HardnBot
Intelligent Server Hardening Software

Project ID: 19_20-J 01

Final Report

G.G.L Anjula IT16022416

Bachelor of Science (Hons) in Cyber Security

Information Systems Engineering Department

Sri Lanka Institute of Information Technology

Sri Lanka

May 2020

HardnBot
Intelligent Server Hardening Software

Project ID: 19_20-J 01

Final Report

G.G.L Anjula IT16022416

Supervisor Mr. Amila Nuwan Senarathne

Bachelor of Science (Hons) in Cyber Security

Information Systems Engineering Department

Sri Lanka Institute of Information Technology

Sri Lanka

May 2020

DECLARATION

I declare that this is my own work and this Final Report does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text. Also, I hereby grant to Sri Lanka Institute of Information Technology, the nonexclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

………………………

G.G.L Anjula (IT16022416)

The above candidate is carrying out research for the undergraduate Dissertation under my supervision.

Supervisor

………………………………….

Mr. Amila Nuwan Senarathne

Table of Contents

LIST OF FIGURES

ABSTRACT

Modern world is trending to perform automation tasks in field of information technology. Simply it provides more comfortable platform in day today life. Applying the security compliances to the servers considering inconsistencies regarding the server security is one of the major challenging task in cyber security. As this is a long process, using one person a single server it takes nearly 8 hours to complete, because, server hardening will deliver more security in proportion to the effort that we put into it. If we can reduce considerable amount of time of the server hardening process while maintaining the security that we expect to have in our servers, it is an added advantage to all of the security engineers.

When considering about the whole server hardening process, most of the projects take large amount of time to collect the backup, as the enormous memory size of the servers. Due to the misconfigurations of this server hardening process, sometimes administrators have to reset the servers after the hardening. By collecting that kind of problems and trying to solve with efficient and smart answers for them should not be a next generation task.

In this report, it describes two methods named Automatic Backup and Intelligence Rollback which are related to server backup and detect misconfiguration issues. In any kind of situation, it is a best practice to have a backup of our system in case of occurrences of any unexpected incidents. Also detecting the misconfigurations of the servers is challenging. Using these two techniques server administrators can reduce the effort that needs to put to have more secure environment.

1. INTRODUCTION

1.1 Introduction

In present-day hackers have the ability to take down an entire network using a single loophole of a server without a matter of second. It could be a weak password which may allow the attacker to launch a brute force attack and take control of the server and then the control of the entire network. Basically, this is the hacker's behavior and it could be surveying variation from the best-case scenario to the worst-case scenario. When considering servers, before experience such kind of situation organizations/server owners tend to prevent those attacks using server hardening. The server hardening is a security process that securing a server by reducing its surface of loopholes/vulnerabilities [1]. In technical terms, in order to serve a most secure/reliable service to their clients, servers should have proper compliance with acceptable policies/compliances. As an example, let's assume a server with open ports which have active running services on them and some of these ports and services are actually unnecessary services for the server. Attackers can easily use those loopholes for launching attacks. Server hardening comes up with a better solution for this kind of situations.

Server hardening has some long, complicated and separated processes to follow forget the better results. Project HardnBot is an approach to collect those process into one central application which has some extra features as well. However, the major goal of this HardnBot project is to take down the overall hardening time, hardening with less user interaction while offering some sort of new features.

This Final Report is created to give a clear idea about the final output and the process of development of the research project "Intelligent Server Hardening Software (HardnBot)" respect to each part of the project to the interest parties.

1.2 Background & Literature Survey

When considering the manual server hardening approach server custodian/owner has responsibilities to manage information level risk. Therefore, accomplish that requirement he/she has to take a backup of the server before it is going to harden. Not only that but also server custodian/owner must ensure that server is working properly after the hardening process, if not that could be an operation level issue and he/she needs to fix it immediately. Let's assume he/she couldn't able to bring server to its normal working behavior, then the only answer is rollback the server to its previous stable state using backup which is previously taken. In the manual approach, all of these tasks take a considerable amount of time and active user interaction as well.

Intelligent Server Hardening Software (HardnBot) is a software that has the capability to automatically detect poor/non-compliant configurations in Linux servers which has CentOS 7 as an operating system and applying industry recommended fixes for them.

Under this final report, the following research components are described:

- Automated backup function
- Intelligence rollback function
- SSH connection to servers

Software is divided into fourteen main parts and all of them are important to the final outcome of this project. This report covers three parts of that. Through this document, these components will be described with validation.

Faraz Shakib, an employee of a Calsoft Private Limited, India and Zoheb Shivani of Pune Institute of Computer Technology, Pune published a research paper in 2006 on "*Snapshot service Interface (SSI), a generic snapshot assisted backup framework for Linux".* In there they were talking about an online backup solution called *"SSI"* for the Linux platform while talking quantitative measure on performance hits incurred due to using SSI framework in lieu of using traditional backup approaches. Mainly SSI is aimed at enterprise-level high-end critical servers (database servers and mail servers) which are providing 24x7 availability. Because that kind of servers doesn't have any downtime periods, especially when taking the backup and maintenance.

In SSI there are 9 major steps covering the whole process of taking a system backup.

1) The SSI initiates from requesting SSI based snapshot-assisted backup by the backup application.
2) SSI asks all the registered business applications to get consistent and waits until all of them respond since they need to maintain an availability of their services. Timeout mechanism is there to prevent unwanted SSI waiting.
3) Make business application's consistent by flushing their buffers.
4) Until the snapshot operation is performed, the application is paused and another timeout mechanism is also available within the business applications to prevent wait forever in case SSI fails to respond.
5) Write to disk is over and consistent file system is available for use.
6) The snapshot provider is then requested to take the snapshot since the file system is consistence. Then snapshot is take as a volume level snapshot and consistent backup can be extract from it.
7) The status of the snapshot (succeed or not) returns by the snapshot provider.
8) SSI in turn the status to the backup application.
9) In this step all the business applications are signaled to continue when the file system writes are resumed.

The major benefit of this SSI framework is that business applications can continuously write data while their snapshot assisted backup is taken [2]

*Since the HardnBot is not an online-based tool, the above approach deviates from our process of taking a backup automatically to the external block device. In the other hand, this method is most suitable for critical servers.
Alireza Tajary and Hamid R. Zarandi of Department of Computer Engineering and Information Technology, Amirkabir University of Technology, Teheran, Iran published a

research paper in 2016 on "An Efficient Soft Error Detection in Multicore Processors Running Server Applications". According to this research, there are three categories for tolerating transient faults in server processors named "redundant execution", "anomaly detection" and "dynamic verification". When considering the redundant execution-based method, it has a high potential to detect any fault, but it reduces the provided throughput of the processor significantly. The impact of the Anomaly detection-based methods on processor performance is very low, but it takes more time to detects faults. When comes to the dynamic verification method it has overhead on each core of the processor negatively and implies dedicated hardware to dynamically verify the execution of instructions.

The proposed method of this research consists of three modules,

1. Configuration manager module
2. Speculative faults detection module
3. Redundant execution-based faults module

Configuration manager module is capable of handling the new threads by changing the configuration of cores. Perform this task it has a table which stores the state of each core. There are four states 1) Free core, 2) Non-coupled busy core, 3) Coupled with right core, 4) Coupled with the left core. Before handle, a new thread, status of the relevant core or cores should be checked.

Enables fault detection mode for each thread which can be run in redundant execution mode is responsible for speculative fault detection module and for that it stores the requests from L1 cache to the L2 cache. By considering the stored value of the L1 cache the proposed solution utilizes a speculation to detect the occurrence of the fault.

The redundant execution-based faults module contains one switch for each core and one comparator module between two adjacent cores. The above switch is used to select the data path between the L1 cache and pipeline. These switches can be configured by the configuration manager module to enable or disable coupling operation.

Using this overall throughput aware redundant execution-based soft error detection method the configuration manager determines the configuration in the arrival of a new thread. If it has not enough available resources one core will be assigned to the above-mentioned thread. Then speculative faults detection method is activated to detect faults. If it has enough available resources it uses redundant execution to run the thread. This proposed method results variate from 70% to 100% with respect to resource availability for redundant execution and speculative detection methods [3].

Olumuyiwa Ibidunmoye, Ewnetu Bayuh Lakew and Erik Elmroth of Department of Computing Science Umeå University, Sweden published a research paper in 2017 on "A Black-box Approach for Detecting Systems Anomalies in Virtualized Environments". In their research, they have been proposed a black-box framework which has an unsupervised prediction-based technique to detect anomalies automatically. This mechanism uses multidimensional resource behavior of datacenter nodes for detecting abnormal behaviors. After that, anomalous nodes across the data center which are related to abnormal behaviors, are ranking using a graph-theoretic technique.

Mainly the system is based on usage of computer, memory, disk IO and network resources. That information collected by distributed lightweight monitoring agents deployed in each node (VMs and PMs) of the virtual infrastructure and sampled at a fixed time interval. The Node Anomaly Detection (NAD) is an independent agent which is deployed in an individual node. The main purpose of this module is adaptively detecting time-points where the node faces abnormal resource usage. It doesn't matter whether it is temporary or extended period of time. To fulfil this target NAD uses three steps. 1) estimates a continuous temporal profile of normal behavior using historical data, 2) predict expected behavior in the immediate future using the profile estimate, 3) Calculate the difference between the forecast residual of the expectation and the baseline in multidimensional space. Then NAD triggers an alarm when an anomaly is detected.

The Global Anomaly Ranking (GAR) module is for cluster-wide observability and its objective is collecting all alarms which are triggering from NAD around distributed nodes at a regular time and rank them based on spatial dependencies or how one anomaly in one node is the effect on other anomalies which are observed within the same period of time. This can facilitate root cause analysis and anomaly mitigation procedures against a large number of alarms generated from NAD [4].

Yogendra Kumar Jain and Sandip S. Patil of Samrat Ashok Tech. Institute, Vidisha, India published another research paper in 2009 on "Design and Implementation of Anomalies Detection System Using IP Gray Space Analysis". They have come up with a methodology which has three steps to detect external anomalous host with their scanning behaviors using IP gray space analysis and to scanning foreign port used by the external anomalous host. IP gray space is a collection of unassigned IP addresses in a considering network. It means IPs which are not assigned to any active host.

Step 01: Identification of anomalous external host using IP gray space and relative uncertainty. In this step, they have been maintained an IP threshold range which includes all the active IPs in it. That kind of threshold setting is called as association rule generation for supervised learning. If the source IP address is coming from this active IP space the host related to that IP is considered as a normal user. If the IP is from gray space, then it should be an anomalous host. Active space can define by our own concerns, ex (192.168.54.1 to 192.168.54.254).

Step 02: Identification of category of the anomaly using dominant scanning port (DSP). In this step its identifying 5 main categories of anomalies using their dominant scanning port (DSP). DSP is the foreign port and port service used by scanning flaws SF(h) of anomalous host for communication with an internal host.

Types of anomalies detected using DSI:

1. Bad Scanner-I: ICMP probes, TCP/UDP scanning activities, searching for well-known ports
2. Bad Scanner-II: TCP/UDP probes on a variety of ports, ICMP ping, TCP connection request on port 113
3. Bad Scanner-III: TCP/UDP probes, receive responses from some live inside hosts, TCP port 80 scanning, performing queries to an inside DNS server, sequential scanning on TCP port 445
4. Focused Hitters: Belongs to a small number of applications like SMTO, web and some targets the web services ports like 80 and 443
5. Mixed Intruders Anomaly: A new category, having hybrid behavior of normal and abnormal behaviors. Required checking Bad scanners and Focused hitters first.

Step 03: Determination of potential behavior of each anomaly using flaws ratio [5].

Ming-Jen Chen, Chia-Chun Shih, Chien-Huei Yang, Gene Hong and Yuan-Sun Chu of Department of Electrical Engineering, National Chung-Cheng University, Chia-Yi, Taiwan have published a research paper in 2011 on “Multi-Layered Monitoring Framework with Indices Relationship Analysis for Distributed Service Performance Evaluation”. As they mention a monitoring framework enables system administrators to identify and exclude abnormal system performance behaviors. There are two monitoring frameworks called centralized monitoring framework and distributed monitoring framework. The proposed multilayer monitoring process provides in-depth performance analysis. This is done by three main processors named cross-layered performance, indices relationship analysis and performance state inference. Using this three in one method administrator can reduce personal cost and increase the reaction speed of anomaly performance.

Cross-layered performance indices relationship analysis

In this approach, the relationship between the unit performance index (UPI) and abstract performance index (API) is identified. Apart from that, it helps administrators to derive APEP according to FPEP. The process of analysis has three phases, initial phase, learning phase, and inference phase. When it comes to the initial phase it’s an administrator responsibility to address the performance issues of system services and define the unit and abstract performance indices (UPIs and APIs), define detection rules of anomaly performance behavior. In the learning phase relationship analysis of the cross-layered performance, indices are processed. Results of the initial phase and the learning phase can be used to pre prediction of new performance abnormal behaviors during the inference phase [6].

Zhe Wang, Jin Zeng, Tao Lv, Bin Shi and Bo Li published a research paper in 2016 on “*A Remote Backup Approach for Virtual Machine Images”.* In there they were talking about virtual backup on cloud storage. When we are considering cloud computing, virtualization is playing a major role, because of hosting several applications and services in virtual machines (VM) which were hosted in cloud environments. Security becomes a prior requirement in virtualized applications. In this research, the main focused area is high availability issue in virtual machines. LiveRB (Live remote backup) is the proposed remote backup approach. The purpose of the Live RB is to save the running state of the VM in an online manner known as "Live Migration". This backup process will happen the background of the hosted cloud applications of the VM and is transparent to them. A virtual block device will be designed and will be used to cache I/O Operations in memory, in order to save the incremental virtual disk data.

LiveRB will be implemented on KVM virtualization platform in order to evaluate effectiveness and efficiency using a set of comprehensive experiments. These experiments are all related to Cloud Computing and the security issues that come along with this and the key points considered in order to have successful cloud computing are security, availability & fault tolerance. The commonly used solution to handle Fault Tolerance & High Availability is using snapshots or checkpoints that periodically record the states of the software for backup and rollback the cloud applications to the previous backup upstate. This procedure will be carried out when encountering Failures or Errors of the original system.

Most currently existing VMs stop the VM to take snapshots. Some VMs need to be shut down to take snapshots which affects the ability to provide the service/ result in abnormal cloud application behavior. Some VMs suspend the current process and save the current progress onto local disks to be transferred onto remote servers later which sometimes result in data loss if a hardware failure is encountered.

The above issues can be resolved using Live RB since it works by not stopping the VM to do the backup process. Results of this process indicate that Live RB can be used on a VM to do the backup task from VM onto a Remote Server with only a slight reduction in performance [7].

In this research, it described a method that used to back up a virtual machine, but when it comes to our research area we have to consider the live server. Therefore, no need of care about any virtual machines, but when we are talking about the remote backup approach used in here, that was Live Remote Back up, so we can consider about this technique when we are dealing with our problem.

L. Farinetti and P. L. Montessoro published a research paper in 1993 which named "*An Adaptive Technique for Dynamic Rollback in Concurrent Event-Driven Fault Simulation*". In here it is discussing automatic rollback based on an adaptive mechanism which is including advanced network/system status recording system. Time can be any time, that means before changing of a system or after changing a system this status recording can be applied. The main feature of this research is the user can define the rate for maximum acceptable level for rollback. This approach takes the average time to a minimum level, that means a very short time of the rollback process.
To come up with the proposed technique, researchers were used existing methods such as incremental backups, journal files, checkpoints, rollback, roll forward which were found on different applications, different operating systems as well as different databases. Mainly the status of the network/system is recorded on the disk and run for the negative time period to analyze the previous status. If needed user can run for a positive time period as well.

Those time periods are for comparison with the current status of the network/system. To make it happens above approaches need some fine tunes as well [8].
According to rollback techniques used by those researchers, we identified some techniques and requirement that should be in our system too. For this part of the software, it is necessary to detect abnormal behaviors of the server after the hardening process is done. For that, we need to record system status after the hardening process. Then compare with the previous status, that means system status before the hardening process, but in our approach, there are pre-defined models for comparison with current system status. Apart from that, the rollback mechanism is going to adopt from this research. What are the things.

Ning Lu and Yongmin Zhao published a research paper in 2018 which named "*Research and Implementation of Data Storage Backup*". In this research, researchers were tried to discuss features of a reliable and secure backup and types of backup. With the use of applications which were dependent on big data, the usage of data storage backups became more important. Therefore, the methods used to backup should be more flexible and can be able to ensure of security and reliability of backup contents and also backup and restore should be in a convenient manner. There are several backup methods such as data backup, system backup, application backup etc. The backup contents are guaranteed to be confidential, complete and effective.

There are several specific performances in a backup,

i) Backup should be upgradable, capacity expansion
ii) Management without affecting other application in the system
iii) Implement a backup storage system combining SAN (storage area networks) and NAS (network attached storage) storage networks.
iv) Provide several backup methods such as data backup, system backup, application backup,
v) Backup contents should be secure and restore operations should be done in a convenient manner.

I. System Backup

Refers to the backup of the end-point operating system, server operating system and other systems. In here core files and system's registry are backed up as a data. In a matter of system crash or operation mistaken the backup can be restored to the previous state.

II. Virtual Tape Library

Virtual tape library (VTL) considered as a world's leading modern technology to create a backup system. It can rapidly backup and rapidly recover a system that we want to backup. The main feature is no manual intervention of this technology. VTL storage media is a SATA disk and its data transfer rate is 150MS/s. That means approximately it takes 10 seconds for transferring 1.5GB data to the backup storage [9].

In our research, one of the main goals is to reduce the overall hardening time. For achieving that task, we should have to minimize the overall backup time to some acceptable level using speed backup mechanism. In this point, we are going to use the technique which is described in this research known as a virtual tape library. If we can adopt this mechanism overall hardening time will reduce averagely by 8 hours to 4 hours.

Teruaki Sakata, Teppei Hirotsu, Hiromichi Yamada and Takeshi Kataoka published a research paper in 2007 which named "*A Cost-effective Dependable Microcontroller Architecture with Instruction-level Rollback for Soft Error Recovery*". This tool is developed to detect soft errors using electronic design automation (EDU) which generates optimized soft error detecting logic circuits for flip-flops. When a soft error is detected that signal goes to a developed rollback control module (RCM). That RCM will reset the CPU and restores the CPU's register file from a backup register file using a rollback program guidance. After that CPU will able to restart from the state which is before the soft error occurred. In here researchers were developed another two modules called error reset module (ERM) that can restore the RCM from soft errors and error correction module (ECM) that corrects errors in RAM after error detection with no delay overhead.
In above mentioned soft error means, which are random transient errors. Those errors are the main cause of failures in microcontrollers which include reversal of a memory element's bit data due to factors such as alpha rays in a package, neutron strike and noise of the environment [10].

D. R. Avresky and M. I. Marinov published a research paper in 2011 which named "*Machine Learning Techniques for Predicting Web Server Anomalies*". The basic idea between servers on the web is to provide requests made by the client through the web using different transmission methods such as Services. Businesses relying on these services require the web servers to have reliability, availability and security in order to provide constant quality in the service provided. This document describes the quality ensured in these services.

The assumption made for this problem is mainly due to Resource Starvation. Resource Starvation is when a process that functions in Concurrent Computing is unendingly denied the necessary resource to continue & process the rest of its work. Resource Starvation is measured by the response time taken to cater requests under artificial workloads while collecting data on other resource parameters. The research provides proof that these recordings gathered from different artificial workloads can be applied to real world entities as well

Machine Learning is used to monitor & correlate the high response time and this is done by observing the system data. The goal of this analysis is to resolve issues of this variety in Web Servers, Operating Systems or in VM (virtual machine) Rejuvenation.

Based on the statistics provided by the Internet World Statistics, we could clearly notice a rapid rise QoS (quality of service) Internet Service Usage users and this gave several companies & industries to exist in the current world. The below listed out Companies/

Industries who gets affected by these figures since their prime business is offering Internet QoS,

- Cloud Computing
- Data Storage
- Hosting Providers
- Content Delivery
- Application Performance Management & other

Due to this high demand and dependence on network QoS, it is important for a particular service to be aware of its own deteriorating quality. Currently there are several self-monitoring network products that ensure that the QoS of services offered through the internet. The goal of this this research is to increase this area.

The benefits taken from this research can be applied to other areas as well and they have been listed down below,

- Proactive Software Rejuvenation
- Web Server Workload Balancing
- Web Server Performance Testing

  Other… [11].

In this research mainly focused about detecting anomalies on a web server using machine learning technique. Hence, we are not going to use machine learning techniques for detecting anomalies in a server this research is not a to good feed for us.

1.3 Research Gap

Server Hardening is one of the most important tasks to be handled on servers. Server hardening, which is also referred to as operating system hardening, is the process of making the server stronger and more resistant to security issues. Server hardening is an inexpensive and simple task to improve the overall operating system for maximum performance and to reduce expensive failures. Hardening is a Process requires many steps, all of which are critical to the success of the hardening system. The more steps a user follows, the safer and more resilient the system will be.

Normally these hardening processes will be done by either network administrators, system administrators, outsourced professionals or server custodians by manually running scripts, commands and queries against the server and it will roughly take more than six hours to completely harden a single server in the infrastructure. Probability of a misconfiguration/abnormal behavior occurrence is higher because hardening will carry out with human interaction. Scenarios where a misconfiguration or abnormal behavior occurs, it may be hard to detect those issues since some issues cannot be identified via an error message. So, in a scenario like that, system administrators, server custodian or network administrators need to go back to the initial state of the server operating system via a backup image which will be a time-consuming task. In the real world, there is no specific software or a tool which has concerned about all the difficulties met when a server is going to harden. Especially when we are talking about the abnormal behavior/misconfiguration detection, it is very hard to detect such kind of things and also very time-consuming approach after the hardening process. Maybe it will take 2 to 3 days' administrator to detect those things.

2. RESEARCH PROBLEM

As mentioned in earlier, in a manual server hardening approach, backup a server which is going to be hardened is much time consuming and a lot of user interaction needed process. We focused on that problem and come up with a solution called "Automated Backup" which takes backup of the server automatically and less time consuming than the manual process.

Instead of wasting time and manpower for detecting abnormal behaviors/misconfigurations, in this research, it gives a proper solution by introducing the main function called "Intelligence Rollback". By introducing the above function for improving the rollback task, rollback full backup image to the initial state will not be necessary up to a certain extent. Rather than detecting abnormal behaviors and server misconfigurations by the server administrator spending more time, the software itself can automatically identify the misconfigured spot or abnormal behavior and rollback only to that point. If those misconfigurations or abnormal behaviors couldn't reverse user has an option to roll back the whole server to its previous stable version using the backup which it already has been taken.

3. RESEARCH OBJECTIVES

3.1 Main Objective

The main objective of this research is to establish a standard and automated platform for server hardening which is more efficient and user friendly. This research is mainly focused on CentOS 7 and RHEL 7 servers which are using most commonly in the industry.

3.1 Specific Objectives

3.1.1 Develop an automated backup function

Provide the capability of automatically backup the server which is going to harden. The ultimate goal is to create an automated backup function which will save more time than the manual backup process.

3.1.2 Develop an intelligence rollback function

Create a function for rollback the server when there are server misconfigurations or abnormal behaviors. Those two parameters supposed to detect automatically and after that user will be able to reset that server misconfigurations or abnormal behaviors through the function. If not, he/she can roll back the server again through the intelligence rollback function.

4. METHODOLOGY

4.1 Automatic Backup

In the manual process, backup is taken to either an external hard drive or mirror server and it is a time-consuming process and needs more human interaction to complete the task. When it comes to the HardnBot we have developed an automated solution for this task. The whole process is done by a shell script named “Backup_Script.sh” which includes all the instructions to collect the backup.

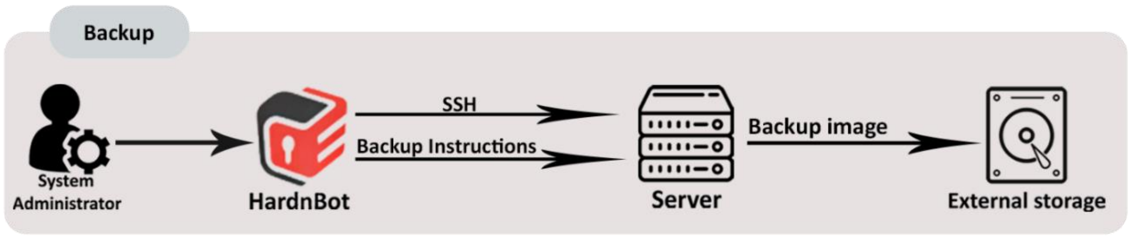

Figure 1 System diagram of automated backup function

When the administrator needs to take a backup, simply what he/she needs to do is connect an external storage device which has enough free space to hold the backup image and select that block device ID (since Unix platform) from the HardnBot backup dashboard. Once the administrator selects that block device ID he/she can launch the backup process by click on the “Backup” button. That’s the big picture of the front end scenario.

A shell script which is used to perform this task contains three main sections.

Initially, it is creating a mount point for mount the block device which is going to store the backup image. In Linux platform, especially when we are talking about the Linux servers, before using any external storage device, it is must create an appropriate file system for the external storage device and mount it with a specific server location.

```
#!/bin/bash

        DATE=`date +%a-%d-%b-%Y-%I:%M:%S-%p-%Z`

        SERVER=`uname -n`

        mkdir /media
        mkdir /media/usb
        mkfs.ext4 -F /dev/$1
```

This is the first half of the “Backup_Script.sh” script and initially in this half, it will collect the date of the backup and the server name for the future reporting purposes. Then it creates two new directories called “media” and “usb” which is inside the media directory for mounting purpose. File system creation is done by “mkfs” command which is short term of make file system. In our approach we used “ext4” file system type because it is more efficient than the other file system see Fig. 1,2; and “-F” for forcefully make the file system without any user interaction. “$1” will be taken as the first command line argument when the “Backup_Script.sh” beginning to run. In here $1 indicates the block device which has been chosen by the administrator earlier. Altogether from last line, it will make the ext4 file system forcefully for the block device which has been chosen by the administrator.

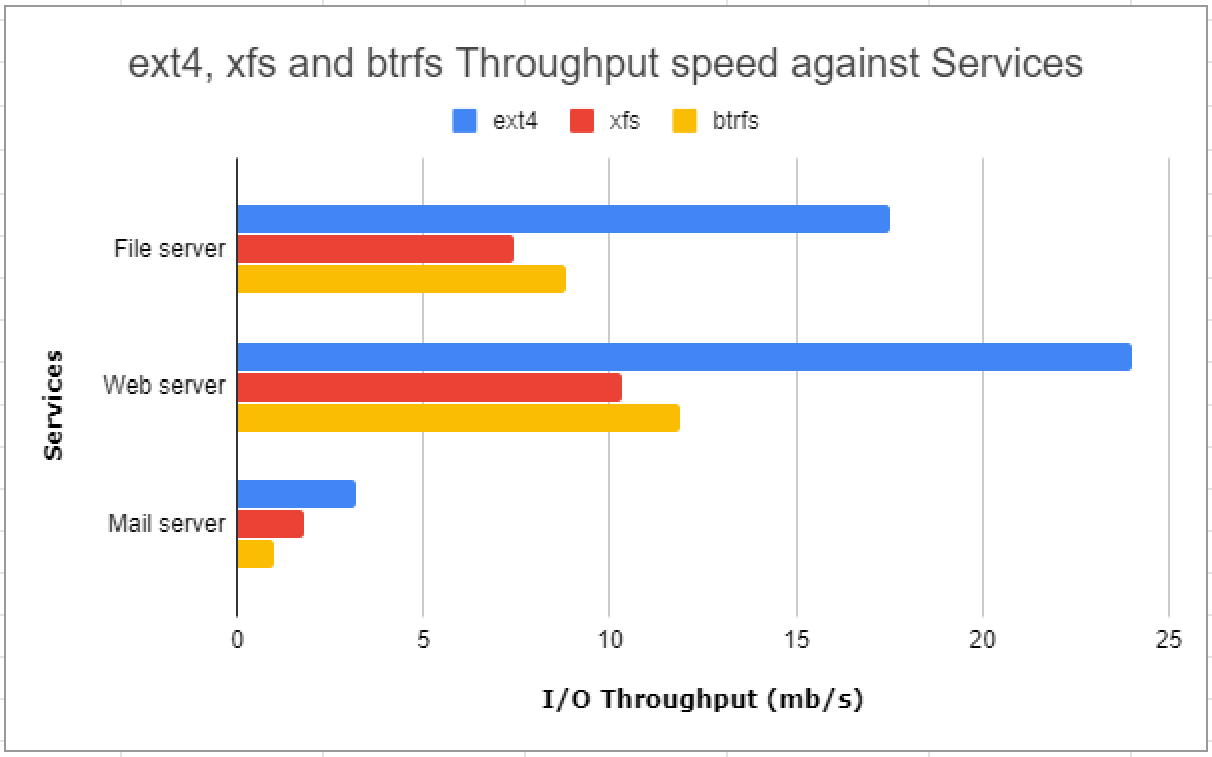

Figure 2 ext4, xfs and btrfs throughput speed comparison against with services

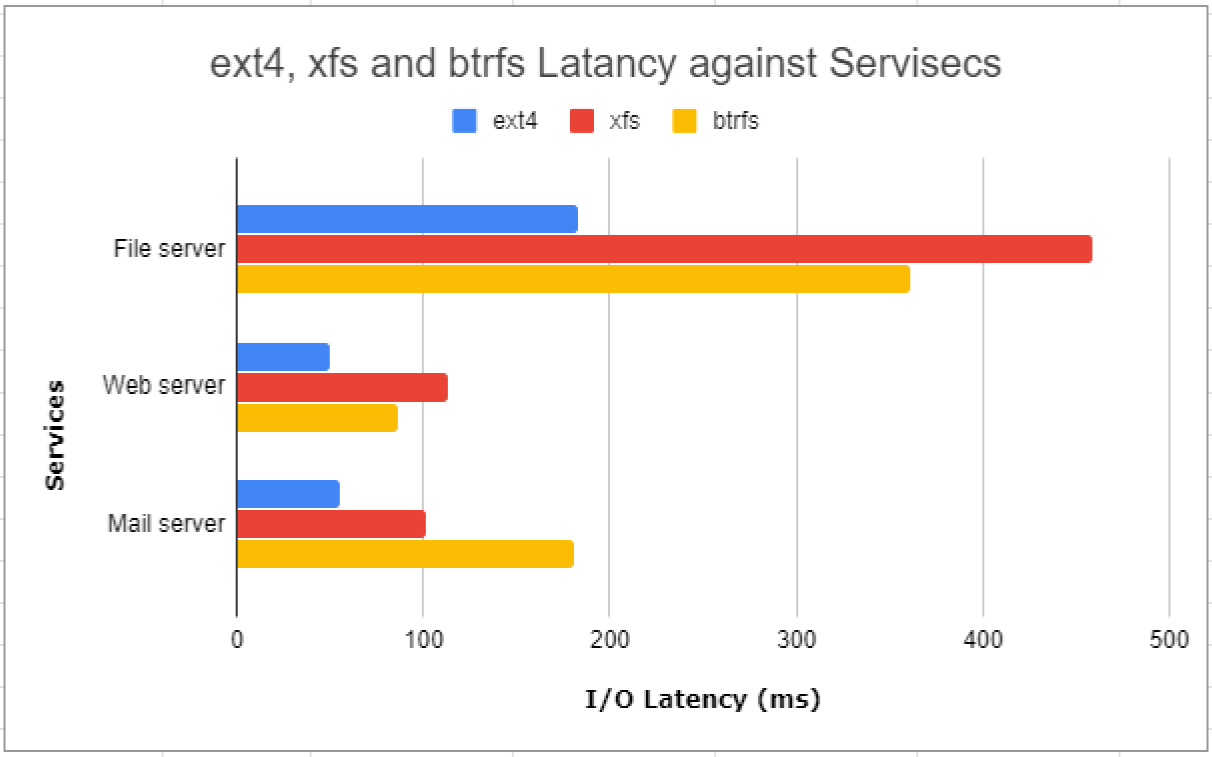

Figure 3 ext4, xfs and btrfs latency comparison against with services

```
echo "/dev/$1 /media/usb ext4 defaults 0 0" >> /etc/fstab
    mount –a
```

Middle part of the script is to mount the block device ($1) to created mount point (/media/usb). This configuration code must save inside the server "/etc/fstab" location to prevent discard of the changes after the reboot of the server. Finally, "mount –a" command will commit the changes.

```
echo "Starting backup for $SERVER..

    dd if=/dev/sda conv=sync,noerror  bs=64K | lz4 -z -f -B4  >
    /media/usb/backup_image.img.lz4

echo "Done"
```

Last part of the script will take the backup using "dd" (disk dump) command and output of the dd command will go through the "lz4" compression algorithm and make a compressed backup image named "backup_image.img.lz4". Backup image will redirect to the mount point which is mounted with external storage device.

Lz4 is a comparatively most efficient and lossless compressing algorithm [12]see Fig. 3; and that is the process which is used to speed backup mechanism other than the file system formatting process.

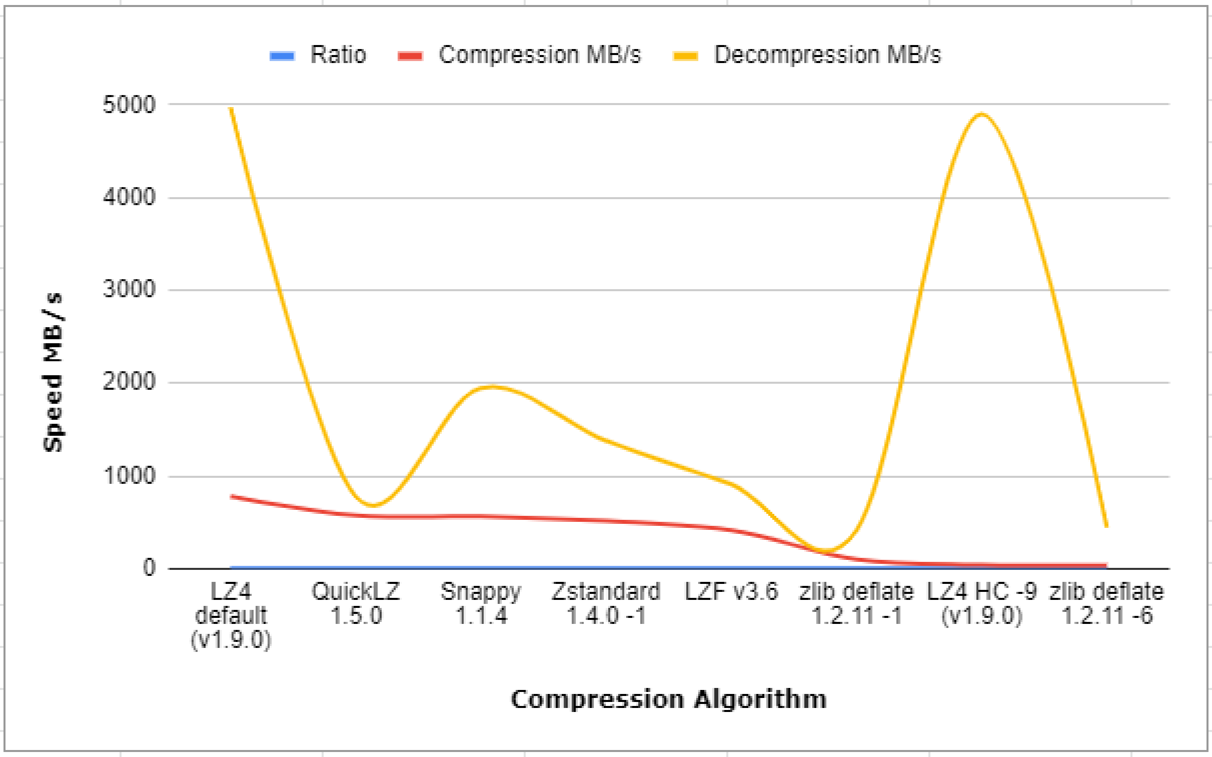

Figure 4 Compression algorithms comparisons against the speed (MB/s)

4.2 Intelligence Rollback

When we are considering the intelligence rollback function whole process is done through altogether 10 shell script. The whole process can be break down to four main parts.

1. Overall RAM usage comparison
2. Service wise RAM usage analysis
3. Check misconfigurations (enabled or disabled services during the hardening process)
4. Rollback the server

4.2.1 Overall RAM usage comparison

Two shell scripts named "int-ram.sh" and "fin-ram.sh" have been used to accomplish this overall RAM usage comparison task. Initially int-ram.sh script is executing before the hardening process is done. Output of the below command will be stored in "ram1.txt" file. Using "free" command user can get the in detail output of the RAM usage. Among them in here HardnBot will extract the total amount of used memory and free memory using "awk" command.

```
free | awk '/Mem/{printf("Used
    Memory:%.2f%\n"),$3/$2*100}/Mem/{printf("Free Memory:
    %.2f%\n"), $4/$2*100}' > ram1.txt
```

Same pipe line command is executing after the hardening process and output will be stored in “ram2.txt” file. When the user wants to compare the RAM usage against before the hardening and after the hardening, he/she can click “Analyze” button and it will display the free and used RAM usages respect to the above two stages (before the hardening, after the hardening) on circular progress bar. User will be able to get an idea about RAM usage of the server by using this function.

4.2.2 Service wise RAM usage analysis

HardnBot is capable of detecting abnormal RAM usages of the services which are running on server. For this it will use three shell scripts named “ramusage.sh, int-ram.sh, fin-service.sh”. int-ram.sh is a same script which is used in Overall RAM usage comparison function as well. That means it utilizes
for both functions (2.2.1 and 2.2.2).

As said in previously int-ram.sh script is running before the hardening process. Inside this script it contains following commands to collect the RAM usage of each services before the hardening.

```
./ramusage.sh > ramusage1.txt
   sed -e 's/ \+/\t/g' ramusage1.txt | cut -f13,14,17 |
grep    -w   RAM > ram-     service1.txt
   sed -e 's/ \+/\t/g' ramusage1.txt | cut -f15,16,19 >>
   ram- service1.txt
```

The ramusage.sh shell script is using inside the int-ram.sh script. First line executes the running command for the ramusage.sh script and output is redirect to the “ramsusge.txt”. By running the ramusage.sh script it provides services vice RAM usage. In here initially it will collect “private” and “shared memory” for each and every service running on the server. Overall RAM usage regarding each service is calculate using above private and shared memory.

```
Pss=`cat /proc/$PID/smaps | grep -e "^Pss:" | awk '{print $2}'|
     paste -     sd+ | bc `

     Private=`cat /proc/$PID/smaps | grep -e "^Private" | awk
'{print   $2}'|       paste -sd+ | bc `

     let Shared=${Pss}-${Private}

     echo -e "Private \t + \t Shared \t = \t RAM used \t Program"
```

Only the used memory and relevant service is extract from the ramusgae1.txt file and that output will redirect to the “ram-service1.txt” file.

Same process will be executing after the hardening process using fin-ram.sh script for collect the RAM usage of each services after the hardening and output will be redirect to the “ram-service2.txt” file.

When the user wants to clarify what are the services which is utilizing more RAM usage than before the hardening process, he/she needs to click on the increased services or decreased services buttons. Then following set of commands will execute and display to the user if there are any services which are utilizing more RAM usage than the way it used before the hardening process or vice versa.

```
awk '
    NR==FNR
    {
               s1[$2]=$1;next
    }

    {
    s2[$2]=$1
    }

    END
    {
      for (value in s2)
      {
          if ((s1[value]!=s2[value] && s2[value] > s1[value]*1.5 ))
               {
                    print s2[value]-s1[value]  "MB" "
    Increased",$2,value
               }
               else if (( s1[value]!=s2[value] && s1[value] >
    s2[value] ))
                    {
                         print (s1[value]-s2[value]) "MB" "
    Decreased",$2,value
                    }
     }
      }
    ' ram-service1.txt ram-service2.txt | column -t > f-outfile
```

Input files are ram-service1.txt and ram-service2.txt. This will get each service and its RAM usages respect to the two stages (before hardening and after hardening). Then compare if they (RAM usages) were increased or decreased and save them in a “f-outputfile” file. Either user clicks on increased services or decreased services button HardnBot will filter relevant services and display to the user with the increased or decreased RAM usage values.

4.2.3 Check misconfigurations (enabled or disabled services during the hardening process)

As mentioned in earlier, server hardening may cause to occur misconfigurations on the server, especially in services. Some disabled services can be enabled as well as some enabled services can be disabled during the hardening approach. We cannot prevent this issue, but we can detect and provide a solution for this issue.

HardnBot provides intelligence solution for this problem. For accomplish this whole task HardnBot uses five shell scripts (initialstate.sh, enfinstate.sh, disfinstate.sh, enable.sh, disable.sh).

At the beginning of the backup (before the hardening) the initialstate.sh script will run and collect the status of all services running on the server. Based on the status of the services they are divided into two parts (enabled and disabled). Basically that two statuses are the main status of any service. Then enabled services list redirect to enlist1.txt file and disabled services list redirect to dislist1.txt file.

```
 mkdir /rollback

   touch /rollback/enlist1.txt
   touch /rollback/dislist1.txt

   systemctl list-unit-files | grep enabled > /rollback/enlist1.txt
   systemctl list-unit-files | grep disabled > /rollback/dislist1.txt
```

Same commands include in the enfinstate.sh and disfinstate.sh script with few changes. Those two scripts used to collect status of the services after the hardening process. Except that HardnBot will identify the enabled and disabled services during the hardening process using above two scripts. Then identified services respectively redirect to the two files named enable.txt and disable.txt.

```
touch /rollback/dislist2.txt

     systemctl list-unit-files | grep disabled >
     /rollback/dislist2.txt

     diff /rollback/dislist1.txt /rollback/dislist2.txt >
     /rollback/disdiff.txt

     awk '{print $2}' /rollback/disdiff.txt > /rollback/disable1.txt
     awk 'NF' /rollback/disable1.txt > /rollback/disable.txt

     touch /rollback/enlist2.txt

     systemctl list-unit-files | grep enabled > /rollback/enlist2.txt

     diff /rollback/enlist1.txt /rollback/enlist2.txt >
     /rollback/endiff.txt

     awk '{print $2}' /rollback/endiff.txt > /rollback/enable1.txt
     awk 'NF' /rollback/enable1.txt > /rollback/enable.txt
```

once the user clicks “check enabled services” or “check disabled services” button from the “intelligence rollback” dashboard, HardnBot will display those services and asks to reset the services to their previous status. If the user wants to reset those services he/she can give the permission to the HardnBot and it will reset above services status using “enable.sh” and “disable.sh” shell scripts.

```
Cat /rollback/disable.txt | while read line

     do

     systemctl disable $line

     done
```

4.2.4 Rollback the server

If the user unsatisfied with the overall server status after the hardening, then he/she can roll back the server to its previous stable status using the backup taken by earlier. This is done by "rollback.sh" script and basically it is almost same to the Backup_Script.sh script with few changes.

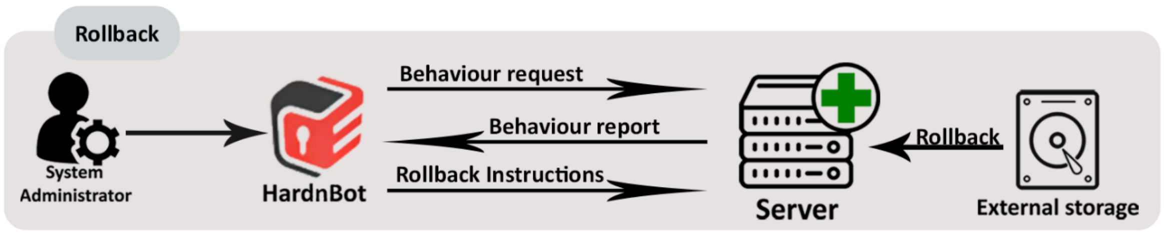

Figure 5 System diagram of intelligence rollback function

5. IMPLEMENTATION & TESTING

5.1 Implementation

When talking about the implementation of the HardnBot automatic backup and intelligence rollback functions, as a developer I have used "bunifu" framework for the design phase and C# for the developing phase. Altogether there are ten main shell scripts are belonging to these two functions and HardnBot needs to send those scripts to the server end time to time. For accomplishing that objective there is a class called "SendFilesToServer" which sends shell script using SCP client. Some final outputs of the above two functions are processed from the client-side application (HardnBot).

5.1 Testing

To attest the functionality and the performance of the HardnBot, some trial series has been run. For testing purpose, we have used Centos 7 and Red Hat 7 servers (virtual machines) which have 20GB hard drives and 2GB RAM on each server and for the client end windows 10 has been used. Both automatic backup and intelligence rollback functions have been tested during this phase.

6. RESULTS & DISCUSSION

6.1 Results

As mentioned in previously here we have utilized the *"dd"* command to get the backup and *"LZ4"* compression algorithm for compressing the backup. As mentioned previously for the experiment purpose, we have used Centos 7 and Red Hat 7 servers (virtual machines) which have 20GB hard drives on each server. When considering the time consumption, servers took 25 - 30 minutes (average) to take a backup.

Furthermore, the rollback function also examined using those virtual machines. Since LZ4 decompression rate is higher than the compression rate it only took less than 20 minutes to take servers to their initial state (rollback). Rollback is depending (not the speed) on abnormal behavior detection. Therefore, pretending as abnormal behavior to execute the rollback function, we have done some configuration changes to the servers. Therefore, it's mandatory to test abnormal behavior detection function as well. After configuring a few changes (appeared to be unusual changes) in servers (virtual machines) we executed an abnormal detection function and it took less than five minutes to identify and correct those unusual changes.

6.2 Discussion

This section contains further discussions on our research outcomes including problems occurred and future work.

6.2.2 Problems

In the testing phase, we had a plan for reach to the real server environment (industrial) and do the tastings on real servers, but unfortunately, the COVID 19 outbreak was there during our testing phase. That was a huge problem which we had faced. Because of that, we had to think another way of conducting a tasting phase. Due to that testing results may little bit differ.

6.2.2 Future works

Once the finished this COVID 19 outbreak, we hope to execute our plan of testing the HardnBot in real environment and do the needful fine-tunes as well. If the product is successful, then we can go for the other Linux servers in future.

7. CONCLUSIONS

By introducing automated backup and intelligence rollback features I was able to create a accurate and resilience system which reduce the user’s work load and make user more free, although there are some areas that still needs to be addressed. Testing phase need a real envirenment. Apart from that overall process is a success one.

8. REFERENCES

[1] "www.resources.newnettechnologies.com," [Online]. Available: https://resources.newnettechnologies.com/lp-hardened-services-guide?keyword=Server%20Hardening&gclid=EAIaIQobChMIicbyxIX-6AIVkHwrCh35XQb8EAAYASAAEgINgfD_BwE.

[2] Faraz Shaikh, Zoheb Shivani , "Snapshot service interface (ssi), a generic snapshot assisted backup framework for linux," in *International Conference on Digital Information Management*, 2007.

[3] Alireza Tajary, Hamid R. Zarandi , "An Efficient Soft Error Detection in Multicore Processors Running Server Applications," in *Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*, 2016.

[4] Olumuyiwa Ibidunmoye, Ewnetu Bayuh Lakew, Erik Elmroth , "A Black-Box Approach for Detecting Systems Anomalies in Virtualized Environments," in *International Conference on Cloud and Autonomic Computing (ICCAC)*, 2017.

[5] Yogendra Kumar Jain, Sandip S. Patil , "Design and Implementation of Anomalies Detection System Using IP Gray Space Analysis," in *International Conference on Future Networks*, 2009.

[6] Ming-Jen Chen, Chia-Chun Shih, Chien-Huei Yang, Gene Hong, Yuan-Sun Chu , "Multi-layered Monitoring Framework with Indices Relationship Analysis for Distributed Service Performance Evaluation," in *International Conference on Technologies and Applications of Artificial Intelligence*, 2011.

[7] Zhe Wang, Jin Zeng, Tao Lv, Bin Shi, Bo Li , "A Remote Backup Approach for Virtual Machine Images," in *International Conference on Cyber Security and Cloud Computing (CSCloud)*, 2016.

[8] L. Farinetti, P.L. Montessoro , "An Adaptive Technique for Dynamic Rollback in Concurrent Event-Driven Fault Simulation," in *IEEE International Conference on Computer Design ICCD'93*, 1993.

[9] Yongmin Zhao, Ning Lu , "Research and Implementation of Data Storage Backup," in *IEEE International Conference on Energy Internet (ICEI)*, 2018.

[10] Teruaki Sakata, Teppei Hirotsu, Hiromichi Yamada, Takeshi Kataoka , "A Cost-effective Dependable Microcontroller Architecture with Instruction-level Rollback for Soft Error Recovery," in *Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07)*, 2007.

[11] M.I. Marinov, D.R. Avresky , "Machine Learning Techniques for Predicting Web Server Anomalies," in *International Symposium on Network Cloud Computing and Applications*, 2011.

[12] "https://lz4.github.io/lz4/," GitHub, [Online]. Available: https://lz4.github.io/lz4/.