HardnBot
Intelligent Server Hardening Software

Project ID: 19_20-J 01

Final Report

R.M.B.B Rathnayake IT16054400

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-J01**

Final Report

R.M.B.B Rathnayake IT16054400

Supervisor:

Mr. Amila Senarathne

Bachelor of Science (Hons) in Cyber Security

Information Systems Engineering Department

Sri Lanka Institute of Information Technology

Sri Lanka

13h May 2020

DECLARATION

I declare that this is my work and this Preliminary Progress Review (PPR) report does not incorporate without acknowledgment 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 acknowledgment is made in the text.

……..
R.M.B.B Rathnayake
IT16054400

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

Supervisor

…….

Mr. Amila Senarathne

Table of Contents

DECLARATION iii
1. Abstract 1
2. Introduction 2
2.1. Definitions, Acronyms, and Abbreviations 2
2.2. Overview 2
2.3. Project scope 3
2.4. Background & Literature Survey 3
2.5. Research Gap 10
3. Objectives 11
3.1. Main Objectives 11
3.2. Specific objectives 11
4. Methodology 13
4.1. Setup connection between HardnBot and the remote server 13
4.2. Define a Benchmark for the Linux server security. 17
4.3. Design formatted configuration libraries/Design Solution playbooks. 18
4.4. Implement Automated Hardening function. 20
4.5. Implement a parameterized Hardening function. 22
5. Test 26
6. References 28

Table of Figures

Figure 1: Proposed Security Compliance Tool 8
Figure 2: Design process of trusted OS based on Linux 9
Figure 3: Get connected interface 13
Figure 4: HardnBot's CLI 14
Figure 5: Intelligent Hardening Interface 20
Figure 6: Hardening verification 21
Figure 7: Hardening in Progress 22
Figure 8: Hardening results 22
Figure 9: Tool tip in main sections 23
Figure 10: Configurations in System maintenance section 24
Figure 11: Enable/Disable main section 24
Figure 12: Hardening results | Parameterized Hardening function 25

Table of Charts

Table 1: Time comparison in hardening function 26

1. ABSTRACT

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. Servers can vary, depending on their usages. As an example, an email server is used to send and store emails over the corporate network. That means each server has its unique operations [1] and to serve the most secure/reliable service to their clients, servers should have proper compliances configured based on acceptable policies. Server hardening is the approach of applying the above-mentioned compliances to a server. 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. Using strong passwords, establish a password lockout policy, establish a data encryption mechanism are sample tasks that are just the tip of the iceberg as far as server hardening is concerned. Many more tasks must be completed, and each is more complex. For proper execution and maximum protection, professional assistance from an expert is needed. The probability of a misconfiguration occurrence is higher because hardening will carry out with human interaction. Scenarios where a misconfiguration 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 this paper, we consider developing an automated server hardening functionality to minimize the system administrator's work with the Ansible orchestration tool. In this realized scenario, more than 10 Linux servers (Red hat and CentOS) were utilized to test this developed functionality. To bring the introduced functionality closer to system administrators, the software was created and thoroughly tested.

Keywords – Server Hardening, Automation, Ansible, Linux

2. INTRODUCTION

2.1. Definitions, Acronyms, and Abbreviations

CIS	Center for Internet Security
GUI	Graphical User Interface
OS	Operating System
PCI DSS	Payment Card Industry Data Security Standard
NIST	National Institute of Standards and Technology
DAC	Discretionary Access Control
MAC	Mandatory Access Control
VPN	Virtual Private Network

2.2. Overview

Massive amounts of data are created daily across the planet. By 2021, the annual global Internet Protocol (IP) traffic is predicted to reach 3.3 zettabytes [2]. To match this huge data environment, the data center industry is anticipating unprecedented growth. Data is the most precious asset in data centers. Data centers require abilities to ensure data service works properly; many technologies are used in data centers to achieve this goal. Data centers are supported to run 24/7/365 without interruption. Planned or unplanned downtime can cause business users serious damage.

Most data centers include Linux servers, Ubuntu Server, Red Hat Enterprise Linux, and CentOS as their base operating system. A data center includes about 170+ live servers and it's very difficult to do the operating system hardening manually.

There are no fully automated hardening platforms implemented yet and, and when performing a hardening process, system administrators, network administrators, server custodians or outsourced expertise need to ensure the security of the operating system that runs on a server, databases, application and other services because a single mistake can affect the whole production line which the server is in. Even though many scanning tools can scan a server and identify compliance failures along with the solutions, the solutions are going to apply with human interaction. So, the probability of mistake occurrence is higher. And manual hardening process consumes much more resources such as time, human, cost likewise. As a solution to the lack of resources, organizations are tending to consider hiring external professionals and assets to perform the hardening task. In a scenario like this, internal critical classified information might have a possible chance to get exposed via outsourced professionals intentionally or unintentionally and leave the server in a critical position of been compromised. And there is a compulsory requirement of the root (administrator) access to the server to scan and perform operating system hardening. Also, the server needs to be temporarily out of living production, because

operating system hardening cannot be performed while the server is in the live production environment. So, when a critical day-to-day serving server is downed for maybe more than six hours to perform operating system hardening, it will be a critical impact on the organization's day-to-day business activities.

To solve these types of difficulties and prevent intentional and unintentional human errors, we developed an automated system hardening solution to explore and highlight the basic security configurations that should be performed to harden the security posture of a default Linux Operating System installation which can provide you with both scheduled or ad-hoc benchmark tests against CIS, PCI DSS and NIST standards. In this paper, we are going to discuss the theoretical approach and comparisons regarding previous researches, experiments we conduct, and obtained results based on those experiments.

2.3. Project scope

Various operating systems are used in server systems; however this research will focus on Linux Operating System, CentOS 6, 7 is selected as the operating system for this project. The basic principles, security guidelines, and best practices may apply to other operating systems as well, however, this project will only focus on the Linux based Operating Systems. The security guidelines, configuration, and settings explored by this research project may address Level 1 configuration profiles that are defined by the CIS Benchmark of a Linux Server.

Items in this Level 1 intend to:

- Be practical and prudent;
- Provide a clear security benefit; and
- Not inhibit the utility of the technology beyond acceptable means [3].

This profile is intended for servers. The project will specifically be focused on the more widely used Linux based Services, not including any custom and 3rd party applications. SSH is the most commonly used tool for remote administration and management of Linux Servers. For the sake of this project, we will explore the basic installation and configuration of OSSEC. This project will also explore and guide on checking what services are running on the server and turning off any unwanted services. This project will not address the security and configurations of all the tools and services available for Linux operating systems and or network security. This project will only cover Linux operating systems and the most common services on Linux Servers. This study alone will not make a Linux server completely secure from attacks or vulnerabilities; however, it will try to point out common settings and configurations that will harden the server security.

2.4. Background & Literature Survey

According to some previous researches done, we found researches that we can theoretically review their approach. These researchers research about various techniques to scan a server for compliances and perform hardening of servers and systems.

A research conducted by Prowse D. In 2010 on "OS Hardening and Virtualization" describes how to perform OS hardening using the OS security audit method. To perform

OS hardening, as a first step they perform vulnerability assessment over windows, then perform the security audit and fully analyze system logs. Further, they explained how important it is to perform periodic security audit over an OS to track vulnerabilities and take relevant countermeasures. As a benefit of doing OS hardening, they pointed out how it helps to reduce the risk, improve the performance, eliminates vulnerable entry points, and mitigate security risks. As this paper states, OS hardening can be done using techniques such as program clean-up, service packs, patch management, group policies, see templates, and configuration baselines. Further, for more user-friendliness, operating systems like windows provide facilities to prioritize vulnerabilities as high, medium, and low. To strengthen the security of OS, they discussed manual technology as well as semi-automated terms under manual techniques. Preparing checklist for security parameters, reviewing security configuration aspects, manually set security configuration, and explaining OS as per configuration parameter included. In a semi-automated way, they are using scripts for audits such as .bat, .ps, set security configuration using a script, exploiting OS scripted payload. In the discussion, they showed how important it is to perform periodic audits to identify security issues, prioritize those, and treat to mitigate risk over operating systems [4].

Amit Nepal published a research paper in 2013 on “Linux Server & Hardening Security” highlighting the basic security configurations that should be performed to harden the security posture of a default Linux Operating System installation. Its main approach was to keep the server as simple as possible, the complexity of the server is reduced. Also, a server with fewer services running is less vulnerable. By exploring the default installation and removing unwanted applications and services, we reduce the vulnerabilities in the server. This research carried out in four phases. In the first phase, it performs some intrusion attempts like brute force attacks, footprinting, etc. and explores the system response. Also it explores the commonly installed services and applications and identifies what applications and services we might not need and remove them. In the second phase, it makes the changes to the default configurations; put some restrictions in place with the common services and operating system. In the third phase, it configures the firewall (IPTABLES) to restrict access to certain ports from certain IP Addresses, so that the services to the server are available only to those who need it. Finally, in the fourth phase, it installs and configures a Host Based Intrusion Detection and Prevention System (OSSEC), which will act as a proactive monitoring and intrusion preventing system [5]. According to the phases conducted by the above researcher, I followed the operating system (OS) footprinting and identified explored the OS. Secondly, the main common services and operating system were identified by following the CIS benchmark. Services were identified whether to be removed, installed, or put some restrictions.

A research conducted by Christine Bresnahan and Richard Blum In 2020 on “Understanding Basic Security” describes how to understand basic security and identify user types. Accounts enable multiple users to share a single computer without causing one another too much trouble. They also enable system administrators to track who is using system resources and, sometimes, who is doing things they shouldn't be doing. Account features help users use a computer and administrators administer it. Understanding these features is the basis for enabling you to manage accounts. Some account features help you identify accounts and the files and resources associated with them. Knowing how to use

these features will help you track down account-related problems and manage the computer's users. Linux permits multiple users to access the computer simultaneously. Most often, this is done using remote access servers such as the Secure Shell (SSH); however, you can use Linux's virtual terminal (VT) feature to log in multiple times with a single keyboard and monitor. Sometimes, you might want to know who is using the computer. Linux is modeled after UNIX, which was designed as a multiuser OS. In principle, you can have thousands of accounts on a single UNIX (or Linux) computer. At least one user, though, needs extraordinary power to manage the features of the computer as a whole [6]. As said that root user privileges are needed to perform server hardening operations.

Douglas Santos and Jéferson Campos Nobre published a research paper in 2019 on "Vulnerability Identification on GNU/Linux Operating Systems through Case-Based Reasoning" highlighting automated rule-based tools are often used to support professionals with little experience in vulnerability identification activities. However, the utilization of rules sets up reliance on designers for the improvement of new standards just as to keep them updated. The inability to update rules can essentially bargain the integrity of vulnerability distinguishing proof outcomes. In this paper, inexperienced professionals are improved in conducting vulnerability identification activities by Case-Based Reasoning (CBR). The motivation behind utilizing CBR is to cause unpracticed experts to get comparable outcomes as experienced experts. Besides, the reliance on rule developers is lessened. A model was created thinking about the GNU/Linux framework to do an experimental evaluation. This assessment exhibited that the utilization of CBR improves the presentation of unpracticed experts as far as the number of identified vulnerabilities [7].

A research conducted by Mohan Krishnamurthy, Eric S. Seagren, Raven Alder, and Eli Faskha In 2020 on "Apache Web Server Hardening" describe that Apache is no different and can be negatively affected by any one of the following problems: poor application configuration; unsecured Web-based code; inherent apache security flaws; and foundational OS vulnerabilities. Apache has many default settings that require alteration for secure operation. Nearly all configuration information for the Apache Web server exists within the httpd. conf file and associated Include files. Web developers are unquestionably more worried about business usefulness than the security of their code. Simply publishing confidential or potentially adverse information without authentication can afford attackers with resources for an attack. There are several means by which hackers can breach or damage an Apache system, such as a denial of service; buffer overflow attacks; attacks on vulnerable scripts; and URL manipulation. Code deficiencies can exist in OSs and lead to OS and application vulnerabilities. It is therefore imperative to fully patch newly deployed systems and remain current with all released functional and security patches. The Apache Web server is a powerful application through which one can deliver critical business functionality to customers. With this power comes the possibility of misuse and attack. To ensure that the Apache server is running securely, a series of steps to harden the Apache application must be followed and these are: preparing the OS for Apache Web server; acquiring, compiling, and installing the Apache Web server software; and configuring the httpd. conf file. According to the research poor Operating system configuration is identified automatically by conducting an audit according to the CIS benchmarks of the relevant operating system [8].

Doug White and Alan Rea published a research paper in 2009 on “Server hardening model development: A methodology-based approach to increased system security”. This research forms a flawless model that combines information on tools, tactics, and procedures that system administrators can use to harden a server against compromise and attack [9]. Kyung Sung’s research paper published in 2020 on “Analysis of Linux firewall based on Firewalld” and “Design and Implementation of Firewall Security Policies using Linux Iptables” research paper published by M. G. Mihalos, S. I. Nalmpantis, Kyriakos Ovaliadis describes A default deny all policy on connections ensures that any unconfigured network usage will be rejected [10], [11]. According to these researches, with a default accept policy the firewall will accept any packet that is not configured to be denied. It is easier to white list acceptable usage than to blacklist unacceptable usage, and Applying host-based firewalls or port filtering tools on end systems, with a default-deny rule that drops all traffic except those services and ports that are explicitly allowed.

A research conducted by Jonas Schneider, Nils Fleischhacker, Dominique Schröder and Michael Backes In 2020 on “Efficient Cryptographic Password Hardening Services from Partially Oblivious Commitments” propose a construction for password hardening services based on a novel cryptographic primitive called partially oblivious commitments, along with an efficient secure instantiation based on simple assumptions. The performance and storage evaluation of our prototype implementation shows that our protocol runs almost twice as fast as Pythia, while achieving a slightly relaxed security notion but relying on weaker assumptions [12]. In my research passwords are cryptographically ensured by the hashing algorithm is SHA-512. The SHA-512 algorithm provides much stronger hashing than MD5, thus providing additional protection to the system by increasing the level of effort for an attacker to successfully determine passwords.

Amith Raj MP, Ashok Kumar, Sahithya J Pai, Ashika Gopal published a research paper in 2016 on “Enhancing security of Docker using Linux hardening techniques” describes that Docker supports the Linux hardening capabilities and Linux Security Modules (LSM) with AppArmor and SELinux for host system hardening. Docker interacts with the kernel security systems and LSMs. In this research work, the security depth of a popular open-source model based on containers and study on other security features and techniques to enhance the security of Docker is performed. Integrating virtualization, automated testing, Deployment tools, and security configurations management will enhance the security capability of Docker on-premise [13]. According to the above mentioned Linux security Modules, SELinux provides a Mandatory Access Control (MAC) system that greatly augments the default Discretionary Access Control (DAC) model. Under SELinux, every process and every object (files, sockets, pipes) on the system is assigned a security context, a label that includes detailed type information about the object. The kernel allows processes to access objects only if that access is explicitly allowed by the policy in effect [14].

The policy defines transitions so that a user can be allowed to run the software, but the software can run under a different context than the user's default. This automatically limits the damage that the software can do to files accessible by the calling user. The user does not need to take any action to gain this benefit. For an action to occur, both the traditional DAC permissions must be satisfied as well as the SELinux MAC rules. The action will not be allowed if

either one of these models does not permit the action. In this way, SELinux rules can only make a system's permissions more restrictive and secure. SELinux requires a complex policy to allow all the actions required of a system under normal operation. Three such policies have been designed for use with RHEL7 and are included with the system: targeted, strict, and mls. These are described as follows:

- targeted: consists mostly of Type Enforcement (TE) rules, and a small number of Role-Based Access Control (RBAC) rules. Targeted restricts the actions of many types of programs, but leaves interactive users largely unaffected.
- strict: also uses TE and RBAC rules, but on more programs and more aggressively.
- mls: implements Multi-Level Security (MLS), which introduces even more kinds of labels (sensitivity and category) and rules that govern access based on these [15].

A research conducted by Sonali Patra, N C Naveen, Omkar Prabhakar in 2016 on "An automated approach for mitigating server security issues" describes servers such as mail server, web servers, application server, etc.., store much sensitive information such as project details, media information, personal data, national security-related information, etc.., if such sensitive data gets into wrong hands. Business and the reputation of the organization will be damaged. Therefore, the need to automate security mechanisms to detect, prevent, and protect the server from the attackers. Security policies play an important role in network security and server security. An Automated Approach for Mitigating Server Security Issues proposes a framework that would ease the work of an administrator. It focuses on designing an automated tool that would perform an audit of the servers and check if it is compliant with all the prescribed security policies. As there are multiple platforms upon which the servers run, the tool is designed to adapt to a heterogeneous environment.

In this paper, an automated security tool as in Fig.1 is proposed that would ease the job of an administrator to check if the server complies with all the security policies of the organization. The security policies which were tested included the following,

1. Checking if the Windows server is running the approved up-to-date anti-malware solution.
2. Checking if the approved antivirus shield is seen in the taskbar for a system running Windows List the version of the Anti-Malware running on the systems for Linux and Windows operating system.
3. Check if any personal removable media present for both Linux and Windows Server, if so list their names and timestamp of the device insertion.
4. List all the security patches applied to the Windows operating system.
5. List the installed versions of software running on the Windows System.
6. Check if the event logs have been enabled or disabled.

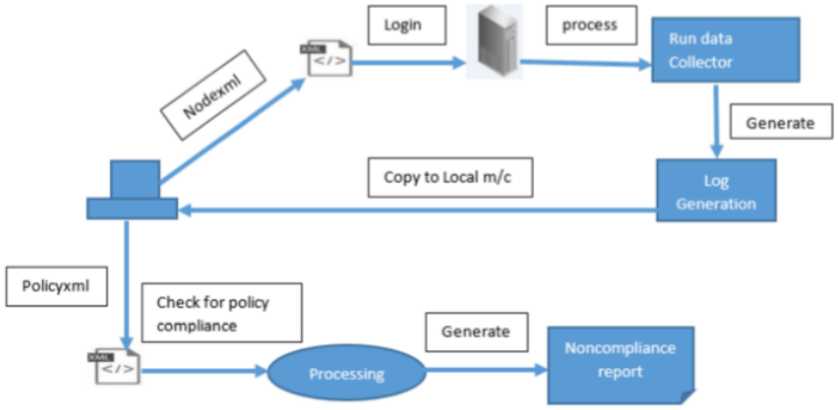

Figure 1: Proposed Security Compliance Tool

The following basic security steps which have defied in this research are applied in my research too,

- Every organization should have their security policies defined.
- There should be an apt network protection mechanism such as firewall technology, anti-virus technology, Intrusion Detection System (IDS), Virtual Private Network (VPN), and data encryption technology.
- Use of automated tools that would keep track of the server's security policies and their compliance.
- Use of secure administration and maintenance processes, which includes application of patches and upgrades, monitoring of all the logs, backing up of the server data, and operating system.
- Installation and configuration of a secure operating system and software in the server.
- Use of vulnerability scanner to perform security testing.
- Malware detection, mainly during the insertion of infected devices through USB.
- Configuring access control [16].

Ayei Ibor, Julius Obidinnu published a research paper in 2019 on “System Hardening Architecture for Safer Access to Critical Business Data” highlighting system hardening and system hardening architecture. This will be a guide to system administrators for implementing multi-layers of in-depth protective mechanism over the stored data.” System hardening is a strategy to increase the security of the system, which applied many different security measures to different layers to detect vulnerabilities of the system layers and defeat them before it damages the system. The proactive protective mechanism of system hardening architecture applied to the host, application, operating system, user, and the physical layers. This system hardening security strategy can be implemented to the organization, to decrease the breaches and also safer access data.

Here they develop a system hardening architecture. Create security functions and combine them independently with the relevant module. These functions are applied independently

and separately, however here they try to implement their security mechanism levels and use those mechanisms to relevant places which the relevant module located in the system. Because of that mechanisms if an attacker breaks the level one security he had to break several security mechanism levels to access the relevant data. Because of the higher number of security levels mechanisms, attackers had spent much more time and also wasting resources they may give up and find some other efficient way to do their malicious activities [17].

A research conducted by Hongjuan Li, Yuqing Lan in 2010 on "A Design of Trusted Operating System Based on Linux" demonstrates that a trusted operating system can help solve the information security problems. A design process of a trusted operating system based on Linux was developed by the China Standard Software Company (CS2C), and it's Still researching furthermore, Double-key authentication and architecture provide in this project. This operating system selects the improve/enhance the method to implement Designing Method. The benefit of a trusted operating system is to offer users a trusted computing environment [18].

<table>
<tr><td rowspan="4">User layer</td><td>shell layer</td><td colspan="3">shell program</td></tr>
<tr><td rowspan="2">utility layer</td><td rowspan="2">general applications</td><td colspan="2">trusted processes</td></tr>
<tr><td>expansion of the original procedures</td><td>new programs</td></tr>
<tr><td rowspan="2">Secure core layer</td><td>system call layer</td><td>security-related system calls</td><td>security-unrelated system calls</td><td>new system calls</td></tr>
<tr><td>core layer</td><td>security-related entities</td><td>security-unrelated entities</td><td>new entities</td></tr>
<tr><td>Hardware layer</td><td colspan="4">Hardware interface</td></tr>
</table>

Figure 2: Design process of trusted OS based on Linux

The architecture of trusted operating system based Linux shown three layers of architecture, they are hardware layer in the bottom, secure core layer in middle, and application layer in the top.

The research on "Automated Provisioning of Application in IAAS Cloud using Ansible Configuration Management" mainly focuses on Cloud. Cloud has become a powerful technology in at present's information technology background and the need arises to stand with the evolving request of clients. This paper focuses on the automation of customer application right from environment provisioning to application deployment. In this paper, the whole architecture of automated application deployment assembled using amazon as IAAS provider and Ansible as the orchestration engine [19]. This Ansible engine is used as the configuration management tool in my research.

2.5. Research Gap

Until now, there has been little work reviewed in the above researches in specifying or detailing the importance of Linux server security and Design and Implementation server hardening models to enhance the security of Linux servers. Based on researches "Automated Provisioning of Application in IAAS Cloud using Ansible Configuration Management" [19] and research on "Automated Provisioning of Application in IAAS Cloud using Ansible Configuration Management" [20] were discussed in length and defined as the following.

Development of an automated application deployment assembled using amazon as IAAS provider and Ansible as the orchestration engine. Using these metrics I introduce an alternative means of measuring its effects and impact on the field of Linux server security. The automated Linux server hardening approach hasn't been reviewed so far. To bring the Full Potential of Ansible Framework functionality closer to system administrators, automated server hardening enabling them to do the configuration changes via the HardnBot's automated hardening functionality.

3. OBJECTIVES

3.1. Main Objectives

This research aims to find an effective way to secure the datacenter Red Hat Enterprise Linux version 6, 7, and CentOS version 7, 6 servers and enhance productivity for the customers and employees with a low cost and few human interactions. Hence there is no fully automated hardening platform implemented yet, in this system implementation a novel automated open-source software is proposed.

Here fully automated free open source hardening platform is developed with the capability to detect poor or non-compliant configurations in a system (OS/DB/Application) and applying industry recommended fixes/configurations and secure systems by reducing its surface of vulnerabilities.

This research composes a great business value as it can cover 90% of an Information Systems Audit and hardening process. For internal audits, this software presents both opportunity and responsibility. By helping the organization understand and control risks and identifying opportunities/industry best practices to embrace. This software also will allow the internal audit to position themselves as trusted advisors.

i. Automatically detect uncompliant configurations of the server

   To make these critical servers more secure, this system can automatically perform a dry run feature. With the Dry Run feature, you can execute the playbook without having to make changes on the server [21]. With Ansible Dry Run you can see if the host is getting a configuration changed or not.

ii. Automated Linux server hardening

   To prevent intentional and unintentional human errors, the automated system hardening solution explores and highlights the basic security configurations that should be performed to harden the security posture of a default Linux Operating System installation which can provide you with both scheduled or ad-hoc benchmark tests against CIS standard.

iii. Bring industry best practices into the system.

   For a particular parameter in the system configuration, there is an industry-recognized value. For example, a password should expire at most in 90 days. A company may not adhere to these standard values; its security policies may not describe them. On such occasions, system administrators may have assigned them with default values. Through our software, we plan to introduce industry-accepted values and configurations into the information systems.

3.2. Specific objectives

i. Reduce manual work

Through this software, we target to reduce the manual effort needed for a complete system hardening. Almost all the parts of the audit and hardening could be performed through this software. Compliance issues are automatically detected. Reports can be generated through this software at the end of the hardening process.

ii. Ease the internal audit

This software is designed in a way that could be easily used by non-technical people. Simple interfaces, pop-up guidelines, and descriptive reports will make the user aware of the functionalities of the software. Even before applying a fix to an issue, the user can read a full detailed report of that issue including its impact and the remediation.

This software package can be handled by a single user. Therefore, the entire system's hardening process can be conducted by a single user. This requires a minimal amount of the organization’s resources. As for the internal audit, this software will cover up to 90% of their work.

4. METHODOLOGY

4.1. Setup connection between HardnBot and the remote server

HardnBot has a separate interface to initiate the connection with a server. This functionality was developed using the SSH.NET library. SH.NET is a Secure Shell (SSH) library for .NET, improved for parallelism and with extensive framework support. It was used to establish a secure shell connection with the server and a new connection to be established, it requires four parameters, IP address, port number, username, and password accordingly.

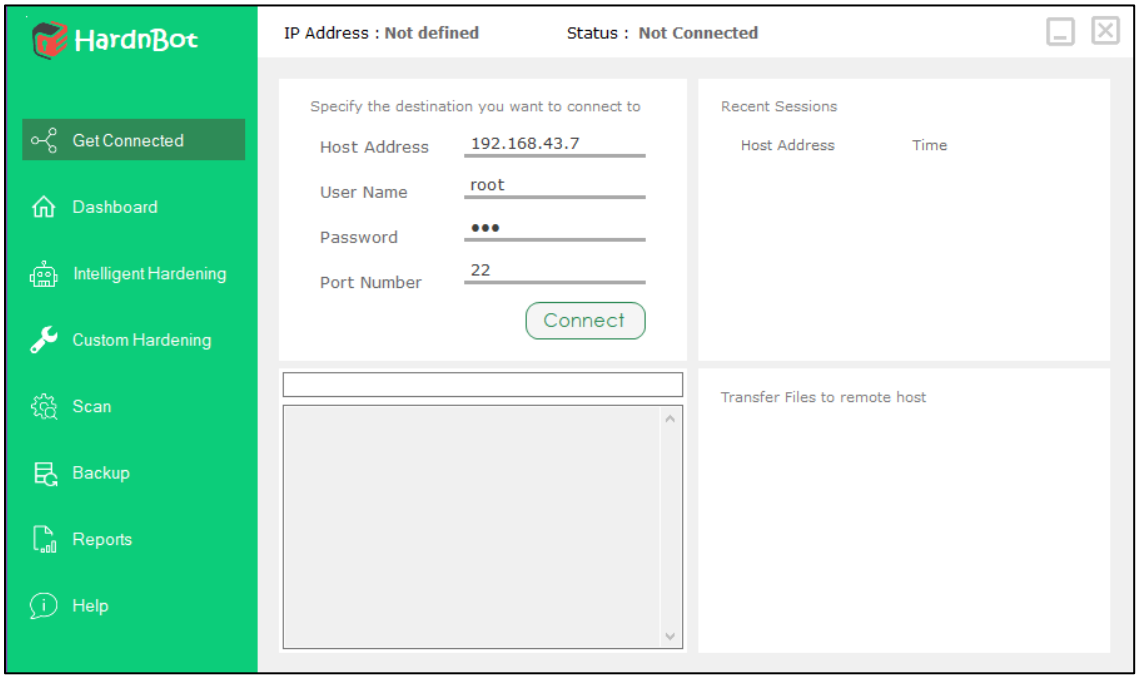

Figure 3: Get connected interface

When developing the scanning interface, we used “Bunifu UI” framework to design the interface. We used C# as our main programming language of the software, but the required server-side scripts are developed using shell.

To get the connection, we used C# library named “Renci SshNet/SSH.NET [10 - 11]” and whenever the system administrator clicks on the “Connect” button HardnBot will get the user inputs (destination IP, root username, root password and the port) provided by the system administrator and pass it to the SshClient object we created. Next HardnBot will connect to the remote server and wait for 120 seconds to get a successful connection. Within this time if a connection failure occurs, HardnBot will notify the user before the 120 seconds we defined.

If the remote connection is successful, HardnBot will save the connection status to a variable, and this variable will be very useful in-order to check the connection status to other functions in HardnBot to function properly. Below is the code segment of this functionality,

```
public static Boolean connectionstat;


this.sshClient = new SshClient(textBox1.Text,Convert.ToInt32(textBox4.Text),textBox2.
Text, textBox3.Text);
this.sshClient.ConnectionInfo.Timeout = TimeSpan.FromSeconds(120);
this.sshClient.Connect();
connectionstat = this.sshClient.IsConnected;
```

After connecting to the remote server, HardnBot will get the server's Operating system information, Kernel and Kernel released date and save them to variable that are used later to display this information to the user.

```
//get OS
SshCommand OS = this.sshClient.CreateCommand("uname -o");
OS.Execute();
os = Convert.ToString(OS.Result.ToString());
//get kernal
SshCommand KERNAL = this.sshClient.CreateCommand("uname -s");
KERNAL.Execute();
kernal = Convert.ToString(KERNAL.Result.ToString());
//get kernal_release-date
SshCommand Release = this.sshClient.CreateCommand("uname -r");
Release.Execute();
kernal_release = Convert.ToString(Release.Result.ToString());
```

After this, all parameters are saved to variables to later use. And then HardnBot will create the shell stream to get user commands form HardnBot connection interface.

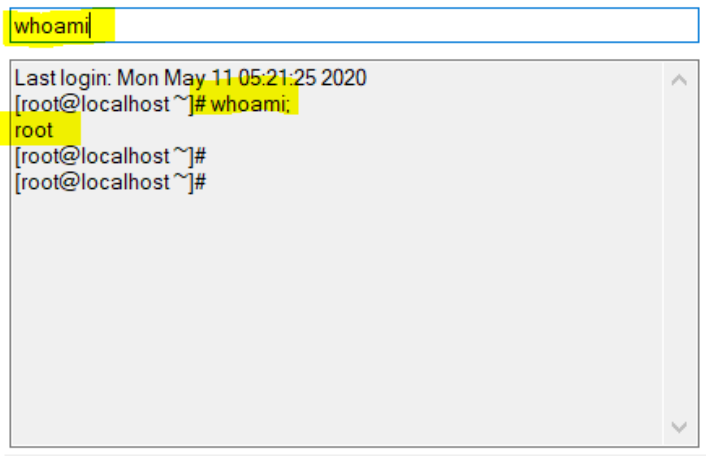

Figure 4: HardnBot's CLI

```
IP = textBox1.Text;
UN = textBox2.Text;
PW = textBox3.Text;
port = textBox4.Text;
this.shellStreamSSH = this.sshClient.CreateShellStream("vt100",80,60,800,600,65536);
connectbtn.Visible = false;
disconnectbtn.Visible = true;
MessageBox.Show("Successfully Connected");
label5.Text = IP;
label7.Text = DateTime.Now.ToString();

//required method for the shell stream.
private void recieveSSHData()
{
while (true)
       {
              try
              {
                if (this.shellStreamSSH != null && this.shellStreamSSH.DataAvailable)
                  {
                     String strdata = this.shellStreamSSH.Read();
                     appendTextBoxInThread(txtSSHConsole, strdata);
                  }
              }catch
               {
               }
                 System.Threading.Thread.Sleep(200);
        }
}

private void appendTextBoxInThread(TextBox t, String s)
{
 if (t.InvokeRequired)
 {
  t.Invoke(new Action<TextBox, string>(appendTextBoxInThread), new oject[] { t, s });
 }
 else
 {
    t.AppendText(s);
 }
}
```

```
private void txtCommand_KeyPress(object sender, KeyPressEventArgs e)
{
    try
     {
           if (e.KeyChar == '\r')
                {
                    this.shellStreamSSH.Write(txtCommand.Text + ";\r\n");
                    this.shellStreamSSH.Flush();
                }
            }
            catch
            {
                throw;
            }
}
```

We also used accessor methods to return private static variables to other classes.

```
//accessor method for kernal
public static String getKN()
{
    return kernal;
}
//accessor method for kernal_release
public static String getKR()
{
    return kernal_release;
}
//accessor method for OS
public static String getOS()
{
    return os;
}
//accessor method for IP
public static String getIP()
{
    return IP;
}
//accessor method for UN
public static String getUN()
{
    return UN;
}
//accessor method for PW
public static String getPW()
{
    return PW;
}
//accessor method for Port
public static String getPort()
{
    return port;
}
```

4.2. Define a Benchmark for the Linux server security.

Center for Internet Security (CIS) Benchmark is referred to as the benchmark which is a consensus-driven security guideline for the Red Hat Enterprise Linux Operating Systems. CIS benchmarks are configuration standards and best practices for securely configuring a system. Each of the control references one or more CIS controls that were developed to help organizations improve their cyber defense competences. NIST Cyber security Framework (CSF) and NIST SP 800-53, the ISO 27000 series of standards, PCI DSS, HIPAA, and many more recognized standards and regulatory frameworks are mapped with CIS controls [22].

As the scope is limited to Red Hat Enterprise Linux Operating Systems version 6 and 7, following latest benchmarks were referenced to design formatted playbooks,

- CIS CentOS Linux 6 Benchmark v2.1.0

- CIS CentOS Linux 7 Benchmark v2.2.0
- CIS Red Hat Enterprise Linux 6 Benchmark v2.1.0
- CIS Red Hat Enterprise Linux 7 Benchmark v2.2.0

CIS benchmarks provide two levels of security settings:

- **Level 1** recommends essential basic security requirements that can be configured on any system and should cause little or no interruption of service or reduced functionality.
- **Level 2** recommends security settings for environments requiring greater security that could result in some reduced functionality.

The scope only focuses to find an automated way to perform all Level 1 configuration profiles of the CIS benchmark.

Items in Level 1 – Server profile intend to:

- be practical and prudent;
- provide a clear security benefit; and
- Not inhibit the utility of the technology beyond acceptable means, this profile is intended for servers.

4.3. Design formatted configuration libraries/Design Solution playbooks.

The playbooks are designed using YAML which is a human-readable data serialization language. Concerning the CIS benchmark level 1 profiles, playbook, designing is performed by six main modules. These modules perform hardening function as shown below;

1. Install Updates, Patches and Additional Security Software
2. Configurations of OS Services
3. Network Configuration and Firewalls
4. Logging and Auditing Configurations
5. System Access, Authentication, and Authorization
6. System Maintenance

HardnBot executes these modules by Ansible. Ansible works against multiple managed nodes or “hosts” in the infrastructure at the same time, using a list or group of lists in the inventory. Once the inventory is defined, we can select the hosts or groups you want Ansible to run against. The default location for inventory is located in `/etc/ansible/hosts`. Following is a basic inventory file in YAML format:

```
all:
  hosts:
    mail.example.com:
  children:
    webservers:
      hosts:
        web1.example.com:
        web2.example.com:
    dbservers:
      hosts:
        db1.example.com:
        db2.example.com:
        db3.example.com:
```

These main sections of playbooks are scripted with CIS benchmark configurations. Before executing these modules firstly it checks OS version and family and Check ansible version and run Preliminary tasks list as follows.

```
- name: Check OS version and family
  fail:
    msg: "This role can only be run against RHEL 7. {{ ansible_distribution }}
     {{ ansible_distribution_major_version }} is not supported."
  when:
    - ansible_os_family == 'RedHat'
    - ansible_distribution_major_version is version_compare('7', '!=')
  tags:
    - always

- name: Check ansible version
  fail:
    msg: You must use ansible 2.1 or greater
  when: not ansible_version.full is version_compare('2.1', '>=')
  tags:
    - always

- include: prelim.yml
  become: yes
  tags:
    - prelim_tasks
    - always
```

Then it executes each above-mentioned playbook one after the other. These playbooks explore the following services and check what services are running on the server and turning off any unwanted services.

avahi_server	rpc_server	snmp_server	named_server	dovecot
cups_server	ntalk_server	squid_server	nfs_rpc_server	samba
dhcp_server	rsyncd_server	Samba server	mail server	squid

ldap_server	tftp_server	dovecot server	bind	net_snmp
telnet_server	rsh_server	httpd_server	vsftpd	autofs
nfs_server	nis_server	vsftpd_server	nfs_server	nfs_server

4.4. Implement Automated Hardening function.

To perform automated hardening Ansible configuration management, and application-deployment tool is used. Ansible is capable of run on many Unix-like systems and can configure both Unix-like systems as well as Microsoft Windows [23]. An interface is designed in HardnBot software to perform this function.

If any compliance gets compliant with the CIS benchmark, it skips and check and fix the rest of compliances by executing all the playbooks.

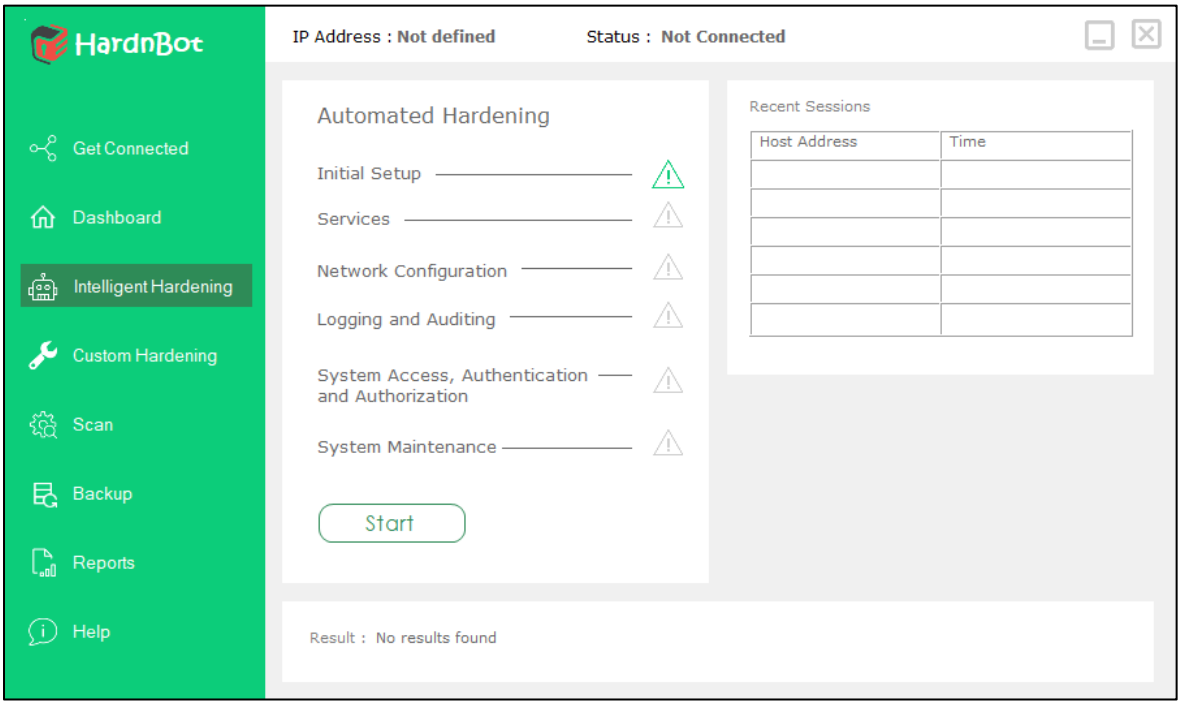

Figure 5: Intelligent Hardening Interface

By this interface, the automated hardening function is called to execute the playbooks via Ansible engine. This intelligent hardening is more efficient for hardening default Linux Operating System installations. In this function, all the playbooks for each section are enabled and all the level 1 configurations will get executed by this function. Following is a sample code of enabled main 6 sections.

rhel7cis_section1: true
rhel7cis_section2: true
rhel7cis_section3: true
rhel7cis_section4: true
rhel7cis_section5: true
rhel7cis_section6: true

These well-formatted ansible playbooks will perform changes by checking every configuration. If an already compliant configuration is met it get skipped to the next configuration.

After navigating to the intelligent hardening interface, system administrator can click on “Start” button and if there is a connection, HardnBot will ask the user to verify the hardening before executing the hardening against the server, see Fig. (6). If the user clicks on “Yes” the intelligent hardening process will begin, see Fig. (7).

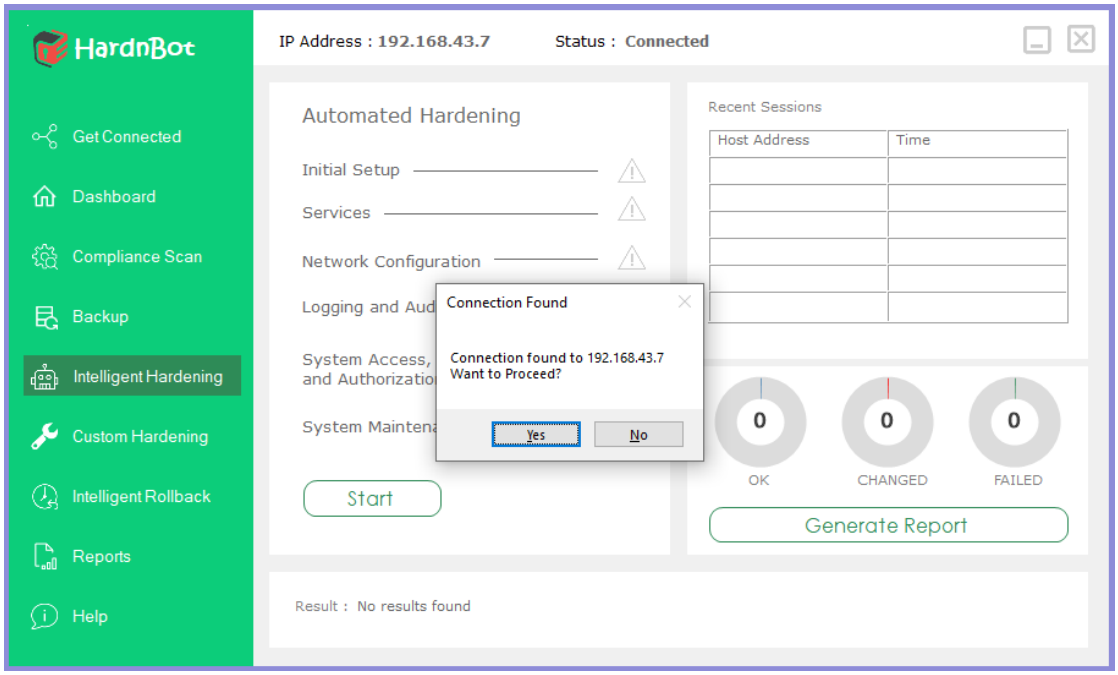

Figure 6: Hardening verification

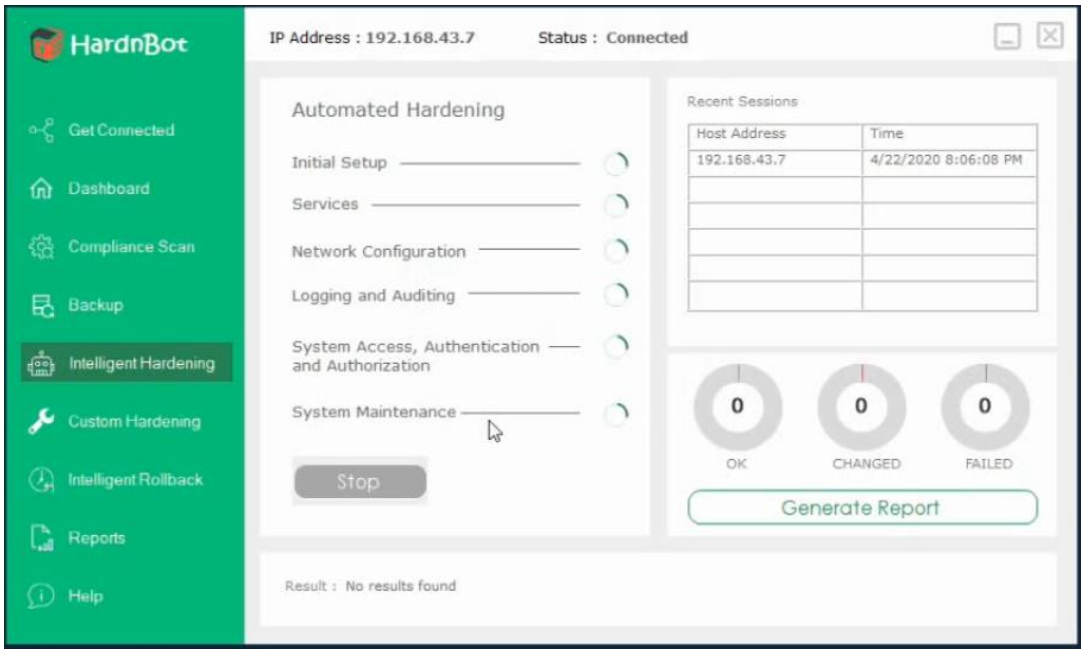

Figure 7: Hardening in Progress

After the hardening, HardnBot will receive the hardening results and display them in the results panel following a message that says the “Hardening Complete”, see Fig. (8).

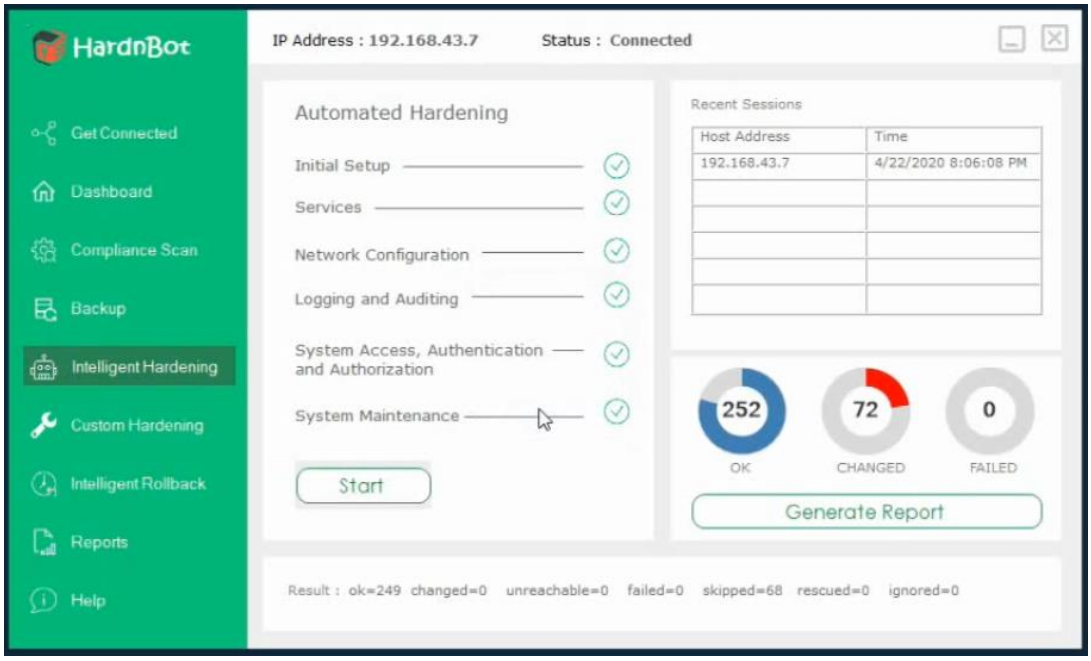

Figure 8: Hardening results

4.5. Implement a parameterized Hardening function.

Parameterized hardening enables the system administrator to input industry-accepted values and configurations. In HardnBot there are interfaces designed to get user inputs to

customize these configurations. Users can either disable or enable main sections or users are given separate interfaces to customize the configurations in each of the sections. For user's sake, when user move the mouse pointer to the relevant section's panel, it shows the configuration types which includes in the specific section, see Fig. (9).

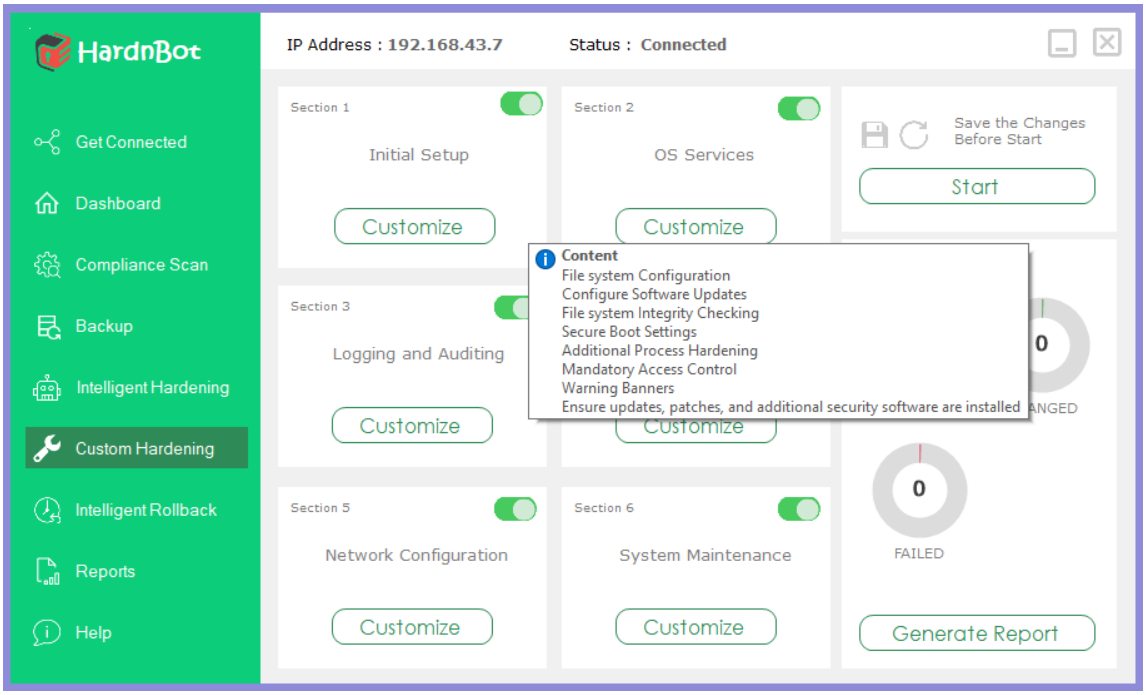

Figure 9: Tool tip in main sections

Users can enable or disable whole sections as their requirements and users can modify the configurations inside these sections, see Fig. (10). the reset function enables users to reset the configurations to its default state.

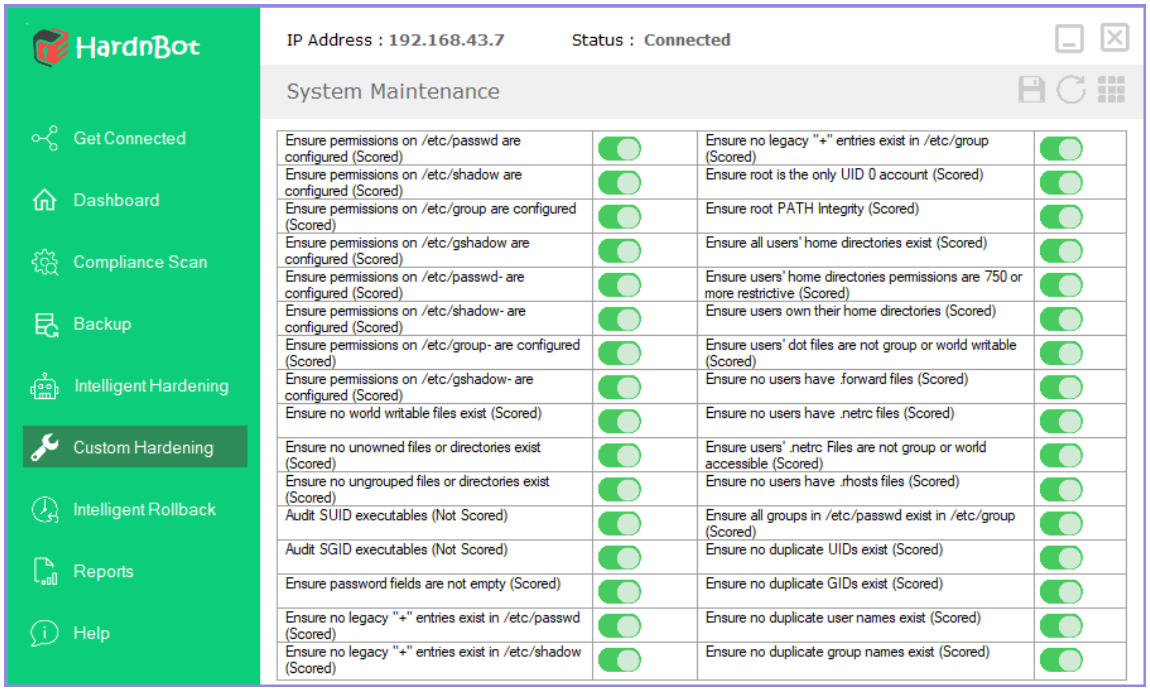

Figure 10: Configurations in System maintenance section

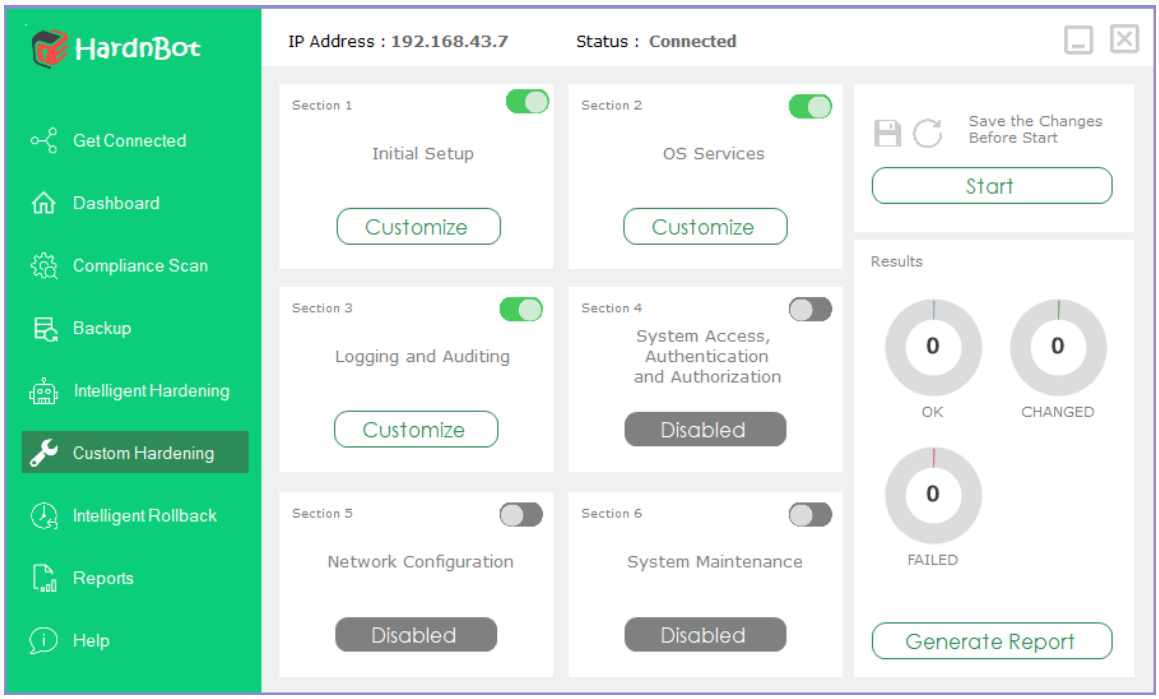

Figure 11: Enable/Disable main section

As in the intelligent hardening function, HardnBot will ask the user to confirm the hardening before executing the hardening against the server and once it is completed HardnBot will receive the hardening results and display them in the results panel following a message that says the “Hardening Complete”, see Fig. (12).

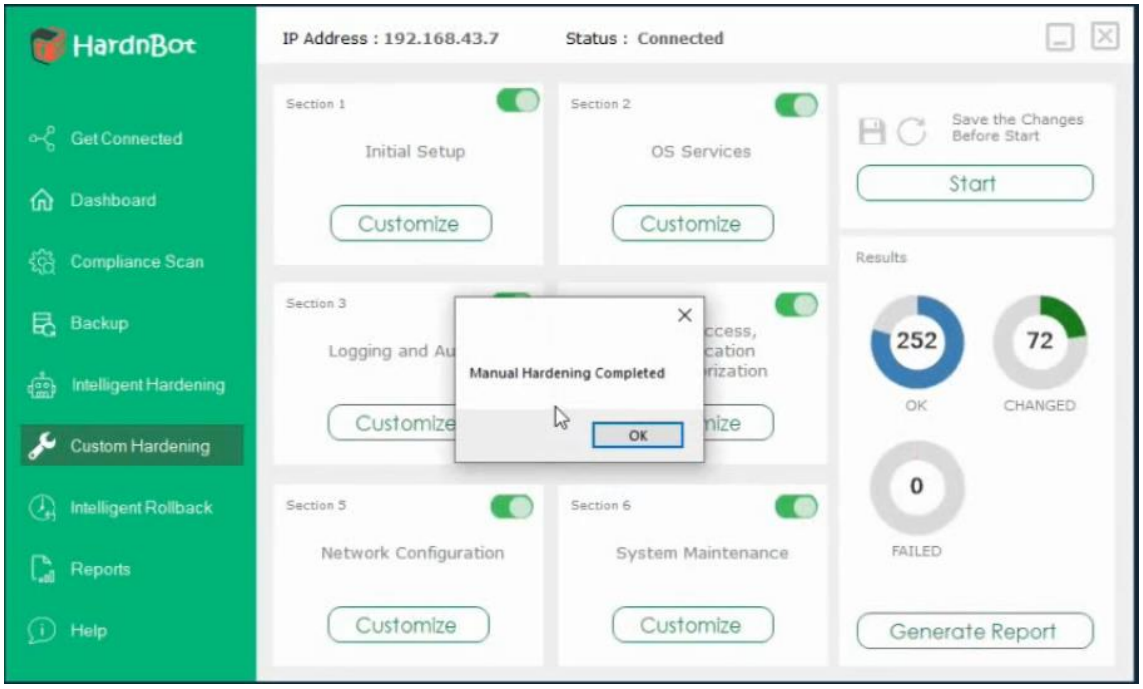

Figure 12: Hardening results | Parameterized Hardening function

5. TEST RESULTS AND DISCUSSION

To attest the functionality and the performance of the Hardening function, some trial series has been run and the trial results are reviewed in this section.

Experimental setup: The intelligent hardening software was run and tested using windows 10 client and it uses a Linux virtual machine as a jump server. The servers that were tested included Centos 7 and Red Hat 7 Linux distributions. These tested servers were virtual machines and were hosted in VMware Workstation 15.5.1.

Effectiveness: Intelligent Hardening was tested with freshly installed Centos 7 and Red Hat 7 servers and successfully hardened within 20-30 minutes. The following chart shows the order of the functionalities evaluated and its performance evaluation compared with the manual process and the HardnBot software. Following chart shows the time evaluated to complete a single server hardening successfully compared with manual process and automated intelligent process.

Table 1: Time comparison in hardening function

	Manual Process	HardnBot
Hardening	< 5 hours	> 30 minutes

6. CONCLUSIONS

In this research, I have blended into the automation of industry server operating system security hardening via scanning a server and identifying operating system compliance failures, classifying them according to the severity levels defined by a proper and thorough search in each and every security compliance in an operating system and then harden those identified issues with an efficient manner which is capable of hardening only required compliances automatically and ignoring unwanted compliances such as compliances related with IP tables manipulation.

This approach save the valuable time and a misconfiguration occurrence by a network administrators, system administrators, outsourced professionals, or server custodian’s. In overall intelligent hardening process and parameterized hardening is proven success.

7. REFERENCES

[1] V.Beal, 2011. [Online]. Available: https://www.webopedia.com.

[2] cisco, "www.cisco.com," [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html.. [Accessed 28 February 2020].

[3] "Center for Internet Security," [Online]. Available: https://www.cisecurity.org/home.

[4] "webopedia," [Online]. Available: https://www.webopedia.com/quick_ref/servers.asp..

[5] A. Nepal, "Linux Server & Hardening Security," in *IEEE*, 2013.

[6] C.Bresnahan, R.Blum, "Understanding Basic Security," in *IEEE*, 2020.

[7] J.Nobre, "Vulnerability Identification on GNU/Linux Operating Systems through Case-Based Reasoning," in *IEEE*, 2019.

[8] M.Krishnamurthy,E.Seagren, "Apache Web Server Hardening," in *IEEE*, 2008.

[9] Doug White,Alan Rea, "Server hardening model development: A methodology-based approach to increased system security," in *IEEE*, 2009.

[10] K. Sung, "Analysis of Linux firewall based on Firewalld," in *IEEE*, 2020.

[11] "Design_and_Implementation_of_Firewall_Security_Policies_using_Linux_Iptables," in *IEEE*.

[12] Jonas Schneider,Nils Fleischhacker,Michael Backes, "Efficient Cryptographic Password Hardening Services from Partially Oblivious Commitments," in *IEEE*, 2016.

[13] Amith Raj MP,Ashok Kumar,Ashika Gopal,Sahithya J Pai, "Enhancing security of Docker using Linux hardening techniques," in *IEEE*, 2016.

[14] N. Petreley, "The ultimate Linux server," in *IEEE*, 2006.

[15] "SELINUX," [Online]. Available: http://www.selinuxproject.org/.

[16] Sonali Patra,Omkar Prabhakar, N C Naveen , "An automated approach for mitigating server security issues," in *IEEE*, Bangalore, 2016 .

[17] Julius Obidinnu,Ayei Ibor, "System Hardening Architecture for Safer Access to Critical Business Data," in *IEEE*, 2015.

[18] Hongjuan Li , Yuqing Lan , "A Design of Trusted Operating System Based on Linux," in *IEEE*, Wuhan, 2010.

[19] Nishant Kumar Singh ,Sanjeev Thakur , Himanshu Chaurasiya ,Himanshu Nagdev , "Automated provisioning of application in IAAS cloud using Ansible configuration management," in *IEEE*, Dehradun, 2015.

[20] Pavel Masek ,Martin Stusek , Jan Krejci ,Krystof Zeman, Jiri Pokorny, Marek Kudlacek , "Unleashing Full Potential of Ansible Framework: University Labs Administration," in *IEEE*, Jyvaskyla, 2018.

[21] admin, "mw[i]," [Online]. Available: https://www.middlewareinventory.com/blog/ansible-dry-run-ansible-check-mode/.

[22] "docs.microsoft.com," [Online]. Available: https://docs.microsoft.com/en-us/microsoft-365/compliance/offering-cis-benchmark?view=o365-worldwide.

[23] "Ansible," [Online]. Available: https://www.ansible.com/.

[24] Ibor, A. , Obidinn,J., SYSTEM HARDENING ARCHITECTURE FOR SAFER ACCESS TO CRITICAL BUSINESS DATA, Nigerian Journal of Technology, 2015.

[25] H. Li , Y. Lan, "A Design of Trusted Operating System Based on Linux," in *IEEE*, Wuhan, 2010.