In recent years, network attacks have exploded in frequency, scale and sophistication [5, 4, 6, 12, 3, 9]. Benign traffic volumes are also increasing exponentially [11], with many applications requiring increasingly stringent performance guarantees. With all these trends, network providers urgently require scalable defenses (handling large amounts of benign and adversarial traffic) that provide both accuracy (i.e., correctly identifying, blocking, and serving traffic) and performance (i.e., quickly dropping adversarial traffic to ensure low latency for benign clients).

Recently, many works have sought to capitalize on performance opportunities offered by high-speed programmable hardware, proposing hybrid hardware-software designs that offload network functions (NFs) from x86 servers to the data plane (e.g., P4-based architectures) [8, 10, 13, 15, 2]. These works propose cache-style designs that exactly partition NFs between hardware and software. Portions of an NF that would benefit from high-speed packet-processing are offloaded to dataplane hardware, creating a cache-style fastpath for optimizing performance, while the non-offloaded portion of the NF remains on the server and acts as the processing slowpath [13]. With such cache-style approaches, only traffic that matches on the subset of the NF in the dataplane "cache" can benefit from cache hits. This optimizes performance for application traffic in the best-case scenario with little to no attacks, but leaves a critical gap in the context of modern security applications. Attacks cannot be predicted to lie within the coverage of a dataplane cache, and such approaches would degrade quickly under worst-case scenarios with large-scale attacks from sophisticated adversaries. Even NF offloading solutions which optimize the fastpath for defenses against large-scale attacks often result in compromising the performance of benign traffic [7, 14].

In this work, we propose Sieve, a novel approach to hybrid hardware-software designs that targets adversarial settings to faithfully provide both accuracy and performance. Unlike prior, cache-style exact offloading designs, Sieve works by refactoring a defense NF into multiple highly optimized layers; each layer is a refinement of the one before and is deployed on the target (e.g., P4-programmable switch, SmartNIC, eBPF kernel subsystem, x86 CPU) that offers the best performance and resource usage, along with the desired accuracy guarantees. Sieve begins with coarse-grained approximations of defense functionality instantiated on high-speed, but memory-limited hardware and moves towards fine-grained exact instantiations in memory-rich, but slower software. The key difference between cache and sieve approaches to hybrid NFs is that caches exactly offload a portion of an NF’s functionality to hardware, while Sieve uses approximate, refactored instantiations of potentially redundant functionality across multiple hardware and software layers. Our key insight is that a hybrid NF can embrace accuracy and performance tradeoffs on different targets to provide defense in depth, accepting some approximation error on early, coarse-grained, high-speed hardware layers, but delivering complete accuracy by the deepest, exact, slower software layer.

We observe that benign and adversarial traffic can often be identified with per-flow state (e.g., connection 5-tuple) or some kind of computed identity verification (e.g., SYN cookie check). With Sieve, a P4 switch can be tasked with the first layer of the defense, approximately identifying and dropping the majority of adversarial traffic early in the path, while ensuring high-speed processing for benign traffic. Because the switch layer is memory-limited and keeping exact state is expensive, state can be kept using probabilistic data structures, engineered to allow a small amount of either false positives or false negatives to fulfill a given defense policy. For example, a Bloom filter can be used as a data structure at the Sieve switch, never dropping benign traffic but accepting some adversarial traffic as false positives, which are handled at a later defense layer [1]. Any traffic that was erroneously passed through the first layer is then detected and correctly dropped at the later layers, providing overall defense accuracy along with high performance. In other words, the Sieve’s first layer accepts all wanted traffic while quickly but coarsely blocking the bulk of unwanted traffic, possibly allowing a small amount of unwanted traffic to trickle down to the next layer where a finer-grained exact netting accurately stops the remainder of the unwanted traffic.

We believe that Sieve is an exciting new direction for unleashing accurate and performant hybrid defenses. Early explorations using Sieve for DDoS defenses have shown vast improvements over prior cache-like offloading approaches, reducing end-to-end application latency by 48-84% and reducing server CPU overhead by 33-100%. Our vision is to create a system that automatically generates network defenses with Sieve. A Sieve compiler would take as inputs the defense NF written in a high-level language, topology of available hardware and software targets, and defense policy, and it would then output a refactored, layered defense with overall accuracy and performance.
References


