

Security hardening using infrastructure as code

Damir Regvart^{1*},  Jasmin Redžepagić², Adriano Bubnjek³,  Robert Petrunić⁴

^{1,2,3,4}Department of System Engineering and Cybersecurity, Algebra University, Zagreb, Croatia; damir.regvart@algebra.hr (D.R.) jasmin.redzepagic@algebra.hr (J.R.) abubnjek@algebra.hr (A.B.) robert.petrunic@algebra.hr (R.P.)

Abstract: This paper examines Infrastructure as Code (IaC) with Ansible to automate and enhance security hardening in Linux environments. As IT infrastructures grow more complex, manual security configurations become error-prone, inefficient, and inconsistent. IaC addresses these issues by allowing organizations to define and deploy infrastructure configurations as code, ensuring a consistent security baseline. Focusing on key settings for Firewalld, SELinux, and SSH, the study demonstrates how Ansible enforces these configurations in a scalable, repeatable, and resilient manner. Results show that using Ansible for security hardening reduces deployment times, minimizes manual errors, and ensures uniform security standards across diverse systems. This research offers a practical foundation for organizations seeking to improve their cybersecurity posture, emphasizing IaC's transformative potential in achieving secure, efficient, and adaptable infrastructure management.

Keywords: *Ansible, Security hardening, Configuration management, Linux automation.*

1. Introduction

Infrastructure as Code (IaC) is emerging as a vital approach for automating and standardizing these environments, helping to meet the dual demands of agility and security. This paper explores using IaC, specifically with the tool Ansible, to implement security hardening measures within Linux-based infrastructures. By focusing on critical security configurations for Firewalld, SELinux, and SSH, this work demonstrates how Ansible can enforce security policies in a repeatable, efficient, and scalable manner. Through practical, detailed examples, this paper illustrates the processes involved in automating these security configurations to mitigate vulnerabilities and minimize the risk of manual misconfigurations.

This paper aims to bridge the gap between traditional security hardening methods and automated infrastructure management, showcasing the potential of IaC to enhance security while improving operational efficiency. Ansible's idempotence, or its ability to apply consistent configurations without creating redundancy, is central to the approach presented here. Furthermore, this paper discusses the challenges and benefits of using Ansible for security hardening, emphasizing the integration of IaC with Continuous Integration/Continuous Deployment (CI/CD) pipelines to maintain real-time compliance and readiness for rapid deployment. In this way, the work demonstrates the potential of IaC to drive secure and agile infrastructure management, aligning with modern DevOps and DevSecOps principles.

This paper's contributions include a practical guide on automating security hardening for Linux systems using Ansible and insights into the benefits of integrating IaC with CI/CD workflows to support continuous security. Additionally, this work highlights the applicability of IaC in enforcing systematic security policies across different environments, addressing both on-premises and cloud-based infrastructures. By examining Ansible's scalability, flexibility, and adaptability in Red Hat-based systems, this paper provides a foundation for further research and adoption of IaC in enterprise security practices.

The rest of this paper is organized as follows. The next section will cover related work to hardening Linux environments, followed by an in-depth look at discussing the usage of IaC in modern IT systems and then configuring FirewallD, SELinux, and SSH using Ansible. The paper concludes with a discussion of potential future research directions and a conclusion.

2. Related Work

System hardening, particularly in Linux-based and critical infrastructure environments, has gained significant attention in recent years as the threat landscape continues to evolve. One approach to improving installation efficiency in hardened Linux systems is highlighted by Ortiz-Garcés, et al. [1] who proposed a methodology using Spacewalk for optimizing installation times in critical infrastructures, ultimately aiding in enhancing system resilience in such environments [1]. Similarly, Rose and Zhou [2] examine system hardening strategies for Infrastructure as a Service (IaaS) platforms, focusing on enhancing security protocols within the shared infrastructure of cloud environments to reduce vulnerability exposure Rose and Zhou [2]. Bosio, et al. [3] address the critical need for security hardening in real-time operating systems to ensure reliability and functionality, particularly in time-sensitive systems where security lapses could lead to substantial operational disruptions [3].

In specialized systems like facial recognition technologies, Cindori, et al. [4] explore hardening mechanisms to safeguard these systems against unauthorized access and data breaches, emphasizing the unique vulnerabilities inherent to biometric systems Cindori, et al. [4]. Stöckle, et al. [5] advance the field by automating the implementation of Windows security configurations, showcasing methods to streamline security guide adherence across enterprise environments [5]. Further advancing automation, He and Vechev [6] leverage large language models to conduct adversarial testing for code security, proposing that AI tools can enhance the effectiveness of hardening efforts in code-based environments [6].

Infrastructure-as-Code (IaC) principles are also applied to open networking, as illustrated by Salazar-Chacón and Parra [7] who use tools like Git and Ansible to streamline security configurations in network environments, particularly in Linux-based systems like Cumulus Linux [7]. Addressing closed-source software, Huang, et al. [8] investigate methods for mitigating vulnerabilities, highlighting the challenges posed by limited source visibility and proposing frameworks for enhancing closed-system security [8]. Kernel security is further enhanced through techniques like profile-guided indirect branch elimination, as suggested by Duta et al., where kernel hardening is achieved through innovative control-flow methodologies aimed at preventing malicious access [9].

In large-scale organizational contexts, Stöckle, et al. [10] discuss Scapolite, a DevOps-based solution designed to simplify and improve security guide authoring and testing, supporting consistent and scalable hardening practices Stöckle, et al. [10]. Amarchand, et al. [11] emphasize foundational Linux security principles, highlighting the importance of basic hardening strategies within Linux systems for widespread security impact [11]. Additional innovations include Franzen, et al. [12] RandCompile, a tool designed to eliminate forensic gadgets from the Linux kernel, further complicating the analysis and exploitation of kernel vulnerabilities [12].

Yin, et al. [13] introduce an audit and monitoring framework based on MQTT for Linux systems, aiming to provide continuous security monitoring capabilities essential for dynamic and interconnected systems [13]. In the domain of cyber-physical systems, Potteiger, et al. [14] integrate moving target defense with control reconfiguration strategies to protect against targeted attacks, especially in infrastructure-critical cyber-physical applications [14]. For web development environments, Wang, et al. [15] focus on API hardening as a means to prevent DOM-based XSS vulnerabilities, advocating for security measures embedded directly into development workflows [15]. For legacy IoT devices, Carrillo-Mondéjar, et al. [16] propose HALE-IoT, a firmware-based solution to retrofit security features into outdated devices, effectively extending their lifespan without compromising security [16]. Lastly, Spensky, et al. [17] address security concerns in physical interfaces of cyber-physical systems,

developing TRUST.IO to enhance interface security. This is a critical consideration, given the rise in physical threats to cyber infrastructure [17].

This body of research illustrates diverse hardening approaches, each tailored to the unique demands of specific systems and environments. These studies highlight the necessity of adopting multifaceted hardening strategies, from automation and AI-driven tools to targeted IoT and kernel security measures, to ensure protection across varying infrastructure layers and technological platforms.

3. Using Infrastructure as Code in Modern IT Systems

Before exploring the main focus of this seminar paper, security hardening, it is essential first to discuss how IaC tools like Ansible are transforming modern IT infrastructures by enhancing agility, flexibility, and automation. These tools enable the rapid deployment and configuration of standardized physical or virtual instances.

In this context, agility refers to the IT system's capacity to adapt swiftly and efficiently to changes in the market, technology, or internal business processes. Tools such as Ansible and Terraform, a commonly used combination, facilitate this agility by allowing IT administrators to quickly provision infrastructure, typically virtual machines, and configure them uniformly. This approach significantly accelerates the provisioning and configuration process, reducing the likelihood of misconfiguration. Additionally, IaC configurations, such as Ansible playbooks, are treated as code, making it possible to leverage version control systems like Git or apply GitOps principles. This practice ensures that changes are trackable, auditable, and reversible when necessary, supporting a more agile and controlled approach to infrastructure management. Ansible further enhances efficiency by allowing community or vendor roles, modules, and collections to streamline the configuration process. For customized needs, IT teams can create private roles and modules stored in private collections within repositories to foster DevOps concepts and methodologies within the organization.

Moreover, IaC integrates seamlessly with CI/CD pipelines, allowing for automated testing and deployment of infrastructure changes. This integration ensures that infrastructure is consistently in a deployment-ready state, supporting rapid development cycles and enabling quicker feature and update delivery.

In terms of flexibility, Ansible adapts to meet the diverse and evolving needs of various environments, including public cloud providers (such as AWS, Azure, and Google Cloud), cloud-native platforms like Kubernetes or OpenShift, on-premises environments (both Linux and Windows), and network devices like routers and switches. Standard configurations can be abstracted into reusable roles or modules, reducing errors and saving time across different projects. However, despite these advantages, implementing IaC, especially at an organizational level, presents challenges due to its steep learning curve. During initial adoption, IT teams (such as DevOps engineers or system administrators) may revert to manual configuration or use Bash or PowerShell scripts integrated into playbooks, which can undermine the intended purpose of Ansible or similar IaC tools.

This section concludes with an overview of IaC and its growing popularity as a modern infrastructure management approach. The next chapter will dive into the technical aspects of Ansible, providing examples of playbooks that can enhance the security of newly deployed RHEL systems.

4. Using Ansible For Linux Machine Hardening

Given the broad scope of security hardening for Linux (or any operating system), which varies depending on the machine's role, installed applications, and its environment, this section will focus on enhancing security through Firewalld, SELinux, and SSH configurations using Ansible. These components will be discussed, followed by playbook examples for practical application.

4.1. Firewalld Configuration

The first component addressed is Firewalld, with the following playbook ensuring its installation and activation:

- ```

- name: Playbook for security hardening
 hosts: webserver
 tasks:
 - name: Ensure Firewalld is installed
 ansible.builtin.dnf:
 state: present
 name: firewalld
 - name: Ensure that firewalld is enabled and started
 ansible.builtin.service:
 name: firewalld
 state: started
 enabled: true

```

This playbook installs the Firewalld package and starts the service. Although Firewalld is pre-installed and enabled by default on RHEL, Ansible's idempotence ensures no repeated changes are made. To test connectivity, additional applications (MariaDB and httpd) are installed as shown below:

- ```

---
...
- name: Ensure httpd and mariadb packages are installed
  ansible.builtin.dnf:
    state: present
    name: "{{ item }}"
  loop:
    - mariadb
    - httpd
- name: Ensure httpd and mariadb are enabled and started
  ansible.builtin.service:
    name: "{{ item }}"
    state: started
    enabled: true
  loop:
    - httpd
    - mariadb

```

After these services are running, testing shows that external connections are blocked on ports 80 and 3306, as shown in Figure 1.

```

[student@servera ~]$ sudo nc -v serverb 22
Ncat: Version 7.91 ( https://nmap.org/ncat )
Ncat: Connected to 172.25.250.11:22.
SSH-2.0-OpenSSH_8.7
^C
[student@servera ~]$ sudo nc -v serverb 80
Ncat: Version 7.91 ( https://nmap.org/ncat )
Ncat: No route to host.
[student@servera ~]$ sudo nc -v serverb 3306
Ncat: Version 7.91 ( https://nmap.org/ncat )
Ncat: No route to host.

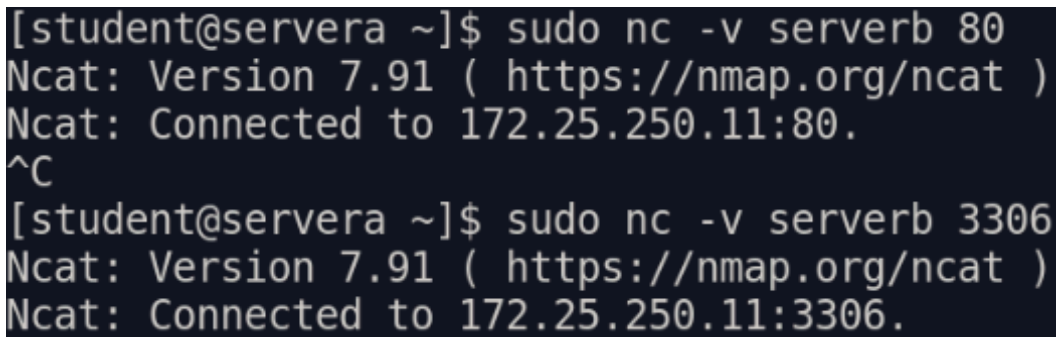
```

Figure 1.
Connection test to port 80 and 3306.

The playbook is then updated to permit these connections:

```
...
- name: Configure firewalld to allow HTTP and MySQL
  ansible.posix.firewalld:
    state: enabled
    immediate: true
    permanent: true
    service: "{{ item }}"
  loop:
    - mysql
    - http
```

Now, we can successfully connect, as shown in Figure 2.



```
[student@servera ~]$ sudo nc -v serverb 80
Ncat: Version 7.91 ( https://nmap.org/ncat )
Ncat: Connected to 172.25.250.11:80.
^C
[student@servera ~]$ sudo nc -v serverb 3306
Ncat: Version 7.91 ( https://nmap.org/ncat )
Ncat: Connected to 172.25.250.11:3306.
```

Figure 2.
Successful connection to ports 80 and 3306.

Additionally, non-essential services like “cockpit” and “dhcpv6-client” are disabled:

```
...
- name: Block non-required services
  ansible.posix.firewalld:
    state: disabled
    permanent: true
    immediate: true
    service: "{{ item }}"
  loop:
    - cockpit
    - dhcpv6-client
```

This concludes the Firewalld subsection of the Linux hardening using Ansible. Next, we will configure the SELinux for HTTPD (Hypertext Transfer Protocol daemon).

4.2. SELinux Configuration for HTTPD

SELinux requires configuration for the HTTPD server to bind to a non-standard port (10080). This involves modifying the httpd.conf file and apply the following play:

```
---
- name: SELinux configuration
  hosts: webserver
  vars:
    selinux_policy: targeted
    selinux_state: enforcing
    selinux_ports:
```

```

- ports: 10080
  proto: tcp
  setype: http_port_t
  state: present
tasks:
- name: Apply SELinux role
  block:
  - include_role:
    name: redhat.rhel_system_roles.selinux
  rescue:
  - name: Check for failure for other reasons than required reboot
    ansible.builtin.fail:
    when: not selinux_reboot_required
  - name: Restart managed host
    ansible.builtin.reboot:
  - name: Reapply SELinux role to complete changes
    include_role:
    name: redhat.rhel_system_roles.selinux
- name: Restart httpd service
  service:
  name: httpd
  state: restarted

```

This play sets SELinux to “enforcing,” configures the port context, and restarts HTTPD. If a reboot is required, it automatically re-applies the SELinux configuration. We will configure SSH (Secure Shell) hardening in the following subsection.

4.3. SSH Hardening

The SSH service is hardened by changing its listening port to 2022 and turning off root login and password authentication. SELinux is also configured to allow SSH on the new port:

```

---
- name: SSH hardening
  hosts: webserver
  vars:
  selinux_ports:
  - ports: 2022
    proto: tcp
    setype: ssh_port_t
    state: present
  tasks:
  - name: Apply SELinux role
    block:
    - include_role:
      name: redhat.rhel_system_roles.selinux
    rescue:
    - name: Check for failure for other reasons than required reboot
      ansible.builtin.fail:
      when: not selinux_reboot_required
    - name: Restart managed host
      ansible.builtin.reboot:
    - name: Reapply SELinux role to complete changes

```

```

    include_role:
      name: redhat.rhel_system_roles.selinux
- name: Configure firewalld
  ansible.posix.firewalld:
    state: enabled
    immediate: true
    permanent: true
    port: 2022/tcp
- name: Configure SSH configuration
  ansible.builtin.copy:
    src: ./ssh-hardened
    dest: /etc/ssh/sshd_config
    owner: root
    group: root
    mode: 0600
  notify: restart ssh
handlers:
- name: restart ssh
  ansible.builtin.service:
    name: sshd
    state: restarted

```

The associated SSH configuration file contains the following settings:

```

Port 2022
Include /etc/ssh/sshd_config.d/*.conf
AuthorizedKeysFile /etc/.rht_authorized_keys .ssh/authorized_keys
ClientAliveInterval 60
Subsystem sftp /usr/libexec/openssh/sftp-server
PasswordAuthentication no
PermitRootLogin no
PubkeyAuthentication yes

```

This section demonstrated how Ansible can improve Linux security by managing FirewallD, SELinux, and SSH configurations.

5. Future Research

Although this paper primarily focused on foundational security configurations for FirewallD, SELinux, and SSH through Ansible, more advanced areas within Infrastructure as Code (IaC) warrant deeper investigation to strengthen security practices across diverse infrastructures.

One area of interest involves developing advanced security policies and implementing Role-Based Access Control (RBAC) within IaC tools like Ansible, Terraform, and Chef. By designing more granular access controls in automated configurations, security can be significantly enhanced in complex, multi-user environments. Future research could also explore integrating Identity and Access Management (IAM) systems with IaC solutions, analyzing the effects on security in both cloud-based and on-premises setups.

Another potential research direction is automated compliance and auditing for IaC-based environments. Adhering to security standards such as CIS, NIST, and ISO is critical for secure infrastructure management. Future studies could focus on creating specialized playbooks and IaC modules that perform compliance checks and remediation tasks. Incorporating these compliance checks within CI/CD pipelines could help organizations maintain continuous security and regulatory adherence in real time, reducing vulnerabilities introduced by non-compliance.

6. Conclusion

This paper demonstrates that IaC represents a significant advancement in the automation and standardization of security hardening practices, particularly within Linux-based environments. By applying IaC tools like Ansible, organizations can address the growing demands for agility, consistency, and security in infrastructure management. By automating crucial security configurations such as FirewallD, SELinux, and SSH, IaC reduces the time and effort required for manual setups and minimizes human error, a common source of security vulnerabilities. Ansible's idempotence allows for repeatable and reliable deployment processes that ensure systems maintain a uniform security baseline, reducing the risk of configuration drift over time. The work underscores Ansible's adaptability and modularity, making it possible to apply security hardening at scale across various environments, from on-premises infrastructure to cloud-based platforms and across different development and deployment stages.

As highlighted in the paper, integrating IaC with CI/CD pipelines further elevates security practices by enabling continuous compliance and rapid adaptation to new security requirements. This integration supports ongoing monitoring and automated testing of configurations and aligns with modern DevOps and DevSecOps practices, facilitating seamless collaboration between development, operations, and security teams. The continuous, automated checks provided by IaC integration within CI/CD pipelines allow for real-time remediation of security issues, fostering a proactive security culture. This capability is precious in dynamic and large-scale environments where frequent changes and rapid deployment cycles can create security gaps if not managed effectively.

Despite its advantages, adopting IaC for security hardening also presents challenges, as implementing automated configurations and achieving seamless integration into existing workflows require specialized knowledge and a cultural shift toward embracing automation and standardization. The paper suggests that future research should explore advanced security policies, such as RBAC and identity management, within IaC frameworks to enhance fine-grained control over infrastructure access. Additionally, it points to the need to develop automated compliance and auditing tools within IaC solutions aligned with security standards like CIS, NIST, and ISO. Such advancements could enable organizations to meet regulatory requirements continuously, reducing risks associated with non-compliance and improving overall security posture.

Transparency:

The authors confirm that the manuscript is an honest, accurate, and transparent account of the study; that no vital features of the study have been omitted; and that any discrepancies from the study as planned have been explained. This study followed all ethical practices during writing.

Copyright:

© 2025 by the authors. This open-access article is distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

References

- [1] I. Ortiz-Garcés, A. Echeverría-López, and R. Andrade, "Methodological proposal for the optimization of the installation times of hardened Linux operating systems through the Spacewalk solution in critical infrastructures," presented at the 2020 International Conference on Computational Science and Computational Intelligence (CSCI). <https://doi.org/10.1109/CSCI51800.2020.00024>, 2020.
- [2] T. Rose and X. Zhou, "System hardening for Infrastructure as a Service (IaaS)," presented at the 2020 IEEE Systems Security Symposium (SSS). <https://doi.org/10.1109/SSS47320.2020.9174202>, 2020.
- [3] A. Bosio, S. Carlo, M. Rebaudengo, and A. Savino, "Toward the hardening of real-time operating systems," presented at the 2022 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT). <https://doi.org/10.1109/DFT56152.2022.9962356>, 2022.
- [4] D. Cindori, I. Tomičić, and P. Grd, "Security hardening of facial recognition systems," presented at the 2021 44th International Convention on Information, Communication and Electronic Technology (MIPRO). <https://doi.org/10.23919/mipro52101.2021.9596961>, 2021.

- [5] P. Stöckle, B. Grobauer, and A. Pretschner, "Automated implementation of Windows-related security-configuration guides," presented at the 2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE). <https://doi.org/10.1145/3324884.3416540>, 2020.
- [6] J. He and M. T. Vechev, "Large language models for code: Security hardening and adversarial testing," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*. <https://doi.org/10.1145/3576915.3623175>, 2023.
- [7] G. Salazar-Chacón and D. M. Parra, "Infrastructure-as-code in open-networking: git, ansible, and cumulus-linux case study," *2023 IEEE 13th Annual Computing and Communication Workshop and Conference (CCWC)*. <https://doi.org/10.1109/CCWC57344.2023.10099084>, 2023.
- [8] Z. Huang, G. Tan, and X. Yu, "Mitigating vulnerabilities in closed source software," *EAI Endorsed Trans. Security Safety*, vol. 8, no. 30, pp. e1-e4, 2022. <https://doi.org/10.4108/eetss.v8i30.253>
- [9] V. Duta, C. Giuffrida, H. Bos, and E. Van Der Kouwe, "PIBE: practical kernel control-flow hardening with profile-guided indirect branch elimination," in *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*. <https://doi.org/10.1145/3445814.3446740>, 2021.
- [10] P. Stöckle, I. Pruteanu, B. Grobauer, and A. Pretschner, "Hardening with scapolite: A DevOps-based approach for improved authoring and testing of security-configuration guides in large-scale organizations," in *Proceedings of the Twelfth ACM Conference on Data and Application Security and Privacy*. <https://doi.org/10.1145/3508398.3511525>, 2022.
- [11] G. Amarchand, P. Brown, and T. Mahoney, "Linux security," *Advances in Engineering Innovation*, vol. 2, pp. 17-20, 2023. <https://doi.org/10.54254/2977-3903/2/2023015>
- [12] F. Franzen, A. C. Wilhelmer, and J. Grossklags, "Rand compile: Removing forensic gadgets from the Linux kernel to combat its analysis," in *Proceedings of the 39th Annual Computer Security Applications Conference*. <https://doi.org/10.1145/3627106.3627197>, 2023.
- [13] J. Yin, Y. Ishikawa, and A. Takefusa, "A linux audit and MQTT-based security monitoring framework," *2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC)*. <https://doi.org/10.1109/COMPSAC57700.2023.00090>, 2023.
- [14] B. Potteiger, Z. Zhang, and X. Koutsoukos, "Integrated moving target defense and control reconfiguration for securing cyber-physical systems," *Microprocess, Microsystems*, vol. 73, p. 102954, 2020. <https://doi.org/10.1016/j.micpro.2019.102954>
- [15] P. Wang, J. Bangert, and C. Kern, "If it's not secure, it should not compile: Preventing DOM-based XSS in large-scale web development with API hardening," presented at the 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE). <https://doi.org/10.1109/ICSE43902.2021.00123>, 2021.
- [16] J. Carrillo-Mondéjar, H. Turtiainen, A. Costin, J. L. Martínez, and G. Suarez-Tangil, "HALE-IoT: Hardening legacy internet of things devices by retrofitting defensive firmware modifications and implants," *IEEE Internet of Things Journal*, vol. 10, no. 10, pp. 8371-8394, 2023. <https://doi.org/10.1109/JIOT.2022.3224649>
- [17] C. Spensky *et al.*, "TRUST.IO: Protecting physical interfaces on cyber-physical systems," presented at the 2020 IEEE Conference on Communications and Network Security (CNS). <https://doi.org/10.1109/CNS48642.2020.9162246>, 2020.