Developing a malware analysis system on distributed environment

Khoi Nguyen Tan

Abstract—Malware is one of the major threats on the Internet today. To protect from the rapid propagation of malware in the network, we need to focus on determining how malware can be analyzed, detected, and blocked analyze and detect. As distributed processing model have been recently developed due to the cloud computing platform and the cluster filesystem, they could be usefully applied to analyzing malware. In this paper, we propose a malware analysis method based on the MapReduce software framework of the distributed processing platform. The proposed solution allows to reduce the time of analyzing and identifying malware. The experimental results show that the MapReduce-based flow analysis method improves the performance when analyzing a large number of malware.

Index Terms—Malware analysis, mapreduce, hadoop, sandbox, distributed processing.

1. Introduction

NOWADAYS, malware is spreading increasingly on the internet, not only due to the increased complexity of the malware itself but also due to the exponential increase of new malware each day [1]. The process of detecting and treatment malware is very complex [4]. There are two major technologies to analyze of malware: the old signature-based methodology and the more advanced method of detecting malware via behavior analysis. The solution currently being addressed is using sandbox and distributed computing to analyze malware. Basically, sandbox is an important technique to create the environment for malware to show all the features while ensuring the safety of outside system.

A number of approaches have been proposed to detect malware on network. Traditional anti-virus tools use static signatures to match patterns, but the effectiveness of syntactic signatures is thwarted by the use of sophisticated techniques that often hide the exploit code contained in malicious files. The method by which malware analysis is performed typically falls under one of two types: static malware analysis and dynamic malware analysis [4][5][Fig 1].

Many approaches and tools have been proposed in recent years to identify and analyze malware [1][5][7]. Most of sandbox systems provided in the Internet, such as Joe Sandbox, Threat Expert, CW Sandbox only allows users to inject one malicious code into the analysis system. Besides, other Sandbox systems also contain their own limitations, represented as Cuckoo Sandbox, Buster Sandbox and Zero Wine Sandbox [7][12]. Buster Sandbox is a closed-source software and the customization of scripts would not be supported [9] [7]. In Zero Wine Sandbox, malware could detect that they are being analyzed with very high probability. Besides, the file size to input on Zero Wine Sandbox is smal, so the analysis of different file types is limited [9] [8]. The Cuckoo system supplies tools to analyze of malware in the sandbox environment, but the process is done sequentially, thus, the efficiency is not high [10].

MapReduce is a processing model designed for treatment large volumes of data in parallel by dividing the work into a set of independent tasks. Therefore, the time to deal with entire data is going to be decreasing [3]. MapReduce model consist of two separate routines, namely Map-function and Reduce-function. The Map-function is responsible for processing one or more chunks of data and producing the output results. The Reduce-function is responsible for consolidating the results produced by each of the Map-functions [8]. Hadoop is a distributed computing framework designed to process very large unstructured data sets. The top level unit of work in MapReduce is a job. The job configuration supplies map and reduce analysis functions and the Hadoop framework provides the scheduling, distribution, and parallelization services. The Fig.1 shows the logical flow of a MapReduce model.

This paper presents the direction of research, design and development of malware analysis and processing system in distributed processing environment using MapReduce model. Our aim is to develop a distributed
malware testing environment by using Sandbox that was used to test an extensive number of malware samples and trace their behavior. The proposed solution helps to detect early and prevent sensitive information from malware. Therefore, this approach completely avoids the impact of malware on the real system.

The main contribution of this study as follow:

- A novel solution that has the ability to detect behaviors of malware by using Map Reduce model and sandbox,
- An analysis technique that automatically produces the report the malware behavior,
- An application system for malware analysis.

The rest of the paper is organized as follows: Section 2 describes some related work; the details of the proposed approach are presented in Section 3; the experimental results are shown in Section 4 and Section 5 gives the conclusion and discussion.

2. Proposed Approach

2.1. Malware Analysis System

We implemented our proposed approach in a system and we used it to detect and analyze malware content. In this section, we describe some of the analyses that our system can perform on malware base on MapReduce model (Fig 2).

![Fig. 2: Logical data flow of MapReduce model](image)

Since the number of malware increases more and more, the manual malware analysis method is not possible due to the limitation of experts and time. To overcome this problem, it is necessary to build a system that automatically analyzes the behavior of malware to further support manual analysis method. Concurrently, this system could quickly offer a warning of potential damage by malware. Fig 3 illustrates the overview of malware analysis system model.

![Fig. 3: OpenStack Cloud and Hadoop integration and architecture](image)

![Fig. 4: Modules of malware analysis system](image)

![Fig. 5: Mapper process at node in Hadoop cluster](image)

Malware analysis system contains a master machine and multiple slaves. This system work on hadoop file system and allows us to choose different file types and can allocate for analysis. We will now describe the process of malware analysis as follows:

1) Users can send a list of potential malware files to master node and requests to analyze them.
2) The master will choose available slaves and schedule them.
3) After that, master will submit each block of malware to slave nodes.
4) Slave nodes try to execute Map and Reduce process in order. Firstly, slave nodes load malwares from HDFS and call $Map()$ function to analyze. Secondly, based on results from $Map()$ function, the Reduce process will continue to gather the data in slaves and return the results to master node for collection.
5) The analysis result is the list of files is infected or not, description of malware behavior. Statistic of malware are analyzed.

Each slave node has one or more virtual machines that are called a sandbox. The purpose of installing a virtual machine is to create an isolation environment for testing malware after downloading malware from master. These sandbox are planned to be able to run automatically (self-startup, automatic clean recovery, self-copying of files for parsing, returning results to slaves, automatic) that no need for intervention from manager. With the snapshot feature, it will help to restore the environment, analyze the configuration in the sandbox quickly or to select different analysis environments to match the malicious code analysis files.
For the process of Map() function in the cluster, slave nodes will accept input as a ⟨key, value⟩ pair where key is the name of the malware, value is its address. Based on this address, local machines will load malware to analyze. The below code show the mechanism of Map() function:

```java
public void map(LongWritable key, Text value, OutputCollector ⟨Text, Text⟩ output, Reporter reporter) throws IOException {
    //TODO Auto-generated method stub
    Configuration conf = new Configuration();
    conf.addResource(new Path(CORE_SITE_PATH));

    // copy file from HDFS to Local
    filename = value.toString();
    FileSystem fs = FileSystem.get(conf);
    fs.copyToLocalFile(new Path(COPY_DIR, filename), new Path(LOCAL_DATA_DIR));

    // Run script which implements the static analysis
    runScript(filename);

    // Run dynamic analysis
    boolean succeed = runDynamicAnalysis(filename);
}
```

After that, the static analysis will be run. Next, malware will be copied to the sandbox to perform dynamic analysis. In sandbox, malware is executed and recorded all behavior. The Fig 4 show the data processing at each slave node.

Thus, there are three main stages at slave node:
- Download malware to do the map from HDFS,
- Perform static analysis,
- Copy malicious code to the virtual machine, execute dynamic analysis.

The analysis process results are output to the collection file. They are going to split, one part is the output for the Map process, another one is saved in HDFS for statistics.

For the process of Reduce() function, slaves will perform the Reduce job after completing the Map task. The import for Reduce function is a pair of (key, value), where key is the malware state (NOT OK, OK, N/A) and value is the malware name. Reduce ones will collect all malware with the same status into each group.

The mechanism of Reduce() function is described as follow:

```java
public void reduce(Text key, Iterator⟨Text⟩ values, OutputCollector ⟨Text, Text⟩ output, Reporter reporter) throws IOException {
    // TODO Auto-generated method stub
    StringBuilder sb = new StringBuilder();
    while(values.hasNext()){
        Text text = values.next();
        sb.append(text);
        if(values.hasNext()) sb.append(", ");
    }
    output.collect(key, new Text(sb.toString()));
}
```

The malware analysis is performed in a sandbox which avoids affecting real systems. In our deployment system, the outside operating system is Ubuntu and the inside environment is Windows XP/Windows 7 which runs malicious codes. The VIX API library enables to write scripts that automate to shutdown virtual machines, transfer files between real/virtual machines. VMware’s snapshot feature restores quickly the previous marked data/configuration, which means that we could select the environment to perform the suitable analysis. This is very useful for malware analysis.

3. Experimental results and Evaluation

In this section, we present some experimental results to prove effective of the proposed solution. All examples are computed on computers running Ubuntu 14.04, Hadoop platform: 1.2.1. We use the malware database include 300 malware sample from [12][13][14]. The MapReduce model which is combined with our system will operate automatically. All analysis steps both automate to run from the process of receiving malicious codes to copy them into the virtual machine, executing them and finally giving a detailed report on malicious behavior without intervention.

In the graphical user interface of program showed in Fig 5, we choose files which contain potential mal-
ware then do the process analysis. This procedure is performed with testing samples which stores 50, 100, 150, 200, 250 and 300 malicious codes accordingly.

The results provide information which imply malware name, MD5 value, malware status (NOT OK, OK, N/A), quantities detected by Antivirus and statistic of analyzed malware amount, number of undetected malware, error counts in the analysis (Fig 6). Fig 7 show dynamic analysis results on system.

TABLE 1: The time to analyze malware in one machine

<table>
<thead>
<tr>
<th>No. Malware</th>
<th>Analysis Time (sec)</th>
<th>Result</th>
<th>Ratio(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>60</td>
<td>44</td>
<td>88.00</td>
</tr>
<tr>
<td>100</td>
<td>113</td>
<td>89</td>
<td>89.00</td>
</tr>
<tr>
<td>150</td>
<td>150</td>
<td>134</td>
<td>89.33</td>
</tr>
<tr>
<td>200</td>
<td>185</td>
<td>173</td>
<td>86.50</td>
</tr>
<tr>
<td>250</td>
<td>222</td>
<td>219</td>
<td>87.6</td>
</tr>
<tr>
<td>300</td>
<td>251</td>
<td>258</td>
<td>86.00</td>
</tr>
</tbody>
</table>

The malware analysis and processing system will provide the convenience of securing the network information. Utilizing the capacity of servers in idle time by setting a schedule that contributes to enhance productivity in the initiative, improving the technique, and significantly shortening the time to analyze malicious code.

If the number of malware are 50 then the analysis time is 60 seconds, thus, the average time to analyze each malware is 1.2 seconds. If the number of malware is 300 then the time is 251 seconds, the average analysis time is only 0.84 seconds (30% reduction on each malicious code).

Therefore, by using distributed processing model with MapReduce, we could decrease the analysis time total and overcome the weaknesses of existing models like as Threat Expert, CW Sandbox. Thus, the performance is improved while reducing the spent time to analyze the large amount of malware.

4. Conclusion and Discussion

In this paper, we have proposed an approach to build the distributed processing model for malware analysis. The application has been implemented in an cloud system, which was evaluated on a large data. This application system helps the analysis and processing of malware is done quickly. The distributed sandbox system enables safety in malware analysis while decreasing the time spent on analysis as well as raising productivity.

In our approach, data will be analyzed in a safe environment and distributed work is done independently and simultaneously. The results of the evaluation show that it is possible to reliably detect malware by using emulation to exercise the behavior of the code. The future work will extend the techniques described here to improve the detection of malware and improve the procedures to identify binary shellcode used in malware.

References


Khoi Nguyen Tan received the B.S. degree in Information Technology from the DaNang University of Technology, Vietnam, in 1997, the M.Sc. degree in Computer Science in 2003, and the Ph.D. degree in Computer Science from the Aix-Marseille University in 2010. He now is lecturer at University of Science and Technology, Danang, Vietnam. His current research interests include Geometric modeling, Distributed processing.