A Modeling Approach to Classifying Malicious Cloud Users via Shuffling *

Yudong Yang
Columbia University
yyd@cs.columbia.edu

Vishal Misra
Columbia University
misra@cs.columbia.edu

Dan Rubenstein
Columbia University
danr@cs.columbia.edu

ABSTRACT

DDoS attacks are still a serious security issue on the Internet. We explore a distributed Cloud setting in which users are mapped to servers where malicious users mapped to the same server can thwart the performance of legitimate users. By periodically shuffling the mapping of users to servers and observing how this affects successfully attacked servers, the malicious users can be identified. We use simple models to understand how to best score these observations to identify malicious users with well-defined levels of confidence.

1. INTRODUCTION

The mapping between client and server can depend on many factors, but in an environment in which there are denial-of-service attacks, one factor driving the mapping is to separate "legitimate" sessions from "attackers", the latter of whom are attempting to prevent continual access by the legitimate sessions. Attackers have various methods to cause an interruption.

Without knowing a priori the subset of sessions who are attackers and the remaining subset that are legitimate, the system must monitor the existing sessions, identify servers that are overloaded, and through this identification, try and assess which sessions are behaving maliciously. After forming a hypothesis regarding how to classify sessions as malicious and legitimate, the system can re-map sessions to servers such that the malicious sessions are "honey-potted": all sent to a small collecion of servers where they continue their attack in the belief that they are thwarting legitimate communication. However, the system, if working successfully, will have assigned legitimate sessions to a separate set of servers so that they can continue their communications in an uninterrupted fashion.

One way to formulate such a hypothesis is to periodically **shuffle** the mapping of sessions to servers, and monitor the communication performance of connections after the shuffle[1, 2]. This shuffle can be used to score sessions based on classification of the server on which they reside. Attacking sessions, by their own nature, should reside more frequently on servers identified being attacked than sessions that are legitimate. Over time, the scores of these two classes of sessions should diverge, in effect revealing the attackers.

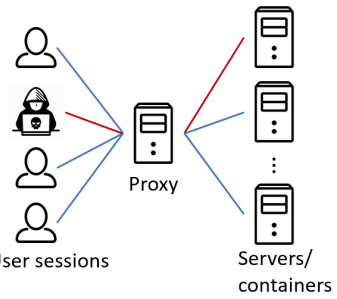

Figure 1: Client sessions map to servers through network proxys.

This paper investigates the scoring process, and uses simple models to understand how effectively the shuffling process is able to identify the attackers.

2. MODEL AND ANALYSIS

We assume a collection of M servers and N sessions, where sessions are mapped to servers such that each server supports approximately the same number of sessions (i.e., N/M for the case where all sessions get mapped to servers). Let $K < N$ be the number of sessions in the **attacking class**, who have malicious intentions. The remaining $N - K$ sessions fall within the legitimate class.

Our initial model assumes that sessions within a class are homogeneous, i.e., each legitimate session adds a load of 1 to a server, while each attacker adds an additional load of $\rho > 0$ (i.e., an attacker's load is $1 + \rho$.) A server who hosts v sessions, ℓ of which are attackers, has a load of $v + \ell\rho$. A server identifies that it is being attacked when its load exceeds some threshold τ. If a server is capable of hosting M legitimate sessions, then clearly $\tau > M$.

Since tracking resource usage per client significantly complicates difficulties system design [1], servers generally only report back whether their workloads are under or over τ, but do not report back the amount. Hence, we do not use the distance from τ as an estimator for the number of attackers: we have only a boolean value for the configuration of the server: attacked or not attacked. We define $A = \left\lfloor \frac{\tau - v}{\rho} \right\rfloor$ to be the server's attacking sensitivity, meaning the maximum number of attack sessions before a server can identify being attacked. Note the model leaves open the possibility that a mix of (attacker and legitimate) sessions may result in an allocation that is above or below the threshold, i.e., sometimes attackers may be mapped in such a way that their attack lies "below the radar".

*This work is supported in part by AFRL Contract HR0011-16-C-0055. Opinions, findings, conclusions, and recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA or the US Government.

Note that in this model, a server that has no attackers should always have its load fall below τ, and a server who serves only attackers should have a load $v(1+\rho) > \tau$. If this were not the case, then either the system would always be attacked, or would never be attacked.

2.1 Shuffling Process

Our shuffling process is inspired by previous work proposed by Jia et al[1, 2] as Shuffling-based moving-target defense. Their idea is to shuffle the clients on compromised servers expecting the attackers will segregate to a smaller subset of servers. This method is further studied in [3], where the cost of moving sessions around is considered into optimization. In [4], the authors study the numerical evaluation of shuffling-scoring defenses by decreasing overloaded server scores by 1, otherwise increases by 1, which is a special case of our work.

We consider a simple shuffling process in which the mapping of sessions to servers is effectively uniformly random. Note that it may be possible to perform subsequent shufflings based on the results of earlier shuffles. However, this "stateful" approach is significantly more complex, and we believe that studying the simpler process to provide a baseline is essential before considering more sophisticated stateful mappings.

As the shuffling process proceeds, after s shuffles, each session i can be described by an s-dimensional vector $v_i = \langle x_1^i, x_2^i, \cdots, x_s^i \rangle$, with $x_j^i \in \{0, 1\}$ with $x_j^i = 1$ indicating that session i was placed on a server that was attacked (above threshold) during the jth shuffle.

After each shuffle, session i can be assigned a score, γ, a function whose input is the session's vector, e.g., $\gamma(x_j^i) = 2x_j^i - 1$, i.e., adding 1 for shuffle over threshold, subtracting 1 for each shuffle under. For simplicity, since x_j^i is a boolean variable, we denote γ_0 and γ_1 to be the value of $\gamma(0)$ and $\gamma(1)$. In order to differentiate the sessions, we require $\gamma_0 \neq \gamma_1$, and for simplicity, we assume w.l.o.g. $\gamma_1 > \gamma_0$. After s shuffles, the score of session i is $\gamma^i = \sum_{j=1}^{s} \gamma(x_j^i)$. We show in Sec.2.5 that the selection of γ does not affect the estimation.

Since attackers should appear with (perhaps slightly) higher frequency upon servers over threshold, the scores of attackers should, over time, diverge from the scores of non-attackers. For the remainder of this paper, we propose answers to the following question:

- What is the right scoring function to use?
- For a given choice of set size (number of sessions) and scoring function, how many shuffles are needed to distinguish the attacking from legitimate sessions?
- Given a threshold value, what is the optimal set size that should be mapped to a server?

2.2 Probablities of identifying attacks

In each shuffle, we first define $a(v, k, N, K)$ to be the probability of having k attackers in a random selected subset of v sessions from a set of size N sessions (with K attacking sessions and $U = N - K$ legitimate sessions). We have

$$\begin{aligned} a(v,k,N,K) &= \binom{v}{k} \frac{K}{N} \frac{K-1}{N-1} \cdots \frac{K-k+1}{N-k+1} \\ &\quad \frac{U}{N-K} \frac{U-1}{N-K-1} \cdots \frac{U-v+k+1}{N-v+1} \\ &= \binom{v}{k} \binom{N-v}{K-k} / \binom{N}{K} \end{aligned} \tag{1}$$

A server can identify as being attacked when the number of attacking sessions is greater than A. For a given attacking session i, the probability that i is shuffled to a non-identified server ($x_j^i = 0$) is

$$q_A = \sum_{k=0}^{A-1} a(v-1, k, U, K-1)$$

while the probability that i is shuffled to a attacked server ($x_j^i = 1$) is $p_A = 1 - q_A$.

For a given legitimate session i, we have the probability of $x_j^i = 0$ to be

$$q_B = \sum_{k=0}^{A} a(v-1, k, U-1, K)$$

and the probability of $x_j^i = 1$ to be $p_B = 1 - q_B$. It is easy to see that $p_A > p_B$, as the attacking sessions have higher probability than legitimate sessions of being detected by the server.

2.3 Random walk model of the scores

We model the process of each session's reputation score as a random walk. Initially, all the sessions have score 0. There are two types of random walk: the attacking sessions P_A and the legitimate sessions P_B. After each shuffle, an attacking session add its score by γ_1 with probability p_A, and γ_0 with probability q_A respectively. A legitimate session will add γ_1 to its score with probability p_B and γ_0 with probability q_B respectively.

We denote $P_A(s, R)$ (or $P_B(s, R)$) to be the transition probability of an attacking (or legitimate) session has reputation score R after s rounds of shuffles.

$$P_A(s, R) = p_A P_A(s-1, R-\gamma_1) + q_A P_A(s-1, R-\gamma_0)$$

$$P_B(s, R) = p_B P_B(s-1, R-\gamma_1) + q_B P_B(s-1, R-\gamma_0)$$

and $P_A(0, 0) = P_B(0, 0) = 1$.

Denote μ_A, μ_B to be the mean score $E[\gamma(x_j^i)]$ of one shuffle. For distribution P_A, after s rounds of shuffles, this score is

$$\mu_A^s = s\mu_A = s(p_A\gamma_1 + q_A\gamma_0) \tag{2}$$

with variance

$$(\sigma_A^s)^2 = s p_A q_A (\gamma_0 - \gamma_1)^2. \tag{3}$$

For process P_B, we have similar results:

$$\mu_B^s = s\mu_B = s(p_B\gamma_1 + q_B\gamma_0) \tag{4}$$

and the variance is

$$(\sigma_B^s)^2 = s p_B q_B (\gamma_0 - \gamma_1)^2 \tag{5}$$

2.4 Accuracy level

Each time we shuffle, processes P_A and P_B drift at rate μ_A and μ_B. Assume w.l.o.g. $\gamma_1 > \gamma_0$, and thus $\mu_A > \mu_B$. As shuffling proceeds, the sessions in P_A and P_B can be separated using a **decision threshold** β. For each session i, if the score $\gamma^i > \beta$, then we can estimate the session to attacking class; otherwise $\gamma^i \leq \beta$ and the session is estimated to legitimate class.

We introduce the **accuracy level** c_A and c_B of the threshold estimation, defined as the probability of a given attacking session (or a legitimate session) classified correctly, e.g., $c_A = 0.99$ meaning that each attacking session have been classified to attacking class with probability 0.99, while the remaining 0.01 probability is being classified to legitimate class by mistake.

As the shuffle process proceeds, the distribution of scores converges to a Gaussian distribution[5]. We estimate the accuracy level using its standard deviation:

$$c_A = \Phi\left(\frac{\mu_A^s - \beta}{\sigma_A^s}\right), \quad c_B = \Phi\left(\frac{\beta - \mu_B^s}{\sigma_B^s}\right)$$

where Φ is the CDF of the standard normal distribution.

2.5 Distinguish scores

In this section, we first answer the question of how many shuffles are needed to separate the attacking and legitimate session.

For given accuracy levels c_A and c_B, we need to find the minimum number of shuffles s and the according decision threshold β which satisfy the accuracy requirements. We define the problem as

$$\begin{aligned} \min \quad & s \\ \text{s.t.} \quad & s > 0 \\ & \beta_A = \mu_A^s - \Phi^{-1}(c_A)\sigma_A^s \\ & \beta_B = \mu_B^s + \Phi^{-1}(c_B)\sigma_B^s \\ & \beta_A \geq \beta_B \end{aligned}$$

where β_A and β_B are two decision thresholds respectively determined by the accuracy level c_A and c_B, e.g., for $c_A = c_B = 97.7\%$, we have $\beta_A = \mu_A^s - 2\sigma_A^s$ and $\beta_B = \mu_B^s + 2\sigma_B^s$. Note that the intervals (∞, β_A) and $(\beta_B, -\infty)$ are the confidence intervals for the attacking and legitimate sessions respectively. When $\beta_A \geq \beta_B$, we can conclude that these intervals do not overlap, and there exists a $\beta, \beta_A \geq \beta \geq \beta_B$ that satisfies both accuracy requirements.

By solving s and substituting (3)(5), we have the solution of s is

$$s^* = (\gamma_1 - \gamma_0)^2 \left(\frac{\Phi^{-1}(c_A)\sqrt{p_A q_A} + \Phi^{-1}(c_B)\sqrt{p_B q_B}}{\mu_A - \mu_B}\right)^2 \quad (6)$$

and the associated decision threshold β is solved by

$$\beta^* = \mu_A^s - \Phi^{-1}(c_A)\sigma_A^s = \mu_B^s + \Phi^{-1}(c_B)\sigma_B^s \quad (7)$$

For the scoring function, we investigate by changing the value of score function γ and see how it impacts the solution s^*. Let $C_0 = (\Phi^{-1}(c_A)\sqrt{p_A q_A} + \Phi^{-1}(c_B)\sqrt{p_B q_B})^2$ and substitute (2)(4) to (6):

$$s^* = C_0 \left(\frac{\gamma_1 - \gamma_0}{(p_A - p_B)\gamma_1 + (q_A - q_B)\gamma_0}\right)^2 \quad (8)$$

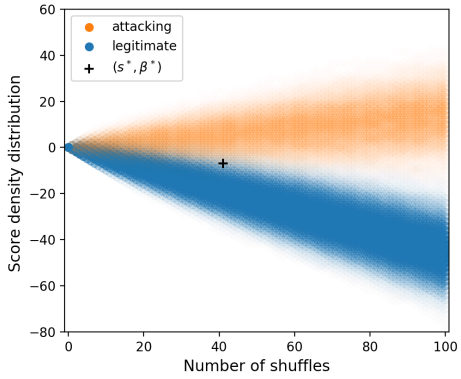

Figure 2: Estimating using s^* and β^*

An immediate observation is that γ is scaling equivalent, i.e., $\forall k \neq 0$, scaling γ_0, γ_1 to $k\gamma_0$, $k\gamma_1$ will have the same solution for s^*. Thus, we fix $\gamma_1 = 1$ and take the derivative

$$\frac{\partial s^*}{\partial \gamma_0} = \frac{(\gamma_0 - 1)(p_A + q_A - p_B - q_B)}{(p_A - p_B) + (q_A - q_B)\gamma_0}$$

Because $p_A + q_A = p_B + q_B = 1$ by definition, we always have $\frac{\partial s^*}{\partial \gamma_0} = 0$. This result indicates that **for any $\gamma_0 \neq \gamma_1$, the solutions are the same**.

Now we consider the question of scaling the number of servers/containers M. Intuitively, too many or too less servers will result in no server or all server detect attacking, which make the attacking sessions less differentiated. Let M^* be the optimal number of servers that minimize the number of shuffles. We define the optimization problem as

$$\begin{aligned} \min \quad & s^*(M^*) \\ \text{s.t.} \quad & 1 \leq M^* \leq N \end{aligned}$$

where s^* is defined in (6), but now as a function of M^*. The probabilities p_A, q_A, p_B, q_B in (6) are also functions of M^*.

3. SIMULATION RESULT

we study a special case where $N = 12000, K = 2000, M = 1000, \gamma_1 = 1, \gamma_0 = -1$. A server can detect attacking when there are more than $A = 2$ attacking sessions. $c_A = c_B = 0.977$, thus $\Phi^{-1}(c_A) = \Phi^{-1}(c_B) \approx 2$. The number of shuffles needed is $s^* = \left(\frac{\sqrt{p_A q_A} + \sqrt{p_B q_B}}{p_A - p_B}\right)^2 \approx 41$ and $\beta^* \approx -6.98$. We plot the score distributions (y-axis) and number of shuffles (x-axis) in Fig.2.

4. REFERENCES

[1] Q. Jia, K. Sun, and A. Stavrou. Motag: Moving target defense against internet denial of service attacks. In *Computer Communications and Networks (ICCCN), 2013 22nd International Conference on*, pages 1–9. IEEE, 2013.

[2] Q. Jia, H. Wang, D. Fleck, F. Li, A. Stavrou, and W. Powell. Catch me if you can: a cloud-enabled ddos defense. In *Dependable Systems and Networks (DSN), 2014 44th Annual IEEE/IFIP International Conference on*, pages 264–275. IEEE, 2014.

[3] Y.-H. Lin, J.-J. Kuo, D.-N. Yang, and W.-T. Chen. A cost-effective shuffling-based defense against http ddos attacks with sdn/nfv. In *Communications (ICC), 2017 IEEE International Conference on*, pages 1–7. IEEE, 2017.

[4] Y. Shan, G. Kesidis, and D. Fleck. Cloud-side shuffling defenses against ddos attacks on proxied multiserver systems. In *Proceedings of the 2017 on Cloud Computing Security Workshop*, pages 1–10. ACM, 2017.

[5] T. Yamane. Statistics: An introductory analysis. 1973.