Statement of Research

Huasong Shan

January 24, 2018

1 General Introduction

I have more than 10 years experience at the internet, IT and telecommunication industry, good command of various technologies which cover performance and security of large-scale web applications, cloud computing, data mining, database and telecommunication. I have a broad research interest in system and security, including measurement and performance analysis, cloud computing, distributed computing, web security, user behavior model, virtual cluster management, etc.

The final goal of my research is to build highly available, scalable, and secure systems for meeting the humans requirements of usability of the computer systems. Common ground is told, usability means that the computer systems should be easy to use, easy to learn, efficient, effective, engaging, error tolerant. This is very grand blueprint, I started my research in distributed systems and cloud computing with a current focus on performance and scalability analysis of large-scale web applications (e.g., Amazon, Facebook).

My work and research philosophy is *"show me the code, show me the data, and show me the graph"*, put everything visible. Based-on the philosophy I have built several tools, which can transform the requirements of project management, log anomaly detection, performance bottleneck detection, intrusion detection into visibility. Overall, my research approaches are experiments-based and data-driven, specifically, using automated infrastructure to visualize real-time performance metrics, using applied mathematics (e.g., queuing theory) to model the systems, using feedback control theory (e.g., Kalman filter) or reinforcement learning to optimize model parameters, using time-series event correlation analysis to diagnose performance bottlenecks, using user-behavior model and machine learning (e.g., SVM) to detect potential attacks.

In the following, I will provide an overview of my research on (1) visibility of performance analysis; (2) a series of stealthy low volume attacks (milli-DDoS attacks); and (3) modeling of n-Tier systems and parameters optimization. I will conclude with a discussion of future research directions.

2 Visibility of Performance Analysis

To get more market and users for websites, providers of web applications try their best to supply rapid responsiveness of their web applications (e.g., e-payment, online-gaming). For example, Amazon reported that an every 100ms increase in the page load is correlated to a decrease in sales by 1% [5]; Google requires 99 percentage of its queries to finish within 500ms [6]. Emerging augmented-reality devices (e.g., Google Glass) need the associated web applications with even greater responsiveness in order to guarantee smooth and natural interactivity.

However, there exists various puzzling performance anomalies that blocks the promotion of emerging techniques, e.g., resource contention in cloud computing. One primary challenge is how to capture performance issues that may hurt the performance of response-time sensitive use-facing applications, e.g., causing long-tail latency problem. Typically, those performance issues are invisible or covert to the providers of web applications. Firstly, performance metrics are too huge, the relationship of various performance metrics are too complicated and undetermined. Secondly, in a large-scale distributed system, log files of all the involved components are distributed in different locations. Therefore, to pinpoint the root-cause of a performance issue typically requires experienced performance engineers. In the following, I highlight some of my work about the analysis approaches of performance bottlenecks and attack detection.

Fine-grained time-line correlation analysis. I developed an automatic log analysis tool [1, 2, 3], which uses fine-grained correlation analysis of performance metrics from infrastructure layer (e.g., TShark, Wireshark, OProfile, Collectl) to application layer (e.g., Apache, Tomcat, MySQL) in time-line graphs. On one hand, this tool can easily plugin any of performance monitoring tools, e.g., hardware performance count from Oprofile, CPU utilization from Collectl, network metrics from Wireshark, response time of Apache, throughput and queue usage from Tomcat and MySQL. On the other hand, this tool can integrate various performance metrics from distributed nodes, and visualize them in a time-series graph. The benefit of time-line graph is that it can supply system administrator with intuitive event correlation of various performance metrics among multi-nodes in distributed systems, pinpoint the root-cause of performance issues more efficiently.

Machine learning based statistical analysis. Except for finding performance bottleneck in time-line graphs, I also use ML techniques to deeply mine potential attacks which may adversarially cause the performance problem [2]. I model the users' behavior of navigating websites, such as, the average and variance of request interval of each IP. Based-on the difference of humans' and bots' navigation behavior, I use ML techniques (e.g., SVM model) to takes into account the features of users' behavior derived from HTTP log data, and classify requesting IP addresses as legitimate or malicious. The benefit of ML-based statistical analysis is that it takes into account the requests as whole not merely in terms of a single page event, prevents intrusion with low false positive and low false negative, and without impact the basic functionality of the systems.

3 milli-DDoS Attacks

Milli-DDoS Attacks originate from the new phenomenon of milli-bottlenecks, also called transient bottlenecks in recent performance analysis of Internet services deployed in Cloud environments [9]. From time to time cloud practitioners have reported that n-tier web applications produce very long response time (VLRT) requests on the order of several seconds, when the system average utilization is only about 50-60%. The VLRT requests themselves do not contain bugs, since the same requests return within tens of milliseconds when no bottleneck exists in the target system. The reason why milli-bottlenecks can turn these normal short requests into VLRT requests is because milli-bottlenecks can cause a large number of requests to queue in the system within a very short time. Due to some system level concurrency constrains (e.g., limited threads) of component servers, additional requests that exceed the concurrency limit of any component server will be dropped, causing TCP retransmissions (minimum time-out is 1 second [7]). The requests encountered TCP

retransmissions become VLRT requests perceived by the end users. To effectively launch milli-DDoS attacks hurting the responsiveness of target systems, I did my research from the following two aspects: external and internal attacks [4].

Distributed System & Security: External Attacks. Due to the ubiquity of milli-bottlenecks (with sub-second duration) and strong dependencies among distributed systems, I assume that the external burst of legitimate HTTP requests can cause erratic fluctuation of resource consumption to be injected into the target system and cause milli-bottlenecks in the weakest node of the whole distributed system, which in turn cause queue overflow and VLRT requests resulting from TCP retransmissions. So I designed a new class of low-volume application layer DDoS attack - Very Short Intermittent DDoS (VSI-DDoS) attacks [1] and Tail attacks [2].

Cloud & Security: Internal Attacks. Cloud elastic model (e.g., Amazon EC2) provides opportunity of auto-scaling computing resources for dynamical user requirements. However, performance interference due to resource contentions among co-located VMs block the promotion of cloud computing. One representative example is the internal memory attack [3]. I exploited two approaches to launch memory attacks: saturating memory bus through memory benchmark tools, and locking memory access through unaligned atomic operations. Further, I exploited this type of memory attacks as the attack bullets to trigger milli-bottlenecks inside the cloud, causing long-tail latency.

Remark on the Stealthiness of milli-DDoS attacks. Unlike the network-layer pulsating attacks which intend to temporarily saturate the bandwidth of network links that connect to the target system, milli-DDoS attacks aim to create very short saturation of the bottleneck resource (usually in CPU or disk I/O) inside the target system, which typically require much less amount of attack traffic to trigger them with a higher level of stealthiness. In additional, milli-bottlenecks typically last tens of milliseconds, from sampling theory, the average resource utilization should be moderate level using coarse-granularity monitoring (e.g., second, minute), thus milli-DDoS attacks can bypass not only the state-of-the-art defense mechanisms but also the elasticity mechanisms of cloud computing, since they are both base-on coarse-granularity monitoring.

4 Systems Modeling & Parameters Optimization

Queueing network theory [8] is commonly used to analyze the performance problems in complex systems, especially for performance sizing and capacity provision. To analyze the performance of distributed systems at scale and under milli-DDoS attacks, I start from queueing network models.

Modeling of Distributed Systems. Firstly, to model Tail Attacks [2] in distributed systems I used a well-tuned queuing network to model n-tier systems, and analyze the sequence of causal events and the impact in the context of milli-DDoS attacks. I proposed two new metrics to quantify the damage and stealthiness of milli-DDoS attacks: damage length during which the new coming requests can be dropped with highly probability, and millibottleneck length during which the bottleneck resources sustain saturation. Secondly, I further model Tail (Response time) Amplification [3] at scale in distributed systems. I analyzed cross-tier queue overflow and tail (response time) amplification under milli-DDoS attacks.

Optimization using Reinforcement Learning and Kalman Filter. Reinforcement learning can provide good sequences of real-time decisions, thus it can be used to continually tune parameters from past measurements and get the optimal parameters in the model. Kalman filter is a classic

feedback control theoretic tool, which also uses historical measurements with statistical noise and measurement inaccuracies, and predicts unknown parameters accurately in the model. To get optimal milli-DDoS attacks parameters, we use these two advanced optimization tools [2] to implement our attack framework, making the attack more effectively and stealthily.

5 Ongoing and Future Direction

My diverse background in computer science and broad experience in the field of system and security with the practical experiments and applied mathematics and statistical learning method has built a solid ground for pursuing my academic career in computer science. In the future, I plan to continue investigating different issues at the intersection of machine learning/deep learning, security, and systems. In the following, I highlight a subset of topics that attract me the most.

Extend Attack Surface of milli-DDoS. My previous work investigated mill-DDoS attacks on web applications. In the future I can extend attack surface of milli-DDoS into different application domains and other emerging techniques (e.g., autonomous vehicles, block chain, AR/VR, Big data Systems, other Internet of Thins (IoT) applications etc). Except for hurting the responsiveness (e.g., long-tail latency), I can hurt the availability of data service in a "Very Short Intermittent" pattern in SaaS cloud platform, such as Dropbox, Google Drive, and Microsoft OneDrive. I expect that milli-DDoS Attacks is a type of advanced persistent threat in the long run, the notable obstacle blocking the rapid development and applications of emerging techniques. Through providing the solution to mitigate milli-DDoS attacks, we can promote these emerging techniques.

Deep Reinforcement Learning. My previous research used reinforcement learning to effectively get optimal parameters of mill-DDoS attacks, achieving attack damage (e.g., long-tail latency) and stealthiness (e.g., average resource usage is moderate). Deep learning enables reinforcement learning to apply in unsolvable issues previously, such as the policies predictions for robots and autonomous vehicles. However, there exists learning performance issues (e.g., optimizing training time and huge number of iterations), accuracy issues (e.g., partial measurement to observe the complete environment) and security issues (e.g., polluted or malicious training data). In the future I would like to research those unsolvable challenges.

References

[1] H. Shan, Q. Wang, and Q. Yan, "Very short intermittent ddos attacks in an unsaturated system," in *Proceedings of the 13th International Conference on Security and Privacy in Communication Systems*. Springer, 2017.

[2] H. Shan, Q. Wang, and C. Pu, "Tail attacks on web applications," in *Proceedings of the 24nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017.

[3] H. Shan, Q. Wang, S. Zhang, and Q. Yan, "Tail Amplification in n-Tier Systems," Submitted to *Proceedings of the 38th IEEE International Conference on Distributed Computing Systems*. IEEE, 2018.

[4] H. Shan, "A Study of Very Short Intermittent DDoS Attacks on the Performance of Web Services in Clouds," *LSU Doctoral Dissertations*. Louisiana State University, 2017.

[5] R. Kohavi and R. Longbotham, "Online experiments: Lessons learned," *IEEE Computer Society*, 2007.

[6] K. Curtis, P. Bodík, M. Armbrust, A. Fox, M. Franklin, M. Jordan, and D. Patterson, *Determining SLO Violations at Compile Time*, 2010.

[7] IETF, *RFC 6298*. "https://tools.ietf.org/search/rfc6298/".

[8] L. Kleinrock, *Queueing systems, volume 2: Computer applications*. John Wiley and Sons, New York, 1976, vol. 66.

[9] Q. Wang, Y. Kanemasa, J. Li, C.-A. Lai, C.-A. Cho, Y. Nomura, and C. Pu, "Lightning in the cloud: A study of very short bottlenecks on n-tier web application performance," in *Proceedings of USENIX Conference on Timely Results in Operating Systems*, 2014.