Reval: A Tool for Real-time Evaluation of DDoS Mitigation Strategies

Rangarajan Vasudevan, Z. Morley Mao
University of Michigan

Oliver Spatscheck, Jacobus van der Merwe
AT&T Labs — Research

Abstract

There is a growing number of DDoS attacks on the Internet, resulting in significant impact on users. Network operators today have little access to scientific means to effectively deal with these attacks in real time. The need of the hour is a tool to accurately assess the impact of attacks and more importantly identify feasible mitigation responses enabling real-time decision making. We designed and implemented *Reval*, a tool that reports DDoS attack impact in real time, scaling to large networks. This is achieved by modeling resource constraints of network elements and incorporating routing information. We demonstrate the usefulness of the tool on two real network topologies using empirical traffic data and examining real attack scenarios. Using data from a tier-1 ISP network (core, access and customer router network) of size in excess of 60000 nodes, Reval models network conditions with close to 0.4 million traffic flows in about 11 seconds, and evaluates a given mitigation deployment chosen from a sample set in about 35 seconds. Besides real-time decision support, we show how the simulator can also be used in longer term network planning to identify where and how to upgrade the network to improve network resilience. The tool is applicable for networks of any size and can be used to analyze other network anomalies like flash crowds.

1 Introduction

Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks are on the increase in today's Internet [1]. They have morphed into extortion tools to attack both small and large businesses alike [2]. Since a large number of customers share the same network infrastructure, attack traffic traversing an ISP network causes service loss not only to the targets of the attacks but also to other customers. Therefore it is important for network operators to mitigate the impact of DDoS attacks to continue providing guaranteed service. For this, network operators first need to understand the resilience of their networks to attacks, and secondly require a method of determining the best mitigation strategies to follow. Both of these need to be performed in *real time* to be of benefit to the customers. As soon as an attack occurs, attack impact analysis is required to provide real-time network information in terms of which links are overloaded, which customers are affected *etc*. This in turn guides network operators in real-time decision making to evaluate and choose the best combination of mitigation responses from the set of mechanisms available in the network.

To perform attack impact analysis in real time, measurement studies on attacks — either direct [3, 4, 5, 6] or indirect [7] — can be conducted. To obtain a complete network-wide view, it would require instrumenting each router in the network with a measurement setup to collect relevant data resulting in excessive overhead and cost. Even if such measurements are instrumented network-wide, there is still a need to aggregate data to generate the combined report for network operators, requiring prohibitive communication overhead and excessive delay. Furthermore, measurement data alone do not provide information on how to mitigate against attacks as defense options still need to be evaluated and ranked.

Our pragmatic approach is to use simulations combined with partial measurement data collected from the network without requiring complete network link-level measurement instrumentation. Simulations coupled with routing analysis provide a holistic view identifying attack impact on the entire network. To meet real-time requirements, we have realized *informed optimizations* to the common-case tasks such as route computation in our tool, and choose the right granularity to provide attack impact information on the fly.

Deciding the best attack mitigation strategies by trial-and-error through actual network deployment is too costly and time consuming to carry out in real networks. To avoid this expense, we argue again that simulation

is a preferred avenue of pursuit, as it allows low-cost, fast analysis of various attack scenarios and network conditions without actually deploying them on the physical network. In short, our simulator provides both *real-time decision support* as well as *long-term network design analysis* capabilities for operators of even large-scale IP networks involving tens of thousands of network elements with an order of a million traffic flows traversing the network simultaneously.

The rest of the paper is organized as follows. We first describe our tool Reval in Section 2. A few simulation scenarios along with their results for the Abilene network are presented in section 3. Then, our experience using the tool in evaluating defense mechanisms for a large ISP network is presented in Section 4. Finally, we discuss related work in Section 5 and conclude.

2 Reval Description

In estimating the impact of attacks in real time, today's network operator has to consider data of many different types like network topology, routing, traffic matrices *etc.* To obtain the right mitigation for an attack, she has to use her experience and intuition in arriving at an optimal answer from multiple impact estimations. Drawing from our experiences with a large tier-1 ISP, we find that this process is currently manual and very complex to realize in real time. The simple solution of static deployment of defense mechanisms does not work in practice as we demonstrate in Section 4. Therefore, we require a tool that automates the process of impact analysis and evaluates desired mitigation scenarios tailor-made to an attack handling the complexity of the whole task as a routine. Using this tool, the network operator needs only intervene in specifying the mitigation mechanism to evaluate. Then, once the tool finishes execution, the network operator can choose the mitigation deployment scenario that offers the desired defense for her network. We provide Reval as the tool to fulfill this role in any network where mitigation deployment is not immediately obvious.

The goals of an attack analysis tool are to: (i) Evaluate attack impact on a given network; (ii) Evaluate different defense mechanisms; (iii) Emulate real conditions as close as possible (routing algorithms, best practices, network element behavior); (iv) Be configurable, user-friendly, and extensible; (v) Be scalable and efficient, providing *real-time* feedback; (vi) Execute on off-the-shelf workstations with modest resources. No existing tools/simulators achieve all the above goals in a satisfactory manner, as we discuss in Section 5. The overall design of Reval is depicted in Figure 1. In brief, it takes as input the topology of the network under analysis, background traffic (not malicious or bad) information, attack traffic information, attack mitigation policy configuration, and other preferences for generating various attack impact metrics. The tool provides capabilities for what-if analyses to understand how well suggested mitigation responses work, and also supports long-term network planning.

2.1 Design Space

Before studying the impact of an attack on a network, we first state the type of attacks we consider and how we calculate their impact. We intended Reval to work with any DDoS attacks that cause impact at the network level — either to access links or core links. DoS attacks like SYN floods and ICMP floods exploit protocol behavior and send large traffic volumes destined towards an end-user or a small set of end-users. If these attacks are not large enough to overload any part of the ISP network, then they are not considered in our tool. Having said that, we can use Reval in numerous scenarios including network management scenarios like capacity planning, overprovisioning; other impact scenarios like flash crowds; analyzing target(s)-specific attack impact and defense mitigation. In this paper, we outline one particular use of Reval while its design readily lends itself to any situation where network-level analysis is required.

Given that ISP networks are set up for profits which predominantly comes from customers who pay to use the network, a reasonable way of defining impact is to relate it with the degree to which customers can communicate without noticing any abnormal performance. We chose to capture *this* way of calculating impact in our tool at a per-customer flow level. That is, network impact of an attack is directly related to the number of customer flows impacted due to performance degradation. By computing attack impact at the flow-level, we avoid the need to carry out time-wise expensive packet-level simulations. Also, we omit the notion of a simulation clock in Reval. The main drawback of these modeling decisions is that the tool is unable to provide time-based fine-grained statistics such as percentage of packets dropped, duration of link and router overloads *etc*. However, as we show in Sections 3 and 4, these statistics are not always required for assessing attack impact.

The data flow of Reval is depicted in Figure 1. Note that the input required for simulation is data that is typically collected by network operators in day-to-day operations. Network topology information is directly extracted from router configuration files; traffic information is directly obtained from Netflow data typically generated by routers in the network. Similarly, the availability of commercial DDoS detection systems [8, 9] allow operators to collect attack traffic information. For example, the tier-1 ISP that we studied has a DDoS detection system deployed at key locations, which, together with Netflow and routing data allows us to construct attack flows for

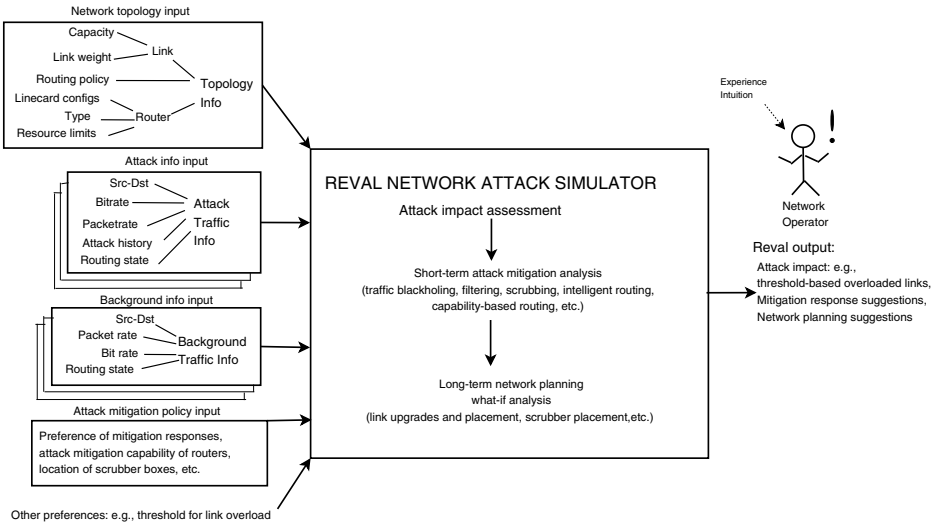


Figure 1: Reval architecture.

the whole network. All this data can be collected from the network and processed in real time while information that changes infrequently, like the network topology, can be updated as required.

2.2 Implementation

In this section, we describe how we build our tool to realize the above goals and capture attack impact at the flow-level. First, we highlight the core functionality of Reval and then discuss various features of the simulator that make Reval an attractive tool for network operators.

2.2.1 Core functionality

We now describe how the information in Figure 1 is used within Reval. The simulator reads in the **topology information** in terms of links and routers and constructs a graph in the Stanford GraphBase [10] format that provides an efficient and well-abstracted representation. We store all additional information of network elements where appropriate by suitably adding to the SGB library data structure. Router types and line card configurations are stored at a per-vertex level while link information of capacity and link weights are stored at a per-arc level. For each network element, resource limits are also stored if applicable.

The **attack information** shown in Figure 1 comprise of multiple traffic flows which together form the ingress-router-interface to egress-router-interface attack traffic matrix. The following procedure is then used in setting up each traffic flow in the matrix: (a) The vertices in the graph corresponding to the ingress and egress routers of a flow are identified; (b) Using the link-state shortest path routing, *e.g.,* as in the OSPF protocol, the shortest path between the source and destination vertices is found. We have implemented the industry best-practice of splitting traffic flows across equal-cost paths as well; (c) The flow is set up along the shortest path(s) updating the state information of each link carrying the flow. This includes properties such as link utilization, amount of traffic carried, and number of flows carried.

As depicted in Figure 1, the **background traffic information** is also specified as a collection of flows specified at a per-router interface level. These traffic flows together form the background traffic matrix. The procedure for setting up background traffic flows is executed once all attack flows are setup and is similar to that outlined above. At every network element where statistics are updated while setting up a flow, we check whether any limits are violated. If so, the element is said to be overloaded and the background traffic flow currently being set up is said to be impacted. The user of the simulator can specify a threshold of when a network element is considered overloaded. Here, a network element could be something as specific as a line card in a router or a backbone link in the core network.

Other input information required is related to the particular use of the simulator like evaluating attack mitigation policies under certain network conditions, and assigning values for parameters of the simulation based on the experimental scenarios desired.

2.2.2 Peripheral functionalities

Routing algorithm support: The default routing algorithm is the shortest path routing which is widely deployed in most IP networks. We also provide implementations of two other routing algorithms that examples of dynamic routing — namely, the max-flow routing algo-

rithm (using ideas from [11]) and a custom load-sensitive routing algorithm.

Defense mechanisms support: We support multiple defense mechanisms that can be enabled via a configurable file-based interface to the simulator. A modified version of the pushback algorithm [12] is supported. In addition, we also provide means of enabling packet filters with varying capabilities across any set of routers. Though we do not simulate traffic flows at the packet-level, this feature is nevertheless useful as it abstracts the inner workings of the filter while maintaining its effect. As we describe in Section 3, we use this filtering ability to realize *selective blackholing of traffic*, in which a subset of routers drop traffic to a particular destination. Selective blackholing is done local to each router.

A third defense feature which we implement is traffic scrubbing. Scrubbers are "boxes" deployed in the network that take in traffic and scrub the "bad" traffic from the "good" traffic [13]. Again, rather than concerning ourselves with the algorithms used for scrubbing, we abstract the effect of these scrubbers. Scrubbers can be enabled at multiple network locations in the simulator. Since scrubbers are physical boxes, there are only a few scrubbers that are deployed in a real network due to cost considerations. An intelligent routing platform such as RCP [14] could be used to realize both the above defenses in a real network.

Reval also provides the right framework and extensibility to evaluate defense mechanisms not yet common in today's networks. For example, it can calculate the amount of bandwidth required for control traffic in a capability-based DoS defense mechanism [15]. Towards end-host protection from DoS attacks, [16] suggest changes to the Internet architecture. However, the effect of some of these changes might be overridden by security policies of ISP core networks. Reval helps quantify the effects of core network security policies.

Analysis support: One of the useful analysis support functions that Reval provides is the simulation of the hypothetical case when the particular attack under consideration had been larger in terms of volume rate. This is achieved by multiplying the traffic volume by a constant factor called the "scaling factor" and then simulating using this scaled-up attack. In the future, we intend to scale other attack properties to obtain a more comprehensive picture. In our simulations described later, we are particularly interested in computing the greatest scaling factor (GSF) by which an attack can be scaled before overloading a core or access link.

Granularity of Simulations for Impact Assessment: Simulation granularity affects the quality of results. As simulations become more fine-grained, one would expect the accuracy of results obtained to improve and, in the limit model the real network *exactly*. The trade-off is higher complexity of simulations. We have explored alternative levels of granularity of simulations, one of which simulates the packet-rate limits of network elements. Routers typically have limitations not only in terms of the volume of traffic they can handle but also in the number of packets serviced owing to a fixed number of packet buffers [17]. In fact, operational experience suggests that the number of packets serviced decreases as more advanced security features are enabled in a router (since more work is required in processing each packet). From an offline study using this simulator, we found that typical attacks impact networks not only via their bit rate but also sometimes due to their packet rate. For the example network in the Section 3, we carry out capacity-based simulations only because details regarding the routers used in that network (and associated packet limits) are not publicly available. However, our simulator has the capability to carry out simulations at more than one granularity level and can be extended to other dimensions of granularity as well.

	Large Tier-1 ISP $>$ 60000 nodes, 0.4 million flows
Real-time	**11 secs, 50 MB**
w/o Vertex Hash Table	400 secs, 49 MB
w/o Dynamic Programming	900 secs, 45 MB
w/o Graph Reduction	$>$ 4 GB
w/o Functional Reuse (for 5 iterations)	11$\times$5=55 secs, 50 MB (versus 4.5+(11-4.5)$\times$5=37 secs)

Table 1: CPU time and memory utilization: demonstrating performance impact of each optimization.

Impact metrics: A sizable number of impact metrics can be enabled at the time of execution of the simulator. Metrics range from network properties like network core link utilization, the GSF, network-geographic distribution of traffic, network throughput *etc*.

Miscellany: The tool is modular, extensible, configurable and robust to input data errors. The simulator also handles network data as input feeds and executes on an automated basis within the tier-1 ISP.

2.3 Simulator optimizations

Below we enumerate the design optimizations of our simulator. Note that the main goal of these optimizations is to reduce simulator runtime. Table 1 presents a summary of the CPU resource utilization for the real-time version of the simulator (highlighted in **bold**) where one iteration of the core functionality is executed (except for the last entry). The table also highlights the contributions provided by the various optimizations. For our simulations, we used a machine with 2 Intel® Xeon™CPU of 3.60 GHz processor speed each sharing an overall RAM of 4 GB running on the RHEL OS with Linux kernel ver-

sion 2.4.21. Program code, written in C, was compiled using gcc (version 3.2.3) and the "-O 4" flag.

Hash Table of Vertices: Network elements are easily accessed in constant time using indices to the vertex array while network input data like Netflow records typically use names to refer to network elements. To avoid converting retrieval into an $O(|\text{Vertices}|)$ operation, we construct a hash table where each element is hashed using its name as the key while the hash value stored is its array index in the graph. The cost of constructing the hash table is amortized over the simulations.

For our implementation of the hash table, we directly adopt the hash algorithm and implementation provided by [18]. In our use of the hash table, we store the number of the vertex with its name as the hashing key. From Table 1, we see that using this hash table has sacrificed a small amount of memory in return for two orders of magnitude improvement in time.

Pre-computation of Shortest Paths: ISP network topologies do change but typically not over short time scales (unless there are failures). Most changes happen to customer connectivity to the access network while the core and access networks themselves change infrequently. Also, ISP networks are hierarchically organized: customers connect to access networks which funnel data towards a hub or a backbone router before entering the core network. This hierarchy invariably creates multiple equal-cost paths across access router interfaces but the multiplicity is inherent in the ISP network only and is independent of the customers.

Both the above observations led us to take the approach of pre-computing shortest paths across routers within the ISP network. When a particular traffic flow needs to be set up between customers, we first determine the respective access routers these customers connect to. In our data structure for the network, this takes a single lookup only, since customers are modeled as connecting to a single access router in the ISP network (we model each access interface as connecting to a unique customer of the ISP network; this does not however reduce the accuracy of simulation since the effects of multi-homing would be reflected in the traffic matrices input to the simulator). The pre-computed shortest path(s) between the routers is then used to set up the flow. This way, we can avoid repeated shortest path computations. A useful side effect of pre-computation is that we can accurately model equal-cost paths by splitting traffic at each router according to the number of equal-cost paths available at that router. This is in relation to the shortest paths computation natively available in the SGB library.

Dynamic Programming to Compute Equal-Cost Paths: The above description shows the benefits of pre-computing shortest paths. However, the process of computing *all* equal-cost shortest paths between each node pair within the ISP network can itself be time-consuming. This is especially true as the typical size of an ISP can be on the order of at least several thousand nodes. So, we adapt the dynamic programming-based all-pairs shortest paths algorithm described in [19]. In particular, we modify it to incorporate the computation of all equal-cost shortest paths of a select set of nodes which we discuss in the next bullet point.

Table 1 illustrates the benefit of dynamic programming over the shortest paths computation native to the SGB software code. We observe significant improvement in run-time in return for a modest 5 MB increase in storage.

Graph Reduction: As stated earlier, we assume that every customer connects to a single access router. So, customer traffic at the time of entering and exiting the network has a single choice of next-hop. Hence, for path computation, it is enough if we consider the core and access network of the ISP alone leaving the customers out of the calculations. This reduces the size of the graph we have to consider for our path computation by two orders — from in excess of 60000 nodes (core, access and customer routers) to little more than 600 nodes. The entry in Table 1 pertaining to this optimization illustrates why, more than just a desired optimization, this is required to execute the simulator on off-the-shelf workstations.

Functionality Re-use: Often the same input data is used in multiple runs of a common piece of simulator code while varying just the parameters fed to the simulations. This could either be within a single scenario or across multiple scenarios. In either case, much time is saved by reusing functionality that is required but remains the same across different iterations. For example, one very useful feature to have is to read in the network topology once only while resetting the network state at each network element every time attack impact needs to be computed given a parameter change only (without changes in network topology). The cost savings in not having to re-read the topology more than offsets the cost involved in visiting every vertex and reinitializing its property variables. Furthermore, storing such data in memory does not make the memory usage prohibitive as is evident from Table 1.

An example where the benefit of functional reuse can be observed is the scenario of computing the GSF value before core network overload. This computation requires changing the scaling factor and re-executing the core functionality code of the simulator. The entry in Table 1 provides the expressions which captures the time required in the two cases when this optimization is and is not enabled (at 5 iterations). For a typical attack on a large tier-1 ISP network, the average value of iterations is 5 which results in an execution time of 37 seconds with optimization and 55 seconds without. Note that in the real-world where topologies change by small

amounts across consecutive days, we can reuse already initialized topology information across multiple simulation runs and simulation scenarios thus amortizing further the one-time computational overhead of 4.5 seconds.

Choice of Data Representation: The Stanford GraphBase (SGB) model [10] provides a useful abstraction of network data where each vertex abstracts a network element and each arc a network link. It also provides a convenient mechanism for storing arbitrarily complex data structures required to maintain state local to each vertex and arc. Moreover, the memory usage is optimum owing to a byte array storage of data. We base our implementation ideas on the GT-ITM project [20] in its use of the SGB model for topology modeling.

Others: Apart from the above techniques, we also tried memory mapping the files comprising the attack and background traffic matrices using the *mmap* command. The reasoning behind this was that a typical traffic matrix file comprises of close to tens of MBs of traffic flows and disk I/O was thought to be a bottleneck in making the simulator run in real time. However, as it turned out, memory mapping proved time-wise slightly more inefficient than the simple read/write functions. This was probably because of a Linux-specific implementation design which rendered read/write operations faster when the file was read sequentially.

3 Case Study: Abilene

The primary use of our tool has been on a large tier-1 ISP but due to the proprietary nature of the data, we cannot reveal detailed results of our experience using the tool. Instead, we discuss at a high-level results from the tier-1 ISP network in Section 4. Here we present a case study use of our real-time simulator on an example network, the Abilene network using publicly available data.

Based on our experience, network operators facing an attack typically choose a particular mitigation strategy without evaluating all available defense options due to lack of time. With Reval, we provide network operators with a systematic way of evaluating available defense options in a real-time manner. We illustrate this point in this case study by considering two real-time attack mitigation mechanisms: traffic scrubbing and selective blackholing (described in Section 2).

The Abilene Network is an education backbone network part of the Internet2 initiative [21]. The core of the network is depicted in Figure 2, and comprises of 11 routers located in various cities across the United States and is believed to be carrying about 1% of all traffic in North America [22]. For our simulation scenarios, we consider the immediate access network of Abilene as well (not depicted in Figure 2). This gives a total of 11 core and 60 access routers. We use the traffic matrices

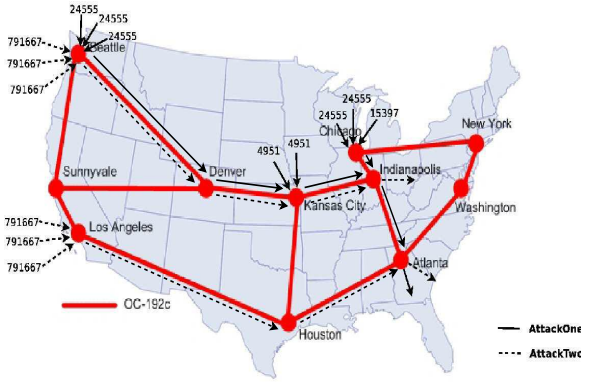


Figure 2: Core of the Abilene Network. The ingress and egress PoPs along with traffic flow rates (in kbps) for attacks used in the case study are also depicted. AttackOne sends traffic from 3 access routers in Seattle, 2 in Kansas City and 3 in Chicago to 1 target access router in Atlanta. AttackTwo sends traffic from 3 access routers in Seattle and 3 access routers in LA to 1 target each in Atlanta and Indianapolis.

generated using the generalized gravity and simple gravity methods discussed in [23, 24]. Since we require traffic matrices at a per access router level, we use the gravity model to interpolate from the PoP-level TMs [23].

In what follows, we first qualitatively analyze the bottleneck points in the Abilene Network. This helps us obtain an understanding on the likely points of attack impact. We then use this information to manually design simulation scenarios to illustrate the use of our tool for evaluating attack mitigation strategies. This is done by designing targeted attacks and tracking the points of attack impact in the network as we incrementally deploy mitigation schemes. The goal here is to illustrate that the Abilene core network is substantially oversubscribed, similar to other IP networks.

3.1 Oversubscription Factor

We seek to study the survivability of the Abilene network to both random and targeted network attacks by calculating the *Oversubscription Factor*. To calculate this factor: for every core router, first add up the capacities of all access router links to this core router to obtain the "maximum incoming flow" value. Then, add up the capacities of all core links from this core router to obtain the "maximum outgoing flow" value. The Oversubscription Factor is then given as: $OversubscriptionFactor = \frac{Max\ Incoming\ Flow}{Max\ Outgoing\ Flow}$. For a PoP, an Oversubscription Factor > 1.0 implies that more traffic could be sent via the access links of the PoP than be accommodated by the core links in the PoP. The numbers for the Abilene network are provided in Table 2. From the table, we see that the New York PoP is the most oversubscribed. If each

PoP-location	Oversubscription Factor
New York City	1.833
Chicago	1.803
Washington D.C.	1.779
Los Angeles	1.508
Seattle	1.5
Rest	<0.5

Table 2: Oversubscription factor values for Abilene.

link from the access routers to the New York backbone router is only 100/1.833 ($\approx$ 54.55) % utilized by an attack, at least one core link out of New York is overloaded.

To corroborate the qualitative understanding just obtained, we simulate attacks considering just the top 2 oversubscribed PoPs in New York and Chicago. For our simulations, for each of the 2 PoPs, we consider a set of 100 artificially generated, random-target attacks where in each attack: every access router of a PoP sends traffic to occupy 1% of its link capacity to targets chosen randomly from among all access routers outside the current PoP so that every flow traverses at least one core link.

From Figure 3, for the New York PoP, we see that the maximum GSF value required across all 100 attacks is about 50. The utilization given this GSF value is 50% which is clearly less than the expected value of 54.55% calculated from the Oversubscription Factor. This corroborates our earlier understanding. Observe that the same is the case with the Chicago PoP. From the starting value of both plots, we gather that there are attacks originating from the New York and Chicago PoP that require to be scaled in volume by only 27 and 15 respectively to overload at least one core link. These minima points correspond to attacks whose targets are such that all attack flows traverse a single core link.

3.2 Mitigation Evaluation

In the previous subsection we performed an analysis of the critical points in the Abilene network. As we saw, carefully targeted attacks overload the core at fairly low scaling factor values, meaning that only a part of access link capacity is required for this. Network operators need to employ mitigation mechanisms to lessen or even eliminate attack impact on the core. To this end, we now consider a sample real-time attack mitigation policy used by ISPs today. Our main thesis is *not* that this sample policy is the panacea to the attack mitigation problem but that our tool offers a convenient framework to test a desired real-time mitigation policy.

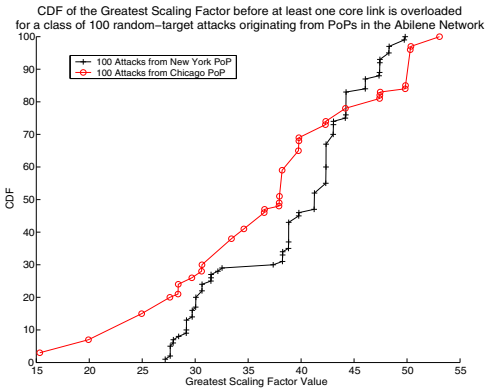


Figure 3: Understanding the strength of an attack originating from the New York and Chicago PoPs required to overload at least one of the respective oversubscribed core links.

3.2.1 A Sample Mitigation Policy

The sample mitigation policy we consider involves a combination of traffic scrubbing and selective blackholing to realize incremental attack mitigation. We aim to answer the following two important questions: given an attack, what is the minimum amount of mitigation required, using a combination of traffic scrubbing and blackholing to:

- *avoid overloading network core*?
- *avoid overloading the target(s) of the attack*?

A flowchart of the policy implementation is shown in Figure 4. A detailed explanation follows. Note that in practice, the steps in the policy are executed until the desired level of mitigation is reached. In addition to the policy, we also consider the mitigation offered by selective blackholing *without* traffic scrubbing for comparison purposes. In this case study, we perform deployment evaluation since we consider particular scrubber deployment scenarios and evaluate this restricted set of options. However, the simulator has the capability of choosing various deployments by itself, in a brute-force manner, and evaluating mitigation options offered in each case.

Step 1 - Ordering attack links: The intuition is that mitigating traffic from links considered in descending order of bit rate offers the best possibility for significant reduction of attack impact.

Step 2 - Traffic Scrubbing: Traffic scrubbers are typically deployed at select locations in the network only. This is because firstly these boxes are expensive, and secondly, their false positive rate is low for large traffic volumes. Scrubbers in general seldom affect genuine traffic resulting in little, if any, collateral damage. At the same time, to realize the same scope of mitigation as blackholing we would need to deploy scrubbers at every router which is expensive. (Henceforth, we refer to "link considered for scrubbing" as "scrubbed link".)

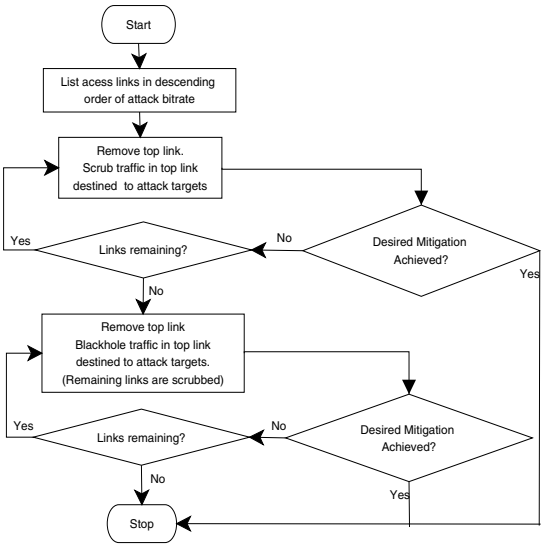


Figure 4: Flowchart of the implementation of our sample mitigation policy involving traffic scrubbing and selective blackholing.

In this step of the policy, given information on the location of the scrubbers, the traffic on a scrubbed link is sent to the closest scrubber in terms of minimum cost paths. This is done by performing a lookup in the precomputed shortest paths table (explained earlier in Section 2) for each scrubber location and identifying the scrubber with the smallest path cost. Once traffic is scrubbed, residual traffic, if any, is then routed to the original destination. This implementation of scrubbing is reasonable and employed in networks today. An important point is that after identifying the attack target(s), *all* traffic in the scrubbed links need to be sent to the scrubber.

Step 3 - Selective Blackholing with Traffic Scrubbing: Selective blackholing attack traffic provides greater mitigation than traffic scrubbing as all traffic to a set of targets is dropped local to the ingress router without entering the ISP network. However, unlike traffic scrubbing, there is no residual traffic after blackholing; thus the collateral damage is absolute. Hence, *a network operator resorts to blackholing traffic only if required*. Traffic that is not blackholed is scrubbed. (We refer to "link considered for blackholing" as "blackholed link".)

In our policy implementation, once all attack links are scrubbed, traffic on the attack links destined for the attack targets are blackholed in the same descending order as before. Again, we do not distinguish between good and bad traffic. The higher collateral cost due to blackholing is confirmed from simulations as is demonstrated in Figure 7.

3.2.2 Attack Scenarios

Taking cues from the analysis in the previous subsection, we consider two particular targeted artificial attack scenarios, referred to as **AttackOne** and **AttackTwo**. Both attacks have flow rates such that each attack target's access router link is just about 95% utilized. The traffic flow rates (in kbps) and the source and target PoPs for these attacks are illustrated in Figure 2. Note that the topology in the figure does not depict access routers while traffic matrices are at the access router level.

We use AttackOne to mainly illustrate the mitigation effects using varying number of scrubbers. AttackTwo on the other hand is a targeted attack aimed at overloading the core links in such a way so that customers on the West Coast of the USA cannot send traffic to those on the East Coast. Despite the large impact of this attack, network cut-off is achieved with a litte more than 30% utilization of the access router link capacities! The exact value for the flows are also depicted in Figure 2.

In both attack scenarios, we geographically co-locate scrubbers with backbone routers and so the cost in reaching them is approximately equal to the cost of reaching the co-located backbone router. Also, we assume a scrubbing capability of 0.99 meaning that 99% of all attack traffic entering the scrubber is dropped (this value was chosen from operational experience,and can be adjusted as desired). Another assumption we make is that when a link in the Abilene network (all OC-192s) is 95% or more utilized, it is said to be overloaded (this value is again reasonable and tunable). We use the simulator feature of scaling and analyzing attacks. In particular, we calculate each time the greatest scaling factor (GSF) value by which a particular attack should be scaled in volume such that a core link is overloaded. In cases where the core cannot be overloaded on scaling, we set the GSF to a ceiling value (1000 for AttackOne and 50 for AttackTwo). Even though these two scenarios seem contrived, targeted attacks have the maximum damage capability (as shown earlier) and use of the tool to analyze these worst-case attack scenarios is instructive.

3.2.3 Results

Experiment Using AttackOne

We start with the scenario of deploying one scrubber and progressively add scrubbers at distinct network locations. We then execute the mitigation mechanism presented earlier on these scenarios independently. In all scenarios, we measure the GSF of an attack before overloading at least one core or target link as the case may be, and set it as the Y-axis on the plots. In the simulator, this value is determined by increasing the GSF starting from a low initial value until link overload occurs.

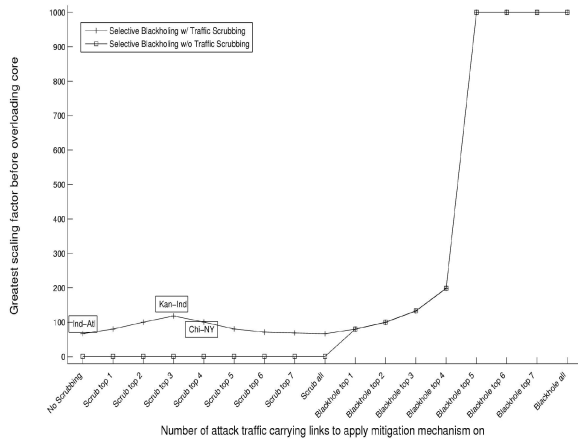


Figure 5: Analysis of effectiveness of mitigation mechanisms on protecting the core network under AttackOne with scrubber at New York. The curve of blackholing without scrubbing in not annotated in the combined plot.

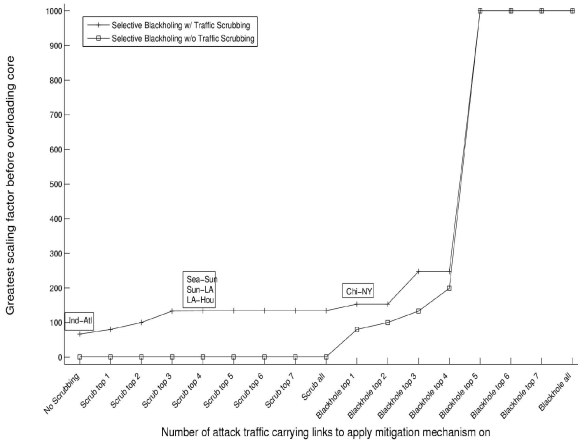


Figure 6: Same scenario as in Fig 5 but with scrubbers at both New York and Houston.

Blackhole top N links	0	1	2	3	4	5	6	7	8
Blackholing Cost (x 10^{-3})	0	0.8	1.6	3.3	4.2	5.9	7.1	7.1	7.1

Figure 7: Table of costs for the blackholing strategy in terms of percentage of background traffic dropped under AttackOne. The columns corresponds the states starting from "Scrub all" to "Blackhole all" in the above plot.

How to interpret the plots? Figure 5 depicts the results from the experiment when a lone scrubber is deployed within the New York PoP under AttackOne. Each point in the plot refers to a particular state of deployment of the mitigation strategy, and this is indicated in the tick labels along X-axis. Since the mitigation deployment we consider is incremental, every successive point denotes the state when one more attack link is scrubbed/blackholed. In the case of the curve 'Selective Blackholing w/o Traffic Scrubbing', no link is scrubbed and this is indicated as a constant zero value in the plots (until the 'Blackhole top' labels). At each state of deployment of the mitigation mechanisms in the network, the plots have been annotated with the set of core edges that are overloaded. So, for instance, at the state when there is no mitigation mechanism (that is, x=0), the attack has successfully overloaded the Ind-Atl link while no other core link is overloaded. Since the deployment happens in increments, if successive states have the same set of overloaded core edges, then only the first state is annotated.

Having explained how to read the plots, we now present an analysis. In Figure 5, until the state when the top 3 attack access router links are scrubbed, even though some traffic is diverted away from the Ind-Atl core link towards the scrubber in the New York PoP, there is still sufficient attack traffic to overload the core network. Since traffic through the congested core link decreases with every attack link scrubbed, we need to scale the attacks by a little more to still be able to overload the core. This is indicated by the steady increase in the GSF value. Then, the scrubbing curve begins to drop off as we scrub the top 4 attack links. At this stage, the congestion point is transferred to the Kan-Ind core link as now a large portion of the attack traffic is ferried via the shortest path from the sources at both Kansas City and Seattle to the scrubber in New York. Observe the trend of steady dip in the scrubbing curve of the GSFs. There is already enough traffic transferred to New York that by the time we scrub some more traffic, due to aggregation the core link leading up to the New York router is overloaded. This is reflected in the plot by the annotation of "Chi-NY". This plot highlights the expected effect of having a single scrubber which becomes the single point of congestion in the network.

Once all attack links are scrubbed, the top attack link is blackholed. Up till when the top 4 attack links are blackholed, blackholing traffic is not sufficient to shift the congestion point from the New York core link. This is indicated by the annotation trend as the GSFs increase. However, when the top 5 attack links are blackholed, the number of attack links still sending attack traffic into the core network is so low that even at full capacity these links together cannot overload any core link in the network. This is indicated by the plateau at the ceiling value 1000.0 at the tail of the blackholing curve. Note that the curve for the blackholing without scrubbing mitigation strategy nearly coincides with the blackholing with scrubbing curve indicating that in this single scrubber situation, the effects of scrubbing are negligible compared to that of blackholing.

Figure 6 shows how the network copes with AttackOne when there are 2 scrubbers deployed at New York and Houston respectively. Since the closest scrubber to

the PoPs on the West Coast is at Houston, traffic from the sources in Seattle and Kansas City are directed towards Houston reducing the load on the New York core links. This is directly observable as the lack of the dip in the scrubbing curve observed in the earlier figure. In this 2-scrubber scenario, the bottleneck links are those along the path from Seattle to Houston which is reflected in the annotation in the plot. Similar to the earlier scenario, as we start blackholing, the only overloaded core link at the GSF is the Chi-NY core link. Also note that since traffic destined for scrubbing is now distributed between 2 scrubber locations not in close network proximity, attack traffic is much more diluted through the network core. So, the attack needs to be scaled up by a higher factor to overload the core.

The best evidence for the benefits of scrubbing at 2 locations is provided by comparing the two blackholing curves in Figure 6. At every state of the network (before x=5), the deployment of 2 scrubbers in addition to blackholing has bought the network operator nearly twice as much network capacity than by blackholing only. This can be seen from the two-fold increase in the GSF values of the 2 curves. Remember that the blackholing without scrubbing curve nearly coincided with the blackholing with scrubbing using one scrubber curve. So, these observations serve to compare indirectly between the 2-scrubber and 1-scrubber deployment scenarios.

We considered other simulation scenarios using more scrubbers under the effect of AttackOne. Due to lack of space, we do not present those plots. In essence, when an additional scrubber is added to the network, the attack traffic in the network becomes more diluted. Of course deploying scrubbers right at the backbone router nearest to the sources almost completely eliminates attack traffic when traffic scrubbing is performed and so we analyze only the contrary deployment. In almost all these simulation scenarios, we observed a similar two-fold improvement in effectiveness of using scrubbing along with blackholing as opposed to using blackholing only.

Using the above, the network operator is able to quantify attack impact of AttackOne as well as mitigation offered by the strategies in the sample policy. Therefore in answering the first of the two questions posed earlier, she can for example choose to defend her network by blackholing the top 4 attack links knowing that the attack now needs to ramp up its volume by more than 200 to be able to still overload the core. Table 7 gives the costs involved in blackholing background traffic. Note that these costs are dependent only on the traffic matrices and are independent of traffic scrubbing. The percentage of background traffic dropped due to blackholing versus total background traffic is the measure of cost captured in the table. Clearly, there is an increase in the cost paid by the network operator in terms of collateral damage as more access links are considered for blackholing. So, she needs to base her final mitigation decision on both the strength of mitigation desired as well as the cost she is willing to pay.

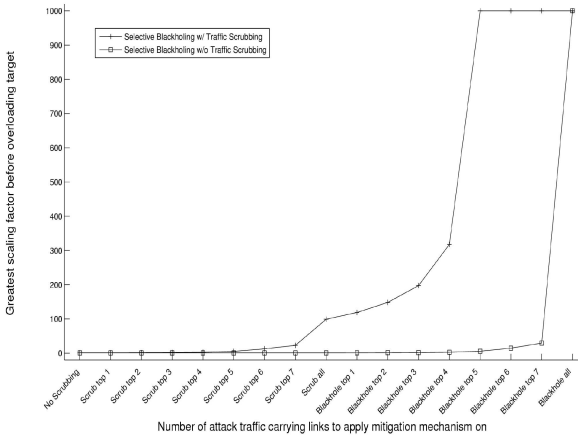
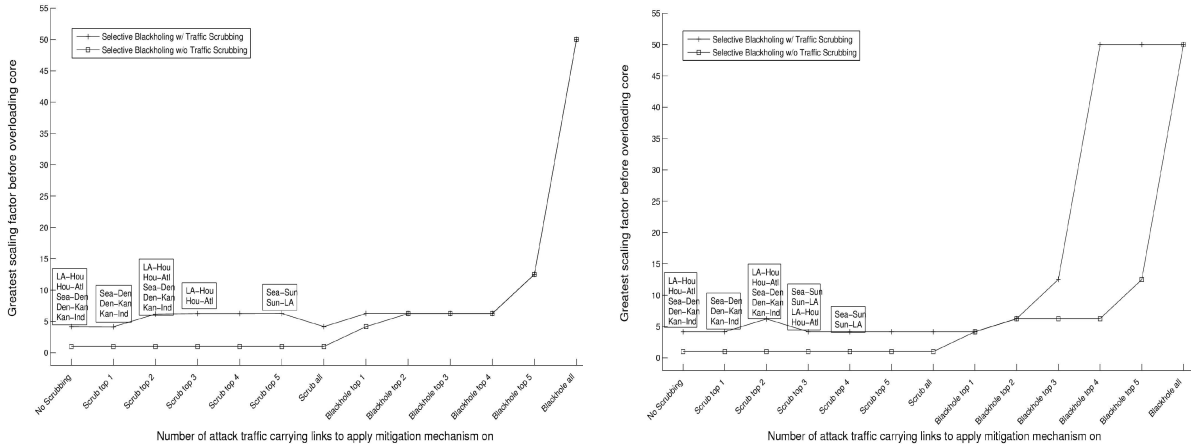


Figure 8: Analysis of effectiveness of mitigation mechanisms on protecting target links under AttackOne.

Figure 8 represents a similar plot as before but with the focus now on avoiding overloading the target link. Recall that the attack at scaling factor of 1 just about overloaded the target access link. The plot depicts the scenario when a single scrubber is deployed in New York. Under the other scrubber deployment scenarios considered earlier, we obtained exactly the same numbers indicating that as far as protecting the target is concerned, scrubber deployment location does not matter (as long as the network core itself is not affected). On comparing the GSFs to overload the core versus that to overload the target link for the same network states, we observe that the latter GSFs are much smaller as expected.

From the curves, it becomes clear that scrubbing is more effective in protecting the target(s) than protecting the core. For a combination of blackholing and scrubbing under AttackOne, Reval is able to provide two benefits simultaneously. First, multiple deployment options are provided based on the level of defense desired. Secondly, for each mitigation option, Reval quantifies the increased attack capacity that the network can now sustain before again overloading the core. Using scrubbing alone, at the very best the network operator can hope to reduce the traffic reaching the target link(s) by a maximum factor of $\frac{1}{1-f}$ where f is the scrubbing accuracy. In our case, this theoretical maximal defense value is $\frac{1}{1-0.99}$=100. The actual value is dictated by when the core network is overloaded due to the diversion in traffic. Reval can provide the network operator with the exact numbers within this maximal range as just illustrated.



(a) With scrubber in Seattle

(b) With scrubber in LA

Figure 9: Choice of scrubber deployment under AttackTwo.

Experiment Using AttackTwo

AttackTwo as described earlier was designed to completely cut off coast-to-coast communication requiring surprisingly small bandwidth to do so. Exploring the full space of mitigation options is a cumbersome task even after restricting ourselves to the two strategies of traffic scrubbing and blackholing. Instead, we ask a rather qualified question: given the availability of one scrubber, where should it be placed so that traffic scrubbing and, if necessary, selective blackholing give the network operator maximum leeway in dealing with the attack?

Since the attack partitions the network into two and all sources of the attack are on the West Coast, it makes sense to deploy the scrubbers in one of the West Coast PoPs. Also, since the sources of attack are at Seattle and LA, deploying the scrubber at other West Coast nodes does not buy the network operator much (this is substantiated by results from the simulator not shown here). Figure 9 then plots the results of the simulator under the two scenarios - scrubber deployed in Seattle and LA respectively. Observe the plots for the GSF values for corresponding states especially paying attention to the blackholing with scrubbing curves. Once the top 3 attack links are considered for blackholing, the effect of scrubbing coupled with the choice of the flows to scrub make LA an attractive choice for scrubber deployment. Also, notice that to protect the core under this attack, with the scrubber deployed at LA, it is enough to blackhole traffic from 4 out of the 6 attack access links while in the case of the deployment at Seattle all 6 attack links need to be considered for blackholing. Thus on both counts, we have LA as the better location for scrubber deployment.

With additional analysis, we discovered the existence of an interesting phenomenon that is brought out by the simulator: in our mitigation strategy, we pick the top attack links starting from the highest. However, a tie breaking mechanism is required to deterministically decide between equally voluminous attack links. We see why this is important now: in AttackTwo since there are 6 attack traffic flows, 3 each from Seattle and LA, all of the same attack volume, there is already a tie. To break the tie, suppose we choose Seattle's attack links before considering LA's attack links, and we consider the deployment of the scrubber at LA first. As a result, because of the order in which we choose to scrub/blackhole attack traffic, we only need to blackhole the top 4 attack links. These correspond to the 3 sources in Seattle (thus relieving the load on the Kan-Ind core link) and one source in LA. The remaining 2 sources are in turn considered for scrubbing performed locally within the LA PoP. On the other hand, when the scrubber is deployed at Seattle, after blackholing the top 4 attack links, we need to send all traffic from 2 attack links in LA to Seattle to be scrubbed. This results in an overload of all core links in the path from LA to Seattle thus requiring to blackhole all 6 attack links in order to protect the core.

Note that by using a purely qualitative study based on the AttackTwo traffic matrix, we are led to believe that both LA and Seattle are equally significant PoPs and deployment at either location yields the same benefit! Such an ad-hoc decision does not factor in practical aspects of mitigation mechanism implementation. This particularly becomes important if the network operator assigns greater priority to traffic from LA than Seattle owing to greater profit potential. Even though equal traffic rate attacks are rare in the real world, the qualitative take-away from this scenario holds. Thus, the network operator using Reval can gain knowledge of both the **effectiveness** and **real-world implementation issues** of the desired mitigation policy.

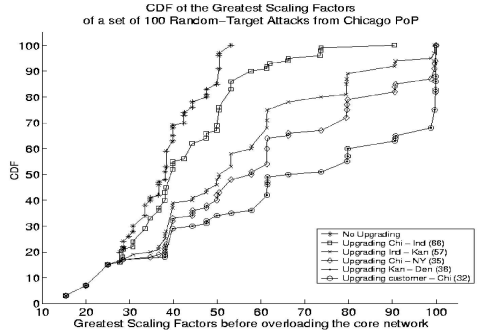


Figure 10: Link Upgrade experiment for a class of 100 random-targeted attacks originating from the Chicago PoP. In the legend, along with the particular link being upgraded, we also present the number of attacks out of the set that overload that link.

3.3 Long-Term Network Planning: Capacity Upgrading

In this subsection, we illustrate a use for our tool in helping a network operator carry out effective long-term network planning. We specifically use the tool to shed light on which links in the network require upgrading to higher capacities. For this, we first choose to upgrade one at a time only the core links that are impacted frequently by attacks (and these are oversubscribed core links). After each link upgrade, all attacks are simulated again to identify the next most frequently impacted link, and so on. Secondly, we upgrade, if at all, a link to have four times its current capacity. This is an industrial practice that amortizes equipment cost over future demands.

Figure 10 shows the result of the link upgrade experiment carried out using the class of 100 attacks from the Chicago PoP described earlier in Section 3.1. We only consider a maximum scaling factor of 100 in the graph without loss of generality. As expected, the most frequently overloaded link is the core link involving the Chicago backbone router. It is interesting to note that on a single link upgrade, nearly 10 attacks out of the 100 required to be scaled by 25 more than before. Also, even after upgrading all core links, more than 60% of the attacks have a GSF value smaller than 100. Another observation is that there is a sizable number of attacks that manage to overload a core link at low scaling factors (15% of attacks have GSF $\leq$ 25). Since these overloaded core links do not occur frequently, they are never upgraded. This illustrates the use of Reval to quantitatively assess which points in the network require upgrading and how much respite from traffic load these link upgrades provide. The analysis also reflects the choices and compromises network operators make in choosing to defend against the common case attacks while still being susceptible to isolated instances.

4 Experience with a Large Tier-1 ISP Network

In this section we present results from evaluating the mitigation strategies of traffic scrubbing and selective blackholing on real data obtained from a large tier-1 ISP. We describe our experience of using the tool over a period of one month. Both background traffic and network topology information are obtained in the form of real-time feeds — the former as Netflow records while the latter as router configuration files. The attack input data for the simulator is obtained from alarms provided by the commercial DDoS detection system. Then these alarms are used to obtain relevant Netflow traffic records from a subset of routers producing the attack traffic matrix. In our network, the commercial detection system can be configured to provide these statistics in real time. Attack information could also be obtained by using real-time anomaly detection systems like [25]. Flow properties required for the simulator in terms of source, target, bit rate *etc* are directly obtained from Netflow.

As mentioned in Section 2, we have automated the execution of our simulator on real data obtained from the tier-1 ISP. Restating the performance numbers, the execution of the core functionality of the simulator on the ISP network of size in excess of 60000 nodes with about 0.4 million traffic flows took 11 seconds approximately.

We considered the simulation scenario of analyzing the effectiveness of the mitigation policy of the previous section with a slight modification. We attempt to analyze the decision a network operator would take in the real world using results from the simulator. The data for this simulation scenario comprised of 144 attacks (as classified by the commercial DDoS detection system) that occurred over the month of September in 2005. For each attack, we obtained statistics on the GSF value before overloading the core under the network state at the time of the attack. This was done for various mitigation deployment states in the two strategies of traffic scrubbing and selective blackholing with traffic scrubbing.

None of the attacks overloaded any core links in their original form, and therefore needed to be scaled up. From preliminary analysis, we found that 77 of the 144 attacks did not have enough volume rate to overload the core. That is, even with maximum scaling (when all ingress access links of an attack are used to their full capacity), these attacks could not individually overload any core link. Also, about 24 of the attacks pertained to instances when the network data collected was not reliable. For the remaining 43 attacks, we fixed a particular scaling factor value for which all of these attacks would overload at least one core link. With this as the base, we then obtained the best mitigation strategy required to avoid overloading the core. By "best", we refer to the

strategy that required the least amount of blackholing as opposed to traffic scrubbing since blackholing results in greater collateral damage.

The number of unique responses offered by the simulator as the best mitigation strategy for these 43 attack instances was 16. Each involved blackholing and/or scrubbing at least one unique network ingress link not considered in any other instance. Thus, Reval revealed that mitigation against attacks in the real-world involves a constantly changing region of deployment of the mitigation mechanisms rather than a fixed deployment. Moreover, the region of deployment is dictated by the nature of the particular attack under consideration.

As further analysis, we investigated the usefulness of the solution by analyzing the scenario of "How much impact would the core network suffer if the second best mitigation strategy was chosen instead of the best?" For these 43 attacks, we found that the second best mitigation strategy overloaded at least one core link for 30 attacks and in a couple of instances, the attack managed to overload 2 core links while the remaining did not overload any core links. So not only did Reval provide the mitigation option that was best in terms of effectiveness but was also better by a fair margin in a large number of attacks.

5 Related Work

The EMIST project [26], a sister project of the DETER test bed, aims to provide a framework and methodology for studying particular classes of network attacks and defense mechanisms. Using a test bed limits the scale of experimentation. Also, defense mechanisms need to be deployed on the test bed either by procuring the right hardware like in the case of traffic scrubbers or implementing functional software. Such test beds are more useful in understanding network attacks in an *off-line* setting unlike our tool.

ns2 [27] is a popular network simulator that provides capabilities for packet-level simulations. However, emphasis is on lower-layer simulations rendering it unscalable to large networks. Also, neither *ns* by itself nor any third-party code address security simulations. Likewise, GloMoSim [28] is a discrete-event mobile network simulator that is currently not used in security research. SSFNet [29] has a similar scalability to that of *ns2* [30]. MicroGrid [31] is a network simulator built to realize realistic large scale online simulations. An associated project is the ModelNet network emulator [32] which also focuses on scalable network simulations. In both works, the goal of the system is to gauge distributed application performance under different network load scenarios, and are not suited for *network* attack impact and mitigation studies.

There have been studies on building simulators adopting a particular model of network traffic. However, almost all these models assume an underlying working theory on patterns in traffic properties. It is not clear how these theoretical traffic models apply to network attack traffic that are inherently non-uniform. Related work in this regard is the network calculus-based approach to scalable network simulations [33]. Other work include the network simulator presented in [34] where a fluid-based model is used to make packet-level simulations more scalable. However, this simulator when run on large IP networks is still far from real time. Moreover fluid models are not viable for light or sporadic traffic.

Parallel to the research on models for faster simulation, there has been considerable work on providing techniques that aim to increase speed of execution in a simulator-independent manner. We mention few of these techniques here while noting that these techniques are complimentary to our simulator. A recent approach to scalable fast large scale simulations is to identify network invariants, preserve them in scaling down simulations and accordingly scaling up the results obtained [35] thus vastly improving on a full-blown packet-level simulation. Another related approach is in the world of building a parallel network simulator with the idea of intelligently parallelizing simulations at a part level and then communicating once in a while to realize system-wide simulations [36].

6 Conclusion

We have presented the design and implementation of *Reval* — an operational support tool that quantifies the impact of attacks on any-sized networks, analyzes the effectiveness of mitigation strategies, and provides comprehensive answers pertaining to a wide range of mitigation deployment scenarios, all in a real-time manner. Moreover, the use of Reval brings to the attention of the network operator potential real-world deployment issues of desired mitigation strategies otherwise observable only after practical experience. Using live data from a tier-1 ISP network of size in excess of 60000 nodes with close to 0.4 million traffic flows simultaneously, Reval executes in about 11 seconds. Given a sample mitigation policy in this real network, Reval identifies the most effect mitigation option after evaluating each option in about 35 seconds.

Though Reval was discussed from the point of view of network attacks in this paper, it could readily be used in a host of network scenarios: analyze network anomalies like flash crowds, routing-driven traffic surges, worm propagation; understand attack impact and mitigation from perspective of customers of the ISP; strengthen the network via capacity planning; study network topology

design using various graph theoretic and other parameters; compare and contrast defense mitigation strategies; quantify the conditions for particular network-level mechanisms like new routing algorithms, defense algorithms *etc.* to work successfully.

It behooves us to state that the simulator we have built is by no means complete or exhaustive in terms of functionalities desired by security researchers. We are currently looking to improve the simulator along a few dimensions including support for packet-level simulations, simulation time, and accurate cost metrics. Reval could be executed on latest network state by incorporating live data feeds that relay changes in the network like IGP changes. Even though Reval evaluates a host of mitigation strategies, choosing the particular strategy is a manual decision taken after making various trade-offs including the level of collateral damage. These trade-offs make the process of attack defense complicated, and difficult to eliminate manual intervention. Nevertheless, Reval is a first step towards its automation.

Acknowledgements

We thank Mukesh Agrawal, Patrick Verkaik, Adrian Cepleanu, the anonymous reviewers and our shepherd Geoff Voelker for the many criticisms and suggestions that have helped shape this paper.

References

[1] R. Richmond, "Firms Join Forces Against Hackers", *Wall Street Journal*, March 28, 2005.

[2] D. Pappalardo and E. Messmer, "Extortion via DDoS on the Rise", http://www.networkworld.com/news/2005/051605-ddos-extortion.html, May 16, 2005.

[3] A. Hussain, J. Heidemann, and C. Papadopoulos, "A Framework for Classifying Denial of Service Attacks", in *Proc. ACM SIGCOMM*, 2003.

[4] P. Barford, J. Kline, D. Plonka, and A. Ron, "A Signal Analysis of Network Traffic Anomalies", in *Proc. ACM SIGCOMM Workshop on Internet Measurement*, 2002.

[5] R. Malan, F. Jahanian, J. Arnold, M. Smart, P. Howell, R. Dwarshius, J. Ogden, and J. Poland, "Observations and Experiences Tracking Denial-Of-Service Attacks Across a Large Regional ISP", http://www.arbornetworks.com/downloads/research37/nanogSlides4.pdf, 2001.

[6] P. Vixie, G. Sneeringer, and M. Schleifer, "Events of 21-Oct-2002", http://d.root-servers.org/october21.txt, 2002.

[7] D. Moore, G. Voelker, and S. Savage, "Inferring Internet Denial of Service Activity", in *Proc. USENIX Security Symposium*, 2001.

[8] "Arbor networks", http://www.arbornetworks.com/.

[9] "Mazu networks", http://www.mazunetworks.com/.

[10] D. Knuth, "The Stanford GraphBase: A Platform for Combinatorial Computing", Addison-Wesley, 1994.

[11] B. Cherkassky and A. Goldberg, "On Implementing Push-Relabel Method for Maximum Flow Problem", *Algorithmica*, vol. 19, pp. 390–410, 1997.

[12] R. Mahajan, S. Bellovin, S. Floyd, J. Ionnadis, V. Paxson, and S. Shenker, "Controlling High Bandwidth Aggregates in the Network", in *ACM CCR*, 2002, vol. 32:3, pp. 62–73.

[13] "Cisco Anomaly Guard Module", http://www.cisco.com/.

[14] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, and J. van der Merwe, "Design and implementation of a Routing Control Platform", in *Proc. NSDI*, 2005.

[15] T. Anderson, T. Roscoe, and D. Wetherall, "Preventing internet denial-of-service with capabilities", in *Proc. Hotnets-II*, 2003.

[16] A. Greenhalgh, M. Handley, and F. Huici, "Using Routing and Tunneling to Combat DoS Attacks", in *Proc. SRUTI*, 2005.

[17] "Cisco 12000 Series Internet Router Architecture: Packet Switching", http://www.cisco.com/.

[18] B. Jenkins, "A Hash Function for Hash Table Lookup", http://burtleburtle.net/bob/hash/doobs.html, 1997.

[19] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*, The MIT Press, 2nd edition, 2001.

[20] E. Zegura, K. Calvert, and S. Bhattacharjee, "How to Model an Internetwork", in *Proc IEEE INFOCOM*, 1996.

[21] "Abilene network", http://www.internet2.edu/abilene.

[22] L. Li, D. Alderson, W. Willinger, and J. Doyle, "A First-Principles Approach to Understanding the Internet's Router-level Topology", in *Proc. ACM SIGCOMM*, 2004.

[23] Y. Zhang, M. Roughan, C. Lund, and D. Donoho, "An Information-Theoretic Approach to Traffic Matrix Estimation", in *Proc. ACM SIGCOMM*, 2003.

[24] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg, "Fast Accurate Computation of Large-Scale IP Traffic Matrices from Link Loads", in *Proc ACM SIGMETRICS*, 2003.

[25] V. Sekar and N. Duffield and J. van der Merwe and O. Spatscheck and H. Zhang, "LADS: Large-scale Automated DDoS Detection System", in *Proc. USENIX*, 2006.

[26] "Deter/Emist Project", http://www.isi.edu/deter/projects.html.

[27] "ns-2", http://www.isi.edu/nsnam/ns/.

[28] "GloMoSim", http://pcl.cs.ucla.edu/projects/glomosim/.

[29] "SSFNet", http://www.ssfnet.org.

[30] D. Nicol, "Comparison of Network Simulators Revisited", http://www.ssfnet.org/Exchange/gallery/dumbbell/dumbbell-performance-May02.pdf, 2002.

[31] X. Liu and A. Chien, "Realistic Large Scale Online Network Simulation", in *Proc. ACM Conf. on High Performance Computing and Networking*, 2004.

[32] A. Vahdat, K. Yocum, K. Walsh, P. Mahadevan, D. Kostic, J. Chase, and D. Becker, "Scalability and Accuracy in a Large-Scale Network Emulator", in *Proc. OSDI*, 2002.

[33] H. Kim and J. Hou, "Network Calculus Based Simulation for TCP Congestion Control: Theorems, Implementation and Evaluation", in *Proc. IEEE INFOCOM*, 2004.

[34] Y. Liu, V. Misra F. Presti, D. Towsley, and Y. Gu, "Fluid Models and Solutions for Large Scale IP Networks", in *Proc. ACM SIGMETRICS*, 2003.

[35] H. Kim, H. Lim, and J. Hou, "Accelerating Simulation of Large-Scale IP Networks: A Network Invariant Preserving Approach", in *Proc. IEEE INFOCOM*, 2006.

[36] Y. Liu B. Szymanski, A. Sastry, and K. Madnani, "Real-Time On-Line Network Simulation", in *Proc. IEEE Intl. Workshop on Distributed Systems and Real-Time Applications*, 2001.