Key Completion Indicators
Minimizing the Effect of DoS Attacks on Elastic Cloud-based Applications Based on Application-level Markov Chain Checkpoints

George Kousiouris
Dept. Of Electrical and Computer Engineering, National Technical University of Athens, 9 Heroon Polytechnioy Str, 15773, Athens, Greece

Keywords: Cloud Computing, elasticity, DoS,eDoS, Markov Chains, DoS Identification as a Service

Abstract: The problem of DoS attacks has significant effects for any computing system available through the public domain. In the case of Clouds, it becomes even more critical since elasticity policies tied with metrics like Key Performance Indicators (KPIs) can lead a Cloud adopter to significant monetary losses. DoS attacks increase the KPIs, which in turn trigger the elastic increase of resources but without financial benefit for the owner of the cloud-enabled application (Economic Denial of Sustainability). Given the numerous scenarios of DoS attacks and the nature of services computing (in many cases involving legitimate automated traffic requests and bursts), networking mitigation approaches may not be sufficient. In this paper, the concept of Key Completion Indicators (KCIs) is provided and an analysis framework based on a probabilistic approach is proposed that can be applied on the application layer in cloud-deployed applications and elasticity policies, in order to avoid the aforementioned situation. KCIs indicate the level of completeness and provided revenue of the requests made towards a publicly available service and together with the KPIs can lead to a safer result with regard to elasticity. An initial architecture of this DoS Identification as a Service is portrayed.

1 INTRODUCTION

In the current IT world, Cloud computing is considered as a revolutionary way to provide software, platform and computational resources as services (SaaS, PaaS, IaaS) in order to reduce IT costs and capital investments for enterprises. The outsourcing of their applications on these infrastructures leads to much simpler management, increased availability and what is more important, elasticity in the needed resources, especially in the case of IaaS/PaaS. Companies do not have to cater for the worst case scenario regarding the usage of their resources but just toggle the amount of reserved virtual servers in a pay-per-use model, following their dynamic need.

In order to identify when this increase/decrease in resources is needed, Key Performance Indicators are used together with service offerings to enable this like AWS Auto Scaling. An example of a KPI may be the response time of a web server to the external requests made by the clients. If this time is over a specific limit, more virtual machines may be deployed in a load balancing way to compensate for the increase in traffic. Due to the fact that Cloud resources are vast, a service can scale up to thousands of servers. This has an effect of course on the overall cost for the owner of the Cloud-based application, but given that along with it the revenues from the usage increase, they are able to make more profit.

The problem in this process is when an application that is available to external clients through the public domain becomes a target for Denial-of-Service (DoS) or Distributed (DDoS) attacks. In that case, the requests generated towards the server do not generate profit for the owner but due to the elasticity policies they increase the cost of deployment, thus leading them to monetary loss and eventually bankruptcy, also known as Economic Denial of Sustainability (Kumar,2012). Due to the severity of the result, not a single false negative detection may be afforded or we must resort to more hard-core solutions, such as budget-based boundaries on elasticity. However the latter greatly cancels the main benefit of Clouds, seemingly

Kousiouris G..
Key Completion Indicators - Minimizing the Effect of DoS Attacks on Elastic Cloud-based Applications Based on Application-level Markov Chain Checkpoints.
DOI: 10.5220/0004963006220628
In *Proceedings of the 4th International Conference on Cloud Computing and Services Science* (CLOSER-2014), pages 622-628
ISBN: 978-989-758-019-2

infinite resources to meet demand peaks.

In this paper, we present a design approach that has as a major goal to identify a DoS/DDoS attack against Cloud-enabled elastic applications. This identification must be applied above the networking layer, in order to avoid false negatives, enable web service-based automated implementations or different networking setups and must be based on the single most definitive criterion: "**Is the noticed burst of requests generating the anticipated profit**?"

In order to do so, we define **Key Completion Indicators (KCIs)**. KCIs are checkpoints introduced inside a cloud-enabled application and indicate the level of completion of the external user requests in the application value chain. Thus, KCIs can indicate the gradual fulfilment of the path towards revenue/value creation. The main argument is that this progress and eventual revenue creation is the single most definitive criterion to identify whether a set of requests (despite their origins) corresponds to a DoS/DDoS attack or a legitimate traffic burst, above and beyond any metric based on networking aspects or usage patterns. This is discussed in detail in the Related Work section. The framework may be integrated with an application through API calls to an external service for the maintenance of the KCIs state.

Each request, as it passes through the various stages of the process that generates revenue, raises the KCIs, thus indicating that it is a legitimate transaction. The approach is based on the assumption that a legitimate user will advance through the various stages of the application, until the point where he/she will produce revenue for the application owner (e.g. credit card payment). An illegitimate user or automated bot will be restricted to a specific area of the application, without completing the full application lifecycle.

Based on the level of these KCIs and the proportion of seemingly legitimate requests over the overall ones, a decision can be made as to whether the owner should enable the elasticity policies or not. This of course does not alleviate from the pain of a DoS attack but the scalability effect on the cost will be avoided. The remainder of the paper is structured as follows. In Section 2, similar approaches in the related field are presented, while in Section 3 an analysis is made on the concept of the approach. Section 4 presents requirements necessary for the implementation of the approach while Section 5 provides the overall conclusions from this study and intentions for the future.

2 RELATED WORK

Elasticity is one of the main benefits of Cloud computing. (Kranas et al, 2012) indicate the usage of this feature in a service oriented framework manner, to enable applications to harvest the benefits of it. From a networking point of view, numerous works exist for identifying patterns that are indicative of a DoS attack. CPM (Wang,2004) uses statistical and time series analysis at the network protocol level (e.g. SYN flooding attack detection) in order to abstract from application behaviour. (Kumar et al., 2011) uses neural network classifiers in order to filter traffic messages that are identified as DDoS packets. The characteristics that are taken under consideration include network level attributes such as UDP echo packets, number of connections with SYN errors, type of service etc. (Wang et al., 2011) uses a fuzzy logic based system in order to evaluate the infection of domain names and IP addresses. (Ahmed et al., 2010) use an IP-based approach in order to detect suspicious addresses and the change in the traffic arrival rate.

In a different approach, QoSSoD (Mailloux et al., 2008) caches incoming requests at a proxy and valuates each request. Requests are then scheduled for execution based on their perceived cost or threat. Usage patterns are collected over time and provide a baseline to compare current request behaviour against nominal behaviour.

In (Pinzon,2010) a different approach is presented in order to obtain time bounded Case-Based reasoning conclusions for attacks on SOAP-based web services, based on classifiers and SOAP specific rules for determining whether a set of requests can be categorized as malicious. (Yang et al, 2008) investigate a credit model and flow control policy for minimizing effects of DDoS attacks on P2P systems including malicious nodes. The most similar to our work is (Cheng et al, 2003), in which an application level approach is considered that utilizes specific API injection calls in the code in order to check common rules regarding aspects of DoS attack requests. However this approach also needs detailed knowledge on the types of attacks and their specificities.

An extensive survey on EDoS attacks and countermeasures can be found in (Sandar and Senai, 2012). The main problem with networking approaches is the fact that in many cases false positives or negatives may influence the decision process. For example, corporate gateways that mask all their traffic to be seen as one IP may be mistakenly interpreted for DoS attacks, if their

employees make a burst of requests towards an external cloud-based application. Furthermore, usually malicious approaches may take advantage of various aspects of networking protocols and defensive tools may focus on only a part of them. Moreover, authentication mechanisms may not always be relevant to a specific application deployed on a cloud infrastructure or may hinder the usability of an application by legitimate bots in a more automated application design/usage or web service system.

So in essence, what is needed is a holistic approach that will abstract from the networking layer and will focus only on the global aspect of revenue creation.

3 KEY COMPLETION INDICATORS APPROACH

As seen in the Introduction and Related work section, modern elasticity approaches allow for a feedback loop between the application and the IaaS provider or incorporate an internal mechanism in the applications to automatically trigger the elasticity rules based on the performance metrics. However, in a classic DoS attack, this means that the KPIs of the application will deteriorate, thus trigger the enactment of elasticity. Then, new resources will be deployed, in an attempt to mitigate this effect, driving the cost of the service very high and without getting any revenue due to the false requests.

In order to avoid this situation, we propose the Key Completion Indicators metric to be inserted in the application source code. These API calls may be generic and may only indicate the increase of a global set of counters, external to the program, each of which indicates a part in the process. These multi-level counters are increased each time a request passes through a specific part of the process(Figure 1). By having at each time the amount of requests in each stage we can apply macroscopic statistical analysis compared to a known interval of known operation.

The reason for having multiple levels of KCIs is to reduce the reaction interval. If there is only one KCI in the end of the process, then the comparison must be made in the end of the average time it takes a user to finish the entire process. By breaking the sequence of actions into more elementary ones we can have multiple levels of completeness and thus identify potential threats sooner.

An example of such an implementation could be a multi-page e-shop. Users navigate through the pages, e.g. entering keywords, surfing through catalogues, selecting products, adding to carts and eventually paying for the kart contents. Thus during this lifecycle they create revenue for the e-shop, mainly after the payment step is completed.

Due to the fact that this is in essence a probabilistic process, a similar approach must be followed. Not all users will eventually buy products (they might just have a look on available products) and not all users spend the same time accessing the intermediate steps (some may be more cautious or selective with regard to their added products). However, from a macroscopic point of view, the percentage of users being in each step can be averaged and used in a probabilistic approach. Thus we may conclude that the amount of time spent by people in the selection of products is a cumulative distribution that follows a specific pattern as time proceeds.

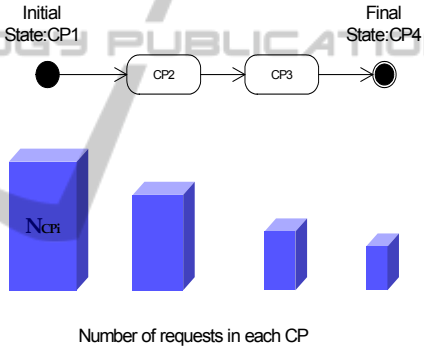

Figure 1: Checkpoints (CPs) through which a request passes until revenue is created in CP4. At each given time, different numbers of requests are in different levels of completion (number of CPs passed).

In order to identify whether a DoS attack is in progress, we can compare the KCI levels of this time interval against the ones from a known normal operation interval. For having the correct coefficients that reflect the normal operation under genuine load, a testing period may be considered, in order to observe application and user behaviour under normal circumstances. By determining the 95% confidence interval of the different KCI levels oscillation, it can be determined during runtime if their observed levels are within the specified intervals.

One problem with this approach is that KCIs will be low also in case of extensive legitimate traffic, due to the low response of the service. We can avoid this false alarm, if we originally enable elasticity for a specific time period (so that the load of the service

is distributed) and then measure the KCI levels. If they do not reach normal levels soon, then the traffic can be considered as illegitimate.

The state diagram for a request across the e-shop appears in Figure 2. A user may navigate through the different pages of the site and these transitions maybe modelled with a probability, as in the case of Markov chains. However, given that this is a stochastic process, the transition probability varies with time, in order to depict different user interactions with the pages (some users may need more than one products, others may be performing other tasks in parallel etc.). Thus, with the use of this method we can determine what would be the probability after a given time for the requests to have reached the intermediate or final stages of the value chain and gradually monitor the anticipated revenue.

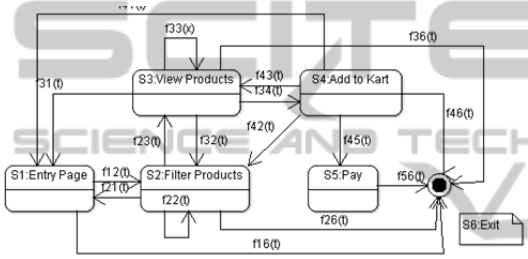

Figure 2: State diagram representing the possible states of a request across the e-shop.

4 APPLICATION AND MODELLING REQUIREMENTS

4.1 Logging Requirements

In order to obtain the necessary data regarding transition statistics, an application should enforce state transition logging. This logging should reflect only the states based on which the Markov model has been designed and it should also support documentation of the source and destination state (e.g. S1->S2, timestamp). The information should be forwarded towards global (to the application threads) counters that will hold the statistics for each transition at any given time.

It is not necessary to have request IDs or similar identifiers since we are not interested in single requests but rather on the macroscopic characteristics in order to detect a potential DoS attack. However it could be helpful to log for example IP addresses in case other countermeasures are jointly applied, in order to reason, isolate and drop packets coming from specific IP addresses that have been identified as malicious through the process. IP address logging could also be helpful during the normal operation to identify a user's path in the e-shop and aid in the modeling analysis.

4.2 Markov Chain Modeling Requirements

In order to create the MC model, we can implement the following process:

- **Step1:** Filter the log files of a known normal operation period in order to extract the probabilities of transitions for each transition case
- **Step2:** Calculate the probability for transitioning from one state to another This would create an array of probability values (Table 1) for the MC analysis that can indicate how many steps are in general necessary (or probable) for a request.
- **Step 3:** Filter the logs of a known normal operation period in order to acquire all the measured timings for a given transition from one state to another
- **Step 4:** Extract the cumulative probability distribution for that given transition and fit a common distribution type (e.g. exponential, normal etc.). This will indicate the probability for a request to go from state A to state B after a specific time interval (Figure 3). By taking the measurements from the normal period of execution we can conclude the time interval after which a transition should have occurred (with a 95% probability).
- **Step 5:** By combining the information from steps 2 and 4, we can macroscopically determine how many requests should have arrived at the final (or even some intermediate) state after a given time interval, based on the initial conditions (e.g. number of pending requests at each stage). This expected value can be compared with the monitored value (potentially incorporating the confidence intervals) in order to conclude whether these requests generate value for the application owner.

It is necessary to stress that steps 1-4, which can be considered in general as time consuming, are needed only in the model preparation phase and not during runtime. Once the model is complete and online, during runtime, only the counters (or KCIs) levels would be needed and probably the calculation of the anticipated value, which is a minor computational task.

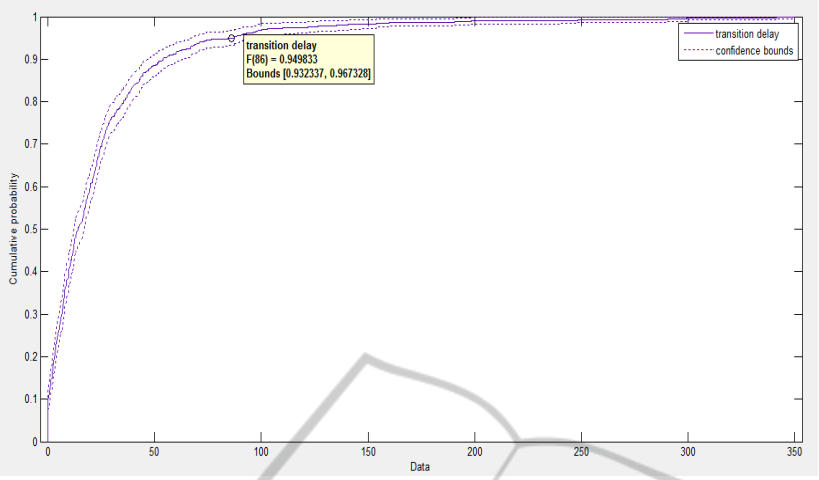

Figure 3: Cumulative Distribution Function of the delay of transition from one given state to another given state (x axis in seconds, y axis the probability for the specific transition to occur within the interval). In the specific example, the 95% probability for this to occur is within 86 seconds.

Table 1: Matrix describing the probability from a transient state to another transient state. From this matrix we can calculate how many steps are probable for a request.

States	S1	S2	S3	S4	S5	S6
S1	0	P_{12}	0	0	0	P_{16}
S2	P_{21}	P_{22}	P_{23}	0	0	P_{26}
S3	P_{31}	P_{32}	P_{33}	P_{34}	0	P_{36}
S4	P_{41}	P_{42}	P_{43}	0	P_{45}	P_{46}
S5	0	0	0	0	0	1
S6	0	0	0	0	0	1

In case one needs to **simplify** this process, then we can reduce the analysis to the absolutely necessary states, which means one state for all cases of surfing through the e-shop and one state that is described as the **"key" decisive state,** the state that generates value. In the aforementioned example the decisive state is S5, in which the user performs the payment. This would however delay the identification process since the interval after which we could check the KCIs level would include all intermediate steps.

4.3 Application Elasticity Logic and Overall Architecture

As mentioned in Chapter 3, the application should enforce elasticity directly in case of a request surge, so that limited resources do not hinder with the progress of the requests and thus contaminate the analysis. For example, if the surge makes the application unresponsive due to scarce resources, valid requests would not be able to reach the end of the process and thus lead to a false positive detection. Thus an elasticity logic should apply the pseudo-code in Figure 4. The sleep interval can be calculated through the model described in Section 4.2 and represent the time after which we would expect requests to have reached the end of the process. The overall system architecture appears in Figure 5.

```
Public class eDoSDetector extends
Thread
.
.
If (elasticity_flag=true)then
    Enable elasticity
    Thread.sleep(interval)
    Check KCIs status
      Run MC model to get anticipated
      KCI values for this interval
      Difference=Predicted-Actual
      If (difference><confidence_limit)
      then
        Disable elasticity
      Else
        Keep elasticity enabled
```

Figure 4: eDoS detector pseudo-code structure.

An interested application will have to inject code at the necessary points, before migrating to the Cloud, that indicate a change of state. That code will be responsible for contacting a web service (DIaaS) to increase the respective KCIs. The probabilistic analyser will have access to the KCI levels and will have created and stored in advance the MC model. Based on the runtime analysis, it may inform the application part (or in a more independent way of communication, the application will query the analyser) that is responsible for enabling elasticity the outcome of the analysis.

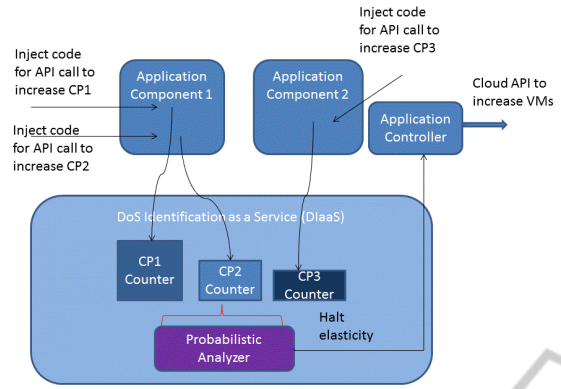

Figure 5: Overall Concept Architecture: the proposed mechanism is offered as a service (DoS Identification as a Service).

5 CONCLUSIONS

DoS attacks on cloud-based applications are more dangerous since they might lead to an unsustainable service and eventual bankruptcy. In order to mitigate this effect and efficiently identifying a DoS attack despite traffic patterns or networking aspects, the Key Completion Indicators concept is proposed in order to model and analyse the transition of requests along the application value chain. The KCIs are accompanied by a probabilistic modelling approach in order to identify whether a surge of traffic is legitimate or not, based only on the generation of wealth by the application. It must be stressed that the approach is for detecting a DoS attack and mitigates only the effect of eDoS, by cutting off the elasticity actions and does not help restore the performance of the server. It is also applicable for DDoS cases, since the analysis is made on the application states transition and not the source of requests.

Given that the approach requires an intervention at the source code level and a similar framework, it can be pursued by PaaS providers as a service offering that ensures the application from such types of attacks. Standard PaaS offering such as Google App Engine already need a certain alteration/adaptation of the source code to the development framework they are using, so it would be easier at this level to support also the concepts presented in this paper. The simplification suggested in Section 4 is also interesting to be investigated in order to minimize the analysis effort but also a potential implementation overhead of the mechanism.

ACKNOWLEDGEMENTS

The research leading to these results is partially supported by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 317859, in the context of the ARTIST Project.

REFERENCES

Naresh Kumar, M.; Sujatha, P.; Kalva, V.; Nagori, R.; Katukojwala, A. K.; Kumar, M., "Mitigating Economic Denial of Sustainability (EDoS) in Cloud Computing Using In-cloud Scrubber Service," *Computational Intelligence and Communication Networks (CICN), 2012 Fourth International Conference on , vol., no., pp.535,539, 3-5 Nov. 2012.*

Haining Wang, Danlu Zhang, Kang G. Shin, "Change-Point Monitoring for the Detection of DoS Attacks," *IEEE Transactions on Dependable and Secure Computing, vol. 1, no. 4, pp. 193-208, Oct.-Dec. 2004.*

P. Arun Raj Kumar, S. Selvakumar, Distributed denial of service attack detection using an ensemble of neural classifier, *Computer Communications, Volume 34, Issue 11, 15 July 2011, Pages 1328-1341, ISSN 0140-3664.*

Kuochen Wang, Chun-Ying Huang, Shang-Jyh Lin, Ying-Dar Lin, A fuzzy pattern-based filtering algorithm for botnet detection, *Computer Networks, Volume 55, Issue 15, 27 October 2011, Pages 3275-3286, ISSN 1389-1286.*

E. Ahmed, G. Mohay, A. Tickle, and S. Bhatia, "Use of ip addresses for high rate flooding attack detection*," in Proceedings of 25th International Information Security Conference (SEC 2010) : Security & Privacy : Silver Linings in the Cloud, Brisbane, Australia.*

M. Mailloux, H. Naim, T. Wayne, "Application Layer and Operating System Collaboration to Improve QoS against DDoS Attack*", Available at: https://wiki.engr.illinois.edu/download/attachments/68780072/QoSSoD.pdf?version=2&modificationDate=1210044601000.*

Pinzón, C., De Paz, J. F., Zato, C., Pérez, J.: Protecting Web Services against DoS Attacks: A Case-Based Reasoning Approach. In: *Graña Romay, M., Corchado, E., Garcia Sebastian, M.T. (eds.) HAIS 2010. LNCS, vol. 6076, pp. 229–236. Springer, Heidelberg (2010).*

Kranas, P.; Anagnostopoulos, V.; Menychtas, A.; Varvarigou, T., "ElaaS: An Innovative Elasticity as a Service Framework for Dynamic Management across the Cloud Stack Layers," *Complex, Intelligent and Software Intensive Systems (CISIS), 2012 Sixth International Conference on* , vol., no., pp.1042,1049, 4-6 July 2012.

Yang J., Li Y., Huang B., Ming J. (2008) Preventing DDoS attacks based on credit model for P2P

streaming system. In*: ATC '08: Proc of the 5th International Conference on Autonomic and Trusted Computing. Springer, Berlin, pp 13–20.*

Sandar S. V., Shenai S. Economic denial of sustainability (EDoS) in cloud services using HTTP and XML based DDoS attacks. *International Journal of Computer Applications 2012;41(20):11–6.*

Cheng Jin, Haining Wang, and Kang G. Shin. 2003. Hop-count filtering: an effective defense against spoofed DDoS traffic. *In Proceedings of the 10th ACM conference on Computer and communications security (CCS '03). ACM, New York, NY, USA, 30-41.*