Machine Learning Based DDoS Attack Detection From Source Side in Cloud

Zecheng He
Department of Electrical Engineering
Princeton University
Princeton, NJ, 08540
zechengh@princeton.edu

Tianwei Zhang
Department of Electrical Engineering
Princeton University
Princeton, NJ, 08540
tianweiz@princeton.edu

Ruby B. Lee
Department of Electrical Engineering
Princeton University
Princeton, NJ, 08540
rblee@princeton.edu

***Abstract*—Denial of service (DOS) attacks are a serious threat to network security. These attacks are often sourced from virtual machines in the cloud, rather than from the attacker's own machine, to achieve anonymity and higher network bandwidth. Past research focused on analyzing traffic on the destination (victim's) side with predefined thresholds. These approaches have significant disadvantages. They are only passive defenses after the attack, they cannot use the outbound statistical features of attacks, and it is hard to trace back to the attacker with these approaches.**

In this paper, we propose a DOS attack detection system on the source side in the cloud, based on machine learning techniques. This system leverages statistical information from both the cloud server's hypervisor and the virtual machines, to prevent network packages from being sent out to the outside network. We evaluate nine machine learning algorithms and carefully compare their performance. Our experimental results show that more than 99.7% of four kinds of DOS attacks are successfully detected. Our approach does not degrade performance and can be easily extended to broader DOS attacks.

***Keywords*-DDOS attack, Machine Learning, Cloud Computing, Virtual Machine Monitor, Cloud Provider**

I. Introduction

There are many attacks on network infrastructures today. These include attacks on the availability of the network, and on the confidentiality and integrity of the network packets and their sources and destinations. Distributed Denial of Service (DDoS) attacks are attacks targeting the availability of networks, hosts and services from multiple attack source machines. These are some of the most dangerous attacks, especially as they are very easily launched, can cause catastrophic loss of service and are difficult to trace back to the true attackers. In this paper, we focus on the network-based DDoS attacks sourced from virtual machines in the cloud.

A. DOS Attacks and DDOS Attacks

Denial of service attacks (DOS) prevent the legitimate users from accessing network and other resources. DOS attacks can be traced back to the 1980s. There are two main categories of DOS attacks: network/transport-level attacks and application level attacks [1]. Network level DOS attacks disable legitimate users' connectivity by exhausting network resources. Application level DOS attacks disable service by exhausting server resources. Reported by McAfee Lab [2], DoS attacks account for more than one-third of all current network attacks in the world. These attacks can disable accesses to a single webpage, or to very large servers, e.g. email, DNS or http servers.

Some DOS defense approaches require the client to solve a challenge as a proof-of-work in advance. In distributed denial of service (DDoS) attacks, instead of using an attacker's single machine, a bunch of (remotely) controlled computers are used to attack the victim. Attackers intrude into the innocent victim computers (also called *secondary victims*, *bots* or *zombies*), take charge of them and use them as botnets to attack the *primary victim*. Botnets also make it very difficult to trace back to the attacker. Attackers can even rent virtual machines in commodity clouds, potentially with a fake credit card, to start DDoS attacks. DDoS attacks can be categorized into three main types: volumetric DDoS attacks, application layer DDoS attacks and state-exhausting DDoS attacks. Volumetric attacks are the most common DDoS attacks. They are more than 65% of the total DDoS attacks, reported by Arbor [3]. TCP flood attacks, UDP flood attacks and DNS spoofing attacks are examples of this kind of attacks. Application level attacks inject into queries or request malicious resources, e.g. very large images, which exhaust the server resources. Distinct from volumetric attacks, these attacks appear innocent. Malicious packages cannot be distinguished unless the contents are disclosed. The last type of DDoS attacks are state-exhausting attacks, e.g. ping-of-death, which are 20% of all DDoS attacks, reported by Arbor [3].

In preparation for launching DDoS attacks, other attacks may be used to intrude into a secondary victim machine to install bot code. These include password guessing attacks. Among these attacks, the majority are dictionary based attacks and personal information based attacks. Dictionary based attacks try commonly used passwords in a password dictionary. These attacks may take a long time and are not easily successful. However, if the attacker can get personal information about the victim, he may only need to try some targeted passwords, e.g. combinations of name or birth date of the victim. In this paper, we also consider detection of such brute-force password guessing attacks.

B. DDoS Attacks from the Cloud

Cloud computing has been flourishing over the years. Compared to physical machines, virtual machines are lower in price and more flexible in ubiquitous, on-demand computational ability. Users rent virtual machines in the cloud and run their jobs on them. Some jobs are very computationally complex, and thus are hard to achieve on personal computers. Portable devices, e.g., mobile phones and laptops, also put their heavy-computing work in the cloud, in order to save power and computing time [4].

In addition to physical bots, DDoS attacks are also launched on commodity cloud platforms. Attackers rent many virtual machines and use them as VM bots to attack the outside world [5]. Because of the virtual machines' computational ability and untraceability, attackers tend to rent them to launch their attacks instead of using their own physical machines. Attacks can be launched from commodity cloud platforms (or education domains). These attacks significantly degrade the reputation of the cloud provider (or education domain) and can cause huge financial losses. A DDoS attack on Oct 21st, 2016 brought down the Internet connection for most of the east coast USA [6]. Earlier, one of the largest and most sophisticated DDoS attacks targeted the American banks, including Bank of America, JP Morgan Chase, Wells Fargo and PNC Bank, from mid September, 2016.

C. DDoS defenses

Based on where the defense is deployed, DDoS defense mechanisms are classified into two main categories: destination side defenses and source side defenses. In destination side defense systems, the detection and responses to DDoS attacks are done at the victim's side. These systems [7] [8] [9] can observe received packages and cut off the connection once an attack is detected. Also, some of them are able to trace back the attacker using carefully designed protocols or router information, e.g., IP traceback using router information [10], management information based traceback [11], or packet marking based tracing [12]. Packet dropping based on the level of congestion [13] is an automatic way of mitigating these DDoS attacks.

However, destination mechanisms have a few significant drawbacks. First, they are passive defenses. Attacks can only be detected after they get to the destination. By that time, network infrastructures have already been under extreme pressure by attack packages. Second, the only response on the destination side toward the detected attacks is cutting off the connection. This cannot preclude the attack from attacking other victims. Third, detecting on the destination side cannot leverage the joint information of multiple virtual machines sourcing the attack.

In contrast, source side detection can overcome these disadvantages with the help of cloud providers. D-WARD [14], [15] is a system comparing inbound and outbound traffic on the source side to detect DDoS attacks. MUlti-Level Tree for Online Packet Statistics (MULTOPS) [16] detects and filters DDoS flooding attacks with the assumption that the rate of inbound traffic is proportional to outbound traffic during normal operations. MANAnet's reverse firewall [17] is a reverse firewall that prevents attack packages from going out.

D. Our Contributions

In this paper, we propose a machine learning based source side DDoS attack detection system. Our main contributions are:

- We propose a new DDoS attack detection system on the source side, in order to detect attacks and mitigate the impact of the attacks from the source side in the cloud.
- We analyze statistical features of different kinds of attacks in our framework, including the most prevalent DDoS attacks: flooding attacks, spoofing attacks and brute-force attacks. This makes our system very scalable.
- We implement a proof-of-concept prototype and test it in real cloud settings. It achieves up to 99.7% accuracy in detecting four types of DDoS attacks.
- We evaluate and compare nine machine learning approaches in our system. We also compare supervised algorithms and unsupervised algorithms.

This paper is structured as follows: Section II defines the threat model, trusted computing base and the attacker's capabilities. Section III describes the attacks and their feature selection and extraction. Section IV shows the implementation of our system. We also show experimental results and analyze them. Section V concludes this work and proposes directions for future work.

II. Threat Model

A. Attacker's Capability and Trusted Computing Base

The attacker's goal is to breach the availability of services for legitimate users. These include DNS services, HTTP services and FTP services. Attacks that target confidentiality and integrity of data are not included in our threat model.

In our threat model, attackers are able to rent virtual machines in the cloud, or compromise virtual machines. They can take full control of these virtual machines and use them as bots to launch attacks. Attackers can use any number of virtual machines as desired, with their identity carefully hidden. Attackers are not able to physically access the virtual machines, because typically servers (and virtual machines) are located in data centers and can only be remotely accessed.

The cloud provider is trusted in our threat model. The cloud provider, as well as the defense system, can monitor the behaviors of virtual machines and Virtual Machine Monitors (VMM). The VMM is also trusted and can provide statistical information about the virtual machines' states and connections.

All network packages, except a few in the beginning of the connections, are encrypted. The defense system can only access the metadata of network packages, e.g. IP addresses and control signals, but not their internal contents. The defense system can also get statistical information about each virtual machine, e.g. how many packages go inbound and outbound. However, the system cannot access the contents of the packages in order to protect legitimate users' privacy.

B. *Attack Categories*

The design of the system focuses on general DDoS attacks in the cloud. We test our design on four very common network attacks: SSH brute-force attacks, ICMP flooding attacks, DNS reflection attacks and TCP SYN attacks. These are very common and representative mechanisms used in DDoS attacks. Our design can be easily extended to other types of DDoS attacks by simply changing the monitored statistical features.

III. Attacks: Feature Selection and Extraction

A. *Overview of Architecture*

Before considering the feature selection and extraction, we first show the architecture of our system in Figure 1. The VMM monitors the status of the virtual machines and gathers statistical network traffic information. The VMM inputs the gathered information to a machine learning engine. The machine learning engine feeds back whether a suspicious action is detected. If the suspicious behavior is only detected on a single VM, terminating this VM is a reasonable way to mitigate the attack. If the suspicious behaviors are detected among multiple VMs, it is highly possible that a distributed DoS attack is ongoing. Therefore, cutting off the network connections of the suspicious servers can be done to defend against the attack.

The machine learning engine has two modules: the pre-trained module and the online learning module. The pre-trained module is trained in advance to determine whether the virtual machine's actions are suspicious. On the other hand, the online learning module is trained in the background to update the pre-trained module. The online learning module takes monitored data as training samples to modify the parameters in the pre-trained module. We choose four representative DDoS attacks to defend against in this paper.

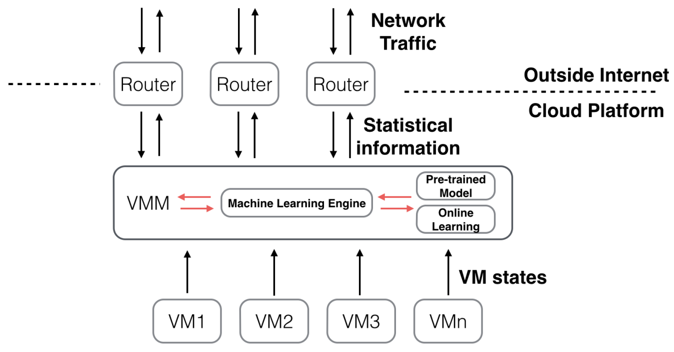

Figure 1. Architecture of proposed system

B. *SSH Brute-force Attack Feature*

The SSH brute-force attack, makes remote brute-force guesses of passwords of legitimate users. It targets taking control remotely of a victim's computer, typically in advance of launching other DDoS attacks. Conventional approaches of mitigating this attack include blocking the user for a while after a predetermined time of failed trials, and disconnecting the channel after three wrong password trials. These two approaches have obvious drawbacks. Blocking user accounts can be misused by attackers to limit the legitimate usage of the machines, itself causing a DoS attack. Also, this protection feature is not provided by official SSH but by third-party firewalls or applications, which may not be trusted. Disconnecting the session after three failed logins is provided by SSH, however, nothing prevents the attacker from connecting again.

We find that the rate of Diffie-Hellman key exchange packages is a good indicator for detecting SSH brute-force attacks. Each time in the SSH establishing stage, a pair of Diffie-Hellman keys is generated and exchanged between the two parties and used to encrypt the following sessions. As mentioned above, three times of authentication failure results in disconnection of the SSH channel. A different pair of Diffie-Hellman keys has to be generated again before the subsequent logins. During an attack, the total number and rate of Dillie-Hellman key exchange packages could be much higher than usual, illustrated in Figure 2. Therefore, we use the number of Dillie-Hellman key exchange packages in a preset interval as a feature to distinguish if there are suspicious SSH brute-force attacks.

Using Diffie-Hellman key exchange packages has many advantages. First, they are the only non-encrypted packages during the session. All the following packages are encrypted with the exchanged keys. Second, Diffie-Hellman key exchanges only happen at the beginning of a session.

This is critical for increasing the Signal-to-Noise Ratio (SNR) in machine learning algorithms. Third, this feature is very distinguishable. As long as the keys have been exchanged and the session has not been closed, no Diffie-Hellman packages will be sent again. Therefore, too many key exchanges during a short period are very suspicious, as shown in Figure 2.

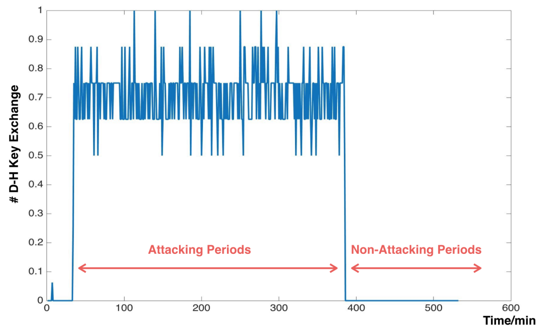

Figure 2. SSH Diffie-Hellman Key Exchange features

C. DNS Reflection Attack Feature

DNS (Domain Name Server) is an essential part of network infrastructures and is responsible for translating domain names to IP addresses. DDoS DNS attacks, especially reflection based attacks, have been big threats for domain name service and network security.

We demonstrate how DNS reflection attacks work in Figure 3. During a DNS reflection attack, the attacker sends out a large amount of DNS requests to DNS servers with the request packages' source IP spoofed as the victim's. These DNS requests ask for heavy-load responses, such as "Give me all your DNS records" or "Give me IP addresses of multiple domain names". These responses overwhelm the victim and exhaust its resources.

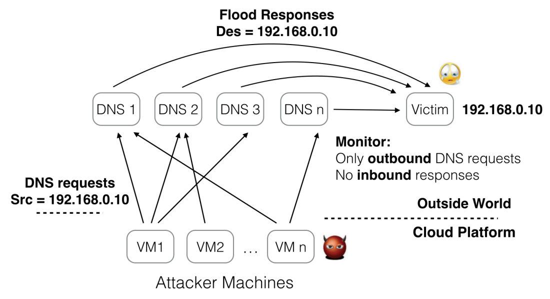

Figure 3. Monitoring ratio of inbound to outbound DNS packages to detect DNS reflection attacks

For DNS reflection attacks, our insight is to monitor inbound and outbound traffic. For normal DNS requests, inbound traffic is approximately proportional to outbound traffic. It is not likely that a huge amount of requests are sent out, but no responses come back. However, during a DNS reflection attack, as the source IP address is spoofed to redirect the response to the victim, no response is returned to the requester. This results in more request packages than response packages. Therefore, we use the inbound/outbound DNS packages ratio to detect this attack. This feature is low-overhead because all we need are two counters in the hypervisor for each virtual machine. Figure 4 verifies our insight of using this feature. As shown in this figure, two attacks can obviously be observed. Both of the attacks have a low inbound/outbound ratio of nearly zero, i.e. only requests but no responses. During non-attack periods, this ratio is very close to 1, i.e. every request tends to have a response.

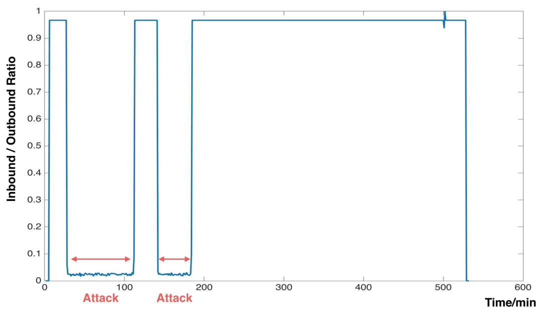

Figure 4. Inbound/outbound DNS packages ratio with or without DNS reflection attacks

D. ICMP Flood Features

The Internet Control Message Protocol (ICMP) is designed to transmit control information, such as error indicators. It is one of the main protocols in the Internet Protocol (IP) suite. ICMP is different from the UDP and the TCP protocols in that ICMP is not used to transfer data. An ICMP flood is an attack where the attacker sends a huge amount of ICMP packages and overwhelms the victim.

Unlike SSH or DNS, there are not many ICMP packages under normal situations. Therefore, we directly use the ICMP package rate as an indicator of an ICMP flood attack. Usually, there are very few ICMP packages, thus a large number of ICMP packages during a short time period is very suspicious. In addition to the number of ICMP packages from a single virtual machine, we can get more accurate results by jointly analyzing multiple virtual machines' ICMP status.

E. TCP SYN Attack Features

The TCP SYN attack is a stateful protocol attack. It leverages a special feature of TCP protocols: three handshakes for establishing a new connection. First, the client sends **SYN x** to the server to ask for starting the connection. Then, the server replies with an **ACK x** and **SYN y** to indicate that the server receives the request and is ready for synchronizing.

Then the client sends an **ACK y** package to the server to acknowledge that he has received the reply package from the server and is ready for synchronized communications.

The server allocates its resources for the connection (CPU, memories, ports, etc) as it sends out the **ACK x/SYN y** package. However if the client does not reply with **ACK y**, these resources will be kept for a long period until timeout. In the SYN attack, the attacker asks for connection but does not reply with ACK y in the third step. Alternatively, the IP address of the attacker is spoofed so the server sends **ACK x/SYN y** to the spoofed IP address, which does not respond with **ACK y** since it never sent the SYN request, or it may even be a non-existent machine. In any case, resources are allocated for unfinished handshakes on the server side. Before timeout, these resources cannot be allocated for other legitimate SYN requests. The server's network resources are quickly exhausted if malicious SYN requests are far more than legitimate ones. Figure 5 shows normal handshakes (left) and a SYN attack (right).

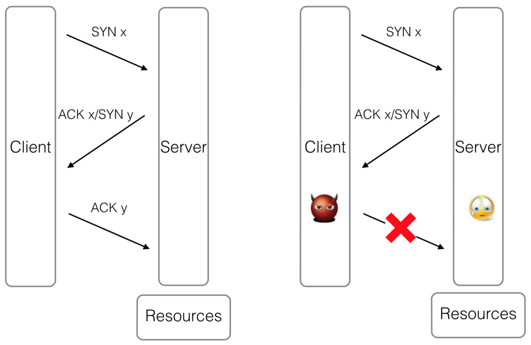

Figure 5. Normal TCP handshakes (left) and SYN attacks (right)

One noticeable feature of the TCP SYN attack is the ratio of TCP packages with ACK tags and SYN tags. During normal periods, packages with ACK tags are much more than packages with SYN tags (at least 100X in our experiments). That is because SYN tags are mainly used only in the session establishment phase, however, ACK packages are also used in data transmission packages. Therefore, it is very abnormal if the SYN/ACK ratio reaches a high ratio. Figure 6 shows the SYN/ACK ratio during a SYN attack. It confirms our intuition that this ratio can be used as a SYN attack indicator.

F. Algorithm for DDoS Attack Detection

Algorithm 1 shows the algorithm we propose for our DDoS detection system. n is the number of statistical features, k is the number of servers, l_i is the number of VMs on server i. Concatinate all F_{ij} vectors on all servers of interest to generate a long feature vector **F** and send it to the machine learning engine. This engine uses a pre-trained machine learning module, M_0, to detect if there are any DDoS attacks.

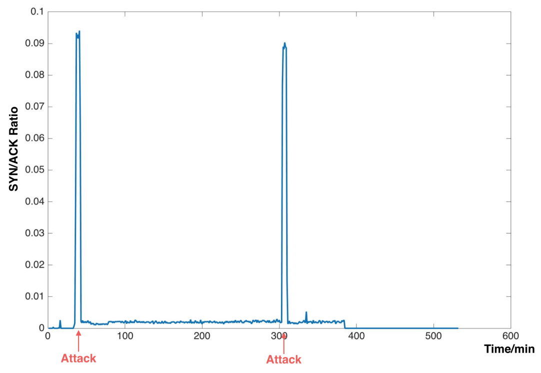

Figure 6. SYN/ACK ratio during attacks (high points) and normal time (low points)

```
Select monitored features $f^1, f^2 ... f^n$
for Server $S_i$, $i = 1, 2...k$ do
    for $VM_{ij}$ on server $S_i$, $j = 1, 2...l_i$ do
        $VMM_i$ reports monitored statistical features $F_{ij}$ of $VM_{ij}$
        where $F_{ij} = \{f_{ij}^1, f_{ij}^2 ... f_{ij}^n\}$
    end
end
```

Algorithm 1: Algorithm for feature generation and classification

The system can also have an online learning mechanism. For example, other machine learning modules, $M_1, M_2 ... M_r$, in the background can be classifying the feature vector **F** as well. If a predefined number k of these modules classify this feature vector **F** as a benign or malicious vector, **F** (with its output label of *not-attack* or *attack*, respectively) is used to update the main testing module M_0.

IV. Implementation and Evaluation

A. Cloud Platform

We implement a prototype of our detection system under real cloud settings. The cloud has six servers ($S_0 ... S_5$) and each server runs multiple virtual machines. Three of the servers work on an Intel Xeon CPU E5-2690 CPU with 64 GB memory. The other three are equipped with Intel Xeon CPU E3-1230 V2 CPU with 32 GB memory. All cloud servers and virtual machines launched on them are running Ubuntu 14.04. Each virtual machine launched on the server runs on one 3.2GHz virtual machine using the OpenStack cloud infrastructure.

We do two experiments to test our defense mechanisms. In the first experiment, we launch four different kinds of DDoS attacks (SSH brute-force, DNS reflection, ICMP flooding and TCP SYN attacks) on virtual machines from S_0. The

Table I
DETECTION RESULTS OF DIFFERENT MACHINE LEARNING ALGORITHMS

Method	Accuracy(%)	FP(%)	FN(%)	Precision(%)	Recall(%)	F1- Score
LR	94.36	0.00	7.85	100.0	92.15	0.9591
SVM Linear Kernel	93.85	1.41	7.92	99.41	92.08	0.9560
SVM RBF Kernel	93.90	2.46	7.51	98.96	92.49	0.9562
SVM Poly Kernel	94.07	3.38	7.23	98.58	92.77	0.9559
Decision Tree	94.24	1.73	7.04	99.29	92.96	0.9602
Naive Bayes	94.92	0.00	7.07	100.00	92.93	0.9293
Random Forest	94.96	0.81	6.60	99.67	93.40	0.9643
K-means (Unsupervised)	64.05	22.93	41.07	86.75	58.93	0.7019
Gaussian EM	63.26	95.88	13.39	69.56	86.61	0.7715

Table II
JOINT DETECTION RESULTS OF THREE VIRTUAL MACHINES

Method	Accuracy(%)	FP(%)	FN(%)	Precision(%)	Recall(%)	F1- Score
LR	97.77	0.37	3.82	99.68	96.18	0.9790
SVM Linear Kernel	99.73	0.068	0.44	99.94	99.56	0.9975
SVM RBF Kernel	98.15	3.78	0.24	96.93	99.76	0.9832
SVM Poly Kernel	99.13	0.40	1.27	99.66	98.73	0.9920
Decision Tree	99.07	0.061	0.0167	99.95	98.33	0.9913
Naive Bayes	98.47	3.07	0.27	97.51	99.73	0.9861
Random Forest	99.53	0.00	0.09	100.0	99.12	0.9956
K-means (Unsupervised)	87.76	0.44	22.05	99.54	77.95	0.8743
Gaussian EM	66.53	13.17	50.37	81.94	49.63	0.6182

victim is a virtual machine on another server S_1 running web service. We deploy our defense system on the server which launches virtual machines running the attacks. Virtual machines on the other servers (except S_0 and S_1) request web service, simulating the legitimate users. In the second experiment, the attacks source from three virtual machines on S_0, S_2 and S_3 to simulate distributed DoS attacks. The victim is the same virtual machine on S_1. The defense systems are deployed on S_0, S_2 and S_3. Our experiments are safe since they run behind a VPN router, so the attack packages never escape to the outside Internet.

B. Data Collection and Machine Learning Algorithms

In our experiments, we collect network packages coming in and going out of the attacker virtual machine(s) for 9 hours. Four kinds of attacks are programmed to randomly start and end. Some of them may be started simultaneously. Our goal is detecting attacks, no matter which category an attack falls into. We evaluate both supervised learning and unsupervised learning algorithms. For supervised classification, we evaluate Linear Regression (LR), SVM (with linear, RBF or polynomial kernels), Decision Tree, Naive Bayes and Random Forest algorithms. We also test unsupervised learning algorithms, k-means and Gaussian-Mixture Model for Expectation-Maximization (GMM-EM). The time interval for collecting statistical features is 60 seconds. Table I shows the results of monitoring a single virtual machine with a pretrained learning module. Table II shows the results of simultaneously monitoring three virtual machines on three servers.

C. Detection Results

We split our gathered data into training samples (80%) and testing samples (20%), and use cross-validation to evaluate our performance. We also apply multi-dimensional analysis on these results. We use several performance metrics. Accuracy indicates the overall correct detection over testing samples. False Positive (FP) and False Negative (FN) indicate false alarms and misses, respectively. Precision shows what portion of alarms are true alarms. Recall shows the portion of attacks that are detected. F1 score is a frequently used criterion to balance FP and FN. The higher F1 score indicates the better performance for an algorithm. We show all results in Table I and Table II. The definition of Precision, Recall and F1 score are:

$$Precision = \frac{TP}{TP + FP}$$
$$Recall = \frac{TP}{TP + FN}$$
$$F1 = 2\frac{Precision * Recall}{Precision + Recall}$$

In the single host server monitoring experiment (Table I), supervised algorithms all achieve over 93% accuracy and over 0.95 F1 scores (except Naive Bayes with F1=0.9293). Among them, Random Forest performs the best, with 94.96% accuracy and 0.9643 F1-score. Also, Random Forrest achieves the highest recall, which means that it detects the most attacks among all algorithms being compared. Naive Bayes achieves zero FP rate and relatively low FN. The last two rows show k-means and Gaussian EM algorithms, which do not perform as well as the others because they are unsupervised learning algorithms, i.e., they learn from unlabeled samples. To improve the performance of these unsupervised learning algorithms, we can 1) use joint data from multiple VMs for training, and 2) retrain the model more frequently because unlabeled data are easier to get than labeled data. Although there are concerns about unsupervised learning, unsupervised learning algorithms can be integrated into our system's online updating mechanism, by first clustering clean samples for online learning.

In the multiple hosts monitoring experiment (Table II), all machine learning algorithms get better results than in the single host monitoring experiment. We achieve the highest 0.9975 F1-Score and 99.73% accuracy using SVM with a linear kernel. Four algorithms (SVM with Linear and Poly kernels, Decision Tree and Random Forest) achieve accuracy greater than 99%. K-means improves by 23% in this experiment. Gaussian EM algorithm only improves by 3%, which indicates that the data do not follow a Multivariate-Normal distribution. These results demonstrate that our design is feasible in real cloud settings.

V. Conclusion and Future Work

In this paper, we propose a DDoS attack detection system based on machine learning to prevent attacks on the source side in the cloud. We extract statistical features of four DDoS attacks and launch real attacks in lab settings for evaluation. Our proposed system is able to detect attacks with high accuracy (99.7%) and low false positives ($< 0.07\%$). By detecting DDoS attacks at the source virtual machines in the cloud, we can "nip the attacks in the bud" and also protect the cloud provider's reputation.

We give some directions for future work:

- Combine different machine learning algorithms for better performance, especially unsupervised learning performance.
- Investigate more DDoS attacks and integrate their features into the current system.

References

[1] S. Ranjan, R. Swaminathan, M. Uysal, and E. W. Knightly, "Ddos-resilient scheduling to counter application layer attacks under imperfect detection." in *INFOCOM*. Citeseer, 2006.

[2] Mcafee lab threat report. [Online]. Available: http://www.mcafee.com/us/resources/reports/rp-quarterly-threat-q1-2015.pdf

[3] Worldwide infrastructure security report. [Online]. Available: https://www.arbornetworks.com/images/documents/WISR2016_EN_Web.pdf

[4] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless communications and mobile computing*, vol. 13, no. 18, pp. 1587–1611, 2013.

[5] R. Miao, R. Potharaju, M. Yu, and N. Jain, "The dark menace: Characterizing network-based attacks in the cloud," in *Proceedings of the 2015 ACM Conference on Internet Measurement Conference*. ACM, 2015, pp. 169–182.

[6] Ddos attack against servers accross the east coast usa. [Online]. Available: http://www.cnbc.com/2016/10/21/major-websites-across-east-coast-knocked-out-in-apparent-ddos-attack.html

[7] S. Noh, C. Lee, K. Choi, and G. Jung, "Detecting distributed denial of service (ddos) attacks through inductive learning," in *International Conference on Intelligent Data Engineering and Automated Learning*. Springer, 2003, pp. 286–295.

[8] T. Shon and J. Moon, "A hybrid machine learning approach to network anomaly detection," *Information Sciences*, vol. 177, no. 18, pp. 3799–3821, 2007.

[9] N. B. Amor, S. Benferhat, and Z. Elouedi, "Naive bayes vs decision trees in intrusion detection systems," in *Proceedings of the 2004 ACM symposium on Applied computing*. ACM, 2004, pp. 420–424.

[10] S. T. Zargar, J. Joshi, and D. Tipper, "A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 2046–2069, 2013.

[11] J. B. Cabrera, L. Lewis, X. Qin, W. Lee, R. K. Prasanth, B. Ravichandran, and R. K. Mehra, "Proactive detection of distributed denial of service attacks using mib traffic variables-a feasibility study," in *Integrated Network Management Proceedings, 2001 IEEE/IFIP International Symposium on*. IEEE, 2001, pp. 609–622.

[12] T. Peng, C. Leckie, and K. Ramamohanarao, "Protection from distributed denial of service attacks using history-based ip filtering," in *Communications, 2003. ICC'03. IEEE International Conference on*, vol. 1. IEEE, 2003, pp. 482–486.

[13] A. Yaar, A. Perrig, and D. Song, "Pi: A path identification mechanism to defend against ddos attacks," in *Security and Privacy, 2003. Proceedings. 2003 Symposium on*. IEEE, 2003, pp. 93–107.

[14] J. Mirkovic, G. Prier, and P. Reiher, "Attacking ddos at the source," in *Network Protocols, 2002. Proceedings. 10th IEEE International Conference on*. IEEE, 2002, pp. 312–321.

[15] ——, "Source-end ddos defense," in *Network Computing and Applications, 2003. NCA 2003. Second IEEE International Symposium on*. IEEE, 2003, pp. 171–178.

[16] T. M. Gil and M. Poletto, "Multops: A data-structure for bandwidth attack detection." in *USENIX Security Symposium*, 2001, pp. 23–38.

[17] Mananet reverse firewall. [Online]. Available: http://www.cs3-inc.com/MANAnet.html