Context Matters: Accurately Measuring the Efficacy of Denial-of-Service Mitigations

Context Matters: Accurately Measuring the Efficacy of Denial-of-Service Mitigations

Samuel DeLaughter
samd@mit.edu
Massachusetts Institute of Technology
Cambridge, MA, USA

Karen R. Sollins
sollins@csail.mit.edu
Massachusetts Institute of Technology
Cambridge, MA, USA

ABSTRACT

Denial-of-Service (DoS) attacks remain a severe and constant threat to the Internet. While various mitigation strategies have been developed and deployed to defend against these attacks, the community lacks adequate metrics for quantifying their efficacy. Metrics used to quantify DoS *attacks* provide a solid starting point, but extending them to the domain of mitigations is non-trivial. Current metrics don't account for a mitigation's overhead outside periods of attack, and fail to capture important context – differences in attack rate, client behavior, network topology, and various other factors can all dramatically alter the impact of attacks and the efficacy of mitigations. This paper provides a methodology and novel suite of metrics designed to enable more meaningful and contextual measurement of DoS mitigations. To illustrate the benefits of these metrics we conduct experiments in the DeterLab network testbed measuring the efficacy of SYN Cookies, a well-known and widely used mitigation against the ubiquitous TCP SYN flood attack. We show that this mitigation is highly effective in certain contexts but can significantly degrade client quality of service in others. Our goal is to help device owners and network operators determine which mitigations are best suited for their particular context, and to help protocol designers and implementers develop a more DoS-resilient Internet.

CCS CONCEPTS

• **Networks** → **Network performance analysis**; **Network measurement**; **Denial-of-service attacks**; *Network experimentation*; *Network simulations*; *Network performance modeling*.

KEYWORDS

denial of service, metrics, measurement, performance analysis, protocol design

ACM Reference Format:
Samuel DeLaughter and Karen R. Sollins. 2022. Context Matters: Accurately Measuring the Efficacy of Denial-of-Service Mitigations. In *Cyber Security Experimentation and Test Workshop (CSET 2022), August 8, 2022, Virtual, CA, USA*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3546096.3546109

CSET 2022, August 8, 2022, Virtual, CA, USA

ACM ISBN 978-1-4503-9684-4/22/08.
https://doi.org/10.1145/3546096.3546109

1 INTRODUCTION

In a Denial of Service (DoS) attack, malicious actors attempt to disable or degrade some networked service, typically by exhausting some set of critical resources on servers, clients, and/or intervening network devices. The Internet has been vulnerable to such attacks since its inception, as they were never considered a threat in the closed designs of the preceding ARPANET and NSFNet [13]. Attacks have only increased in frequency and severity, with hundreds or even thousands of unique attacks now being launched every day.

The rise of IoT-enabled botnets has caused a trend toward highly distributed and extremely high-volume attacks. These "DDoS" floods first reached the 1 Tb/s threshold in the infamous 2016 Mirai attacks [24], and Amazon reported the first attack over 2 Tb/s less than four years later [1]. Kaspersky has reported record-breaking numbers of DoS attacks in the last two quarters; in Q1 of 2022 alone they observed 91,052 attacks, surpassing the previous quarter by 1.5x and surpassing Q1 of 2021 by 4.5x [12]. These trends are incredibly worrying, as our society has become increasingly reliant on networked services for even its most basic and essential functions.

A wide variety of strategies have been proposed for dealing with these attacks, including traffic filtering and rate limiting, overprovisioning, and modifications to network protocols. Due to the imminence of the threat and the slow pace of standardization, deployment of these mitigations has been largely ad hoc. DoS protection is a growing billion dollar industry [20], and while providers have proven remarkably effective at maintaining service the costs of doing so are poorly understood – users shouldn't need to pay a premium for DoS-safe Internet access. Similarly, kernel developers have implemented non-standard mitigations like SYN Cache and SYN Cookies with little or no empirical evaluation of their overhead or efficacy. At the application layer, many websites rely on mitigations like Cloudflare's Browser Integrity Check [5] which adds multiple seconds of latency to an HTTP request, as well as human-level challenges like CAPTCHA [4] and its successor reCAPTCHA [10] which reportedly take the average user 32 seconds to complete, totaling an estimated *500 years per day* of wasted time [16].

Without proper analysis we have no way of knowing whether the DoS resilience these mitigations claim to provide actually justifies the delays and monetary costs they impose. The few attempts that have been made to conduct such evaluation all suffer from a lack of adequate metrics. Through our own separate efforts to develop and measure DoS mitigations we have identified two key factors that prior work has failed to consider: that a mitigation's efficacy is highly context-dependent, and that deploying a mitigation imposes some amount of overhead which may *reduce* client quality of service (QoS) in certain contexts. The term "context" here is intentionally

vague, as there are innumerable variables that may impact the efficacy of any given mitigation, including but not limited to: the composition and rate of attack traffic, clients' usage patterns and choices of applications, the number of hops between clients and the server, the location of attackers in the network, the capacity and latency of network links, the hardware and operating system used by each device, etc.

This paper presents a novel suite of metrics designed to help account for these factors, thereby enabling more accurate measurements of DoS mitigations both within and across contexts. Context-specific metrics can assist device owners and network operators in determining whether a mitigation is worth deploying in their system, while cross-context metrics can assist protocol designers and Internet architects in determining whether a mitigation warrants widespread deployment or even standardization as a core component of the network stack.

To further motivate our proposed metrics we conduct a set of experiments involving DoS attacks and mitigations in a controlled network testbed. The goal of these experiments is not to evaluate any particular attacks or mitigations – rather, we aim to provide a blueprint for others to conduct such analysis in their own networks, and to highlight subtleties that existing metrics fail to capture and corner-cases they fail to handle. We demonstrate that seemingly minor changes in contextual variables can dramatically alter a mitigation's effects and apparent efficacy. Our hope is that the metrics we define will enable more robust evaluation of DoS mitigations, ultimately leading to reduced costs, improved QoS, and a generally more DoS-resilient Internet.

1.1 Paper Structure

The remainder of this paper is structured as follows. We provide additional background information on the threat of DoS attacks, strategies for mitigation, and existing metrics in Section 2, and define our own metrics in Section 3. Section 4 describes the physical testbed we use for DoS experimentation and presents results of experiments designed to showcase our metrics. We discuss limitations of our metrics and potential future work in Section 5, and conclude with a summary of our findings and recommendations in Section 6.

2 BACKGROUND

This section provides background information required to understand and motivate our metrics and experiments. We overview the threat posed by DoS attacks (§2.1), general approaches to mitigating attacks (§2.2), and current state-of-the-art metrics used to evaluate DoS attacks (§2.3) and mitigations (§2.4).

2.1 DoS Attacks

The goal of a DoS attack is to deny access to some networked service, or failing that to degrade the quality of service users experience. These attacks come in many different varieties, but can be roughly divided into two main categories: **targeted** attacks which exploit vulnerabilities in network protocols and/or bugs in the implementations of said protocols, and **volumetric** attacks which overwhelm servers and network devices with so much traffic that they are forced to drop packets, delay packet handling, close connections, or otherwise degrade service for legitimate clients.

The boundary between these categories is not perfectly clear. Some targeted attacks *require* low-volume communication (such as those that exploit the TCP retransmission timeout [15]), but many simply become more damaging at higher volumes. For example, SYN floods may be considered a targeted attack that exploits the TCP handshake, forcing a server to allocate state for half-open connections that will never be used. Yet SYNs are also one of the most commonly used packet types in high-volume attacks.

We assert that the distinguishing factor between these two manifestations of the same attack is the resource they target – a low-rate SYN flood targets server memory resources, but once the traffic volume crosses a certain threshold the bottleneck shifts to the bandwidth and CPU resources required to receive and forward packets. A key goal of our metrics is to better understand these tipping points, so that protocols can be designed and implemented in ways that preserve constrained resources by leveraging abundant ones.

2.2 DoS Mitigations

Approaches to mitigating DoS attacks also come in many different varieties, which can be divided into three main categories: **traffic filtering** (§2.2.1), **over-provisioning** (§2.2.2), and **protocol changes** (§2.2.3). Our metrics can be applied to evaluate mitigations of any type. Note that we consider efforts to *prevent* attacks from being launched (e.g. by detecting/removing malware or holding responsible individuals legally accountable) as distinct from efforts to *mitigate* attacks that are launched. Preventative efforts are highly valuable but fall outside the scope of this work.

2.2.1 Traffic Filtering. Traffic filtering mitigations attempt to detect malicious packets and divert or drop them before they can cause harm. This is the ideal approach in theory, but can be difficult or even impossible in practice. Examining packets requires computational resources and adds latency to the system, particularly if deep packet inspection (DPI) is employed. It also suffers from a risk of both false positives and false negatives. A false positive means dropping traffic from a legitimate client, effectively causing denial of service in an attempt to prevent it. False negatives are inevitable, as there's always a possibility of some new zero-day attack vector being exploited before filters can be configured to detect it.

Additionally, some forms of attack are fundamentally indistinguishable from legitimate traffic – a malicious SYN typically looks identical to a regular one. It's even possible that malware on some compromised device could attack a networked service while a legitimate process on that device simultaneously tries to access the same service, precluding any attempts to filter by source address. Our mitigation metrics can aid in tuning a filter towards more false positives or false negatives, depending on which bias produces the optimal QoS for legitimate clients.

2.2.2 Over-provisioning. A common approach to DoS resilience employed by content delivery network (CDN) operators like Akamai and Cloudflare is to replicate or over-provision a service so that it can handle spikes in demand, whether they are caused by malicious floods or a genuine shift in user behavior. This approach

has proven highly effective, maintaining service even under massive Tb/s-scale attacks, but its cost is obvious. Say you normally need five servers to maintain adequate QoS but provision 50 in case an attack occurs. You're well prepared for disaster, but spending ten times what's likely necessary. This is a difficult and important trade-off to evaluate – one goal of our metrics is to help determine how much over-provisioning is suitable for a particular service based on the expected scale and frequency of attacks.

2.2.3 Protocol Changes. Some approaches to mitigation operate by modifying network protocols themselves. Notable examples include SYN Cache and SYN Cookies, which tweak the TCP handshake process to reduce or delay (respectively) the allocation of server memory until a client's source address can be verified [8]. These interventions have not been incorporated into the TCP standard but have seen widespread ad hoc deployment, sometimes with a negative impact on performance or even a loss of interoperability. Another example is the randomization of initial sequence numbers in TCP, which has now been in the process of standardization for over 26 years and counting [2, 9].

This delay may seem excessive, but the standardization process is long and complex for good reason. Even the most minuscule modification to or deviation from a specification can have disastrous unintended consequences. Yet mitigations in this category are often deployed without thorough empirical analysis, due in large part to a lack of adequate metrics for quantifying their effects. Even protocols that *are* standardized almost invariably lack sufficient consideration of potential DoS vulnerabilities.

Our hope is that the novel metrics proposed in Section 3 will assist the IETF and similar organizations in deciding what protocols and protocol changes to standardize, and assist device owners and network operators in deciding whether non-compliant mitigations are worth the risk of deploying. They can also allow us to compare the DoS-resilience of two disparate protocols or network architectures, potentially motivating migration from IPv4 to IPv6, HTTP to QUIC, or even TCP/IP to proposed future internet architectures like Named Data Networking (NDN).

2.3 Existing Attack Impact Metrics

In the simplest terms, a mitigation's efficacy is the amount by which it reduces an attack's damage. Therefore, any mitigation metric must be defined relative to some attack impact metric. Prior work by Mirkovic et al. [17–19] makes a strong case for using metrics that capture client quality of service (QoS) as directly as possible. Since different client applications have different resource requirements[1], so-called "legacy" metrics like round-trip-time (RTT), throughput, packet loss, and jitter are not reliable indicators of end user experience. A more direct approach is to use transaction-oriented metrics, wherein clients attempt to complete as many sequential transactions as possible over some period of time. For example, simple transactions may include: establishing a TCP connection, performing a DNS lookup, or transferring a file.

The metrics we present in Section 3 all require *some* method of quantifying client QoS, but can be constructed equally well from legacy DoS impact metrics, transaction-oriented metrics, or even some weighted combination of multiple metrics. All are equally usable, but not equally useful. Different QoS metrics will yield different measurements of mitigation efficacy, and the selection of an appropriate QoS metric is also context-dependent – ideally it should be based on a transaction or set of transactions that models typical behavior of real clients in some context of interest.

[1]Real-time audio requires low latency but also low throughput, while high-definition video streaming requires high throughput but can tolerate higher latency by buffering.

2.4 Existing Mitigation Metrics

Existing research into DoS mitigation efficacy is limited, and the metrics employed are typically a simple side-by-side comparison of performance with/without a mitigation deployed while under attack. Other analyses show timeseries data for some QoS metric and indicate a point in time at which a mitigation is enabled. These simple analyses may yield interesting insights in specific contexts, but often ignore mitigation overhead outside periods of attack as well as key contextual variables, making them insufficient for decision making about when and where a given mitigation should be deployed.

For concrete examples we point to the prior work evaluating SYN Cookies. They are one of the most well-known mitigations against one of the most common DoS attacks, yet we have found only three papers analyzing their efficacy, all of which have shortcomings. First was a 2008 study by Smith and Watrawy comparing different OS implementations – this is an important contextual variable to measure, but their analysis failed to consider the mitigation's overhead outside of attack, used only a single low-volume attack rate (80 kB/s), and tested on an unrealistic network topology with only two devices [22]. A 2018 paper by Echevarria et al. measured SYN Cookies' efficacy in the specific context of network-constrained devices, with similar issues – no consideration of mitigation overhead, a single slow attack rate of 200 packets/second, and a topology of just three devices connected via a single switch [7]. The most thorough analysis of SYN Cookies to date is presented by Scholz et al. in a 2020 pre-print [21]. They compare efficacy for different attack rates, client request rates, kernel versions, and even different variants on the mitigation. We see this sort of cross-context measurement as a large step in the right direction, but it still leaves important variables unexplored: they test on a single simple network topology and measure a maximum attack rate of just 1.4 MB/s.

3 METRICS

This section defines a novel suite of metrics for quantifying DoS mitigations. We begin with simple, existing metrics and build up gradually more complex combinations and formulations thereof. Our starting point in developing these metrics was the observation that all mitigations impose some amount of overhead to deploy, and that the balance between that overhead and the reduction of attack impact is extremely context dependent. To that end we first present metrics for better describing a mitigation's efficacy within a single context in Section 3.1, followed by cross-context metrics in Section 3.2. The former can be used by device owners and network operators to determine which mitigations are best-suited for their specific contexts, while the latter can be used to determine whether a mitigation warrants widespread deployment or even standardization as an integral part of a network protocol.

3.1 Context-Specific Metrics

This section presents metrics for quantifying the efficacy of a DoS mitigation within a specific context, with static attack type, client type, network topology, device hardware, etc. The starting point for all of our metrics is the selection of some basic indicator of client QoS. This could be a legacy metric like RTT, throughput, loss, or jitter, or something more complex. Following prior work by Mirkovic et al. [17–19] we recommend the use of transaction-oriented metrics as described in Section 2.3. QoS can then be defined as either the number of transactions a client can complete in a given period of time, or inversely as the average time it takes to complete a single transaction. Since our aim is to *maximize* QoS, we find the former framing of transactions-per-second (TPS) more intuitive.

The next step is to run four discrete experiments, to control for both separate and combined effects of the attack and the mitigation. We refer to these experiments as follows:

- **UB** (Unmitigated Baseline): No attack, No mitigation
- **UA** (Unmitigated Attack): Attack, No mitigation
- **MB** (Mitigated Baseline): No attack, Mitigation
- **MA** (Mitigated Attack): Attack, Mitigation

We take measurements of our chosen client QoS metric during each experiment, and from these four values we can derive several different metrics for describing attacks and mitigations. First, we refer to the result of the UB experiment as the **baseline** QoS – this represents the ideal scenario. We define the mitigation's **overhead** as $UB - MB$, the amount by which the mitigation *reduces* QoS in the absence of an attack.[2] We define an attack's **threat** as $UB - UA$, the amount by which it reduces QoS when no mitigation has been deployed. We define an attack's **damage** as $MB - MA$, the amount by which it reduces QoS *with* the mitigation deployed. Finally, we define a mitigation's **efficacy** (for a specific context) as the percentage of an attack's potential threat that the mitigation prevents:

$$E = \frac{threat - damage}{threat} * 100 = \frac{(UB - UA) - (MB - MA)}{UB - UA} * 100 \quad (1)$$

The ability to describe a mitigation's efficacy with a single numeric value simplifies the process of determining whether a mitigation is worth deploying. The definition provided in Equation 1 above is already an improvement over current metrics, as it vitally incorporates data about the mitigation's overhead, the baseline QoS, and the threat posed by the attack. Yet this still only applies to a single, extremely specific context.

3.2 Cross-Context Metrics

In this section we extend the efficacy metric defined in Equation 1 above to begin analyzing mitigation efficacy *across* disparate contexts. There are innumerable factors that may constitute a meaningful change in context, including variations in attack type and rate, client application, network topology, device hardware, etc.. We divide these factors into two categories which lend themselves to different forms of analysis: **categorical variables** and **numerical variables**.

[2] Any mitigation with overhead ≤ 0 would more accurately be referred to as an "optimization."

3.2.1 Measuring Across Categorical Variables. Examples of categorical variables include client protocols, device hardware and operating systems, as well as some topology changes. The simplest way to evaluate such factors is through a side-by-side comparison of results from separate sets of the four experiments described above. For a concrete example, let's say we want to compare the efficacy of SYN Cookie implementations in the Linux and BSD kernels. We run a full suite of experiments (with and without SYN floods being launched, and with and without SYN Cookies deployed) using a Linux-based server, then re-run the same experiments with a BSD server. It may be helpful to see raw data from these two scenarios side-by-side, or we can compute their difference to make statements like "the Linux implementation has $X\%$ higher overhead than the BSD implementation" or "the Linux implementation is Y times more effective than BSD's."

To generalize across values of a categorical variable we can assign a weight to each category and compute a weighted average of efficacy or other context-specific metrics. For example, if Linux, BSD, and Windows have (respectively) X%, Y%, and Z% of the OS market share, we might assign their weights as $w_L = \frac{X}{100}$, $w_B = \frac{Y}{100}$, and $w_W = \frac{Z}{100}$. We then multiply each of these by the efficacy observed for the corresponding implementation ($E_{L/B/W}$), sum the result and divide by the sum of the weights to compute a **weighted average of efficacy with respect to OS implementation**:

$$\widetilde{E}_{OS} = \frac{w_L E_L + w_B E_B + w_W E_W}{w_L + w_B + w_W} \quad (2)$$

The utility of such analysis is admittedly questionable. For certain variables it is impossible to enumerate every category, let alone assign them all reasonable weights. Additionally, the observation of an extreme outlier or a drastic difference between two categories is likely to be of more interest than a global average.

3.2.2 Measuring Across Numerical Variables. Numerical variables are those with a range of values (such as attack rate, client request frequency, and bottleneck link capacity), such that we can reason about increases and decreases in value. Our metrics apply equally to both continuous and discrete numerical variables – we can only run a finite number of experiments, so continuous variables must be discretized in practice. Given the robust definition of efficacy in Equation 1, which crucially accounts for the mitigation's overhead and is defined relative to the attack's actual threat and the baseline QoS, accurate cross-context analysis becomes surprisingly simple. A general approach for any numerical variable is to plot a line graph showing the mitigation's efficacy as a function of the variable in question. Simultaneous analysis across two variables can be illustrated with a heat map or by simply adding additional lines.

This gives us a clear tool for quantifying a mitigation's efficacy and its dependence on contextual variables, but doesn't necessarily help us understand *why* such dependencies exist. In some cases, changes in efficacy may be a direct result of changes in baseline QoS, attack threat, or mitigation overhead, so a good first step toward explaining dependencies is to plot line graphs for those metrics as a function of the contextual variable as well. For example, if increasing the client request rate significantly increases a mitigation's overhead, that could explain an inverse correlation between request rate and efficacy. In other cases the root cause may be less obvious

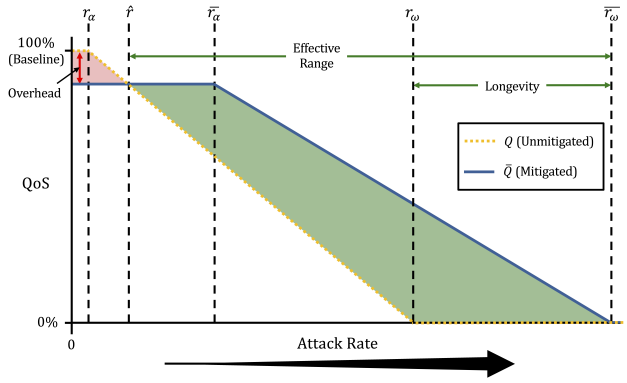

Figure 1: A simplified example of how we might expect a DoS mitigation to change the way QoS depends on attack rate. This dependency can be defined by some function Q parameterized by the attack rate r, which morphs to function $\bar{Q}$ when a mitigation is deployed. Deploying a mitigation imposes some amount of *overhead* on the system, but hopefully improves QoS under some *effective range* of attack rates. A mitigation may also provide some *longevity*, increasing the maximum rate of attack that can be withstood before service is denied completely, but under extremely high rates of attack client QoS will approach zero regardless of the mitigation(s) deployed. We define the mitigation's *efficacy with respect to attack rate* as the area of the dark green shaded region to the bottom right of $\hat{r}$, minus the area of the light red shaded region to the upper left.

– here we suggest turning to traditional forensic metrics such as utilization of CPU and memory[3], or counts of network errors and retransmissions.

The rate of attack is a particularly noteworthy numerical variable which warrants more complex analysis, for two reasons. First, it has universal importance across all contexts, at least for volumetric attacks[4]. We define mitigation efficacy relative to the attack's threat, which is expected to be directly correlated with the attack's rate. Second, since an attack rate of zero is equivalent to our Baseline experiments (UB and MB), we can simply compare the results of UA and MA experiments as a function of attack rate, without needing to separately account for the mitigation's overhead.

Consider Figure 1, which shows a simplified example of how client QoS (depicted as a percentage of the baseline QoS) typically depends on attack rate (Q), and how a mitigation may alter the function describing that dependence ($\bar{Q}$). There will always be some rate of attack r_α below which no measurable degradation of QoS can be observed – here a mitigation's efficacy is undefined since the threat is zero. There is also some rate r_ω above which the threat is so severe that client QoS effectively drops to zero, and a (hopefully higher) rate $\bar{r}_\omega$ at which QoS is zero even with the mitigation in place (because the system is so congested with attack traffic that client requests never reach the server). Between the extremes of r_α and $\bar{r}_\omega$ exists some rate $\hat{r}$, the lowest attack rate for which a mitigation's benefits begin to outweigh its overhead.

The mitigation's **efficacy with respect to attack rate** is visualized as the area of the dark green shaded region to the bottom right of $\hat{r}$, minus the area of the light red shaded region to the upper left. More formally, we can define this as:

$$E(r) = \int_0^{\bar{r}_\omega} \bar{Q}(r)\,dr - \int_0^{\bar{r}_\omega} Q(r)\,dr \tag{3}$$

We refer to the interval $(\hat{r}, \bar{r}_\omega)$ as the mitigation's **effective range**. Ideally a mitigation should *only* be deployed when attack rates are in this range – for lower rates the mitigation's overhead will make it counter-productive, and for higher rates it will have no effect. Yet in practice it may be infeasible to toggle a mitigation on and off precisely when attack rates cross these thresholds, for two reasons. First, attack traffic may be indistinguishable from legitimate traffic which makes it impossible to determine the attack rate – a sudden surge in traffic rates could simply be the result of a demand spike from legitimate clients. Second, the process of enabling and disabling a mitigation may be non-trivial, perhaps requiring a system restart or other configuration change that would disrupt ongoing connections.

In cases where toggling a mitigation on and off is infeasible, we need some way to determine whether a mitigation is worth enabling permanently (or at least for some extended period of time that sufficiently amortizes the cost of toggling). The first step is to identify the distribution of attack rates that the server *expects* to receive. In most cases this will likely be a long-tail distribution, with no attack the vast majority of the time, somewhat frequent low-rate attacks, and very rare instances of extremely high-rate attacks. This distribution is of course impossible to predict exactly, but it can be approximated by monitoring traffic patterns over time. Any such effort should at least yield a better approximation than the uniform distribution.

After defining this expected distribution, we multiply the two functions in Figure 1 by it. We then integrate the resulting functions and take their difference to compute the mitigation's **adjusted efficacy with respect to the expected attack rate**. More precisely: let $Q(r)$ be the function describing QoS with respect to attack rate (r) without the mitigation deployed; let $\bar{Q}(r)$ be the function describing QoS with respect to attack rate (r) with the mitigation deployed; and let $P(r)$ be the function describing the probability distribution of attack rates. Then the adjusted efficacy with respect to expected attack rate distribution is defined as:

$$\widetilde{E}(r) = \int_0^{\bar{r}_\omega} P(r)\bar{Q}(r)\,dr - \int_0^{\bar{r}_\omega} P(r)Q(r)\,dr \tag{4}$$

Assuming $P(r)$ is a long-tail distribution as described above, this formulation will appropriately emphasize the overhead imposed by the mitigation in the common case where no attack is present, and de-emphasize benefits the mitigation provides in rare instances of high-volume floods.

[3]In our experience, system-wide CPU and memory measurements are rarely insightful. Ideally measurements should be specific to the processes involved in the experiment – for memory this means slab- or page-level measurements of memory allocation and utilization, and to truly understand CPU utilization we recommend measuring the portion of cycles spent on each individual kernel function. An exceptional resource for collecting and visualizing these kernel stack traces with the Linux perf tool is made available by Brendan Gregg [11].

[4]Certain targeted attacks may only function at a specific (set of) attack rate(s), as discussed in Section 2.1.

This adjusted efficacy is perhaps the best single-value indicator of whether a mitigation is worth deploying across the contexts of differing attack rates, but still only applies to a single broader context (a single network topology, client application, etc.). After computing $\widetilde{E}(r)$ and/or other complex metrics describing the mitigation's relation to attack rate, we can return to our simpler approaches to cross-context analysis. For instance, how does the mitigation's effective range depend on bottleneck link capacity? How do its $\hat{r}$ and $\widetilde{E}(r)$ values depend on the choice of client application? Answering such questions is vital to understanding when and where a mitigation should be deployed.

4 EVALUATION

This section showcases the metrics we defined in Section 3 through experimentation. Our goal is to demonstrate why these metrics are useful and how they can be used in practice. We do not intend to provide conclusive analysis of any particular mitigations, attacks, protocols, or network topologies. Section 4.1 describes the physical infrastructure of the DeterLab testbed on which our experiments are conducted, including hardware and software specifications of our test devices. Section 4.2 describes the other aspects of our initial experimental context, and presents results from our analysis of SYN Cookies in that context. We then present results showcasing how context changes can alter the mitigation's apparent efficacy in Section 4.3.

4.1 Testbed Environment

Our experiments are all conducted on the DeterLab network testbed. We chose this setting for two main reasons: safety and realism. Safety is a result of DeterLab's sandboxing features – packets destined to the outside Internet are dropped to avoid unintended side-effects of malicious traffic. DeterLab enables us to conduct realistic experiments because it provides access to real physical devices and network links. This yields more accurate measures of DoS attack impact and mitigation efficacy than simulation or emulation environments because it imposes realistic resource bottlenecks.

The main limitation of using a physical testbed is scalability – it's much more expensive to spin up 100 physical servers than 100 virtual machines. DeterLab's resources are time-shared among researchers across the globe, so we are limited to a relatively small number of nodes for our experiments. This precludes us from launching attacks on the scale of newsworthy real-world distributed DoS (DDoS) attacks, but we compensate by using a small set of clients and a commodity server, essentially constructing a microcosm of real Internet activity. A small but accurate test environment is more valuable than a large inaccurate one, at least for our purposes.

All of the nodes we deploy on DeterLab run on dedicated Microcloud instances, each of which has an Intel Xeon E3-1260L 2.4 Ghz quad-core processor and 16GB of RAM. All nodes run Ubuntu 18.04LTS with the Linux kernel upgraded to version 5.16.10 (the latest stable release at the time we started conducting experiments). Naturally, these hardware and software specifications can have a tremendous influence on mitigation efficacy. Specification changes like increasing RAM or the number/speed of CPUs can be considered mitigations themselves, in the form of over-provisioning

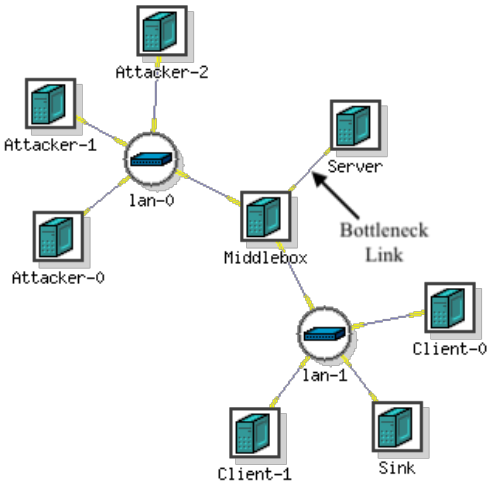

Figure 2: Our network topology. This image was generated automatically by DeterLab upon creating our experiment. We've annotated the bottleneck link, which starts at 1 Gb/s in the initial context but is constrained in later experiments.

discussed in Section 2.2.2. Upgrading to a newer kernel version may add patches that obviate certain attacks or introduce bugs that facilitate new ones.

4.2 Context-Specific Analysis

Our analysis begins by selecting an initial context of interest, and a mitigation we want to measure in that context. For this simple example we run two client nodes which each attempt to establish and then immediately (but cleanly) shut down TCP connections with a single server. They repeat this process for five minutes, attempting to complete as many transactions as possible. We deploy three attacker nodes which launch a TCP SYN flood at a combined rate of approximately 280 Mb/s. Figure 2 shows the network topology of this initial context. In addition to the three device roles mentioned above (server, client, and attacker), we also deploy a **middlebox** and a **sink**.

Middleboxes may be used as vantage points for measurement or to deploy mitigations, but for our purposes it simply performs standard packet forwarding. However, it also allows us to add a realistic bottleneck to the system, which we adjust in our cross-context analysis in Section 4.3 below. The middlebox has three network interfaces: one connected to the client LAN, one connected to the attacker LAN, and one connected to the server via a 1 GB/s duplex link with 10ms latency and drop-tail queueing. Consolidating traffic through this intermediary device before delivering it to the server more closely resembles a real-world distributed attack than connecting all devices on a single LAN, or delivering client and attack traffic directly to separate server interfaces.

The sink is used to preserve realistic effects of the attack's backscatter traffic. Our attacker nodes employ source address spoofing, which means SYN-ACK responses generated by the server have destinations on the public Internet. We need to prevent this backscatter from actually reaching those destinations, but relying

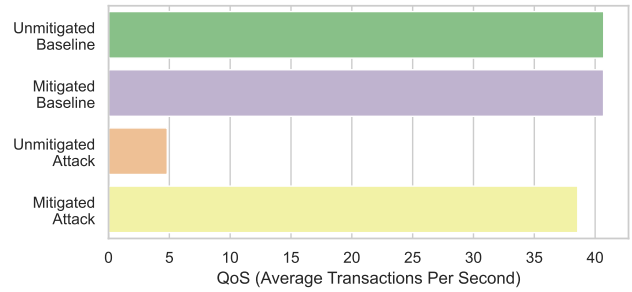

Figure 3: Results of the four primary experiments for our initial context-specific analysis. The attack is a 280Mb/s SYN Flood, and the mitigation is SYN Cookies.

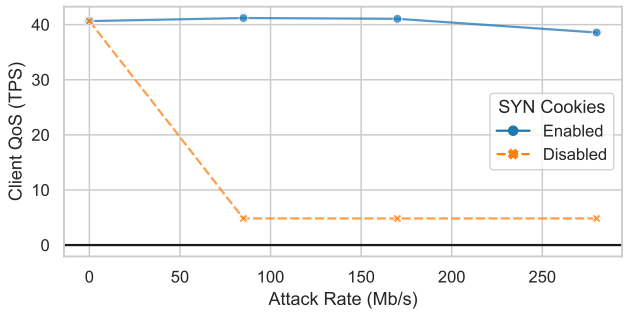

Figure 4: The impact of attack rate on client QoS in our initial context, with and without SYN Cookies.

on Deter's built-in sandboxing would drop SYN-ACKs before the server actually sends them. SYN-ACKs are an important though often overlooked component of a SYN flood – they consume server resources to send, as well as network resources en route to their destination. Our solution is to first route any packets destined to external IPs towards the sink, where they are then dropped via the normal mechanism. Placing the sink closer to legitimate clients will cause SYN-ACKs to share more resources with client connections, likely leading to reduced QoS. At a minimum, the sink should be positioned such that attack backscatter traverses any bottleneck links.

The mitigation we test is SYN Cookies, which prevents TCB state allocation until the the source address can be validated, upon receipt of a corresponding ACK. This is primarily designed to mitigate against spoofing attacks – spoofers cannot receive SYN-ACKs and therefore cannot echo back the cryptographic cookie they contain. We toggle SYN Cookies in the Linux kernel using the command `sysctl -w net.ipv4.tcp_syncookies=?`, where `?` is `0` to disable and `1` to enable.

Our analysis starts with the four experiments described in Section 3.1, measuring client QoS with and without both the attack and the mitigation. The results of these experiments are shown in Figure 3. The ideal, baseline client QoS is 40.71 transactions per second (TPS). As expected we see a small amount of overhead from deploying the mitigation (0.01 TPS), a significant threat posed by the attack (35.88 TPS), and encouraging mitigation efficacy of 94.09%, with the mitigation reducing actual attack damage to just 2.12 TPS.

4.3 Cross-Context Analysis

Our first step is to measure results across different attack rates. We accomplish this by simply repeating the experiments described above with 2, 1, and zero[5] attacker devices instead of 3. We've found that attempting to rate-limit an attacker in the traffic generation script itself typically produces inconsistent and/or unacceptably slow flood rates. If attackers were connected via individual duplex links we could use DeterLab's traffic shaping to rate limit them, but we've been unable to rate-limit LAN connections with sufficient granularity. A detailed analysis should use a much wider range of attack rates, but our resources are limited and this is merely an example.

Figure 4 provides these results in the same manner as Figure 1 – with Q and $\bar{Q}$ corresponding to the lines for SYN Cookies being disabled and enabled, respectively. Surprisingly, the picture painted here is even simpler than our contrived example – since overhead is near zero so is $\hat{r}$, and the effective range extends across the entire distribution of attack rates measured. This is roughly the extent of analysis we have observed in prior work – it paints a fairly rosy picture of SYN Cookies' efficacy, yet one that is far from complete. Our maximum attack rate of 280 Mb/s still pales in comparison to real-world floods, and there are myriad other contextual variables to evaluate.

We lack the physical resources required to test higher-rate attacks, but we do have the means to induce a precise change in topological context by constraining the bandwidth of the link between the server and middlebox. Here DeterLab's traffic gives us full control over what values we can test (within the upper bound of 1 GB/s imposed by hardware). We began with a logarithmic distribution, measuring 1, 10, 100, and 1000 Mb/s bottlenecks, and plotted efficacy as a function of bottleneck capacity for each number of attackers (1, 2, and 3) to quickly scan for *any* noticeable impact. We then ran additional experiments iteratively, testing different values in each case depending on where more granularity seemed most needed. Specifically, we added data points in between existing ones with adjacent x-values and drastically different y-values, as well as on either side of inflection points. Essentially our goal was to smooth each curve as much as possible without having to run an excessive number of experiments. The full set of results is depicted in Figure 5, which represents over 40 hours of total experimentation time (not including time spent on setup or data analysis).

We set out to define and motivate metrics and methodologies, not to draw conclusions about any specific mitigations, but there are a number of undeniably interesting findings here. First and most obvious is that the efficacy of SYN Cookies appears to be severely dependent on the bottleneck link capacity, and drops below zero at surprisingly high capacities for all attack rates tested with a worst-case efficacy of -11.2%. We observe counter-productive effects with bandwidths as high as 400 Mb/s, which is faster than the connections of many real-world servers. The line between effective

[5]It is not strictly necessary to perform separate zero-attacker experiments since their UA and MA results should be identical to the UB and MB results from experiments with one or more attackers.

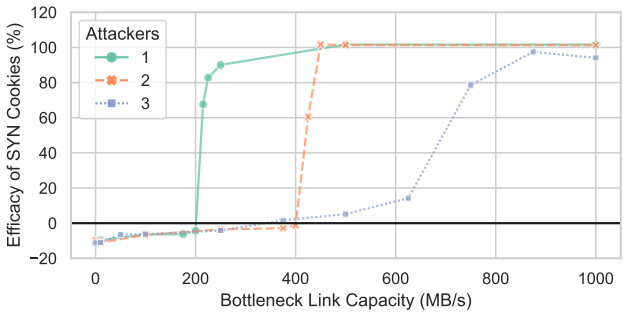

Figure 5: The combined impact of attack rate and bottleneck link capacity on the efficacy of SYN Cookies (as defined in Equation 1).

and counter-productive contexts can be remarkably thin – in the 2-attacker context we observe 101.67% efficacy with a 450 Mb/s bottleneck, 60.55% at 425 Mb/s, and -1.2% at 400 Mb/s.

Perhaps our most interesting finding is that *any* contexts exhibit efficacy over 100%, which indicates that *client QoS improved during the attack*, exceeding even the pre-mitigation baseline. At this time we cannot offer any possible explanation for these results. The performance gain is small (at most 101.67% efficacy), but it appeared consistently over multiple hours of testing and even persisted across a full re-deployment onto a different set of physical devices in DeterLab. We do not observe significant variations in baseline, overhead, or threat metrics across any of our experiments, so the underlying cause of these extreme and varied efficacy values remains unclear. The Internet is an extraordinarily complex distributed system, in which even the slightest changes can have drastic and unpredictable results.

5 DISCUSSION

In this section we discuss limitations of our metrics (§5.1) and potential future work in this domain (§5.2).

5.1 Limitations

The primary limitation of our metrics is that because they average results over extended periods of time, they may gloss over interesting *variations* that occur over the course of a single experiment. For instance, some attacks have a gradual ramp-up that we may want to exclude or analyze separately. Others send traffic in bursts – in this case we may want to evaluate how long it takes QoS to return to the baseline value after a traffic spike, to assess what resources must be allocated to tolerate bursts of a given frequency and severity. If bursts follow a cyclic pattern, experiments should should be run for some multiple of the cycle's duration. The benefit of averaging results over time is that it allows us to abstract away details and view higher-layer trends more cleanly. This is not mutually exclusive with the more traditional approach of presenting raw time-series data – both methods have their own benefits and drawbacks and both may serve as valuable tools for different purposes.

A potential middle-ground would be to include some form of error bars, confidence intervals, or other statistical analysis in plots of our high-level metrics to illustrate the variability of results. However, this is not as straightforward a task as it may sound – a single metric not only averages over time, but also over multiple clients, multiple trials, and potentially multiple contexts as well. There is bound to be some variability in results across each of these dimensions, and it is not clear how to combine them in a coherent manner. As we continue to expand and refine this paper's metrics in future work, we will explore ways to incorporate more statistical analysis.

It is also important to note that our metrics require certain data which are not presently available in all contexts. Perhaps most challenging to obtain is an accurate distribution of expected attack rates, which is required for the $\widetilde{E}(r)$ metric defined in Section 3.2. Many ISPs, CDNs, and other network service providers release periodic whitepapers with aggregated statistics for observed DoS attacks, as well as occasional press releases following particularly notable attack incidents, yet external researchers lack access to the raw packet traces and other fine-grained data from which these reports are generated. The UCSD Network Telescope operated by CAIDA [3] captures backscatter traffic from attacks that utilize randomly spoofed source addresses, but this accounts for only a small fraction of traffic from a small fraction of attacks and does not typically include any of the original attack packets themselves, only the responses they elicit. Though network operators can gain some benefits from analyzing attack traffic they observe in isolation, a complete understanding of global attack patterns will require sharing data across organizations and with third-party researchers. We understand that for network providers to release raw traffic logs would pose serious privacy concerns for their users, but even traces which have been anonymized or filtered to include only malicious packets would be tremendously useful.

It would also be helpful for the research community to establish a set of "honeypot" devices which are intentionally allowed to be infected with malware and used as part of a botnet. This would provide valuable information on the typical behavior of individual attackers, most notably: what protocols and specific packet types they use, whether they attempt source address spoofing, at what rate they generate traffic, and whether that rate is limited artificially or by some resource scarcity on the device. This could allow us to redesign protocols in ways that restrict compromised devices to less dangerous behaviors. For example, if we observed that attackers are commonly bandwidth-limited and that the overhead incurred by attack traffic is primarily per-packet (rather than per-bit), that could make a case for increasing minimum packet sizes to further limit flooding rates. Of course, knowingly allowing a device to participate in a flooding attack would raise serious ethical concerns. Such honeypot devices would need to be sandboxed in a way that allows them to receive instructions from a botnet operator and generate attack traffic for researchers to observe, but prevents that traffic from actually reaching its target.

Without access to such data (from real-world attackers or their targets), our metrics must rely on artificially generated attacks in controlled environments. We implore standards bodies to stress-test new protocols before deployment. Such tests should consider well-known attack vectors, as well as randomly generated packet compositions that *may* prove dangerous.

5.2 Future Work

One essential piece of future work is to expand on our experiments in Section 4.3, performing a comprehensive analysis of SYN Cookies to determine where they should be deployed and why they degrade QoS so heavily in certain contexts. Other protocols that employ a similar mechanism to mitigate spoofing attacks should also be re-examined – notable examples include QUIC Retry packets [14] and the 4-way handshake used by SCTP [23]. Though these are both standard components of their respective protocols, our methodology can be applied by simply framing a change of transport protocol *as* a mitigation.

In addition to properly analyzing and comparing currently deployed protocols and mitigations, we encourage those designing new network protocols and Internet architectures to analyze their DoS resilience as thoroughly as possible before standardization or deployment. We hope to work with the IETF to incorporate our metrics and methodology into their standardization process.

The cases in which we observed efficacy exceeding 100% also warrant further investigation. Any mechanism that can reliably increase client QoS above the baseline would obviously be extremely desirable. How a flooding attack could possibly play a beneficial role in such a mechanism is unfathomable, and yet that is what our results appear to suggest.

We have made our code and experiment data publicly available to facilitate reproduction of our results as well as any relevant future work [6].

6 CONCLUSION

As our results in Section 4.3 clearly demonstrate, context matters for DoS mitigation. A protocol change or other intervention that fully mitigates an attack in one environment may be ineffective or even harmful when deployed elsewhere. We have provided a detailed methodology and robust metrics for analyzing this context-dependent behavior, by providing a definition of mitigation efficacy which accounts for baseline performance, mitigation overhead, and attack threat. As the threat posed by real-world DoS attacks continues to rise, we encourage the thorough empirical analysis of all mitigation strategies to ensure they are deployed in appropriate contexts.

REFERENCES

[1] AWS Shield. 2020. *Threat Landscape Report - Q1 2020.* White Paper Q12020. Amazon Web Services. 9 pages.

[2] Steven M. Bellovin. 1996. *Defending Against Sequence Number Attacks.* Request for Comments RFC 1948. Internet Engineering Task Force. https://doi.org/10.17487/RFC1948

[3] CAIDA. 2018. The UCSD Network Telescope. https://www.caida.org/projects/network_telescope/.

[4] CAPTCHA. 2022. The Official CAPTCHA Site. http://www.captcha.net/.

[5] Cloudflare. 2022. Understanding the Cloudflare Browser Integrity Check. https://support.cloudflare.com/hc/en-us/articles/200170086-Understanding-the-Cloudflare-Browser-Integrity-Check.

[6] Samuel DeLaughter and Karen Sollins. 2022. samd/CSET22. https://github.mit.edu/samd/CSET22.

[7] Juan Jose Echevarria, Pablo Garaizar, and Jon Legarda. 2018. An Experimental Study on the Applicability of SYN Cookies to Networked Constrained Devices. *Software: Practice and Experience* 48, 3 (2018), 740–749. https://doi.org/10.1002/spe.2510

[8] Wesley M. Eddy. 2007. *TCP SYN Flooding Attacks and Common Mitigations.* RFC 4987. Internet Engineering Task Force.

[9] Fernando Gont and Steven Bellovin. 2012. *Defending against Sequence Number Attacks.* Request for Comments RFC 6528. Internet Engineering Task Force. https://doi.org/10.17487/RFC6528

[10] Google. 2022. reCAPTCHA. https://www.google.com/recaptcha/about/.

[11] Brendan Gregg. 21. CPU Flame Graphs.

[12] Alexander Gutnikov, Oleg Kupreev, and Yaroslav Shmelev. 2022. *DDoS Attacks in Q1 2022.* White Paper Q12022. Kaspersky Securelink.

[13] IAB, Mark J. Handley, and Eric Rescorla. 2006. *Internet Denial-of-Service Considerations.* Request for Comments RFC 4732. Internet Engineering Task Force. https://doi.org/10.17487/RFC4732

[14] Jana Iyengar and Martin Thomson. 2021. *QUIC: A UDP-Based Multiplexed and Secure Transport.* Request for Comments RFC 9000. Internet Engineering Task Force. https://doi.org/10.17487/RFC9000

[15] Aleksandar Kuzmanovic and Edward W. Knightly. 2003. Low-Rate TCP-targeted Denial of Service Attacks: The Shrew vs. The Mice and Elephants. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '03).* ACM, New York, NY, USA, 75–86. https://doi.org/10.1145/863955.863966

[16] Thibault Meunier. 2021. Humanity Wastes about 500 Years per Day on CAPTCHAs. It's Time to End This Madness. http://blog.cloudflare.com/introducing-cryptographic-attestation-of-personhood/.

[17] Jelena Mirkovic, Alefiya Hussain, Sonia Fahmy, Peter Reiher, and Roshan K. Thomas. 2009. Accurately Measuring Denial of Service in Simulation and Testbed Experiments. *IEEE Transactions on Dependable and Secure Computing* 6, 2 (April 2009), 81–95. https://doi.org/10.1109/TDSC.2008.73

[18] Jelena Mirkovic, Alefiya Hussain, Brett Wilson, Sonia Fahmy, Peter Reiher, Roshan Thomas, Wei-Min Yao, and Stephen Schwab. 2007. Towards User-Centric Metrics for Denial-of-Service Measurement. In *Proceedings of the 2007 Workshop on Experimental Computer Science (ExpCS '07).* Association for Computing Machinery, New York, NY, USA, 8–es. https://doi.org/10.1145/1281700.1281708

[19] Jelena Mirkovic, Peter Reiher, Sonia Fahmy, Roshan Thomas, Alefiya Hussain, Stephen Schwab, and Calvin Ko. 2006. Measuring Denial Of Service. In *Proceedings of the 2Nd ACM Workshop on Quality of Protection (QoP '06).* ACM, New York, NY, USA, 53–58. https://doi.org/10.1145/1179494.1179506

[20] Mordor Intelligence. 2022. DDOS Protection Market | 2022 - 27 | Industry Share, Size, Growth - Mordor Intelligence. https://www.mordorintelligence.com/industry-reports/ddos-protection-market.

[21] Dominik Scholz, Sebastian Gallenmüller, Henning Stubbe, Bassam Jaber, Minoo Rouhi, and Georg Carle. 2020. Me Love (SYN-)Cookies: SYN Flood Mitigation in Programmable Data Planes. https://doi.org/10.48550/arXiv.2003.03221 arXiv:2003.03221 [cs]

[22] Craig Smith and Ashraf Matrawy. 2008. Comparison of Operating System Implementations of SYN Flood Defenses (Cookies). In *2008 24th Biennial Symposium on Communications.* IEEE, Kingston, ON, Canada, 243–246. https://doi.org/10.1109/BSC.2008.4563248

[23] Randall Stewart. 2007. *Stream Control Transmission Protocol.* RFC 4960. Internet Engineering Task Force.

[24] Nicky Woolf. 2016. DDoS Attack That Disrupted Internet Was Largest of Its Kind in History, Experts Say. https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet.