Design and Implementation of Network Intrusion Detection System on the Apache Hadoop Platform

Vladimir Ciric, Dusan Cvetkovic, Ivan Milentijevic

Abstract — **Growing influence of computer networks and Internet to everyday life, with more and more devices connected to global network, opens a new possibilities for malicious activities, while exposing the users to attacks even more, including their data and privacy. Due to the amount of data that need to be processed in order to detect such activities, intrusion detection and prevention systems become a challenging topic. The goal of this paper is the implementation of Intrusion Detection System (IDS) on the Apache Hadoop platform. The Hadoop implementation will enable task parallelization on multi-core processors. The proposed system will be evaluated and compared with popular Snort IDS on a two-core i3 processor. The obtained results show that proposed Hadoop based IDS is about 25% faster then the Snort IDS.**

Index Terms — **Network Security, IDS Systems, Hadoop MapReduce, Myers algorithm.**

I. Introduction

In the era of Internet of Things (IoT), where the goal, in a nutshell, is to connect every device and sensor that can be connected to the global network and build an application around them, the security of the devices and data that they produce becomes one of the key requirements [1]. According to The Statistical Portal “Statista”, the number of connected devices installed worldwide in 2017 is about 23 billion, and it is expected that the number will reach 75 billion in 2025 [2].

Unfortunately, software security testing is a commonly misunderstood and underestimated task [3]. Dynamic market often put tough timing requirements and deadlines that influence the development to reuse untested or poorly tested libraries and components, leaving potential doors open for attacks. Even the “well tested” libraries are not a guaranty that the exploit will not be found in the future [4]. In order to protect the system against threats in such an environment, a robust security architecture should be built.

Legacy security architectures usually were limited to firewall as a key component that can permit or deny the traffic to or from specific host or protocol [5]. However, todays exploits and attacks are far more sophisticated, and require deep network packet analysis. Examples include various distributed DoS attacks, zombie networks, etc.

Deep packet analysis can be performed by Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS), which are commonly used components in the security architectures today [6]. An intrusion detection system (IDS) is a device or software application on network boundary that transparently monitors network traffic for malicious activity or policy violations. Any malicious activity detected by IDS is typically reported to the network administrator and recorded into log files. The IPS, in addition to IDS, is proactive and acts to prevent the malicious activity, for example, by adjusting the firewall rules.

Taking into the account the growing number of devices and typical bandwidths on the network boundaries, the amount of data that need to be analyzed for malicious signatures becomes challenging. Several researchers proposed IDS implementations with aim to speed up network packet analysis [7-12]. Different approaches to task and data parallelism were exploited [9,10]. In [11] authors proposed GPU accelerated implementation of IDS. Some of the implementations aim to accelerate the pattern matching operation through parallelization using Phoenix++ and MAPCG MapReduce frameworks for multi-core CPUs [12].

The goal of this paper is design and implementation of IDS system using Apache Hadoop MapReduce framework. The Apache Hadoop is a framework for development of distributed applications, which offers both task parallelization on multi-core processors, and distributed application execution. The IDS will be implemented using Myers pattern-search algorithm as a core for signature-based packet analysis. The design of MapReduce IDS workflow will be described in details. The proposed system will be evaluated and compared with popular Snort IDS on a two-core i3 processor. The obtained results will show that proposed Hadoop based IDS is about 25% faster then the Snort IDS.

The paper is organized as follows. Section 2 gives a brief introduction to IDS and Myers algorithm. Section 3 is devoted to the MapReduce framework, as a basis for the proposed Apache Hadoop implementation of IDS. Section 4 is the main section and presents the design of the IDS workflow on the Hadoop framework. Section 5 is devoted to the system evaluation, while in Section 6 concluding remarks are given.

Vladimir Ćirić is with the University of Niš, Faculty of Electronic Engineering, Aleksandra Medvedeva 14, 18000 Niš, Serbia (e-mail: Vladimir.ciric@ elfak.ni.ac.rs).

Dusan Cvetković is with the University of Niš, Faculty of Electronic Engineering, Aleksandra Medvedeva 14, 18000 Niš, Serbia (e-mail: cvetkovicdusan@outlook.com).

Ivan Milentijević is with the University of Niš, Faculty of Electronic Engineering, Aleksandra Medvedeva 14, 18000 Niš, Serbia (e-mail: ivan.milentijevic@ elfak.ni.ac.rs).

II. Classification of IDS and the role of Myers algorithm

IDSs are classified, based on their network packet analysis model, into two categories: pattern (or signature) matching and anomaly detection [6,12,13]. The signature matching IDS monitors the network activity for a known misuse pattern that was previously identified as a malicious attempt [7,8]. The anomaly-detection IDS makes the decision based on a profile of a normal network behavior. The network "baseline" is often constructed using statistical or machine learning techniques [13,14].

Signature matching IDSs utilize a database with malicious signatures that are prepared in advance. This leads to fast and reliable pattern matching operation, commonly used in the majority of commercial systems [14]. However, anomaly-detection based IDSs are able to detect new attacks that have not been seen before. The drawback of this category of IDSs is the occurrence of false positives.

Snort is a widely used open-source signature matching IDS [12]. It has a large and publically available database of rules, which covers known attacks, and it grows with each discovered attack. Many commercial and experimental systems use the snort rule syntax, due to its flexibility, and ease of new rules creation. Fig. 1 shows the basic elements of a Snort's rule.

```
alert tcp any any -> 160.99.14.0/24 25 (content: “mail from:
root”; msg: “root attempts to send an email”; sid:12345)
```

Fig. 1. Example of Snort rule: left side of the rule contains network related parameters; the right side of the rule in the brackets contains rule's parameters such as pattern, alert message, etc.

Any signature based IDS checks the presence of a malicious signature in the incoming packet sequence and act as instructed by the corresponding rule. The pattern matching algorithm must be fast enough in order to meet the increase in both the number of signatures and the link speed. Signature based IDS systems, publically available or experimental, usually differ in the pattern search algorithm, on one hand, and the implementation technology, on the other hand [8,9,10,11,12]. Since the release of version 2.0 in 2002, Snort has utilized a high-speed multi-pattern search engine [15]. Before Snort 2.8.0, the default string-pattern matching algorithm was ACF, whose speed of packet processing was faster than AC-BNFA that is currently used, but it consumed more memory [15].

In this paper we explore Myers pattern search algorithm with rules in Snort syntax, as a core for the proposed IDS. The Myers algorithm is an approximate string matching algorithm, which uses the Levenshtein distance to compute the matches [12]. The algorithm matches a large text *t* of length *n* with a short pattern *p* of length *m* allowing up to *k* differences, where *k* is a chosen threshold error. The example of the approximate string matching is shown in Fig. 2. The matrix in Fig. 2 is formed in three steps:

1) Fill the text *t* into the first row, and the pattern *p* into the first column. The matrix will be of the order *n* x *m*.
2) Fill the second row with zeros, and the second column with numbers 1,2,3,…,*m*.
3) Calculate all the values *v* in the matrix using the formula:

$$v_{i,j} = \min\begin{cases} v_{i-1,j} + 1 \\ v_{i,j-1} + 1 \\ v_{i-1,j-1} + \delta_{i,j} \end{cases},$$

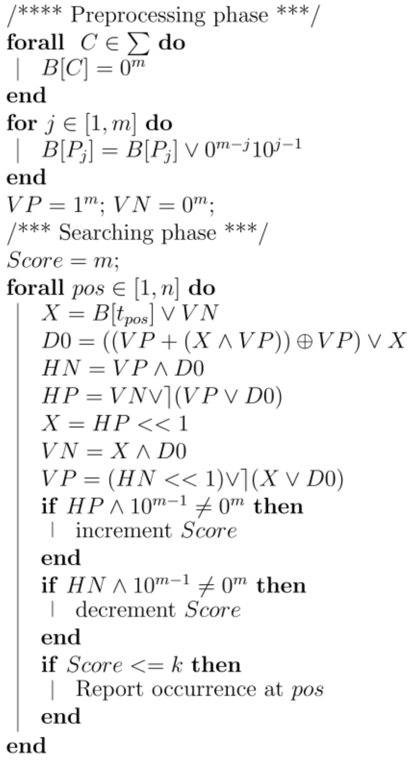
where $\delta_{i,j} = 0$ if the characters in *i*-th row and *j*-th column are the same, and $\delta_{i,j} = 1$ otherwise.

	I	**C**	**E**	**T**	**R**	**A**	**N**	**P**	**A**	**L**	**I**	**C**	**T**	**R**	**A**	**N**
	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
E	**1**	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
T	**2**	2	1	0	1	2	2	2	2	2	2	2	1	2	2	2
R	**3**	3	2	1	0	1	2	3	3	3	3	3	2	1	2	3
A	**4**	4	3	2	1	0	1	2	3	4	4	4	3	2	1	2
N	**5**	5	4	3	2	1	0	1	2	3	4	5	4	3	2	1

Fig. 2. Example of approximate string matching

The distance of the search pattern from the particular positions in the text can be seen in the last row. As the example in Fig. 2 shows, there is one exact match (value 0). However, there are few positions with the distance 1, etc.

The Myers algorithm encodes the columns by integer numbers, and calculates the *i*-th column in iterative manner using the value of the *i*-1 –th column and integer arithmetic. The Myers algorithm is given in Fig. 3 [12].



```
/**** Preprocessing phase ***/
forall $C \in \sum$ do
    $B[C] = 0^m$
end
for $j \in [1, m]$ do
    $B[P_j] = B[P_j] \vee 0^{m-j}10^{j-1}$
end
$VP = 1^m; VN = 0^m;$
/*** Searching phase ***/
$Score = m;$
forall $pos \in [1, n]$ do
    $X = B[t_{pos}] \vee VN$
    $D0 = ((VP + (X \wedge VP)) \oplus VP) \vee X$
    $HN = VP \wedge D0$
    $HP = VN \vee \rceil (VP \vee D0)$
    $X = HP << 1$
    $VN = X \wedge D0$
    $VP = (HN << 1) \vee \rceil (X \vee D0)$
    if $HP \wedge 10^{m-1} \neq 0^m$ then
        increment $Score$
    end
    if $HN \wedge 10^{m-1} \neq 0^m$ then
        decrement $Score$
    end
    if $Score <= k$ then
        Report occurrence at $pos$
    end
end
```

Fig. 3. Myers pattern searching algorithm

As it is shown in Fig. 3, the Myers algorithm has a preprocessing phase, followed by the search phase. In the preprocessing phase it generates a bitmasks B, which are used later on in the pattern search phase. One bitmap is generated for each character *C* from the alphabet (text and pattern). The generated bitmap *B* is equal to 0 if the character *C* is not in the search pattern (Fig. 3).

III. Hadoop MapReduce model

In this section we give a brief introduction to the Apache Hadoop platform and MapReduce model, as a basis for IDS implementation.

The Apache Hadoop is a framework for distributed processing of large data sets on clusters of computers using MapReduce programming model, where each computer, or so called "node" offers local computation and storage [16]. There are two main components of Hadoop system: Hadoop Distributed File System (HDFS), used for distributed data storage, and MapReduce computing framework for data manipulation. The HDFS is a layer above existing file system of every node in cluster, and Hadoop uses its blocks to store input files or parts of them. Large files are split into a group of smaller parts called blocks (default block size is 64MB) [16]. Typical Hadoop execution has 4 parts: transferring input data from Client host to HDFS, processing data using MapReduce framework on the slave nodes, storing results by Master node on HDFS, and reading data by Client host from HDFS.

In essence, MapReduce technique consists of two transformations that can be applied many times on input files: Map transformation, and Reduce transformation. During the Map transformation, every Map task processes a small part of the input file (input split) and passes the results to the Reduce tasks (Fig. 4). After that, during the Reduce transformation, Reduce tasks collect the intermediate results of Map tasks and combine them in order to get the output.

During the execution of the Mapper, the Mapper calls a Map function, which performs required computations. Precisely, Map function transforms input dataset into the set of output values (*key*, *value*). After that, intermediate data with the same key are grouped and passed to the same Reduce function. At the end, Reduce function summarizes all data with the same key in order to get the final result (Fig. 4).

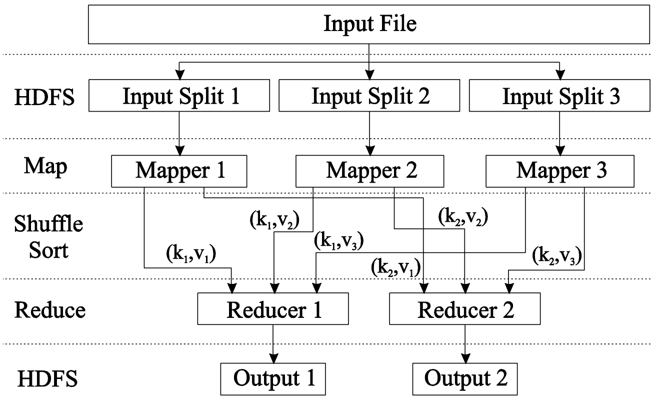


Fig. 4. The MapReduce execution

IV. Design of MapReduce suitable IDS

The workflow of the proposed IDS is shown in Fig. 5. The workflow has three phases for network traffic fetching and data format preparation (phases 1, 2 and 3 in Fig. 5), and one central phase "pattern search" (phase 4 in Fig. 5). The framework is designed in such manner to encapsulate the pattern search into the phase that can be parallelized using *Hadoop* framework from Fig. 4, and to prepare and provide the data in required format for the central phase.

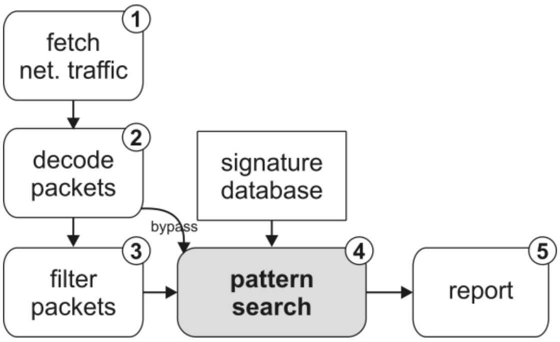
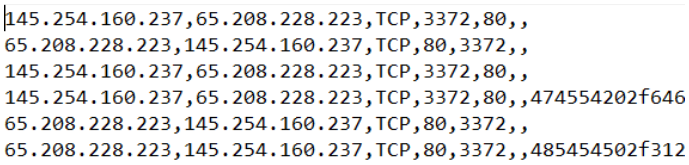


Fig. 5. The workflow of the proposed IDS

The intrusion detection starts with network packet fetching (phase 1 in Fig. 5). For fetching the network packets we use *libpcap* library. Other tools like Wireshark can be used, too. The output of this phase is binary data collected from the network in *raw* format. In order to simplify manipulation, the *raw* data are decoded, and the most significant information, such as IP addresses, ports and content, are extracted and stored as strings. For this purpose we use *tshark* Linux command line tool. The example of *tshark* tool usage is:

```
tshark -r <pcapfilename> -T fields -E separator=, -e ip.addr -e _ws.col.Protocol -e tcp.port -e udp.port -e data > output.txt
```

A part of the output of the *tshark* tool is shown in Fig. 6. Each line in Fig. 6 contains the information from one fetched network packet.



```
145.254.160.237,65.208.228.223,TCP,3372,80,,
65.208.228.223,145.254.160.237,TCP,80,3372,,
145.254.160.237,65.208.228.223,TCP,3372,80,,
145.254.160.237,65.208.228.223,TCP,3372,80,,474554202f646
65.208.228.223,145.254.160.237,TCP,80,3372,,
65.208.228.223,145.254.160.237,TCP,80,3372,,485454502f312
```

Fig. 6. The output of the *tshark* tool

The output can be filtered in order to remove the traffic which is not of the interest for IDS, such as local traffic, etc. (phase 3 in Fig. 5). Filtered or not, the output from Fig. 6 is well prepared for the pattern search phase (phase 4 in Fig. 5).

The pattern search is performed on the Apache Hadoop. The input data from Fig. 6 is divided in input splits and fed into Hadoop Map tasks, which are executed in parallel. Each mapper from Fig. 4 executes the Myers algorithm from Fig. 3 on its input split. There are no data dependencies, thus the mappers can be executed in parallel.

It should be mentioned that the preprocessing phase of Myers algorithm is performed only once for all Snort rules, and the bitmaps are stored for the further use.

A mapper from Fig. 4 reads one line of input data at the time, i.e. one network packet, and executes the Myers algorithm. If a Snort rule pattern is found, the mapper emits <*key*,*value*> pair, where the *key* stands for a network flow attack identification, while the *value* is constant 1. The *key* is in the format

<*sid*>,<*SourceIP*>,<*SourcePort*>,<*DestIP*>,<*DestPort*>,

where *sid* is Snort rule ID, and the rest of the fields are the identifications of the network flow.

Having the same key, the results from the same malicious flow go to the same reducer (Fig. 4), which counts the malicious packets on the flow and outputs the result.

V. Implementation and evaluation results

The proposed IDS is evaluated using 1GB of fetch network data containing 2.966.346 packets. The data is obtained by merging the data from the network forensics site *asecuritysite.com* and the data obtained in our lab environment. The complete set of input data contained 400 malicious packets of the following types: Heartbleed attack (sid=100000), three types of WonnaCry attacks (sid = 2024217, sid = 2024218, sid=2024220), and two malicious flows with the same FTP brute force attack (sid=491) [4]. The rules are obtained from the network forensics site *asecuritysite.com*, too.

The evaluation is performed on single processor PC with two-core i3 6006U CPU and 8GB of RAM. The results are given in Table 1. The both systems were evaluated with the same input data set. We used Apache Hadoop 2.9.0 and Snort 2.9.7.0 GRE (build 149). The file sizes in Table 1 differ due to the fact that the Snort uses the *raw* data in the binary format as an input, while the proposed system first decodes the packets into a text format.

TABLE I
Evaluation results

IDS	The input data		Processing time
	Size in GB	# of malicious packets	
The Snort	1	400	2 min
The proposed Hadoop IDS	1.6	400	1 min 34 sec

From Table 1, it can be seen that the proposed Hadoop IDS is faster than the Snort for about 25%. The gain in the speed obtain by involving both processor cores in this case is compensated by the usage of robust framework and by data preparation.

VI. Conclusion

In this paper the implementation of Intrusion Detection System (IDS) on Apache Hadoop platform is proposed. The design and the implementation of the proposed IDS are given in details. Using the Hadoop platform we utilized parallelization on multi-core processors and speedup the system. The proposed system is evaluated and compared with popular Snort IDS on a two-core i3 processor. The evaluation results show that the proposed Hadoop IDS is about 25% faster than Snort IDS.

Acknowledgment

The research was supported in part by the Serbian Ministry of Education, Science and Technological Development (Project TR32012).

References

[1] Li Da Xu, Wu He, and Shancang Li, “Internet of things in industries: A survey. IEEE Transactions on industrial informatics, 10(4), 2014, pp.2233-2243.

[2] “Internet of Things connected devices installed base worldwide”, The Statistics Portal, URL: www.statista.com/statistics/471264/, Accessed in 2018.

[3] Potter, Bruce, and Gary McGraw. "Software security testing." IEEE Security & Privacy 2.5 (2004): 81-85.

[4] Tsoutsos, Nektarios Georgios, and Michail Maniatakos. "Trust No One: Thwarting" heartbleed" Attacks Using Privacy-Preserving Computation." VLSI (ISVLSI), 2014 IEEE Computer Society Annual Symposium on. IEEE, 2014.

[5] Hunt, Ray. "Internet/Intranet firewall security—policy, architecture and transaction services." Computer Communications 21.13 (1998): 1107-1123.

[6] Endorf, Carl, Eugene Schultz, and Jim Mellander. Intrusion detection & prevention. Emeryville, CA: McGraw-Hill/Osborne, 2004.

[7] Aldwairi, Monther, and Duaa Alansari. "Exscind: Fast pattern matching for intrusion detection using exclusion and inclusion filters." Next Generation Web Services Practices (NWeSP), 2011 7th International Conference on. IEEE, 2011.

[8] Xu, Dongliang, Hongli Zhang, and Yujian Fan. "The GPU-based high-performance pattern-matching algorithm for intrusion detection." Journal of computational information systems 9.10 (2013): 3791-3800.

[9] Kharbutli, Mazen, Monther Aldwairi, and Abdullah Mughrabi. "Function and data parallelization of Wu-Manber pattern matching for intrusion detection systems." Network Protocols and Algorithms 4.3 (2012): 46-61.

[10] Su, Xiong, Zhenzhou Ji, and Xiaoyang Lian. "A Parallel AC Algorithm Based on SPMD for Intrusion Detection System." Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering. Atlantis Press, 2013.

[11] Vasiliadis, Giorgos, et al. "Gnort: High performance network intrusion detection using graphics processors." International Workshop on Recent Advances in Intrusion Detection. Springer, Berlin, Heidelberg, 2008.

[12] Aldwairi, Monther, Ansam M. Abu-Dalo, and Moath Jarrah. "Pattern matching of signature-based IDS using Myers algorithm under MapReduce framework." EURASIP Journal on Information Security 2017.1 (2017): 9.

[13] Jeong, Hae-Duck J., et al. "Anomaly teletraffic intrusion detection systems on hadoop-based platforms: A survey of some problems and solutions." Network-Based Information Systems (NBiS), 2012 15th International Conference on. IEEE, 2012.

[14] Aljarah, Ibrahim, and Simone A. Ludwig. "Mapreduce intrusion detection system based on a particle swarm optimization clustering algorithm." Evolutionary Computation (CEC), 2013 IEEE Congress on. IEEE, 2013.

[15] Yoshioka, Atsushi, Shariful Hasan Shaikot, and Min Sik Kim. "Rule hashing for efficient packet classification in network intrusion detection." Computer Communications and Networks, 2008. ICCCN'08. Proceedings of 17th International Conference on. IEEE, 2008.

[16] Lam, Chuck. Hadoop in action. Manning Publications Co., 2010.