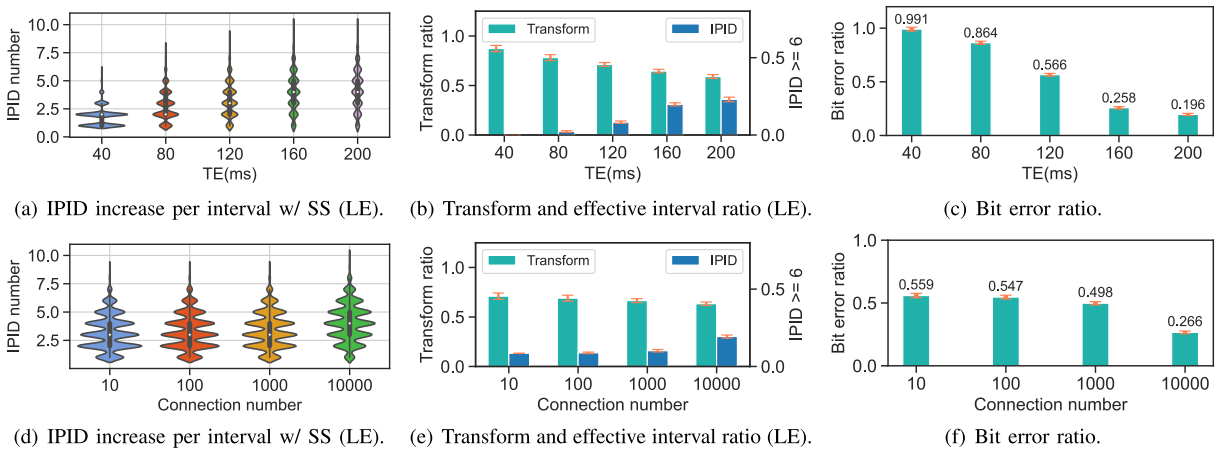
Metrics. The metric to evaluate the mitigation effectiveness of each attack is attack-specific. StateShield thwarts each attack by breaking its condition for the attack to succeed, so the key for measuring the mitigation effectiveness of StateShield is to measure whether the necessary condition for attack success is broken with StateShield's defense. We summarize the condition for each attack to succeed and the corresponding mitigation effectiveness metrics in Table III.

Parameter setting for T_{expire} of DAM. Based on our measurement of RTT distribution (see details in Appendix D in the Supplementary Material), we set T_{expire} as 80 ms, which is sufficient to cover the elapsed time for the switch to send an address-converted packet to the target and the target to send a response back to the switch. Note that the average RTT in our testbed subnet is about 8ms.

Results. We elaborate the results with a focus on the relatively complex attack variants (see Figure 6) .[5]

Figure 6(a) to Figure 6(c) show the effectiveness of mitigating LE attack tested in four rounds. Figure 6(a) shows the distribution of the increase in each IPID counter of the target when the defense is turned on. It shows that for the 14,000 spoofed IPs sent by the attacker in each round, most of the IPID counters are only triggered with an increase of 4. Note that the condition for attack success requires each IPID counter to increase by at least M (with M = 6) [1], which indicates that the attack has failed. It can be seen from Figure 6(b) that 80% of the packets from the attacker have undergone address compressor, and only about 2% of the spoofed IPs can trigger IPIDs to increase continuously over 6. Figure 6(c) shows the bit error rate of the covert channel when the defense is turned on and off. Compared with the 0.1% error rate (calculated based on the original paper [1]) when there is no defense, StateShield can increase the channel error rate to 87%, which strongly shows that the attack is impossible to succeed.

[5] *SS* is the abbreviation for StateShield in the figures.



(a) IPID increase per interval w/ SS (LE). (b) Transform and effective interval ratio (LE). (c) Bit error ratio.

(d) IPID increase per interval w/ SS (LE). (e) Transform and effective interval ratio (LE). (f) Bit error ratio.

Fig. 7. Evaluation of defense robustness.

TABLE V
SERVICE-PRESERVING PROPERTY

Protocol	Arriving Ratio (S_t, wo SS)	Response Ratio (S_l, wo SS)	Arriving Ratio (S_t, w SS)	Response Ratio (S_l, w SS)
UDP	100%	99.999%	100%	99.999%
ICMP	100%	100%	0%	100%

Figure 6(d) to Figure 6(f) the effectiveness of mitigating WE attack. Figure 6(d) shows the number of unique source IPs reaching the protected host with the defense on and off. StateShield can effectively compress the address space of packets that probes the target for about 12 times. Figure 6(e) shows that more than 90% of the attack packets have undergone address compressor, and the number of unique source IPs arriving at S_t is only about 8% of the number of spoofed IPs in the original attack. Figure 6(f) shows that when StateShield is turned on, the bit error rate of the covert channel increases by more than 100 times, which means the attack fails.

For ICMP-based ILCs (e.g., LF1, LF2, OF1, etc.), our evaluation shows that the attacks can be detected within 0.1s after StateShield is initialized, and the latency from the detection of an attack to the activation of IRA defense in the data plane is within 0.01 seconds. With the activation of IRA defense, the switch automatically replies to the ICMP request, and the shared IPID counter of the target is no longer probed by the attacker, in which case the attack fails.

D. Evaluating Service-Preserving Property for Legal Packets

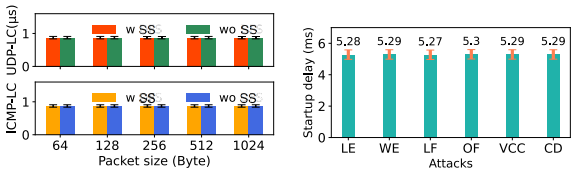
We use *Packet Arriving Ratio* at S_t and *Packet Response Ratio* at S_l measured on the communication between the protected target host and legitimate server to evaluate the service-preserving property of StateShield for legitimate packets.

Specifically, we let S_a spoof legitimate server's IP (i.e., S_l) and send probing attack packets to the protected host S_t while S_l and S_t are communicating. We evaluate the proportion of request packets received by S_t from S_l and the proportion of response packets received by S_l from S_t under UDP attack (LE, WE) and ICMP attack (LF, OF) scenarios, respectively. We repeated 20 experiments with StateShield turned on and off, and the results are shown in Table V. For arriving ratio, it can be seen that since the StateShield does not drop packets when the defense is enabled, 100% of the UDP request packets sent from S_l can reach S_t. Since StateShield will handle ICMP packets and send responses on behalf of the target, the ICMP request packet will not reach S_t when the defense is enabled. For response ratio, regardless of whether the defense is enabled or not, the ICMP requests sent by S_l can get 100% replies, and the UDP requests sent by S_l have a reply ratio as high as 99.999%, which indicates that StateShield does not incur additional dropping of packets.

E. Evaluating Robustness of Defense

We measure the mitigation effectiveness under varying T_{expire}, varying intensity of background UDP connections (i.e., number of UDP sessions), varying intensity of background traffic rate and varying target configurations to evaluate the defense robustness of StateShield. We elaborate on the result with a focus on the most complicated case, i.e., LE.

Figure 7(a) to Figure 7(c) show the result on LE under different T_{expire} (denoted as *TE* in the figures). Figure 7(a) shows that as T_{expire} expands, the distribution of continuous IPID growth varies. However, the majority of data points remain below the expected threshold (i.e., 6). Note that in LE attack, the attacker anticipates that the IPID growth triggered by each spoofed IP will be consistently above 6 with high confidence. Thus, the attack fails since this condition expected by the attacker is disrupted by the StateShield defense. For more details, Figure 7(b) shows the proportion of attack packets undergoing address compressor and the proportion of spoofed IPs that trigger IPID to increase continuously over the expected threshold (i.e., 6) under different T_{expire}. Even with a T_{expire} as large as 200ms, over 60% of the packets can be successfully transformed, and more than 80% of the spoofed IPs cannot achieve the attacker's expected state. The variation of the bit error rate shown in Figure 7(c) indicates that when TE is 200ms, the bit error rate can also be as high as 19.6% (200 times higher than when there is no defense). The above results show that StateShield can effectively mitigate attacks under different T_{expire} and has good robustness. Note that using excessively large values for T_{expire} would result in an attacker's single transient packet occupying register buckets for extended periods, leading to inefficient utilization of the limited register space. Therefore, we do not recommend setting T_{expire} to excessively large values.



(a) Packet processing latency (w/ and w/o SS.) (b) Defense startup delay.

Fig. 8. Evaluation of time overhead.

Figure 7(d) to Figure 7(f) show the result of defense effectiveness against LE under varying background UDP connections. According to [40], the 99th percentile value of Internet RTT is approximately 100ms, so we selected 120ms as a suitably large value for T_{expire}. We let S_a send attack packets to S_t, and maintain a large number of connections with S_t during each round of the attack. We evaluate the defense effectiveness when maintaining 10, 100, 1000, and 10000 connections, respectively. Figure 7(d) illustrates that as the number of background connections increases, the distribution of continuous IPID growth varies. For the majority of data points, it remains below the expected threshold of 6 (i.e., condition for attack success is disrupted). Figure 7(e) shows that when there are 10,000 background connections, 63.4% of the attack packets undergo address compression, and only 19.4% of the spoofed IPs can trigger the number of IPIDs to continuously increase over 6. Figure 7(f) illustrates changes in the bit error rate. When the number of connections is below 1,000, the bit error rate exceeds 50%. With 10,000 connections, the bit error rate remains notably high at 26.6%, which is nearly 300 times higher than the rate with no defense. Therefore, when there is high-intensive background UDP sessions, StateShield's defense is robust.

We also verify that the defense is robust under varying background traffic rates and different target OS configurations. More details are provided in Appendix E.

F. Evaluating Time Overhead

We evaluate the processing latency of StateShield using different-sized packets sent from S_l to S_t. As shown in Figure 8(a), the average latency for processing UDP and ICMP packets is about $0.873\mu s$ and there is no noticeable processing latency change at the microsecond level for a StateShield-enabled programmable switch.

We further evaluate the StateShield startup delay to illustrate the real-time property of StateShield. For each attack, we repeated the experiment 20 times to measure the corresponding delay, and the result is shown in Figure 8(b). For different attacks, there is no obvious difference in StateShield startup delay, with an average of about 5.3ms, which is fast enough to respond to ILCs.

VIII. Discussion

Security of StateShield. It is challenging for attackers to perceive the presence of StateShield's defenses since DAM and IRA are dynamically enabled only when ILCs are detected, and their defenses don't generate distinct traffic patterns that can be easily observed by adversaries. Furthermore, StateShield includes specific design features intended to create challenges for potential attacks against it. Firstly, it's difficult to manipulate DAM to create additional potential side channels because the security of DAM is strengthened through periodic random switching of hash functions and virtual IP address pools for mapping. Secondly, manipulating IRA to create other information leakage attacks is not easily achievable since IRA generates ICMP echo reply messages without direct interaction with the protected target hosts.

Difference between DAM and NAT. DAM incorporates unique and innovative designs specifically aimed at addressing novel ILCs, setting it apart from the conventional Network Address Translation (NAT) in terms of both design and functionality. Firstly, *Dynamic and on-demand address mapping* of DAM *vs. Static and persistent translation* of NAT. DAM is only activated when detecting the state of SrcIP Storm in a target, which is a dynamic and on-demand security service, while NAT only provides static address translation service. Secondly, *Unknown and unlimited address space* of DAM *vs. Fixed and limited address space* of NAT. DAM has to handle unknown and unlimited address space because the spoofed address space used by attackers is unknown and unlimited, while NAT only needs to handle fixed and limited address space of the internal network. These distinctions significantly amplify the complexity and challenges in designing DAM.

Distinctions Between StateShield and Existing ML-based Methods. StateShield differs from existing ML-based methods in two key aspects. First, it employs attack-type-specific indicators, which allows for more accurate detection by capturing the unique characteristics of ILCs. In contrast, methods like Kitsune [6] and Whisper [5] classify traffic based on general anomalies without identifying specific types of attacks, which limits their granularity. Second, many ML-based methods focus solely on detecting attacks and rely on dropping packets or flows identified as anomalous, leading to unavoidable disruptions of legitimate traffic. StateShield, however, uses a state-obfuscation-based defense mechanism that avoids packet dropping, thereby mitigating attacks without affecting legitimate traffic or causing service degradation (See Appendix G for additional experimental evidence).

Handling False Positives in StateShield. In the context of ILC scenarios, it is inevitable for the detection module to generate false positives, especially when ILC attackers spoof connectionless packets for probing. In addition to using ILC-specific features to enhance detection accuracy, StateShield improves upon existing solutions by avoiding the unintended dropping of false positives. By employing state-obfuscation-based defenses instead of dropping packets, StateShield ensures that legitimate traffic is preserved even if mistakenly flagged as malicious, maintaining service continuity and minimizes negative impacts on legitimate traffic (Additional experimental evidence is provided in Appendix G).

Extending StateShield to protect legacy devices of global IPID counter. Since global IPID counter is an insecure and outdated IPID allocation mechanism that mainly exists in uncommon legacy devices [23], StateShield primarily focuses on addressing attacks against non-global IPID allocation mechanisms due to their prevalence and increasing significance in modern networks (as highlighted in [23]), which are currently the prevailing choice among mainstream operating systems [1]. Although global IPID falls outside the problem scope defined in our threat model, StateShield can still be extended to defend against global

IPID-based ILCs when necessary, requiring only minor adjustments within its existing framework. These adjustments only involve integrating a straightforward IPID statistical analysis component to identify global IPID target hosts in the protected subnet and implementing dynamic hash-based mapping operations on IPIDs generated from these targets. The hash functions utilized in DAM can be reused, and StateShield's defense mechanisms against non-global IPID attacks remain unchanged. Our preliminary experiments suggest that these adjustments can automate the discovery of potential legacy devices with global IPIDs and transform their IPIDs from sequential to dynamic and non-sequential (the occurrence of sequential IPIDs becomes negligible, reaching a ratio of 0 percent in our tests utilizing a basic CRC for mapping).

Future Work. StateShield can be extended to defend against other attacks that rely on similar attack vectors over connectionless protocols. For example, the attack vectors of smurf attack (ICMP), ICMP flood, and UDP flood closely resemble the ones studied in this paper, differing mainly in their higher intensity and distinct consequences. The required modifications for expanding StateShield for those attacks include implementing DDoS-adapted detection thresholds and mitigation mechanisms to handle the overwhelming flooding packets. We plan to explore these extensions in future work.

IX. Related Work

ILCs based on Shared Resources. Shared resources such as IPID counter, ICMP rate limit, global challenge ACK rate limit, etc., have long been abused to construct side channels and covert channels [1], [2], [3], [4], [41], [42], [43]. The major solutions proposed for the ILC vulnerabilities are fixing the vulnerable protocol with OS patches, and some old vulnerabilities have gradually been fixed as time goes by. Our work focuses on a novel type of ILCs, *i.e.*, the recently proposed ILCs based on connectionless protocols(*e.g.*, [1], [2], etc.), which is challenging to defend and can be exploited by attackers to cause significant security consequences on the current Internet.

Network Timing channels, Storage channels, and their Mitigation. Network timing channels [15], [16], [44], [45], [46], [47], [48] and storage channels [17], [18], [19], [49], [50], [51] have been extensively studied for a long time and belong to traditional network covert channels that are different from connectionless ILCs. Existing work has developed a series of defenses to mitigate storage and timing channels [11], [19], [20], [21], but they are incapable of mitigating connectionless ILCs in our problem scope due to the huge difference in attack behaviors.

Attack detection. The academic community has proposed a series of solutions for attack detection(*e.g.*, [5], [6], [31], [32], [33], [34], [35], [36], [52]). They use various features such as flow [31], [34], packets [6], [35], and frequency domain [5], [33]. However, these methods neither cover ILCs nor provide fine-grained defense against ILCs.

Defenses on the programmable data plane. The programmable data plane has been used for network security applications, such as Netwarden [11], Poseidon [7], Jaqen [8], Ripple [9], and Mew [10]. Netwarden is designed for mitigating network storage and timing channels, Poseidon and Jaqen are designed to mitigate volumetric DDoS attacks, and Ripple and Mew are designed to mitigate link flooding attacks. All of those defenses are orthogonal to StateShield with respect to problem scope and mitigation method.

X. Conclusion

In this paper, we present StateShield, a real-time defense system against state-of-the-art information leakage channels over connectionless ICMP and UDP. It is the first in-network system to mitigate connectionless ILCs in real-time while inflicting minimal unintended dropping of legitimate packets. StateShield achieves this with efficient detection and on-demand defense components against ILCs in the data plane. In the core of StateShield, two novel defense components based on state obfuscation are designed to mitigate the risk of ILCs automatically without dropping packets. Our evaluation demonstrates that StateShield can effectively mitigate various ILCs without hurting the services over legitimate connectionless packets, and the defense provided by StateShield is robust under high-intensive background traffic. StateShield moves the first step towards real-time and service-preserving mitigation against connectionless ILCs.

References

[1] A. Klein, "Subverting stateful firewalls with protocol states," in *Proc. 29th Netw. Distrib. Syst. Secur. Symp.*, 2022, pp. 1–18.

[2] X. Feng, C. Fu, Q. Li, K. Sun, and K. Xu, "Off-path TCP exploits of the mixed IPID assignment," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2020, pp. 1323–1335.

[3] L. Pan et al., "Your router is my prober: Measuring IPv6 networks via ICMP rate limiting side channels," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2023, pp. 1–18.

[4] K. Man, X. Zhou, and Z. Qian, "DNS cache poisoning attack: Resurrections with side channels," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2021, pp. 3400–3414.

[5] C. Fu, Q. Li, M. Shen, and K. Xu, "Realtime robust malicious traffic detection via frequency domain analysis," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2021, pp. 3431–3446.

[6] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2018, pp. 1–15.

[7] M. Zhang et al., "Poseidon: Mitigating volumetric DDoS attacks with programmable switches," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2020, pp. 1–18.

[8] Z. Liu et al., "Jaqen: A high-performance switch-native approach for detecting and mitigating volumetric DDoS attacks with programmable switches," in *Proc. USENIX Secur. Symp.*, 2021, pp. 3829–3846.

[9] J. Xing, W. Wu, and A. Chen, "Ripple: A programmable, decentralized link-flooding defense against adaptive adversaries," in *Proc. USENIX Secur.*, Jan. 2021, pp. 3865–3881.

[10] H. Zhou, S. Hong, Y. Liu, X. Luo, W. Li, and G. Gu, "Mew: Enabling large-scale and dynamic link-flooding defenses on programmable switches," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2023, pp. 1625–1639.

[11] J. Xing, Q. Kang, and A. Chen, "NetWarden: Mitigating network covert channels while preserving performance," in *Proc. 29th USENIX Secur. Symp.*, 2020, pp. 2039–2056.

[12] J. Postel, "Rfc0793: Transmission control protocol," Internet Eng. Task Force (IETF), Wilmington, DE, USA, Tech. Rep. RFC 793, 1981.

[13] J. Postel, "Internet control message protocol," Internet Eng. Task Force (IETF), Wilmington, DE, USA, Tech. Rep. RFC 792, 1981.

[14] J. Postel, "Rfc0768: User datagram protocol," Internet Eng. Task Force (IETF), Wilmington, DE, USA, Tech. Rep. RFC 768, 1980.

[15] S. Cabuk, C. E. Brodley, and C. Shields, "IP covert timing channels: Design and detection," in *Proc. 11th ACM Conf. Comput. Commun. Secur.*, Oct. 2004, pp. 178–187.

[16] K. S. Lee, H. Wang, and H. Weatherspoon, "PHY covert channels: Can you see the idles?" in *Proc. 11th USENIX Symp. Networked Syst. Design Implement. (NSDI)*, Apr. 2014, pp. 173–185.

[17] X. Luo, P. Zhou, E. W. W. Chan, R. K. C. Chang, and W. Lee, "A combinatorial approach to network covert communications with applications in Web leaks," in *Proc. IEEE/IFIP 41st Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2011, pp. 474–485.

[18] C. H. Rowland, "Covert channels in the TCP/IP protocol suite," *First Monday*, vol. 2, no. 5, May 1997. [Online]. Available: https://firstmonday.org/ojs/index.php/fm/article/download/528/449

[19] D. M. Dakhane and P. R. Deshmukh, "Active warden for TCP sequence number base covert channel," in *Proc. Int. Conf. Pervasive Comput. (ICPC)*, Jan. 2015, pp. 1–5.

[20] G. Fisk, M. Fisk, C. Papadopoulos, and J. Neil, "Eliminating steganography in Internet traffic with active wardens," in *Proc. Inf. Hiding*, Noordwijkerhout, The Netherlands. Cham, Switzerland: Springer, Oct. 2003, pp. 18–35.

[21] G. Lewandowski, N. B. Lucena, and S. J. Chapin, "Analyzing network-aware active wardens in ipv6," in *Proc. 8th Int. Workshop Inf. Hiding*, Alexandria, VA, USA. Cham, Switzerland: Springer, 2007, pp. 58–77.

[22] F. Gont, "Security assessment of the Internet protocol version 4," Internet Eng. Task Force (IETF), Wilmington, DE, USA, Tech. Rep. RFC 6274, 2011.

[23] F. Salutari, D. Cicalese, and D. J. Rossi, "A closer look at IP-ID behavior in the wild," in *Proc. Int. Conf. Passive Active Netw. Meas.*, Berlin, Germany. Cham, Switzerland: Springer, Mar. 2018, pp. 243–254.

[24] V. Paxson, "Bro: A system for detecting network intruders in real-time," *Comput. Netw.*, vol. 31, nos. 23–24, pp. 2435–2463, Dec. 1999.

[25] G. Cormode and S. Muthukrishnan, "An improved data stream summary: The count-min sketch and its applications," *J. Algorithms*, vol. 55, no. 1, pp. 58–75, Apr. 2005.

[26] K.-Y. Whang, B. T. Vander-Zanden, and H. M. Taylor, "A linear-time probabilistic counting algorithm for database applications," *ACM Trans. Database Syst.*, vol. 15, no. 2, pp. 208–229, Jun. 1990.

[27] C. Estan, G. Varghese, and M. Fisk, "Bitmap algorithms for counting active flows on high speed links," in *Proc. ACM SIGCOMM Conf. Internet Meas.*, 2003, pp. 153–166.

[28] WIDE. *MAWI Working Group Traffic Archive*. Accessed: Apr. 2023. [Online]. Available: http://mawi.wide

[29] J. Touch, "Rfc 6864: Updated specification of the ipv4 id field," Internet Eng. Task Force (IETF), Wilmington, DE, USA, Tech. Rep. RFC 6864, 2013.

[30] WIDE. *Data Set for IMC 2010 Data Center Measurement*. Accessed: May 2024. [Online]. Available: https://pages.cs.wisc.edu/

[31] B. H. Ali, N. Sulaiman, S. A. R. Al-Haddad, R. Atan, and S. L. M. Hassan, "DDoS detection using active and idle features of revised CICFlowMeter and statistical approaches," in *Proc. 4th Int. Conf. Adv. Sci. Eng. (ICOASE)*, Sep. 2022, pp. 148–153.

[32] D. Barradas, N. Santos, L. Rodrigues, S. Signorello, F. M. V. Ramos, and A. Madeira, "FlowLens: Enabling efficient flow classification for ML-based network security applications," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2021, pp. 1–8.

[33] C. Fu, Q. Li, M. Shen, and K. Xu, "Frequency domain feature based robust malicious traffic detection," *IEEE/ACM Trans. Netw.*, vol. 31, no. 1, pp. 452–467, Feb. 2023.

[34] G. Zhou, "Netbeacon: An efficient design of intelligent network data plane," in *Proc. Secur. USENIX*, 2023, pp. 6203–6220.

[35] J. Holland, P. Schmitt, N. Feamster, and P. Mittal, "New directions in automated traffic analysis," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2021, pp. 3366–3383.

[36] G. Siracusano et al., "Re-architecting traffic analysis with neural network interface cards," in *Proc. 19th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2022, pp. 513–533.

[37] S. Zhu et al., "You do (not) belong here: Detecting DPI evasion attacks with context learning," in *Proc. 16th Int. Conf. Emerg. Netw. Exp. Technol.*, Nov. 2020, pp. 183–197.

[38] R. Tang et al., "ZeroWall: Detecting zero-day Web attacks through encoder–decoder recurrent neural networks," in *Proc. IEEE Conf. Comput. Commun.*, Jul. 2020, pp. 2479–2488.

[39] D. Arp, "Dos and don'ts of machine learning in computer security," in *Proc. 31st USENIX Security Symposium (USENIX Security)*, 2022, pp. 1–19.

[40] S. Sengupta, H. Kim, and J. Rexford, "Continuous in-network round-trip time monitoring," in *Proc. ACM SIGCOMM Conf.*, 2022, pp. 473–485.

[41] R. Ensafi, J.-C. Park, D. Kapur, and J. R. Crandall, "Idle port scanning and non-interference analysis of network protocol stacks using model checking," in *Proc. USENIX Secur. Symp.*, Aug. 2010, pp. 257–272.

[42] R. Ensafi, J. Knockel, G. Alexander, and J. R. Crandall, "Detecting intentional packet drops on the Internet via TCP/IP side channels," in *Proc. 15th Int. Conf. Passive Act. Meas. (PAM)*, Los Angeles, CA, USA. Cham, Switzerland: Springer, Mar. 2014, pp. 109–118.

[43] Y. Cao, Z. Qian, Z. Wang, T. Dao, S. V. Krishnamurthy, and L. M. Marvel, "Off-path TCP exploits: Global rate limit considered dangerous," in *Proc. 25th USENIX Secur. Symp.*, 2016, pp. 209–225.

[44] S. Cabuk, "Network covert channels: Design, analysis, detection, and elimination," Ph.D. dissertation, Dept. Elect. Eng., Purdue Univ., West Lafayette, IN, USA, 2006.

[45] S. Gianvecchio, H. Wang, D. Wijesekera, and S. Jajodia, "Model-based covert timing channels: Automated modeling and evasion," in *Proc. 11th Int. Symp. Recent Adv. Intrusion Detection*, Cambridge, MA, USA. Cham, Switzerland: Springer, Sep. 2008, pp. 211–230.

[46] X. Luo, E. W. W. Chan, and R. K. C. Chang, "TCP covert timing channels: Design and detection," in *Proc. IEEE Int. Conf. Dependable Syst. Netw. FTCS DCC (DSN)*, Jun. 2008, pp. 420–429.

[47] G. Shah, A. Molina, and M. Blaze, "Keyboards and covert channels," in *Proc. USENIX Secur. Symp.*, Jul. 2006, p. 64.

[48] X. Wang and D. S. Reeves, "Robust correlation of encrypted attack traffic through stepping stones by manipulation of interpacket delays," in *Proc. 10th ACM Conf. Comput. Commun. Secur.*, Oct. 2003, pp. 20–29.

[49] J. Giffin, R. Greenstadt, P. Litwack, and R. Tibetts, "Covert messaging through TCP timestamps," in *Proc. Int. Workshop Privacy Enhancing Technol.*, Berlin, Germany, 2002, pp. 194–208.

[50] A. Hintz, "Covert channels in TCP and IP headers," *Presentation DEFCON*, vol. 10, no. 16, pp. 1–40, 2002.

[51] E. Jones, O. L. Moigne, and J.-M. Robert, "IP traceback solutions based on time to live covert channel," in *Proc. 12th IEEE Int. Conf. Netw. (ICON)*, vol. 2, Nov. 2004, pp. 451–457.

[52] H. Li, Y. Zhao, W. Yao, K. Xu, and Q. Li, "Towards real-time ML-based DDoS detection via cost-efficient window-based feature extraction," *Science China Inf. Sci.*, vol. 66, no. 5, 2023, Art. no. 152105.

Haibin Li received the Ph.D. degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 2024. This work was conducted during his Ph.D. studies at Tsinghua University. He is currently an Assistant Professor with the College of Information and Communication, National University of Defense Technology. His research interests include network and system security.

Qi Li (Senior Member, IEEE) received the Ph.D. degree from Tsinghua University. He is currently an Associate Professor with the Institute for Network Sciences and Cyberspace, Tsinghua University. His research interests include network and system security, particularly in internet security, the IoT security, and AI security. He is an Editorial Board Member of IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, ACM TOPS, and ACM DTRAP.

Yingying Su received the Ph.D. degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 2024. Her research interests include routing security and RPKI.

Xuewei Feng received the Ph.D. degree from Tsinghua University, Beijing, China, in 2022. His research interests include network security and software vulnerability detection.

Chuanpu Fu (Graduate Student Member, IEEE) received the B.E. degree from the Department of Networking Engineering, Dalian University of Technology, in 2020. He is currently pursuing the Ph.D. degree with Tsinghua University. His research interests include machine learning for security, and network and system security.

Ke Xu (Fellow, IEEE) received the Ph.D. degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China. He is currently a Full Professor with the Department of Computer Science and Technology, Tsinghua University. He has published more than 200 technical articles and holds more than ten U.S. patents in the research areas of next-generation internet, blockchain systems, the Internet of Things, and network security. He is a member of ACM. He has won the IWQoS 2024 Best Paper Award and the Distinguished Paper Award from USENIX Security 2023 and 2024.