AnomBench: A Benchmark for Volume-Based Internet Anomaly Detection

Shashank Shanbhag and Tilman Wolf
Department of Electrical and Computer Engineering
University of Massachusetts, Amherst, MA, USA
{sshanbha,wolf}@ecs.umass.edu

***Abstract*—Developing algorithms to detect anomalies in network traffic is an important goal to achieve secure and efficient operation of the Internet. To evaluate different algorithms, it is crucial to have a set of standardized test cases. We propose a benchmark suite called "AnomBench" that consists of sixteen different traffic scenarios that contain various different traffic anomalies. We describe why these scenarios are representative and show the results of a prototype implementation on DETERlab.**

I. Introduction

Anomaly detection is an important aspect of network operation in the Internet. Network operators need to be aware of unusual traffic that may case security breaches, performance degradation, and other problems. Unlike intrusion detection systems (IDS), which attempt to match a set of given rules against traffic, anomaly detection algorithms apply heuristics to identify unusual traffic patterns. Since it is inherently difficult to determine what scenario constitutes an anomaly, there exists no anomaly detection system than can always make a completely accurate decision.

Inaccuracies in anomaly detection lead to false positives (alert without cause) and false negatives (no alert despite anomaly), both of which are undesirable from an operational point of view. Therefore, an important metric for determining the quality of an anomaly detection algorithm is the percentage of false positive and false negative answers (often measured by a receiver operating characteristic (ROC) curve [6]).

To compare different anomaly detection approaches and to judge their accuracy, it is imperative that they are evaluated under identical traffic conditions. Often, a particular algorithm is more sensitive to a specific anomaly than another. Without a broad evaluation of algorithms on the same traffic scenarios, it may be possible that the quality of an anomaly detection algorithm may be misjudged. To address this problem, we present a novel benchmark for anomaly detection called "AnomBench" in this paper.

Our benchmark is intended to cover typical scenarios related to *volume-based network anomalies*. The purpose of AnomBench (and practically any other benchmark) is to generate a realistic workload for an anomaly detection system enabling the measurement of false positive and false negative rates, detection latency, overhead, etc. The scenarios in AnomBench vary in terms of attack pattern, attack intensity, traffic mix, and attack timing. Testing an anomaly detection algorithm with this broad set of attack patterns provides a comprehensive evaluation of the algorithms performance. To make the benchmark broadly useful, we have developed traffic generation scripts based on DETERlab, which can be parameterized for different baseline traffic rates and traffic destinations. The results from this prototype implementation show the operation of AnomBench in the context of two specific algorithms.

The remainder of this paper is organized as follows. Section II discusses related work. AnomBench is introduced in Section III. A prototype implementation of AnomBench is discussed in Section IV and evaluation results are presented in Section V. Our work is summarized in Section VI

II. Related Work

Anomaly detection algorithms have been developed using a variety of different techniques, for example, time series (e.g., Holt-Winter forecasting [4]) and machine learning (e.g., one-class neighbor machine [1]) to name a few. Some of these approaches are surveyed in [5]. The evaluation of these anomaly detection algorithms is done differently in each paper and comparisons across papers are difficult.

Packet traces from different known anomalies have been used to evaluate some anomaly detection algorithms. Examples of such traces are Los Nettos traces [9] and MIT Lincoln Labs DDoS traces [11]. While traces provide some level of comparability, they are limited to a small collection of attack scenario. Also, trace-specific characteristics (e.g., baseline traffic rate, duration, etc.) limit the usefulness for exploring anomaly detection accuracy under different scenarios (e.g., increased bandwidth).

In terms of traffic anomaly benchmarks, Mirkovic et al. [12] have proposed a set of rules that can be used to specify characteristics of a denial-of-service attack. These rules can be used as guidelines to generate attack traffic. Our work goes a step further and provides specific scenarios (and corresponding traffic generation scripts), which is essential to achieve comparability of results. Brauckhoff et al. [3] discuss the design of a tool, called FLAME, for injecting simulated flow-level anomalies into a background trace. The background trace is modified to make injection possible according to the type of anomaly desired: additive (additional flows merged with existing baseline flows), subtractive (removal of flows from trace) and interactive (identification and removal of flows modified due to injection of anomaly). The authors however acknowledge that FLAME does not consider anomalies, if any, already present in the trace. In contrast, AnomBench's focus is on different anomaly scenarios that vary in terms of attack pattern, intensity, mix and timing to provide with a base using which different anomaly detection algorithms can be tested.

Benchmarks have long been an essential component in computer system design based on engineering principles. Most well-known is probably the SPEC CPU benchmark suite [15], which has been widely used in computer architecture. Other benchmarks have been developed more targeted application domains (e.g., MiBench for embedded systems [8], MediaBench for multimedia networking [10]) including our own work on benchmarks for network processors [18] and BGP routers [19]. Based on the positive impact that benchmarks had on other fields to enable quantitative and objective comparisons, we believe it is important that such a benchmark is developed for the anomaly detection domain.

III. AnomBench

The goal of any benchmark is to condense a large number of different workloads that may potentially be used on a system into a small set of representative scenarios. In AnomBench, we identify sixteen different scenarios that represent a variety of different volume anomalies.

A. Anomaly Scenarios

All traffic scenarios in AnomBench are based on baseline background traffic (see details below) on which different anomalies are overlaid. In each trace, each anomaly is repeated multiple times (i.e., the anomaly is "turned on and off" repeatedly). Between scenarios, the anomaly traffic is varied based on combinations of several aspects that characterize anomalies. These aspects are:

- **Shape:** The shape of the anomaly can vary between a *pulse* and a *ramp*. Pulses characterize sudden onsets of anomalous traffic, where as ramps represent slowly changing scenarios. This aspect explores how algorithms react to sudden and slow changes.
- **Duration:** The duration of an anomaly can vary between *short* and *long* pulses. Spikes can be treated as extremely short duration pulses. This aspect characterizes how long an anomaly persists and checks an algorithm's ability to quickly detect anomalies.
- **Intensity:** Intensity characterizes the relative size of an anomaly relative to the background traffic. The intensity can vary between *high* and *low* relative to the background rate. This aspect helps determine how sensitive an anomaly detection algorithm is to changes.
- **Target:** The target of an anomaly can vary between *random* ports on the end-system or a *single* port. The target port choice characterizes if an anomaly is focused on a single destination or if it scans across multiple destinations. This aspect characterizes if an anomaly detection algorithm is sensitive to attacks on particular ports or only to the overall traffic volume.

TABLE I
AnomBench Scenarios.

Scenario	Shape	Duration	Intensity	Target
Scenario 1	pulse	long	high	random
Scenario 2	pulse	long	high	single
Scenario 3	pulse	long	low	random
Scenario 4	pulse	long	low	single
Scenario 5	pulse	short	high	random
Scenario 6	pulse	short	high	single
Scenario 7	pulse	short	low	random
Scenario 8	pulse	short	low	single
Scenario 9	ramp	long	high	random
Scenario 10	ramp	long	high	single
Scenario 11	ramp	long	low	random
Scenario 12	ramp	long	low	single
Scenario 13	ramp	short	high	random
Scenario 14	ramp	short	high	single
Scenario 15	ramp	short	low	random
Scenario 16	ramp	short	low	single

TABLE II
Background Traffic Distributions for Some of the Example Services.

Traffic Type	Parameter	Distribution
Web	Request Interarrival Time(s)	Exponential
	File Size(Bytes)	Pareto
DNS	Request Interarrival Time(s)	Exponential
Telnet	Request Interarrival Time(s)	Exponential
	Response Size(Bytes)	Gamma
	Session Duration(s)	Gamma
	Time between key strokes(s)	Exponential
	Time between sessions(s)	Exponential
FTP	Request Interarrival Time(s)	Exponential
	File Size(Bytes)	Pareto
IRC	Request Interarrival Time(s)	Exponential
	Message Size(Bytes)	Gamma
Ping	Request Interarrival Time(s)	Exponential

Given these four aspects with two choices, we obtain $2^4 = 16$ different combinations. Each combination represents on AnomBench scenario. The sixteen scenarios are enumerated in Table I.

B. Background Traffic

It is important to consider the characteristics of background traffic. Volume anomaly detection is challenging due to the fact that background traffic is highly variable. Moreover, state of the art anomaly detection systems specialize in detecting small anomalies compared to the variation in the normal background traffic. To facilitate realistic background traffic generation that reflects traffic in a typical network, we use the distributions shown in Table II, which are drawn from measurements of actual real-world traffic [13].

C. Benchmark Parametrization

A unique aspect of benchmarks in the networking domain is the need for adaptability to different link data rates. A traffic scenario that uses a particular bandwidth may be meaningful for one system, but not for others. To reflect this need for configurability, we introduce several parameters that make AnomBench adaptable to a diverse set of networking scenarios (see Figure 1):

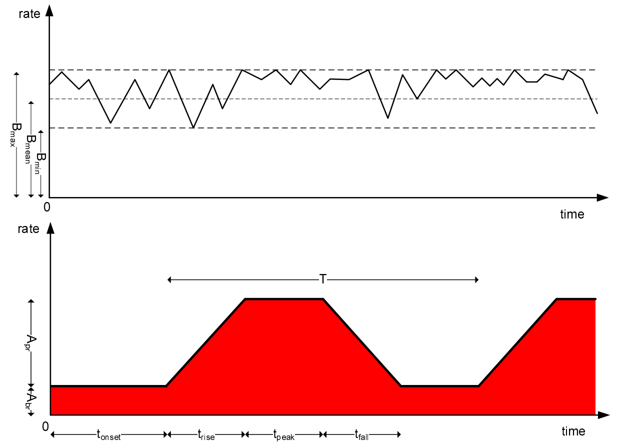


Fig. 1. AnomBench Benchmark Parametrization for Anomaly and Background Traffic.

- **Baseline Bandwidth (B_{mean}):** The bandwidth parameter specifies the data rate of the baseline traffic. This allows an adaptation of AnomBench to different link rates. The parameter β is used to set the amount of variation in baseline traffic. For example, setting $\beta = 0.1$ causes the baseline traffic to vary between +10% and -10% of B_{mean}.
- **Anomaly Base Rate (A_{br}):** The anomaly base rate is defined as the anomaly traffic rate between attacks. It is given as a fraction of the baseline bandwidth: $A_{br} = \alpha_{br} B_{mean}$, where $0 \leq \alpha_{br} \leq 1$.
- **Anomaly Peak Rate (A_{pr}):** This is the peak rate of anomaly and is a multiple of the anomaly base rate. It is defined as the multiple of the baseline bandwidth: $A_{pr} = \alpha_{pr} B_{mean}$, where $\alpha_{pr} \geq \alpha_{br}$.
- **Temporal Parameters:** The temporal parameters along with the anomaly base and peak rates determine the pattern of the anomaly. The temporal parameters are: Time to onset (t_{onset}) defines the time period before the onset of the first anomaly. Rise-time (t_{rise}) is the time taken by the anomaly traffic to reach the anomaly peak rate (A_{pr}) from the anomaly base rate (A_{br}). Thus, t_{rise} along with A_{pr} decides the slope in case of a ramp anomaly. Peak-time (t_{peak}) is defined as the time for which the anomaly is maintained at the peak rate. Fall-time (t_{fall}) is the time taken by the anomaly traffic to drop to the anomaly base rate from the peak rate. Period (T), in case of periodic anomalies, defines the frequency at which the anomalies occur.
- **Span:** The span parameter, S, determines the total duration of the scenario. This allows an adaptation of AnomBench to experiments that require different amounts of time.
- **Packet characteristics:** The packet characteristics parameter permits the generation of packets with different lengths and packet header flags. This adaptation is useful when using AnomBench for performance testing, where traffic with short packets stresses the I/O and processing components.

These parameters can be changed arbitrarily, but remain fixed for one particular benchmark scenario. When using AnomBench, these parameters should be recorded and reported with the results to permit reproduction of generated traffic and comparison to other work. We identified the complete set of parameters for all sixteen scenarios in our benchmark as discussed in Section IV.

D. Benchmark Metrics

Benchmarks can be used to obtain number of different metrics. In many cases, the actual benchmark results are less important than the workload that is generated on the system (e.g., to measure secondary metrics like power consumption). However, in the context of anomaly, there is one essential metric that dominates: *detection accuracy*. When comparing anomaly algorithms, this metric is likely the most important quantity to consider. We propose to measure this metric in one of two ways:

1) ROC curve: Anomaly detection algorithms often use an internal threshold decision to obtain a binary output that indicates the presence of an anomaly. The choice of the threshold value has a direct relation to the tradeoff between false positive and false negative rates (e.g., a low threshold increases the false positive rate while reducing the false negative rate). A receiver operating characteristic (ROC) curve illustrates the sensitivity of the binary classifier for different threshold values [6]. The area under the curve is often used to quantify the discrimination ability of the detector.
2) False positive/false negative rate pair: Obtaining ROC curves requires the ability of change the internal threshold value of the anomaly detection algorithm. In cases where this is not possible (e.g., black-box anomaly detector) or not applicable (e.g., non-threshold-based detection algorithm), a single pair of false positive and false negative rates can be reported. This pair represents a single point on the ROC curve.

Using one of these two accuracy metrics, anomaly detection algorithms can be compared. However, it should be noted that the false positive/false negative rate pair presents a very limited amount of information. Thus, a comparison may not always provide a strict winner.

IV. Prototype Implementation

We have implemented AnomBench on DETERlab, a testbed for network security experiments [2]. The DETERlab technology is based on Emulab, a widely used experimental infrastructure in the networking domain [17]. Thus, AnomBench can be easily used by other researchers in the community.

The AnomBench implementation consists of a set of scripts that instantiate traffic according to the sixteen scenarios discussed in Section III. We use the SEER tool [14] to translate our scenario specification into actual network traffic. The AnomBench parameters that represent these scenarios and that are used in the prototype system are shown in Table III.

TABLE III
AnomBench Parameters for Prototype Implementation. Scenarios with odd numbers target random ports and scenarios with even numbers target a single port.

Scenario	α_{pr}	t_{rise}	t_{peak}	t_{fall}	T
Scenario 1/2	4	0s	60s	0s	120s
Scenario 3/4	2	0s	60s	0s	120s
Scenario 5/6	4	0s	15s	0s	30s
Scenario 7/8	2	0s	15s	0s	30s
Scenario 9/10	4	60s	0s	2s	120s
Scenario 11/12	2	60s	0s	2s	120s
Scenario 13/14	4	15s	0s	2s	30s
Scenario 15/16	2	15s	0s	2s	30s

Baseline: $B_{mean} = 20$Mbps, $\beta = 0.1$, $t_{onset} = 60$s, $A_{br} = 0$, $S = 250$s, packet characteristics: 100–1000 bytes, SYN attacks.

V. Evaluation

We have evaluated two anomaly detection algorithms using AnomBench. This evaluation illustrates the different AnomBench scenarios and shows accuracy metric results for each algorithm. For the evaluation, we use the following volume-based anomaly detection algorithms:

- Holt-Winter forecasting [4]: This forecasting method maintains several internal variables that characterize trends, seasonal variation, and confidence bands. Based on the history of traffic rates observed, an estimated value for the next time period is computed. If the actual observed value lies outside the estimated values (plus/minus the confidence bands), then an anomaly is reported.
- Cumulative sum [16]: This algorithm averages a window of past observed data rates to obtain an estimate for the next interval. The amount of traffic that exceeds this estimate is added towards a cumulative sum. If this sum exceeds a threshold, an anomaly is reported and the cumulative sum is reset to zero.

The inputs to each algorithm are the packet rates for 1-second intervals collected on the DETERlab implementation of AnomBench.

Our results are shown in Figures 2 and 3. Figure 2 shows the pulse scenarios and Figure 3 shows the ramp scenarios. In both figures, the traffic only the odd-numbered scenarios (random target) are shown. Targets are not distinguished by either algorithm, and thus results are the same for single target. (More sophisticated algorithms that distinguish traffic by port numbers (e.g., entropy-based anomaly detection [7]) would show difference in those cases.) Each subfigure for a scenario shows three lines of information along one x-axis that represents time. The first line shows the data rate of traffic that is generated by AnomBench for this scenario. The colored bars indicate periods of baseline traffic (green) and periods of anomalies (red). The second line shows the results for Holt-Winter and the third line shows the results for cumulative sum. In both cases, the binary algorithm output is shown as a pulse (positive when anomaly is reported, negative when no anomaly is reported). The color of the pulse indicates if the report by the algorithm matches with what AnomBench actually generated.

TABLE IV
Accuracy of Anomaly Detection Algorithms for AnomBench Scenarios.

AnomBench scenario	Holt-Winter false positive	Holt-Winter false negative	Cumulative Sum false positive	Cumulative Sum false negative
1	9.60%	47.10%	0.71%	0.00%
2	8.21%	48.36%	0.00%	0.00%
3	10.36%	53.28%	0.00%	0.00%
4	7.77%	51.69%	2.83%	0.00%
5	2.93%	6.25%	1.10%	0.00%
6	2.93%	3.91%	1.56%	0.00%
7	3.52%	0.00%	0.70%	0.00%
8	3.20%	5.83%	0.36%	0.00%
9	7.55%	7.26%	0.00%	13.71%
10	9.68%	6.56%	0.00%	10.66%
11	10.39%	24.59%	0.00%	45.08%
12	8.60%	26.23%	1.08%	29.51%
13	2.48%	15.00%	0.00%	10.00%
14	2.86%	13.22%	0.36%	14.10%
15	2.14%	16.53%	1.79%	18.18%
16	2.87%	14.63%	2.51%	19.72%

We can observe that in general, both anomaly detection algorithms perform correctly. However, there are several areas where accuracy could be improved:

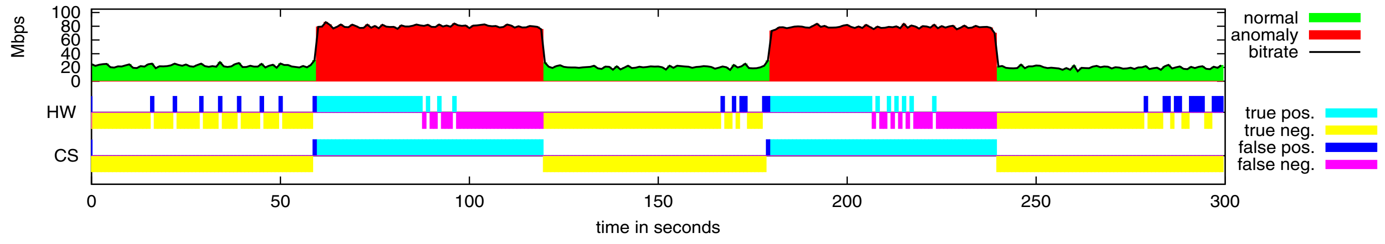
- Long pulses: Pulses that last for a long time (scenarios 1 and 3) cause the algorithms to adapt their reference baseline. Thus, they do not report anomalies after they have persisted for a while.
- Initial ramp phases: Ramps change the rate of traffic very slows. Initially, it is difficult for algorithms to distinguish between variations in the baseline and the beginning of a ramp.

These observations from AnomBench not only help in determining shortcomings of particular algorithms, but they also show what anomaly scenarios are particularly challenging to detect. This information can help in determining where efforts in anomaly detection algorithm development should be targeted.

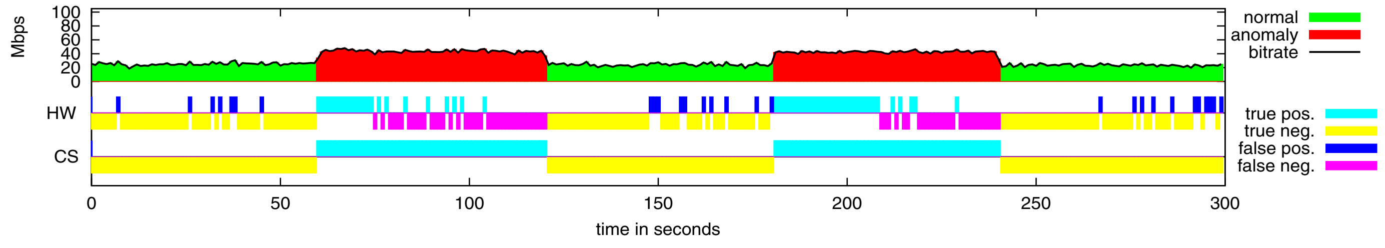
The accuracy of each algorithm for each scenario is shown in Table IV. The results are reported as false negative/false positive rate pair (ROC curves not shown due to space limitations). As observed in the plots above, Holt-Winter performs well on some scenarios (short pulses and long, high-intensity ramps), but poorly on others. With the information obtained from AnomBench, it can be inferred that the algorithm adjusts quickly to new baselines and thus does not detect attacks with long durations or small slopes. For cumulative sum, we can observe that it performs nearly perfectly for pulses. For ramp attacks, it performs nearly as poorly as Holt-Winter.

VI. Summary

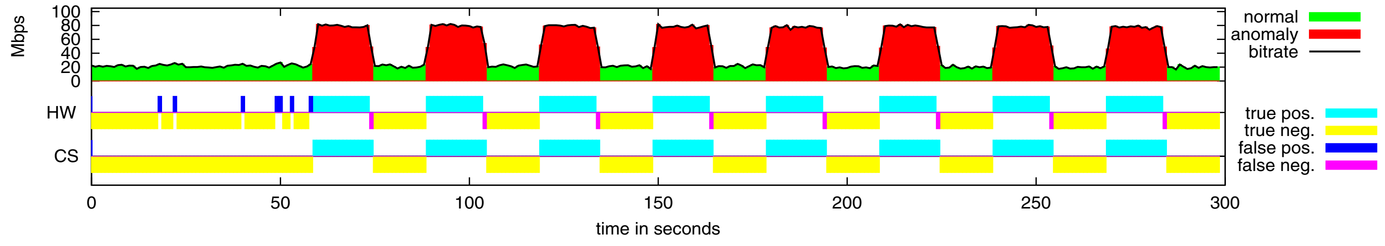
In this paper, we argue for the importance of a benchmark for anomaly detection for comparable and reproducible research in this area. We present AnomBench, a benchmark suite consisting of sixteen different scenarios that encompass a range of volume-based network anomalies. The benchmark is parameterizable to allow for adaptation to different bandwidth and experiment durations. We present result from our



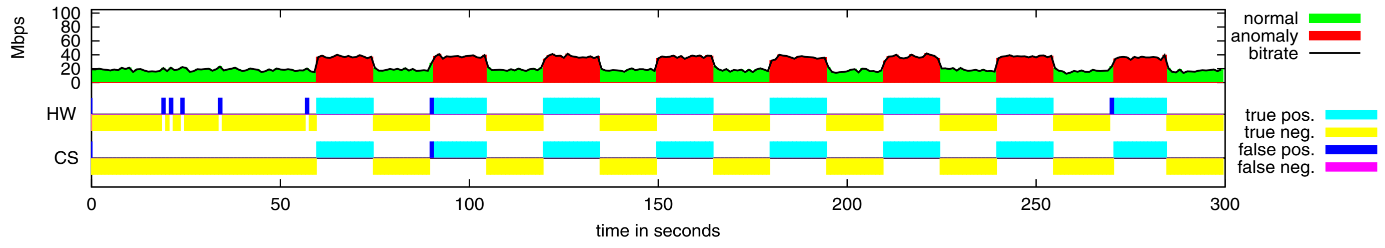
(a) Scenario 1



(b) Scenario 3



(c) Scenario 5



(d) Scenario 7

Fig. 2. AnomBench Scenarios for Pulse Anomalies. Only scenarios with random target ports are shown. Holt-Winter anomaly detection is shown as HW, cumulative sum as CS.

prototype implementation of AnomBench on DETERlab using two different anomaly detection algorithms. The scripts used to run the benchmark as well as traffic traces are available at http://www.ecs.umass.edu/ece/wolf/data/AnomBench/.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. CNS-0325868.

References

[1] T. Ahmed, B. Oreshkin, and M. Coates. Machine learning approaches to network anomaly detection. In *Proc. of Second Workshop on Tackling Computer Systems Problems with Machine Learning Techniques (SysML)*, Cambridge, MA, Apr. 2007.

[2] T. Benzel, R. Braden, D. Kim, C. Neuman, A. Joseph, K. Sklower, R. Ostrenga, and S. Schwab. Design, deployment, and use of the DETER testbed. In *DETER: Proceedings of the DETER Community Workshop on Cyber Security Experimentation and Test on DETER Community Workshop on Cyber Security Experimentation and Test 2007*, Boston, MA, 2007.

[3] D. Brauckhoff, A. Wagner, and M. May. FLAME: a flow-level anomaly modeling engine. In *Proc. of the Workshop on Cyber Security Experimentation and Test (CSET)*, pages 1–6, San Jose, CA, July 2008.

[4] J. D. Brutlag. Aberrant behavior detection in time series for network monitoring. In *Proc. of the 14th Systems Administration Conference*, pages 139–146, New Orleans, LA, Dec. 2000.

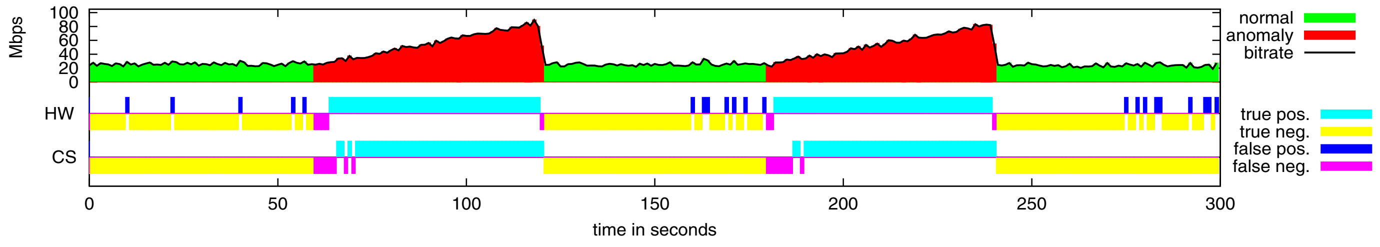
[5] J. M. Estevez-Tapiador, P. Garcia-Teodoro, and J. E. Diaz-Verdejo. Anomaly detection methods in wired networks: a survey and taxonomy. *Computer Communications*, 27(16):1569–1584, Oct. 2004.

[6] T. Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, June 2006.

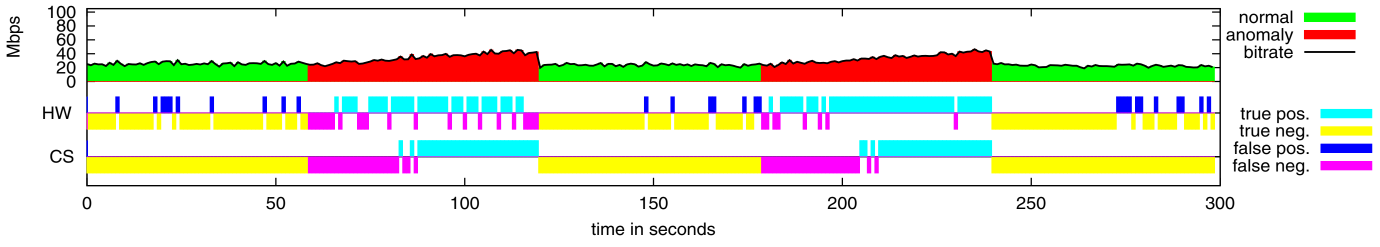
[7] Y. Gu, A. McCallum, and D. Towsley. Detecting anomalies in network traffic using maximum entropy estimation. In *Proc. of the 5th ACM SIGCOMM Conference on Internet Measurement (IMC '05)*, Berkeley, CA, Oct. 2005.

[8] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown. MiBench: A free, commercially representative embedded benchmark suite. In *Proc. of IEEE 4th Annual Workshop on Workload Characterization*, Austin, TX, Dec. 2001.

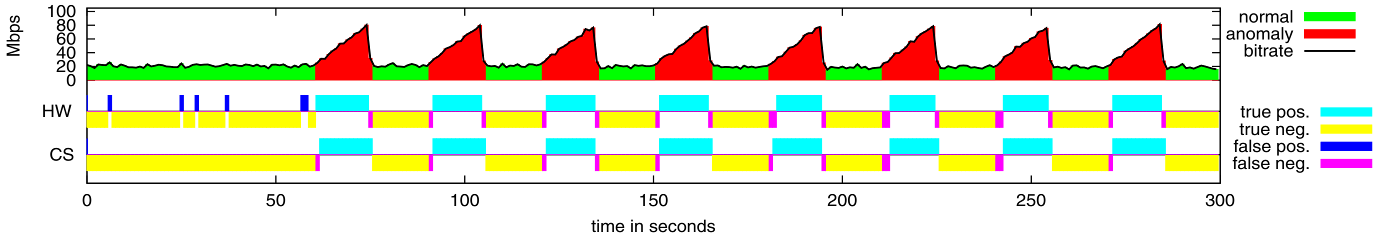
[9] A. Hussain, J. Heidemann, and C. Papadopoulos. A framework for classifying denial of service attacks. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures,*



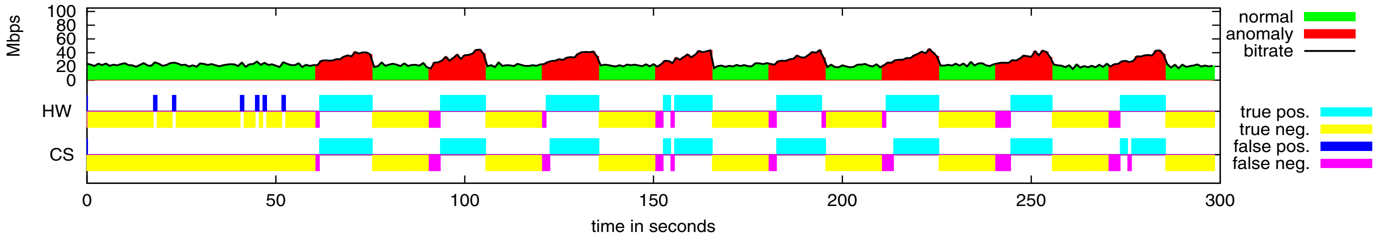
(a) Scenario 9



(b) Scenario 11



(c) Scenario 13



(d) Scenario 15

Fig. 3. AnomBench Scenarios for Ramp Anomalies. Only scenarios with random target ports are shown. Holt-Winter anomaly detection is shown as HW, cumulative sum as CS.

and protocols for computer communications, pages 99–110, Karlsruhe, Germany, Aug. 2003.

[10] C. Lee, M. Potkonjak, and W. H. Mangione-Smith. MediaBench: A tool for evaluating and synthesizing multimedia and communications systems. In *Proc. of IEEE MICRO-30*, pages 330–335, Research Triangle Park, NC, Dec. 1997.

[11] R. P. Lippmann, D. J. Fried, I. Graf, J. W. Haines, K. R. Kendall, D. McClung, D. Weber, S. E. Webster, D. Wyschogrod, R. K. Cunningham, and M. A. Zissman. Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation. In *Proc. of DARPA Information Survivability Conference and Exposition (DISCEX '00)*, volume 2, pages 12–26, Hilton Head, SC, Jan. 2000.

[12] J. Mirkovic, E. Arikan, S. Wei, S. Fahmy, R. Thomas, and P. Reiher. Benchmarks for DDOS defense evaluation. In *Proc. of the 2006 IEEE Conference on Military Communications (MILCOM)*, Washington, DC, Oct. 2006.

[13] J. Mirkovic, A. Hussain, B. Wilson, S. Fahmy, P. Reiher, R. Thomas, W.-M. Yao, and S. Schwab. Towards user-centric metrics for denial-of-service measurement. In *ExpCS '07: Proceedings of the 2007 Workshop on Experimental Computer Science*, San Diego, CA, June 2007.

[14] S. Schwab, B. Wilson, C. Ko, and A. Hussain. SEER: a security experimentation environment for DETER. In *DETER: Proceedings of the DETER Community Workshop on Cyber Security Experimentation and Test on DETER Community Workshop on Cyber Security Experimentation and Test 2007*, Boston, MA, 2007.

[15] Standard Performance Evaluation Corporation, http://www.spec.org/. *SPEC CPU2006 - Version 1.1*, Aug. 2008.

[16] H. Wang, D. Xhang, and K. G. Shin. Change-point monitoring for the detection of dos attacks. *IEEE Transactions on Dependable Secure Computing*, 1(4):193–208, Oct. 2004.

[17] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An integrated experimental environment for distributed systems and networks. In *Proc. of the Fifth Symposium on Operating Systems Design and Implementation*, pages 255–270, Boston, MA, Dec. 2002. USENIX Association.

[18] T. Wolf and M. A. Franklin. CommBench – a telecommunications benchmark for network processors. In *Proc. of IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 154–162, Austin, TX, Apr. 2000.

[19] Q. Wu, Y. Liao, T. Wolf, and L. Gao. Benchmarking BGP routers. In *Proc. of IEEE International Symposium on Workload Characterization (IISWC)*, Boston, MA, Sept. 2007.