How to cite this paper:

Adilah Sabtu & Nurulhuda Firdaus Mohd Azmi. (2017). Enhancing security elements for mapreduce processing with whitelist in Zulikha, J. & N. H. Zakaria (Eds.), Proceedings of the 6th International Conference of Computing & Informatics (pp 49-55). Sintok: School of Computing.

ENHANCING SECURITY ELEMENTS FOR MAPREDUCE PROCESSING WITH WHITELIST

Adilah Sabtu[1] and Nurulhuda Firdaus Mohd Azmi[1,2]

[1]*Advanced Informatics School (UTM AIS), Universiti Teknologi Malaysia, Kuala Lumpur*
adilah.sabtu@gmail.com, huda@utm.my
[2]*UTM Big Data Centre, Ibn Sina Institute, Universiti Teknologi Malaysia, Skudai, Johor*

ABSTRACT. Big data requires new ways and technologies of how data is harnessed, managed and applied to create values that offer insights for better decision making. An exploration of MapReduce model which reliably accommodates big data processing requirements reveals that data traversing through nodes inside clusters during processing are exposed to security and privacy breaches. Further examination identifies elements in security task that impact MapReduce challenges. This paper concerns with the experimentation on how Whitelist access control element can enhance security within MapReduce environment using Hadoop platform. Datasets are executed through series of Whitelist coding/scripts. The enhancement is measured on a basis of Whitelist capability and effectiveness of reducing False Positive Rate error in different scenarios, comparing different sizes of applied Whitelists, key strengths used for filtering and the execution time. The results yield reduced False Positive Rate for Whitelist, supporting claim of an enhanced security but the execution time have increased, indicating lower overall performance.

Keywords: Big Data Programming, Hadoop Platform, MapReduce, Data Security, Whitelist

INTRODUCTION

Big data processing is basically characterized by the scalability for large scale data-intensive computing and fault tolerance, iterative refinement, parallel processing, adaptation in diverse environments, support schema-free format and real-time large batch processing (Grolinger *et al.*, 2014). Hence, new emerging system which incorporates programming model like MapReduce capable of tackling aspects of big data is needed. MapReduce programming model allows the processing of massive amount of data in parallel through clustering across a distributed system. This process can be applied to both structured and unstructured data. MapReduce model is composed of basically 2 parts which are Map procedure and Reduce procedure (Grolinger *et al.*, 2014; Garcia, 2013) with amplifying steps between such as Split, Combine and Shuffle and Sort. For MapReduce processing, data are replicated across multiple nodes with high-speed and parallel processing nodes running on very large sets of

data. These big data clusters share almost similar vulnerabilities as web applications and traditional warehouses. Sensitive data uploaded by user must be protected from any unauthorized access to ensure the integrity, authenticity and privacy of the data. It is important that data at rest, data in transit and nodes are managed securely.

This paper is concerned with the enhancement of security task elements for integration into a MapReduce model. It explore the use of Whitelist as a fresh approach that can enhance security in a MapReduce processing environment by filtering False Positive entries and as a surveillance tool for intrusion detection. This paper will explain the development and experimentation of MapReduce processing with Whitelist access control for security element. The discussion in this paper will include analysis of the experiment which is done in Hortonworks Hadoop platform.

SECURITY AND PRIVACY CHALLENGES FOR MAPREDUCE IN BIG DATA

According to Garcia (2013), MapReduce inspired by Google is a functional programming of two structural transformation functions, Map and Reduce, composed together in practice to provide a programming interface for implementing algorithms and to perform wide variety of computations. Xu *et al.* (2014) described MapReduce as a parallel programming model designed based on divide-and-conquer concept and follows a master/slave paradigm. Qin *et al.* (2012) described MapReduce as a general execution engine that ignores storage layouts and data schema, and the runtime system automatically parallelizes computation across a large cluster of machines, handles failure and manages disk and network efficiency. Zhang *et al* (2013) described MapReduce as a scalable and fault-tolerant data processing framework that is capable of processing huge volume of data in parallel with many low-end commodity computers. Fadika *et al*. (2011) described that MapReduce model is anchored around 3 central principles of data management, synchronization/ parallelization abstraction and fault-tolerance. Grolinger *et al.* (2014) stated that the popularity of MapReduce can be accredited to its high scalability, fault-tolerance, simplicity and independence from the programming language or the data storage system. Studies by Grolinger *et al.* (2014) categorized the challenges on MapReduce in Big Data into 4 categories: data storage, data analytics, online processing and security and privacy. The security issue for MapReduce in Big Data are about accountability of having someone responsible for actions performed during big data processing and keeping tracks for auditing. They suggested having the requirement of optimization for any access control approach in reasonable amount of time (Grolinger *et al.*,2014).

Studies by Qin *et al.* (2012) mentioned the challenge of providing guaranteed privacy and security in data intensive computing environments arise especially when running big data analytics in the cloud. In addition, Liu *et al.* (2013) stated that the privacy protection of data output is due to the openness of the internet. Almost online big data applications are used for storage and processing where third parties are involved to host private information and perform computation tasks on these data. Therefore, this information might vulnerable to cyber attackers to intrude and cause harm (Xu et al., 2014).

Whitelist Access Control Mechanism for MapReduce

Whitelist is a 'trust centric' access control security approach comprised of a list of genuine and safe entities or registry that is given authorization and privilege to access an entry. Its basis is that the number of entries allowed into a system is smaller than the number of entries blocked. It is considered relatively easy to maintain and offers administrative control. Alt-

hough whitelist is commonly built for email and web services, the technique can also be used for LAN, program, application and software. The purpose of whitelist is to reduce False Positive Type 1 error. During security filtering, all confirmed legitimate entities registered in the list is granted access into a system. This also prevents accidental exclusion of important entities. Plus, it can potentially alert the system of any illegitimate entities or entirely block them from intrusion. Thus, whitelist can be considered as a first defensive line of a defense in depth for access control mechanisms, whilst providing that aura of concealment (Sabtu et.al, 2015). Further section will explain the development and experimentation of MapReduce processing with Whitelist access control for security element. Later it will discuss about the analysis of the experiment which is done in Hortonworks Hadoop platform.

EXPERIMENTAL SETUP AND IMPLEMENTATION OF WHITELIST

Figure 1 illustrates the Whitelist implementation workflow of the experimentation. The experimentation uses Hadoop architecture in Hortonworks platform which is widely used for big data analytics and has its environment built with MapReduce and Hadoop Distributed File System (HDFS) as its core. For this paper, the experimental setup is based on a single node cluster which involved a single machine. The reason of using the single node is to have a preliminary experiment on whitelist implementation in MapReduce for security element. Later, we plan to further the experiment in real-time environment with multi nodes.

This experimentation uses an available Server Log dataset and Whitelist datasets created to fit the experimental needs; therefore it does not replicate a real-time scenario. In terms of programming model, firstly, pig scripts containing specifics for Whitelisting operations, the Whitelist and Server Log datasets are run into MapReduce jobs on the Apache Hadoop cluster. Then, the experiment results data stored in Hadoop Distributed File System (HDFS) and data from executed scripts are extracted and downloaded for content analysis and measuring the capability and effectiveness of Whitelist implementation. Finally, the captured results will be analyzed to identify the outcome of the experiment.

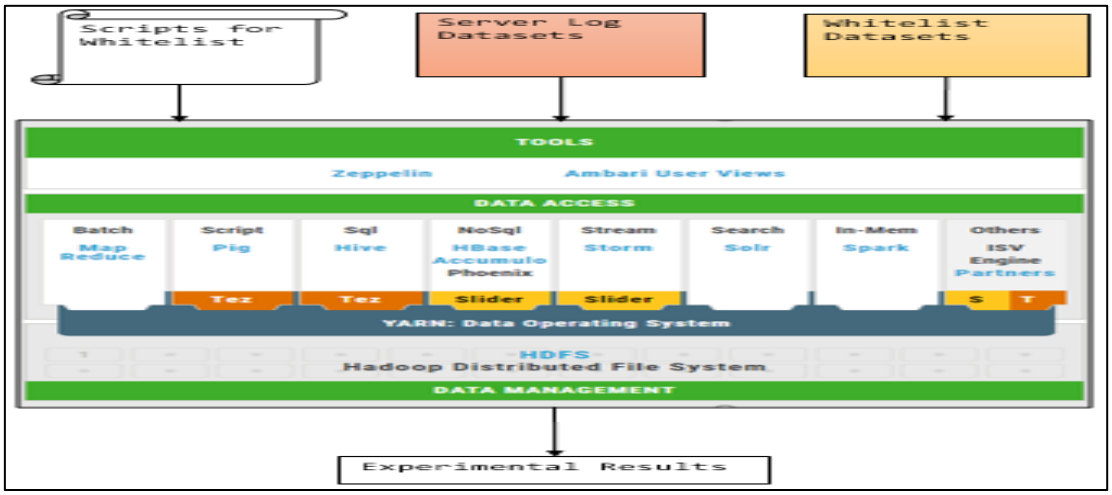

Figure 1: Experimentation Work Flow

For data input, this experimentation will be based from two sets of ready-made Whitelists which will be created. The proposed Whitelists contains two fields which are commonly associated and used to control access: User ID and IP Address. The fields could be any identified security risk elements which intruders use as targets to attack or breach security and force access. For the Whitelist, initially a unique User ID field is assigned as primary key. The assumption is that one user is assigned to only one User ID. For a more stringent measure, IP address can also be assigned as primary key. Note that a user might have more than one IP address assuming the possibility of working from different machines or locations. In one of the experiment scenarios, for the case where IP addresses do not match Whitelist, it would be filtered into unlisted file for system administrator to control, maintain and decide later for Whitelist update.

Pseudo Codes of Whitelist Scripts

The pseudo codes for the Whitelist operations algorithms or scripts are presented in Algorithm 1 until Algorithm 3:

```
LOAD data from Whitelist AS User ID, IP
LOAD data from Server Log AS User ID, IP, Time, Country
JOIN Whitelist and Server Log BY User ID to filter True Negative entries
        GENERATE AS User ID, IP, Time, Country
COGROUP Whitelist and Server Log BY User ID
        FILTER BY Whitelist = null to filter False Positive entries
        GENERATE FLATTEN to un-nest tuple
STORE into WhitelistChecked for True Negative entries
STORE into UnlistedFiltered for False Positive entries
```

Algorithm 1: Pseudo Code for Whitelist Script Using 1 Primary Key/Field – User ID for Filtering

```
LOAD data from Whitelist AS User ID, IP
LOAD data from Server Log AS User ID, IP, Time, Country
FOREACH Whitelist CONCATenate User ID, IP to create Combo Key
FOREACH Server Log CONCATenate User ID, IP to create Combo Key
JOIN new Whitelist and Server Log BY Combo Key to filter True Negative entries
        GENERATE AS User ID, IP, Time, Country
COGROUP Whitelist and Server Log BY Combo Key
        FILTER BY Combo Key = null to filter False Positive entries
        GENERATE FLATTEN to un-nest tuple
STORE into WhitelistChecked for True Negative entries
STORE into UnlistedFiltered for False Positive entries
```

Algorithm 2: Pseudo Code for Whitelist Script Using 2 Primary Keys/Fields – User ID and IP Address for Filtering

```
LOAD data from UnlistedFiltered USING PigStorage
LOAD data from Server Log USING PigStorage
False positive = COUNT UnlistedFiltered number of entries
Total log entries = COUNT Server Log number of entries (True negative+ False positive)
False Positive Rate = False positive / Total log entries
DUMP to execute
```

Algorithm 3: Pseudo Code for Whitelist Script That Computes False Positive Rate

RESULT AND DISCUSSION

False Positive rate is calculated and Filtered entries data taken from different Whitelist scenarios and results are summarized in Table 1 for comparison. The results of False Positive Rate are further visualized in a graph in Figure 2. In a nutshell, the desired result would be low False Positive rates for both Whitelist implementations (10 and 50 Whitelist entries) which imply reduced Type 1 error and can be considered effective security.

Table 1: False Positive Rate and Filtered Entries of Different Whitelist Scenarios

	No Whitelist	Whitelist1		Whitelist2	
		False Positive Rate	Filtered Entries	False Positive Rate	Filtered Entries
1 Key - User ID	1	0.772	386	0.006	3
2 Keys - User ID, IP	1	0.792	396	0.134	67

Whitelist1: 10 recorded entries
Whitelist2: 50 recorded entries
Server Log: 500 recorded entries

From Figure 2 below, it shows that False Positive Rates for Whitelists are less than No Whitelist. The rate for Whitelist1 containing 10 entries yields 0.772 and 0.792 whereas Whitelist2 containing 50 entries produce rates of 0.006 and 0.134. Both of these Whitelists results are less than the baseline value of 1. This clearly indicates that Whitelist implementation delivers a good impact on security enhancement for the system.

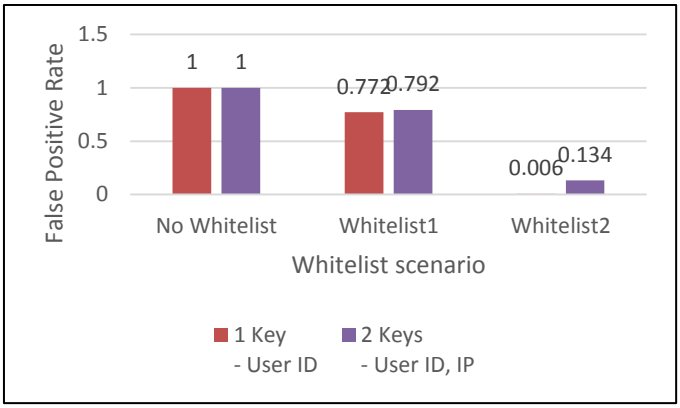

Figure 2: Comparison of False Positive Rate among Different Whitelist Scenarios

CONCLUSION

This paper identifies MapReduce security elements from which enterprises could use as basis of selecting security solutions. The experimentation of Whitelist access control for MapReduce processing may contribute to better security choices for a MapReduce model. The outcome of the study may well trigger interests from MapReduce practitioners, big data community and the data security field. However, there is constant need to continue finding solutions for MapReduce that effectively handle big data security concerns. Most existing models are specifically tailored to structured-based requirements.

ACKNOWLEDGEMENT

The authors would like to thank Advanced Informatics School (AIS), Universiti Teknologi Malaysia (UTM) for the support of the resources. This work is currently funded by GUP Tier 2 from UTM (Vot Number:14H08).

REFERENCES

Fadika, Z., Dede, E., Govindaraju, M. and Ramakrishnan, L. (2011). Benchmarking MapReduce Implementations for Application Usage Scenarios. *12th IEEE/ACM International Conference*. Lyon, pg. 90 – 97.-

Garcia, C. (2013). Demystifying MapReduce. *Procedia Computer Science*. pg. 484 – 489.

Goss, R.G. and Veeramuthu. K. (2013). Heading towards big data building a better data warehouse for more data, more speed, and more users. *24th Annual SEMI*. pg.220–225.

Grolinger et al. (2013). Data management in cloud environments: NoSQL and NewSQL data stores. *Journal of Cloud Computing: Advances, Systems and Applications*. 2(22).

Grolinger, K., Hayes, M., Higashino, W.A., L'Heureux, A., Allison, D.S. and Capretz, (2014). Challenges for MapReduce in Big Data. *IEEE World Congress on Services (SERVICES)*. pg. 182–189.

Hu, H., Wen, Y., Chua, T-S. and Li, X. (2014). Toward Scalable Systems for Big Data Analytics: A Technology Tutorial. *IEEE Access*. 2, pg. 652 – 687.

Kim, Y-G., Lee, M., Cho, S. and Cha, S. (2012). A Quantitative Approach to Estimate a Website Security Risk Using Whitelist. *Security Comm. Networks.* 5, 1181 - 1192.

Liu, Z., Yang, P. and Zhang, L. (2013). A Sketch of Big Data Technologies. *2013 Seventh International Conference on Internet Computing for Engineering and Science (ICICSE)*. 26–29.

Lu, T., Guo, X., Xu, B., Zhao, L., Peng, Y. and Yan, H. (2013). Next Big Thing in Big Data: The Security of the ICT Supply Chain. *2013 International Conference on Social Computing (SocialCom).*1066 – 1073.

Pandey, S. and Tokekar, V. (2014). Prominence of MapReduce in Big Data Processing. *2014 Fourth International Conference on Communication Systems and Network Technologies (CSNT).* April 2014. 555 – 560.

Sabtu, A., Mohd Azmi, N.F. and Yuhaniz, S.S. (2015). Enhancing Security and Privacy Protection for MapReduce Processing: The Initial Simulation Work Flow. *Int. J. Advance Soft Compu. Appl.* 7(3), 72 – 84.

Zhang, X., Liu, C., Nepal, S., Yang, C., Dou, W. and Chen, J. (2013). Combining Top-Down and Bottom-Up: Scalable Sub-Tree Anonymization over Big Data Using MapReduce on Cloud. *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom).*501 – 508.