Research Statement of Prof. Anat Bremler-Barr, January 2022

My primary research interests are networking and network security. I am motivated by the desire to improve the reliability and efficiency of the Internet. My research expands the knowledge on the fundamentals of network algorithmics and networking device design, especially of middleboxes and virtualized middleboxes (i.e., network functions).

In recent years I have focused on the following four topics: 1) Designing Deep Packet Inspection for next generation network devices. 2) Network function virtualization and offloading, with special focus on building efficient network functions in software-defined network environments. 3) Accelerating algorithms using TCAM - I focus on accelerating commonly used algorithms in network devices using ternary content addressable memories. 4) Distributed Denial of Service attacks - I focus on understanding how to build network systems that are resilient to DDoS attacks and how to build efficient DDoS scrubbers. Specifically, I research DNS resilience, which is a weak point in today's Internet and on new cloud architectures, such as Kubernetes. 5) IoT security, focusing on IoT network level security.

This document presents only my main research contributions *in the last ten years* that are aligned with my major research directions. A complete detailed list of my publications appears in my CV, and is available on my homepage. I am also the founder and director of the Deepness Lab which focuses on designing reliable and efficient networks and network devices .

1 Deep Packet Inspection

Deep packet inspection (DPI) is one of the main tasks performed by middleboxes (e.g. Firewall, IDS) in contemporary networks. DPI engines match the packets' payload against a set of patterns (a.k.a. signatures), which, for example, indicate malicious activity. While the general problem of pattern matching is fundamental in computer science, traditional algorithms suffer from limited scalability, narrow solutions to compressed traffic, and security vulnerabilities in the DPI engine. I received an ERC starting grant on this topic, and in my works I tackle various aspects of these important challenges.

1.1 Scalable DPI

First, I tackle the problem of the continual increase in Internet traffic rates, by proposing significantly more scalable design in terms of speed and memory usage.

Compact DFA: Scalable Pattern Matching Using Longest Pattern Matching (ToN 2015): In a joint work with David Hay and PhD student Yaron Koral [26], we have shown how DPI engines can leverage advances in other networking devices to boost their performance. DPI solutions are often based on constructing and traversing a Deterministic Finite Automaton (DFA) that represents the patterns. While this approach ensures deterministic time guarantees, modern IDSs need to deal with hundreds of patterns, requiring us to store very large DFAs, which usually do not fit in fast memory. We proposed a novel method to compress DFAs. Then, the problem of pattern matching is reduced to the well-studied problem of Longest Prefix Match (LPM), which can be solved either in TCAM, in commercially available IP-lookup chips, or in software.

DPI as a Service (CoNext2014): In a joint work with David Hay and PhD students Yotam Harchol and Yaron Koral [24], we designed a system that treats DPI as a service to the middleboxes. DPI is a common task in almost all middleboxes that deal with L7 protocols. Today, traffic is inspected from scratch by all the middleboxes on its route (i.e., in the service chain). With DPI as a service, the traffic is scanned only once, but against the data of all the middleboxes that use the service. We have shown in the paper that DPI as a service significantly boosts performance, improves scalability and robustness, and acts as a catalyst for innovation in the middlebox domain.

Leveraging Traffic Repetitions for High Speed DPI (INFOCOM 2015): In a joint work with David Hay, Post-doc student Shmirit Tzur-David, and PhD student Yotam Harchol [30], we designed a proof-of-concept system that shows that DPI engines can leverage repetitions in the traffic to increase their speed. Our new mechanism makes use of these repetitions to allow the repeated data to be skipped rather than scanned again. The mechanism consists of a slow path and data path. In the slow path frequently repeated strings are identified and stored in a dictionary, along with some succinct information for accelerating the DPI process. In the data path, the traffic is scanned byte by byte but strings from the dictionary, if encountered, are skipped. Upon skipping, the data path recovers to the state it would have been in had the scanning continued byte by byte. Our solution achieves a significant performance boost, especially when data is from the same content source (e.g., the same website).

1.2 DPI on compressed traffic

The large share of compressed web traffic (e.g., due to the uses of mobile devices) produces additional challenges with regard to time and space. Time-wise, decompression must be performed prior to pattern matching. Space-wise, the last 32KB for each parsed session (GZIP requirement) must be stored in order to uncompress the following traffic in the session.

Accelerating Multi-pattern Matching on Compressed HTTP traffic (ToN 2012): In a joint work with my then Master's student, Yaron Koral [25], we have shown that it is faster to perform pattern matching on the compressed data with the penalty of decompression, than to perform it on regular traffic. This surprising result led to the development of algorithm that takes advantage of information gathered by the decompression phase in order to accelerate the pattern matching algorithm. By analyzing real HTTP traffic and real web application firewall signatures, we have shown that up to 84% of the data can be skipped in the string matching scan.

Space Efficient Deep Packet Inspection of Compressed Web Traffic (Computer Communications 2012):

In a joint work with Yehuda Afek and PhD student Yoran Koral [6], we have introduced new algorithms and techniques that drastically reduce the space requirement for performing DPI on compressed traffic. Our proposed scheme improves space by almost 80% and the time performance by over 40%, thus making real-time compressed traffic inspection a viable option for networking devices.

Accelerating Regular Expression Matching Over Compressed HTTP (Infocom 2015): While the above work was focused on string-matching algorithms, in a joint work with Yaron Koral, Michela Becchai, David Hay and Master's student Omer Kochba, we have extended this approach to also work with regular expression algorithms [14].

Decompression-Free Inspection: DPI for Shared Dictionary Compression over HTTP (Infocom 2012): The above works dealt with GZIP compression, which is the most common technique for compressed HTTP. However, in 2008, Google introduced Shared Dictionary Compression over HTTP (SDCH), which was used natively in Google Chrome and Android. In a joint work with David Hay, PhD student Yaron Koral and Post-doc student Shmirit Tzur David [20], we presented a novel pattern matching algorithm that inspects SDCH-compressed traffic without decompressing it first. Our algorithm relies on offline inspection of the shared dictionary, which is common to all compressed traffic and it marks the dictionary with auxiliary information to speed up the online DPI inspection.

We have shown that our algorithm works near the rate of the compressed traffic, implying a speed gain of SDCH's compression ratio (which is around 40%).

1.3 Resilient DPI

Finally, DPI solutions must be resilient to cyber-attacks that aim to knock down the DPI engine. This is especially crucial in the common case where DPI engines are part of a Network Intrusion Detection System.
Making DPI Engines Resilient to Algorithmic Complexity Attacks (ToN 2016): In a joint work with Yehuda Afek, David Hay and PhD students Yotam Harchol and Yaron Koral [2], we built a DPI resilient to complexity attacks. In such attacks an attacker carefully crafts "heavy" messages (or packets) such that each heavy message consumes substantially more resources than a normal message. We demonstrated the vulnerability of existing systems and then designed a novel framework for mitigating such attacks in a multi-core and NFV setting. In our architecture, cores quickly identify such suspicious messages and divert them to a fraction of the cores that are dedicated to handling all the heavy messages. This keeps the rest of the cores relatively unaffected and free to provide the legitimate traffic the same quality of service as if no attack has taken place.

1.4 Signature extraction for the DPI engine

Finding attack signatures is a crucial to protecting Internet sites from Worm attacks and Distributed Denial of Service (DDoS) attacks.
Automated Signature Extraction for High Volume Attacks (ANCS 2013, ToN 2019): In a joint work with Yehuda Afek and PhD student Shir Landau Feibish [7, 8], we presented a pioneering system for zero day attack signature extraction for the DPI engine. Given two large sets of messages - messages captured in the network during peace-time and those captured during attack time - we presented a tool for extracting a set of strings frequently found only in attack time. Using our system, a yet unknown attack can be detected and stopped within minutes of its start time. This algorithm finds popular strings of variable length in a set of messages, through clever use of the classic heavy-hitter algorithm as a building block.

2 Network Function Virtualization and Offloading

Network function virtualization (NFV) aims to reduce the cost of ownership and management of middleboxes by making NFs virtual appliances, running on top of a hypervisor or in a container. While NFV improves on-demand scaling and provisioning, it does not solve other problems such as the limited and separate management of each NF.

OpenBox: Enabling Innovation in Middlebox Applications (SIGCOMM 2016): To cope with these problems, in a joint work with David Hay and our PhD student Yotam Harchol, we have proposed a general framework, called OpenBox [21]. OpenBox is a software-defined framework for developing, deploying, and managing network functions (NFs). The framework decouples the control plane of NFs from their data plane, merges the logic of multiple NFs, and in doing so allows smart NFV deployment in large-scale networks. Network traffic nowadays usually traverses a sequence of NFs (a.k.a. a service chain), many of which process the packet using very similar processing steps (DPI, packet classification). In order to enhance performance, we proposed a novel algorithm for merging the core logic of multiple NF applications in the control plane, such that computationally intensive procedures are performed only once for each packet. OpenBox promotes innovation in the NF domain, since developers can develop and deploy new NFs as OpenBox applications, using basic building blocks (e.g., header classification) provided by the framework.

Another venue of my research is network function offloading to SDN switches. Before the SDN revolution, there was no vendor agnostic API to the routers and switches. In SDN, the switches can be controlled using a standard protocol, commonly the Openflow protocol. The new SDN paradigm opens the door to innovative solutions that are formally required to reside in the middleboxes (or in the network function) and which can now be offloaded to the SDN switches and routers, thus reducing CAPEX and network complexity and improving performance. In my research, I work on the following challenges:

Detecting Heavy Flows in the SDN Match and Action Model (Computer Networks 2018): In a joint work with Yehuda Afek and PhD students Liron Schiff and Shir Landau-Feisbish [1], we proposed efficient algorithms and techniques to detect and identify large flows in a high throughput traffic stream in the SDN match-and-action model. Our large flow detection methods provide high accuracy and present a good and practical trade-off between switch-controller traffic, and the number of entries required in the switch flow table. We use different parameters to differentiate between heavy flows, elephant flows, and bulky flows. We present efficient algorithms to detect flows of the different types. Finally, we have shown how our algorithms can be adapted to a distributed monitoring SDN setting with multiple switches, and how they can be easily scaled with the number of monitoring switches.

Efficient Round-Trip Time Monitoring in OpenFlow Networks (INFOCOM 2017): In a joint work with Master's student Alon Atary, we presented efficient RTT measurements in SDN networks [13]. OpenFlow, the common SDN protocol, does not support RTT monitoring as part of its specification. In this work, we leveraged the ability of OpenFlow to control the routing, and we presented GRAMI, the Granular RTT Monitoring Infrastructure. GRAMI uses active probing from selected vantage points for efficient RTT monitoring of all the links and any round-trip path between any two switches in the network. GRAMI was designed to be resource efficient. It requires only four flow entries installed on every switch in order to enable RTT monitoring of all the links. Moreover, GRAMI uses a minimal number of probe packets and does not require controller involvement during online RTT monitoring.

Network Anti-Spoofing with SDN Data plane (INFOCOM 2017): In a joint work with Yehuda Afek and Master's student Lior Shafir, we proposed network anti-spoofing [11]. Traditional DDoS anti-spoofing scrubbers require dedicated middleboxes, thus adding CAPEX, latency and complexity in the network. In this work we showed that the current SDN match-and-action model is rich enough to implement a collection of anti-spoofing methods. We also developed and advanced methods for dynamic resource sharing to distribute the required mitigation resources over a network of switches. None of the earlier attempts to implement anti-spoofing in SDN directly exploited the match-and-action power of the data switch plane. Our solution requires a number of flow-table rules and switch-controller messages proportional to the legitimate traffic.

Load Balancing Memcached Traffic Using Software Defined Networking (Networking 2017): In a joint work with David Hay, Liron Schiff and Master's student Idan Moyal, we proposed load balancing memcached traffic in SDN [27]. Memcached is an in-memory key-value distributed caching solution, commonly used by web servers for fast content delivery. Keys with their values are distributed between Memcached servers using a consistent hashing technique, resulting in an even distribution (of keys) among the servers. However, as a small number of keys are significantly more popular than others (a.k.a. hot keys), even distribution of keys may cause a significantly different request load on the Memcached servers, which, in turn, causes substantial performance degradation. In a nutshell, our suggested solution, MBalancer, runs as an SDN application and duplicates the hot keys to many (or all) Memcached servers. The SDN controller updates the forwarding tables of the SDN switches and uses SDN ready-made load balancing capabilities. MBalancer offloads requests from bottleneck Memcached servers, and our experiments show that it achieves significant throughput boost and latency reduction.

ORange: Multi-field Openflow-based range classifier (ANCS 2015): In a joint work with Yehuda Afek and PhD student Liron Schiff, we presented the multi-field openFlow based Range classifier [31]. Configuring range based packet classification rules in network switches is crucial to all network core functionalities, including firewalls and routing. However, OpenFlow, the leading management protocol for SDN switches, lacks the interface to configure range rules directly and only provides mask based rules. In this work we presented ORange, the first solution to multi-dimensional range classification in OpenFlow. Our solution is based on paradigms used in state of the art non-OpenFlow classifiers and is designed in a modular fashion, allowing future extensions and improvements. We considered switch space utilization as well as atomic update functionality, and in the network context we provided flow consistency even if flows change their entrance point to the network during policy updates, a property we term cross-entrance consistency

3 Accelerating Algorithms Using TCAM

Ternary content addressable memories (TCAMs) have become highly popular in networking equipment and network processing units. TCAMs are used for high-speed IP lookup and packet classification in switches and routers. Software defined networking (SDN) schemes such as OpenFlow rely on TCAM as the main hardware for their data path. TCAM enables parallel matching of keys against multiple ternary entries. I use this unique memory in order to enhance the performance of traditional computer science problems:

Encoding Short Ranges in TCAM Without Expansion: Efficient Algorithm and Applications (ToN 2018): In a joint work with David Hay, Toky Hel-Or, and PhD student Yotam Harchol, we presented a method for encoding short ranges in TCAM without expansion [23]. Efficient expression of ranges rules is difficult in TCAM. Over the last decade, extensive research has been conducted on range encoding in TCAM but all the proposed methods require that a range be encoded using several TCAM entries (e.g., row expansion). In this work our novel encoding scheme does not impose row expansion, and uses bits proportionally to the maximal range length.

In addition, we have shown how the same technique could be used to efficiently solve other hard problems, such as the nearest-neighbor search problem and its variants [22]. Similarity search, and specifically nearest-neighbor search (NN), is widely used in many fields of computer science such as machine learning, computer vision and databases. However, in many settings such searches are known to suffer from the notorious curse of dimensionality, where running time grows exponentially with d. We devised a novel algorithm and encoding that finds a $2\sqrt{d}$-approximation of this problem and requires only a single TCAM lookup, regardless of the size of the problem (as long as our encoded problem fits in memory). Our nearest neighbor implementation on a TCAM device provides search rates that are up to four orders of magnitude higher than previous best prior-art solution.

Recursive Design of Hardware Priority Queues (Computer Networks 2014): In a joint work with Yehuda Afek and PhD student Liron Schiff, we presented hardware priority queues [10]. We showed how TCAMs can be used to implement priority queues, which are primarily used for scheduling (e.g., within routers or operating systems). Our TCAM implementation requires a small amount of memory and achieves excellent throughput of 100 Gb/s, using TCAM hardware available today. On the theoretical side, our priority queue implementation can be used to sort elements in linear time.

4 Distributed Denial of Service Attacks

DDoS attacks consume the resources of the victim (a server or a network), causing a degradation in performance or even a total failure. I am very interested in the evolution of DDoS attacks, having followed this area from the time I was the chief scientist and co-founder of Riverhead, a company that presented a solution for ISPs to cope with DDoS attacks and was acquired by Cisco in 2014. We are now witnessing DDoS evolution with new types of DDoS attacks. In the early years of DDoS attacks, the attacks were simple high-bandwidth DDoS attacks. That is, these attacks used simple brute force flooding, where the attacker sends as much traffic as possible to consume the network resources, without using any knowledge of the system design. With time, we are witnessing more sophisticated low-bandwidth DDoS attacks that use less traffic and increase their effectiveness by aiming at a weak point in the victim's system design. Such attacks are accomplished by sending traffic that consists of requests that are complicated for the system.

In a series of works together with the Hanoch Levy and PhD student Udi Ben-Porat, we have studied sophisticated DDoS attacks [18, 17, 16].

Vulnerability of Network Mechanisms to Sophisticated DDoS Attacks (ToC 2013) : In [17], the objective of our work was to provide fundamental understanding of sophisticated DDoS attacks, with the goal of making future computer/network designs resilient to them. Our approach is based on a metric that evaluates the vulnerability of a system. We then use our vulnerability metric to evaluate a data structure commonly used in network mechanisms – the Hash table data structure. We show that Closed Hash is much more vulnerable to DDoS attacks than Open Hash, even though the two systems are considered to be equivalent by traditional performance evaluation.

On the Exploitation of CDF based Wireless Scheduling (Computer Networks 2013): In [16], we evaluated different scheduling algorithms in wireless networks and showed that a malicious or a selfish user can harm the fairness and throughput of the system.

Not only have the attack types changed over the time, but so have the DDoS mitigation solutions and the protected systems.

DDoS Attack on Cloud Auto-scaling Mechanisms (Infocom 2017): In a joint work with Eli Brosh and Master's student Mor Sides [19], we studied the auto-scaling mechanism as an important line of defense against DDoS in the cloud. Using auto-scaling, machines can be added and removed in an on-line manner to respond to fluctuating load. It is commonly believed that the auto-scaling mechanism turns DDoS attacks into Economic Denial of Sustainability (EDoS) attacks. Rather than suffering from performance degradation up to a total denial of service, the victim suffers only from the economic damage incurred by paying for the extra resources required to process the bogus traffic of the attack. In refutation of this belief, we presented and analyzed the Yo-Yo attack, a new attack against the auto-scaling mechanism that can cause significant performance degradation in addition to economic damage. In the Yo-Yo attack, the attacker sends periodic bursts of overload, thus causing the auto-scaling mechanism to oscillate between scale-up and scale-down phases. We have demonstrated the attack on Amazon EC2, and analyzed protection measures the victim can take by reconfiguring the auto-scaling mechanism. Recently, in a joint work with the master's student Ronen Ben David, we investigated the resilience of Kubernetes to a YoYo Attack [15]. As containerized cloud applications using Kubernetes gain popularity, they are replacing the VM-based architecture of recent years. We present experimental results on the Google Cloud Platform, showing that even though the scale-up time of containers is much lower than VM, Kubernetes is still vulnerable to the YoYo attack because VMs are still involved. Finally, we evaluate ML models that can accurately detect the YoYo attack on a Kubernetes cluster.

4.1 DNS Resilience to DDoS attacks

Domain Name System (DNS) is a huge, critical, and efficient directory system that (1) became a prime target for new and old fashioned cyber attacks, and (2) is being utilized for additional new applications, not all of which are expected or good. For example, one Distributed Denial of Service (DDoS) attack, known as the Mirai attack, occurred on October 2016. It was sourced from nearly a million IoT devices with a peak of 1.2 Tbps and took down the DYN DNS servers, which in turn made many services such as Netflix and Spotify inaccessible. Unintended and unexpected uses of the DNS system include: key value store, which in turn is used for file verification where the hash of a file is stored as a non-routable IP address; a mechanism for members of a botnet to rendezvous with their hidden command and control (CC) server; and load balancing, which is a good use. A key motivation behind my research is the realization that the additional and unintended new applications of the DNS system affect its vulnerability. First, they make the system more vulnerable and second, they make the mitigation of some attacks harder. In my research, I analyze and model the DNS system with all its recent new uses and mechanisms to understand its bottlenecks and weak points. I focus on fortifying and improving critical components to strengthen and possibly remove the weak points. Finally, I suggest mechanisms to detect and mitigate attacks on the DNS system and protect it from harmful abuses. In a series of works [32, 9, 1, 12], I study the DNS and sophisticated DDoS attacks on DNS.

Mitigating DNS Random Subdomain DDoS Attacks by Distinct Heavy Hitters Sketches (HotWebs 2017): In a joint work with Prof. Yehuda Afek, Edith Cohen, and PhD student Shir Landau-Feibish, we studied a sophisticated attack on the Domain Name System (System): the Random Subdomain DDoS Attack [1]. In these attacks, many queries are sent to a single or several victim domains, yet they include highly varying non-existent subdomains generated randomly. (The recent Mirai attack on DYN is one example.) This attack harms the caching mechanism of the DNS. We designed and implemented novel and efficient algorithms for distinct heavy hitters (dHH). Our algorithms are superior to previous designs in both the asymptotic (theoretical) sense and practically. We used these algorithms to build and implement a system for the detection and mitigation of Random Subdomain DDoS attacks.

NXNSAttack: Recursive DNS Inefficiencies and Vulnerabilities (USENIX 2020):

In a joint work with Prof. Yehuda Afek and the student Lior Shafir, we exposed a new vulnerability and introduced a corresponding attack, the NoneXistent Name Server Attack (NXNSAttack) that disrupts and can paralyze the DNS system [12]. The NXNSAttack generates a storm of packets between DNS resolvers and DNS authoritative name servers. The storm is produced by the response of resolvers to unrestricted referral response messages of authoritative name servers. The attack is significantly more destructive than NXDomain attacks (e.g., the Mirai attack): i) It reaches an amplification factor of more than 1620x on the number of packets exchanged by the recursive resolver. ii) In addition to the negative cache, the attack also saturates the 'NS' section of the resolver caches. To mitigate the attack's impact, we proposed an enhancement to the recursive resolver algorithm, MaxFetch(k), which prevents unnecessary proactive fetches. We implemented the MaxFetch(1) mitigation enhancement on a BIND resolver and tested it on real-world DNS query datasets.

5 Internet of Things

I focus on solutions at the network level for IoT security problems, such as designing a network function that can protect the IoT. This is especially critical since IoT devices themselves are too weak in terms of CPU and memory to run security solutions and protect themselves. The Manufacturer Usage Description (MUD) is an IETF white-list protection scheme that formalizes the authorized network behavior in a MUD file; this MUD file can then be used as a type of firewall mechanism.

NFV-based IoT Security for Home Networks using MUD (NOMS 2020)

In a joint work with Prof. David Hay and Prof. Yehuda Afek, and several students [3, 4] we presented a new scalable ISP level system architecture to secure and protect all IoT devices in a large number of homes. The system is based on white-listing, as in the Manufacturer Usage Description (MUD) framework, and implemented as a VNF. Unlike common MUD suggestions that place the whitelist application at the home/enterprise network, our approach is to place the enforcement upstream at the provider network, combining an NFV (Network Function Virtualization) with router/switching filtering capabilities, e.g., ACLs. The VNF monitors many home networks simultaneously, and therefore, is a highly-scalable managed service solution that provides both the end customers and the ISP with excellent visibility and security for the IoT devices at the customer premises. In [5] we enhanced the MUD framework to deal with P2P traffic, while preserving the current MUD specification. Typically, MUD includes a set of legitimate endpoints, specified either by domain names or by IP addresses, along with the legitimate port numbers and protocols. While these descriptions are adequate when IoT devices connect (as clients) to servers (e.g., services in the cloud), they cannot adequately describe the cases where IoT devices act as servers to which endpoints connect (i.e., P2P traffic).

IoT or NoT: Identifying IoT Devices in a Short Time Scale (NOMS 2020):

Whenever a new device connects to the network, it must be quickly managed and secured using the relevant security mechanism or QoS policy. Thus, a key challenge is to distinguish between IoT and NoT devices in a matter of minutes. Unfortunately, there is no clear indication whether a device in a network is an IoT. In [28], a joint work with Prof. Zohar Yakhini and the master's student Haim levy, we proposed different classifiers that identify a device as IoT or non-IoT, in a short time scale and with high accuracy. Our classifiers were constructed using machine learning techniques on a seen (training) dataset and were tested on an unseen (test) dataset. They successfully classified devices that were not in the seen dataset with an accuracy above 95%.

One MUD to Rule Them All: IoT Location Impact (NOMS 2022):

Analyzing the network behavior of IoT devices, including which domains, protocols, and ports the device communicates with, is a fundamental challenge for IoT security and identification. Solutions that analyze and manage these areas must be able to learn what constitutes normal device behavior and then extract rules and features to permit only legitimate behavior or identify the device. In a recent work [29], we demonstrated that learning what is normal behavior for an IoT device is more challenging than expected. In many cases, the same IoT device, with the same firmware, can exhibit different behavior or connect to different domains with different protocols, depending on the device's geographical location. We present a technique to generalize MUD files. By processing MUD files that originate in different locations, we can generalize and create a comprehensive MUD file that is applicable for all locations.

References

[1] Afek, Y., Bremler-Barr, A., Feibish, S. L., and Schiff, L. Detecting heavy

flows in the sdn match and action model. *Computer Networks 136* (2018), 1–12.

[2] Afek, Y., Bremler-Barr, A., Harchol, Y., Hay, D., and Koral, Y. Making dpi engines resilient to algorithmic complexity attacks. *IEEE/ACM Transactions on Networking 24*, 6 (2016), 3262–3275.

[3] Afek, Y., Bremler-Barr, A., Hay, D., Goldschmidt, R., Shafir, L., Avraham, G., and Shalev, A. Nfv-based iot security for home networks using mud. In *IEEE/IFIP Network Operations and Management Symposium* (2020), IEEE Press.

[4] Afek, Y., Bremler-Barr, A., Hay, D., Shafir, L., and Zhaika, I. Demo: Nfv-based iot security at the isp level. In *IEEE/IFIP Network Operations and Management Symposium* (2020), IEEE, pp. 1–2.

[5] Afek, Y., Bremler-Barr, A., Hay, D., and Shalev, A. Mudirect: Protecting p2p iot devices with mud. In *IEEE iThings* (2021).

[6] Afek, Y., Bremler-Barr, A., and Koral, Y. Space efficient deep packet inspection of compressed web traffic. *Computer Communications 35*, 7 (2012), 810–819.

[7] Afek, Y., Bremler-Barr, A., and Landau Feibish, S. Automated signature extraction for high volume attacks. In *Proceedings of the ninth ACM/IEEE Symposium on Architectures for Networking and Communications Systems* (2013), pp. 147–156.

[8] Afek, Y., Bremler-Barr, A., and Landau Feibish, S. Automated signature extraction for high volume attacks, 2019.

[9] Afek, Y., Bremler-Barr, A., and Peleg, N. Poster: Measuring open resolvers negative caching. In *ACM Internet Measurement Conference* (2021).

[10] Afek, Y., Bremler-Barr, A., and Schiff, L. Recursive design of hardware priority queues. *Computer Networks 66* (2014), 52–67.

[11] Afek, Y., Bremler-Barr, A., and Shafir, L. Network anti-spoofing with sdn data plane. In *INFOCOM 2017-IEEE Conference on Computer Communications, IEEE* (2017), IEEE, pp. 1–9.

[12] Afek, Y., Bremler-Barr, A., and Shafir, L. {NXNSAttack}: Recursive {DNS} inefficiencies and vulnerabilities. In *29th USENIX Security Symposium (USENIX Security 20)* (2020), pp. 631–648.

[13] Atary, A., and Bremler-Barr, A. Efficient round-trip time monitoring in openflow networks. In *INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications* (2016), pp. 1–9.

[14] Becchi, M., Bremler-Barr, A., Hay, D., Kochba, O., and Koral, Y. Accelerating regular expression matching over compressed http. In *INFOCOM 2015 IEEE Conference on* (2015), pp. 540–548.

[15] Ben David, R., and Bremler-Barr, A. Kubernetes autoscaling: Yoyo attack vulnerability and mitigation. In *CLOSER* (2021).

[16] Ben-Porat, U., Bremler-Barr, A., and Levy, H. On the exploitation of cdf based wireless scheduling. *Computer networks 57*, 10 (2013), 2193–2205.

[17] Ben-Porat, U., Bremler-Barr, A., and Levy, H. Vulnerability of network mechanisms to sophisticated ddos attacks. *IEEE transactions on computers 62*, 5 (2013), 1031–1043.

[18] Ben-Porat, U., Bremler-Barr, A., and Levy, H. Computer and network performance: graduating from the "age of innocence". *Computer networks 66* (2014), 68–81.

[19] Bremler-Barr, A., Brosh, E., and Sides, M. Ddos attack on cloud auto-scaling mechanisms. In *INFOCOM 2017-IEEE Conference on Computer Communications, IEEE* (2017), IEEE, pp. 1–9.

[20] Bremler-Barr, A., David, S. T., Hay, D., and Koral, Y. Decompression-free inspection: Dpi for shared dictionary compression over http. In *INFOCOM, 2012 Proceedings IEEE* (2012), IEEE, pp. 1987–1995.

[21] Bremler-Barr, A., Harchol, Y., and Hay, D. Openbox: a software-defined framework for developing, deploying, and managing network functions. In *Proceedings of the 2016 ACM SIGCOMM Conference* (2016), pp. 511–524.

[22] Bremler-Barr, A., Harchol, Y., Hay, D., and Hel-Or, Y. Ultra-fast similarity search using ternary content addressable memory. In *Proceedings of the 11th International Workshop on Data Management on New Hardware* (2015), ACM, p. 12.

[23] Bremler-Barr, A., Harchol, Y., Hay, D., and Hel-Or, Y. Encoding short ranges in tcam without expansion: Efficient algorithm and applications. *IEEE/ACM Transactions on Networking 26*, 2 (2018), 835–850.

[24] Bremler-Barr, A., Harchol, Y., Hay, D., and Koral, Y. Deep packet inspection as a service. In *Proceedings of the 10th ACM International Conference on Emerging Networking Experiments and Technologies (CoNext)* (2014), pp. 271–282.

[25] Bremler-Barr, A., Hay, D., Hendler, D., and Roth, R. M. Efficient detection of errors in associative memory, Nov. 11 2014. US Patent 8,887,026.

[26] Bremler-Barr, A., Hay, D., and Koral, Y. Compactdfa: Scalable pattern matching using longest prefix match solutions. *IEEE/Acm Transactions On Networking 22*, 2 (2014), 415–428.

[27] Bremler-Barr, A., Hay, D., Moyal, I., and Schiff, L. Load balancing memcached traffic using software defined networking. In *IFIP Networking Conference (IFIP Networking) and Workshops, 2017* (2017), IEEE, pp. 1–9.

[28] Bremler-Barr, A., Levy, H., and Yakhini, Z. Iot or not: Identifying iot devices in a short time scale. In *IEEE/IFIP Network Operations and Management Symposium* (2020), IEEE, pp. 1–9.

[29] Bremler-Barr, A., Meyuhas, B., and Shister, R. One mud to rule them all: Iot location impact. In *IEEE/IFIP Network Operations and Management Symposium* (2022), IEEE.

[30] Bremler-Barr, A., Tzur-David, S., Harchol, Y., and Hay, D. Leveraging traffic repetitions for high-speed deep packet inspection. In *INFOCOM* (2015), pp. 2578–2586.

[31] Schiff, L., Afek, Y., and Bremler-Barr, A. Orange: multi field openflow based range classifier. In *Proceedings of the Eleventh ACM/IEEE Symposium on Architectures for networking and communications systems* (2015), IEEE Computer Society, pp. 63–73.

[32] Shafir, L., Afek, Y., Bremler-Barr, A., Peleg, N., and Sabag, M. Poster: Dns negative caching in the wild. In *Proceedings of the ACM SIGCOMM 2019 Conference Posters and Demos* (2019), pp. 143–145.