An Efficient Threat Detection Framework for Docker Containers using AppArmor Profile and Clair Vulnerability Scanning Tool

MSc Research Project
Cloud Computing

Teena Kohli
Student ID: 20226845

School of Computing
National College of Ireland

Supervisor: Vikas Sahni

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Teena Kohli
Student ID:	20226845
Programme:	Cloud Computing
Year:	2022
Module:	MSc Research Project
Supervisor:	Vikas Sahni
Submission Due Date:	15/08/2022
Project Title:	An Efficient Threat Detection Framework for Docker Containers using AppArmor Profile and Clair Vulnerability Scanning Tool
Word Count:	3580
Page Count:	17

Signature:	Teena Kohli
Date:	14th August 2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

An Efficient Threat Detection Framework for Docker Containers using AppArmor Profile and Clair Vulnerability Scanning Tool

Teena Kohli
20226845

Abstract

With the increased demand for continuous delivery, faster deployments and cost-saving solutions, containers have gained traction in Cloud Computing. Today, containerisation has become a de facto of the IT industry that leverages various capabilities like orchestration, fault tolerance etc. Over a few years, Docker containers have been widely accepted package solutions across IT and therefore subjected to numerous threats like malware, image vulnerabilities and host exploitations. Thus, this paper marks the contribution toward efficient research on a secure framework for Docker container solution following a Threat Detection-Defence mechanism using AppArmor Custom Profiling and Clair scanner as image vulnerability assessment tool. To evaluate the effectiveness of the suggested framework, publicly available Word-Press's website and its aspects regarding backup plugins' security have been leveraged. The results show that the proposed implementation significantly improves the website's security and restricts the upload of any plugin from outside the world compared to the default security measures.

1 Introduction

Containers are the new essentials of Cloud Computing. Talking briefly about Virtualization, System virtualization was the progressive innovation to solve the problem of underutilized hardware resources by creating a virtualization layer between hardware components and the user allowing them to create a virtual Machine (VM) as virtual computers that can run in multiple on a single set of hardware.In recent years,as an alternative to Virtual Machines, Containers emerged as lightweight alternative to virtualization since they are based on OS virtualization and do not require the entire configuration for their dependant components. Since containers are lighter, easier to deploy, allow version control, and have other benefits over virtual machines (VMs), they are now viewed as a more viable option for microservices. In Fig 1 Sultan et al. (2019)depicts the comparison between container and virtual machine architectural design.

1.1 Motivation and Background

With applications ranging from IoT to smart cars to fog computing, Docker is a widely adopted Containerization concept paving path to Container As A Service (CaaS) in IT

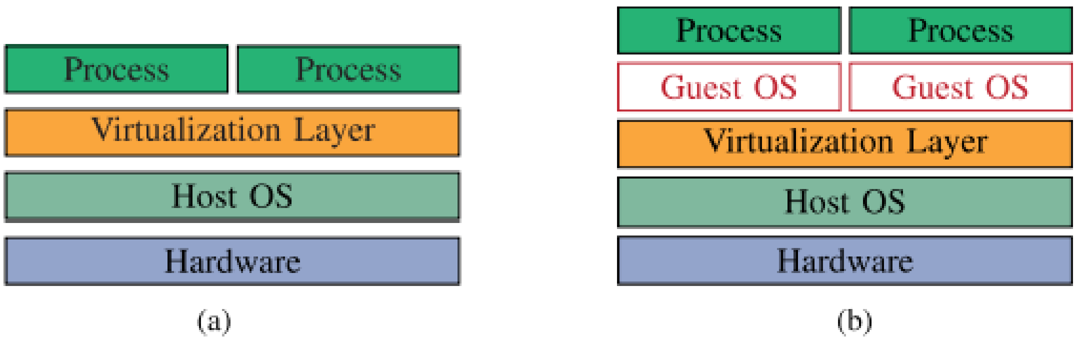


Figure 1: Design Comparison between a)Containers and b)Virtual Machines

industry.However, there is a growing worry that all of the ideas and advances will undermine the current security measures. Are the conventional tools effective enough to defend against contemporary cloud threats, or should a new discipline be established to identify dynamic Cloud weaknesses? Although much work has gone into making containers more secure, containerized environments still have a significant attack surface, which it is considered should be constantly exploited for the benefit of the software industry.

1.2 Research Question

Can the application hosted on docker container be more secure with improved operational capabilities, by implementing an enhanced defense mechanism using AppArmor profiling and Clair image vulnerability assessment tool.

1.3 Research Objective

As quoted by Bruce Schneier [1], a known cryptographer, ”Security is not an end product. It’s a continuous process” and it is to be believed that security and cyberattacks are the two sides of the same coin and are regarded as one of the main sections of cloud computing challenges. Building a strong defense system requires the proper motivation, which can only come from comprehending the threat. It is understood that variety of approaches have already been studied in the past, but the majority have shown to be out-of-date. Therefore a concrete defensive mechanism with the most recent malware detection and unified support should be created for end-users. Additionally, runtime vulnerabilities and a robust tool to enforce customer-required compliance policies to secure data is needed for an hour.

2 Related Work

The main purpose of this section is to provide a theoretical overview of published researches related to Docker container security and threat classification in order to identify

[1]https://www.schneier.com/blog/about/

gaps in existing systems to allow ideas for new innovations.The selection criteria focused on Docker architectural overview,security features, threats,tools and alternative methodologies. The proposed research examines several problem related documents to develop an optimal solution by comparing different results and suggested methods.

2.1 Docker Architecture

Virtualization has been one of the dominant technologies in the hypervisor and container-based categories for the last decade. It is widely adopted by organizations as a modern data center virtualization solution.Bui (2015) sees Docker as a lightweight container alternative to usual virtualization that can be easily integrated with third party tools like Jenkins Ensemble etc. And further divides the term into two main components - Docker Engine and Docker-Hub. Docker Engine is an open source application container technology that follows a client-server API interaction mechanism. Whereas Docker-Hub acts as a central repository for the IT community to develop store and download their code in containers.
However in recent years, Bashari Rad et al. (2017) have proposed a well-defined entry theory by splitting the framework on Docker into different components including Docker images Docker containers and repositories.In the literature review the authors also presented other Docker performance evaluation experiments that includes similar CPU performance where compact iterations also raise security concerns due to direct access to untrusted kernel images etc powerful enough to investigate whether modern packaged approaches are safe from cyberthreats.

2.2 Docker Security Overview and Threats classification

Following the docker container based approach, Combe et al. (2016) proposed the theory based on docker usage and associated security challenges. The Docker daemon guarantees isolation in a typical Docker host using Kernel capabilities and a network security concept. Researchers have also described the seriousness of threats in place of docker usage, noting that docker assaults are not restricted to the docker process or hosted container. The host operating system, inter-container communication, the network, and image registries are all listed as being vulnerable to assaults. Although a different defense for safeguarding the Docker development lifecycle was presented, no comprehensive remedy was suggested.Because containers are the first thing an attacker will go at, they carry a wide range of risks.
Jian and Chen (2017) looked at the details of the Docker escape assault while highlighting the namespace, capabilities, and seccomp CGroups features of the Linux kernel. They then presented a secured technique based on namespace expansion.
Anticipating other attacks,Luo et al. (2016) conducted a test in which they asserted that, contrary to Red Hat documentation [2], the default Dockersystem lacks the appropriate method to configure and restrict the system calls. With specific capabilities and fine-grained access, Seccomp, a way to limit system calls to the kernel, is more likely to be safe since it restricts access to the system directly, reducing the attack surface.The use case of a DOS assault is another crucial scenario appropriate for host-container or inter-container protection. Understanding the other Kernel feature's capability: Cgroup

[2]https://access.redhat.com/documentation/enus/red_hat_enterprise_linux_atomic_host/7/html/container_security_guide/linux_capabilities_and_seccomp

is in charge of restricting the resources in a container, preventing the container from using system resources in an unethical manner by limiting the container's capacity to access those resources. However, memory DOS attacks are conceivable, according to experiments by Chelladhurai et al. (2016) and Chen et al. (2019), necessitating additional practical developments.

2.3 Docker security AppArmor profile generator

Containers operate directly on the host kernel in contrast to virtual machines, which incorporate their kernel. Containers are lightweight because the kernel and kernel-sharing property are absent. One issue is that containers aren't as safe as virtual machines due to the kernel-sharing nature.
An initial theory proposed by Mattetti et al. (2015) referred to LSM tool called LiCShield,framework for securing of linux containers and their workloads via automatic construction of rules describing the expected activities of containers spawned from a given image. However the author themselves iterated in their conclusion as LicSheild cannot be termed as derieving force as it only stems from the based of MAC (mandatory access control) and doesn't not protect against network intrusion etc and therefore can be best suited as complimentary tool with other competing technology.
In conjunction to same theory, Docker-sec is a recently suggested Linux Security Module (LSM) by Loukidis-Andreou et al. (2018). Limiting capabilities and network accesses inside containers, Docker-sec is a user-friendly security module based on AppArmor that secures Docker containers during their entire lifecycle. To be more precise, Docker-sec gives users the option to automatically create initial container profiles from configuration data supplied during container initialization. The authors of Docker-sec demonstrated that the suggested LSM is effective at shielding containers from zero-day vulnerabilities with little performance overhead.
Additonally, MP et al. (2016)have also demonstrated through their research that improving container security increases Docker security. Any LSM can be integrated using appropriate profiles for on-premise secure cloud orchestration and deployments to achieve this. On-premises deployments error can be avoided for standard Debian-based distributions like Ubuntu, Debian, etc., by customizing AppArmor profile.

2.4 Security Evaluation of Wordpress Backup Plugins

For every organization, protecting the online applications is a crucial duty. The most popular content management system among businesses is WordPress.Cernica and Popescu (2018) clearly stated on blogging function of wordpress which draws attackers looking for security flaws. Finding issues is made simpler by examining the installed plugins and themes rather than the source code of the core WordPress program.
Koskinen et al. (2012) in their research affirmed that although some of plugins are following security bestpractises however Only one vulnerability in one of those plugins or themes might result in a partial compromise of the operating system and the leakage of all data from Wordpress. This can result in ideal situation for attackers is one in which they may compromise more servers by taking advantage of other network systems and so amass more data.
Cernica et al. (2019) through security evaluation research of Wordpress plugin they confirmed that majority of backup plugins don't use robust cryptographic techniques for

creating backup names.These backup plugins exploit time to build file names, endangering the confidential information on Wordpress website sent. Their experiment clearly demonstrated that all the exposures made on the backup plugins might have a significant effect on Wordpress websites and therefore a more secure framework is much required.

2.5 Docker Image Vulnerability tool assessment

Registry and image risk are categorized by risks such as image vulnerabilities, registry insecure connections, untrusted images, and insufficient authentication. For storing and disseminating Docker images, a stateless, scalable server-side application called the registry is employed. Docker advises using a secure version of Registry over TLS, however numerous experiments have been carried out to make use of a secure approach. According to Jain et al. (2021), the open source Docker Image tool was mostly used for static image scanning. With a theoretical understanding of the picture scanning mechanism, the study demonstrated theory on 10 open source image scanning tools. However, the paper's scope was extremely limited with the only option of static scanning.
A docker image verification Framework (DIVA) based on the Clair image vulnerability detector tool was suggested by Shu et al. (2017) in contrast. Researchers reported that after analyzing 356.218 photos, 90% of the images had significant vulnerabilities, including trustworthy images that were heavily promoted from parent to child image distribution.
Martin et al. (2018) present the docker ecosystem of vulnerabilities, using a top-down approach to categorize the vulnerabilities in five categories, despite the fact that various workarounds have been discovered for the vulnerabilities. The exploitation and suggested remedies were also clearly outlined by the model. Threats might also affect the package management. Therefore, Docker provided shared trust between the client and registry; but, due to technical difficulties, the theory didn't gain any momentum in the developer community.
An interesting report on evaluating container vulnerability tools is presented by Javed and Toor (2021) based on actual implementation, where the quality of scanners is based on their accuracy and vulnerabilities coverage. The tools available on the market, such as Clair, Anchore, Microscanner, etc., claim to identify OS and nonOS Vulnerabilities; however, according to an experiment done for the report, detection coverage for all of them was surprisingly low. For this reason, the authors encourage researchers to focus their efforts on improving and developing better container vulnerability detection tools.

2.6 Research Niche

Containers are the advanced technology for elastic and scalable cloud computing applications as cloud applications are subject to dynamic workloads.This paradigm shift in Industry hopes to be a highly parallel, event-driven and fast. Various Experiences and developments therefore provide opportunities for safer practices. There are lot of research on container/docker security, based on framework model guidelines and various vulnerabilities however as Security is a constantly evolving process, comprehensive analysis alone is not enough.Therefore, it should always be reviewed.This literature review included both academic and industry research with the aim of covering all theoretical and practical applications of container security and defense framework. Upon focusing on some of the recent research like side channel attacks Sprabery et al. (2018), orches-

Author	Defense Mechanism	Host Inspection	Application Inspection
Jian and Chen (2017)	dynamic detection of status namespace inspection	YES	NO
Shu et al. (2017)	Image Vulnerability scanning framework, studied 356,218 images from dockerhub	NO	YES
Javed and Toor (2021)	Different Image Vulnerability Tools are compared based on the defects they cover and how accurate they are	NO	YES
Proposed Framework	Threat Detection Defence Framework	Yes	Yes

Table 1: Summary of Literature Review and Research Niche

tration secure mechanism Vaucher et al. (2018) and vulnerability tools comparisonJaved and Toor (2021) the intention was to present an endorsed theory for a comprehensive understanding of Docker security and threats exemplifications .Table 1 summarizes previous research solutions and how the suggested framework in 4 will make the contribution to enhanced secure mechanism for containerized application security.

In Table 1 we therefore represents The proposed container solution and defense mechanism for vulnerability coverage summarizes the research based on its objectives.

3 Methodology

Security is a constant endeavor until it stabilizes the process and not just the choice. Despite the availability of numerous vulnerability tools and various mechanisms as discussed in section 2.5 despite the various measures that are accessible, some are either outdated or ineffective. This research proposes a finite elemental model following an enhanced defence system for containerized applications unlike limited static assessments from past. A step-wise complete software life cycle secure process from container build to deployment, the recent trend of DevOpsSec inspires this approach secures the process through the entire life cycle development-to-deployment phase. With the enforcement of custom-based rules and the proper permission, the model will a) restrict the host access policies for the containerisation deployment process by coding AppArmor Profile for container defence mechanism and b) vulnerability assessment for Docker images to ensure enhanced security from any possible exploitation of OS/non-OS packages and finally c)deployed on AWS instance, accessible publicly.

3.1 Tools and Techniques used

1.Amazon Ubuntu Platform - A t2.micro free tier eligible AMI built on Ubuntu with an expanded EBS volume was used for the project installation. Canonical continuously monitors and updates Ubuntu images to ensure stability and security for workstations and containers run, which is the purpose behind choosing the Ubuntu platform.

2.AppArmor The default Kernel Security Module, AppArmor, restricts what programs can read and write to the system and network shares. AppArmor uses MAC as its foundation (mandatory access control). Custom profile codes can be loaded, but the application profile still determines the authorisation. Additionally, AppArmor Profile includes two modes: complain and enforced. When AppArmor is in enforce mode, it actively blocks and logs in dmesg anything beyond the parameters of the docker default profile, whereas complain mode means it will only log messages to dmesg, which is outside the boundary of the Apparmor profile attached[3].

3.Vulnerability scanning Tool After exploring different tool and techniques from research section 2.6, commercial tools like Prisma Cloud had to be discarded as these tools were not available for the trial version. Many of the features are also not provisioned outside the Licensed version. Clair has thus been employed as a vulnerability assessment tool in this research. It is an open-source tool from CoreOS that examines container images for known security flaws. Clair has primarily been used to scan images in CoreOS's proprietary container registry, Quay.io. However, it can investigate Docker images as well.The evaluation has been divided into two case experiments to evaluate the proposed framework. a) One with Standalone static image - Nginx Docker image for Static Host vulnerability. This case experiment is inspired by the use case of protecting the host from containers exemplified in Sultan et al. (2019). Through malicious containers, attackers can aim for Host resources; therefore, it is crucial to disable the container's capacity to attack the host components. The second case experiment is b) WordPress Docker composes application plugin vulnerability. This case experiment is inspired by the WordPress plugin vulnerability, which led to the Panama Paper leak attack.

4.Nginx and WordPress Docker Image - The evaluation has been divided into two case experiments to evaluate the proposed framework. a) One with Standalone static image - Nginx Docker image for Static Host vulnerability. This case experiment is inspired by the use case of protecting the host from containers exemplified in Sultan et al. (2019). Through malicious containers, attackers can aim for Host resources; therefore, it is crucial to disable the container's capacity to attack the host components. The second case experiment is b) WordPress Docker composes application plugin vulnerability. This case experiment is inspired by the WordPress plugin vulnerability, which led to the Panama Paper leak attack.[4].

3.2 Proposed strategy in general Structure

Figure 2.presents the research methodology based on collective steps of Development, Testing and Deployment, where the Development phase is AppArmor Program Profile Coding to implement advanced Docker security. First, the installed AWS ubuntu Host shipped with the default version of AppArmor is set up. Then, AppArmor's profile is coded based on the case study and requirement to ascertain the files and permissions

[3]https://docs.docker.com/engine/security/apparmor/

[4]https://www.wordfence.com/blog/2016/04/panama-papers-wordpress-email-connection/

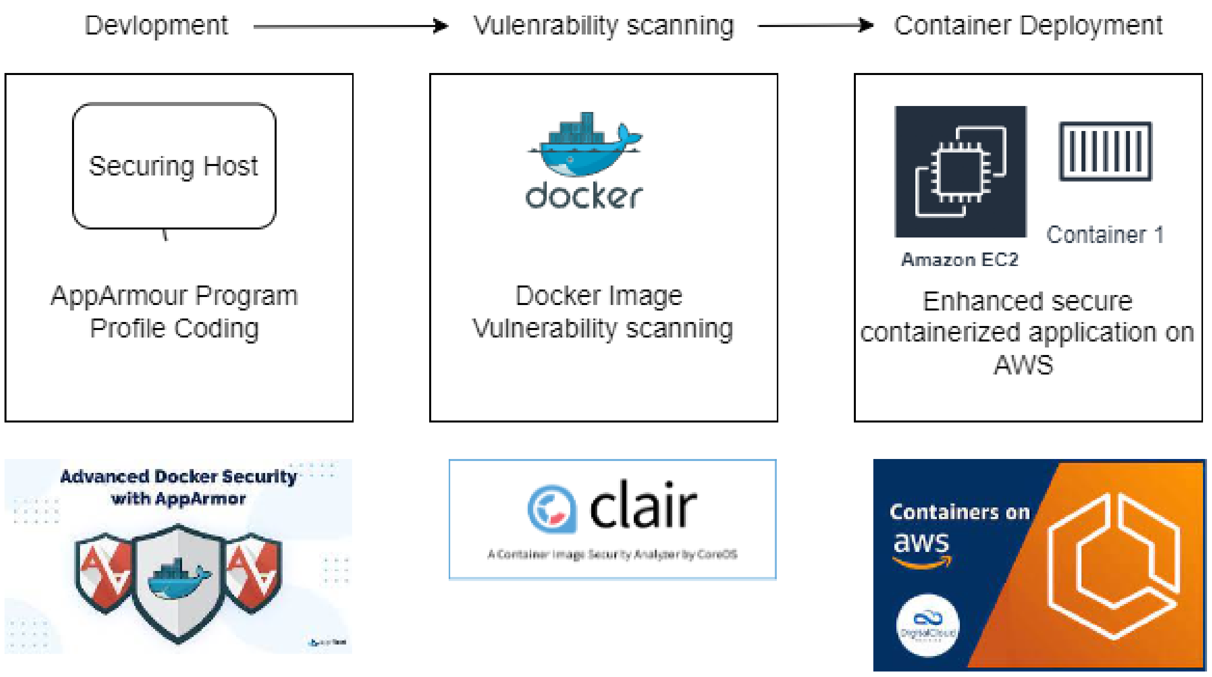

Devlopment
Vulenrability scanning
Container Deployment
Securing Host
AppArmour Program Profile Coding
docker
Docker Image Vulnerability scanning
Amazon EC2
Container 1
Enhanced secure containerized application on AWS
Advanced Docker Security with AppArmor
clair
A Container Image Security Analyzer by CoreOS
Containers on aws

Figure 2: Enhanced Secure Defense Framework

necessary for the experimental evaluation.

In the second step, For the vulnerability detection phase, a series of pre-requisites installation on Ubuntu host, Docker latest community version, and Docker compose to be installed for Clair configuration. A PostgreSQL back-end server, a middleware server, and an end-user tool — in this case, clairctl, a command-line interface checking tool, make up the Clair system ready for our implementation.

In the third step, aims to bring the application up by attaching AppArmor Profile with respective containers. The AppArmor profile needs to be loaded into the system and added to the application docker-compose.yaml for it to be in enforced mode for auditing the security behaviour. For Nginx Docker container evaluation through console logging and WordPress Application deployed on ec2 instance, publicly accessed, is then tested against the plugin upload vulnerability and measured against the suggested framework claims.

4 Design Specification

Figure 3 describes the detailed design of the Threat Detection-Defence Framework. The diagram and the associated definitions are intended as an aid in clarifying and presenting the principles that underlie the implementation and the related requirements in the following manner: -

1. Custom AppArmor Profiling - The overarching approach of any defence mechanism is the ability to protect hosts by extending default rules and policies to provide a secure foundation and stricter rules during dynamic monitoring. The host configuration-based approach proposed through this research is a new security mechanism that creates an additional layer of security compared to Docker's default using AppArmor, which leverages a custom profile as an additional layer of security. Usually, Apparmor starts

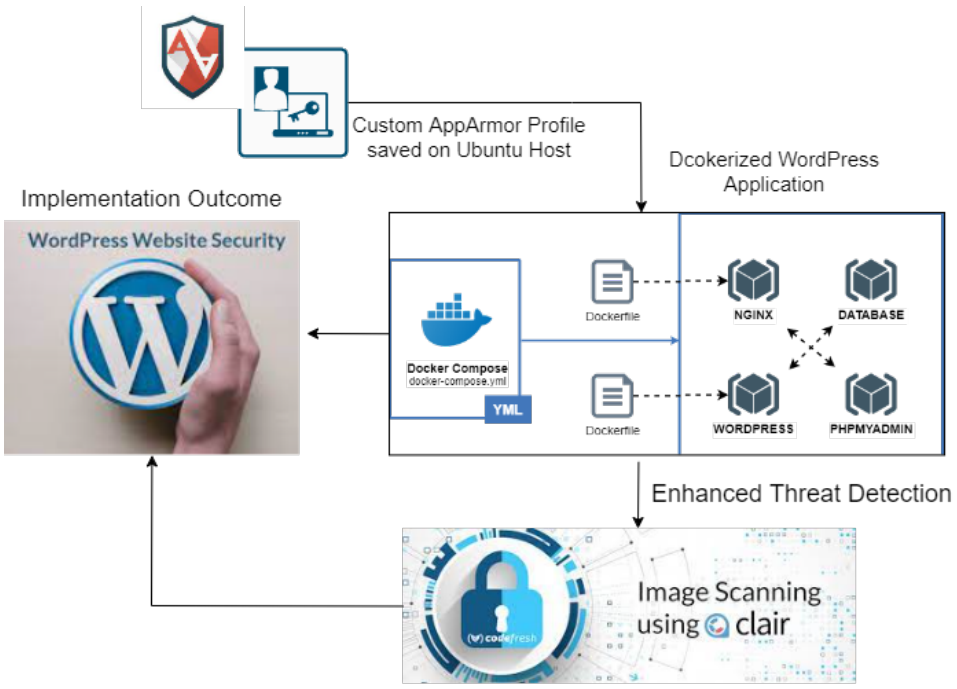

Custom AppArmor Profile saved on Ubuntu Host
Dcokerized WordPress Application
Implementation Outcome
WordPress Website Security
Docker Compose
docker-compose.yml
YML
Dockerfile
Dockerfile
NGINX
DATABASE
WORDPRESS
PHPMYADMIN
Enhanced Threat Detection
Image Scanning using clair

Figure 3: Threat Detection-Defence Framework WordPress Application Design

with docker daemon under default profile protection but still no protection and only for specific permissions. That is why AppArmor further and extensively expands the ability in the runc profile code to create a separate secure profile before the initial receiver runs any process. This mechanism defines a set of privileges and capabilities required for a Docker container to be a secure automated container.

```
sudo apt install apparmor-profiles --Installation
sudo apparmor-status -- Status
sudo apparmor_parser -r /etc/apparmor.d/profile.name - To load new profile
```

Options considered for AppArmor profile customisation are denied rules limiting HOST file's use. Since under AppArmor, they take precedence over allow rules and access control rules - R, W, M, K, L, which stand for reading, write, memory map as executable, and lock files (creation hard links) implemented on files accordingly. Although additional customised rules may also be applied depending on the situation, the research implementation's primary focus is encoding mount and network paths for secure access.

```
deny mount,
deny /sys/kernel/security/** rwklx,
deny /etc/** wl,
deny @{PROC}/* w,
# deny write to files not in /proc/<number>/** or /proc/sys/**
```

2. Dockerized Clair Scanner, Nginx and WordPress Application- A dedicated Clair server are generally required for the Clair scanner to function. Since having a dedicated Clair server is not practical for the research case, Al's alternative and robust solution is to run the tool as an independent docker container. A prefiled database and

Clair docker image have been fetched from the official Github link, and a Clair scanner with local DB are being used for this job.

```
docker run -p 5432:5432 -d --name db arminc/clair-db:latest
docker run -p 6060:6060 --link db:postgres -d --name clair
/clair-local-scan:latest
```

Similarly, an Nginx official image has been pulled through the docker pull command from its central repository. Finally, the WordPress application is executed using docker-compose, a multi-container application running WordPress, PHP and mysl database container as one multi-container docker application for the case experiment.

5 Implementation

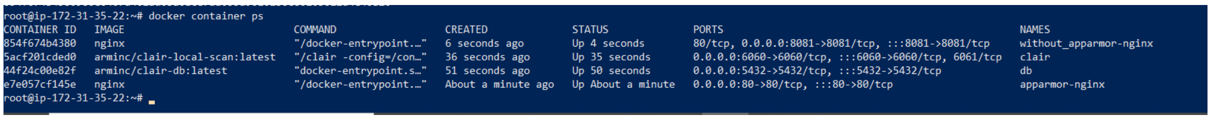
Threat Detection Framework is a model where different lines of defence work are applied together to enhance the overall defensive capability of the application. Based on the experimental study of containers and their perspective of a sensitive data leak, also emphasising the vulnerabilities of Applications, the implementation aims to represent two different case studies: 1)Host Protection against Containers and 2) Application Protection against attacks. Docker uses many Linux technologies such as AppArmor seccomp and capabilities to build a robust defence system. An Apparmour profile is built a top of, with AppArmor enforced mode. It employs two mechanisms, a) When a container is created, an initial static access rule is built based on the container configuration parameters, and b)The initialisation package has been modified to more tightly constrain the capabilities of the container to reflect the actual functionality of the application when the container is running. Clair Scanner's use for improved threat detection complemented security assessment of static vulnerabilities on Hosted Distribution and images.
With the proposed architecture following are the steps followed for the case-wise implementation.

- **a) Protect and restrict the use of Nginx docker container in order to increase the security of host resources** - The first step in putting the first case study into practice is to pull a Docker image using Docker Pull.
  In the second stage, the vulnerability assessment report is produced using a command like the program Clair-scanner using the Clair scanning tool.
  Finally, an Apparmor profile has been generated and afterwards attached while initialising the Nginx image docker for evaluation purposes based on vulnerability identification from the report for the last phase.
  Fig4 demonstrate Clair,Clair-local-db and 2 Nginx containers with AppArmor and witout AppArmor, respectively, concluding the system is prepared for evaluation.

- **b) Protect Dockerized WordPress Application from the malacious Plugin upload attack vector using AppArmor Profile**-As the first step to creating a multi-container WordPress application, two container services have been created in docker-compose yaml, which includes WordPress's latest image container, MySQL DB container and related environment variables[5].
  In the second step, an AppArmor Profile for WordPress application is created to

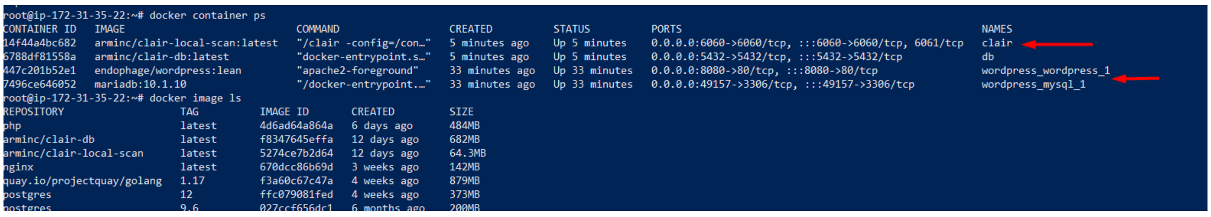
[5]https://hub.docker.com/$_/mysql$

deny any plugin upload from the internet.
In the final step, for Apparmor to attach to WordPress docker-compose, the security option parameter has been defined in yaml. Once the WordPress docker-compose is up, the security option mentioned in yaml will automatically load the AppArmor Profile to the application, thus making it ready for evaluation.
Fig 5 that Clair, Clair-local-db and WordPress multi-containers are running. A concluding system is prepared for evaluation. Fig 6 represents deployment of WordPress website on public AWS host from Cloud.ncirl.ie, ready for evaluation in upcoming section 6.2



```
root@ip-172-31-35-22:~# docker container ps
CONTAINER ID   IMAGE                            COMMAND                  CREATED              STATUS              PORTS                                                  NAMES
854f674b4380   nginx                            "/docker-entrypoint.…"   6 seconds ago        Up 4 seconds        80/tcp, 0.0.0.0:8081->8081/tcp, :::8081->8081/tcp      without_apparmor-nginx
5acf201cded0   arminc/clair-local-scan:latest   "/clair -config=/con…"   36 seconds ago       Up 35 seconds       0.0.0.0:6060->6060/tcp, :::6060->6060/tcp, 6061/tcp    clair
44f24c00e82f   arminc/clair-db:latest           "docker-entrypoint.s…"   51 seconds ago       Up 50 seconds       0.0.0.0:5432->5432/tcp, :::5432->5432/tcp              db
e7e057cf145e   nginx                            "/docker-entrypoint.…"   About a minute ago   Up About a minute   0.0.0.0:80->80/tcp, :::80->80/tcp                      apparmor-nginx
root@ip-172-31-35-22:~#
```

Figure 4: Docker Container for Clair and Nginx Application



```
root@ip-172-31-35-22:~# docker container ps
CONTAINER ID   IMAGE                            COMMAND                  CREATED          STATUS          PORTS                                                  NAMES
14f44a4bc682   arminc/clair-local-scan:latest   "/clair -config=/con…"   5 minutes ago    Up 5 minutes    0.0.0.0:6060->6060/tcp, :::6060->6060/tcp, 6061/tcp    clair
6788df81558a   arminc/clair-db:latest           "docker-entrypoint.s…"   5 minutes ago    Up 5 minutes    0.0.0.0:5432->5432/tcp, :::5432->5432/tcp              db
447c201b52e1   endophage/wordpress:lean         "apache2-foreground"     33 minutes ago   Up 33 minutes   0.0.0.0:8080->80/tcp, :::8080->80/tcp                  wordpress_wordpress_1
7496ce646052   mariadb:10.1.10                  "/docker-entrypoint.…"   33 minutes ago   Up 33 minutes   0.0.0.0:49157->3306/tcp, :::49157->3306/tcp            wordpress_mysql_1
root@ip-172-31-35-22:~# docker image ls
REPOSITORY                  TAG       IMAGE ID       CREATED         SIZE
php                         latest    4d6ad64a864a   6 days ago      484MB
arminc/clair-db             latest    f8347645effa   12 days ago     682MB
arminc/clair-local-scan     latest    5274ce7b2d64   12 days ago     64.3MB
nginx                       latest    670dcc86b69d   3 weeks ago     142MB
quay.io/projectquay/golang  1.17      f3a60c67c47a   4 weeks ago     879MB
postgres                    12        ffc079081fed   4 weeks ago     373MB
postgres                    9.6       027ccf656dc1   6 months ago    200MB
```

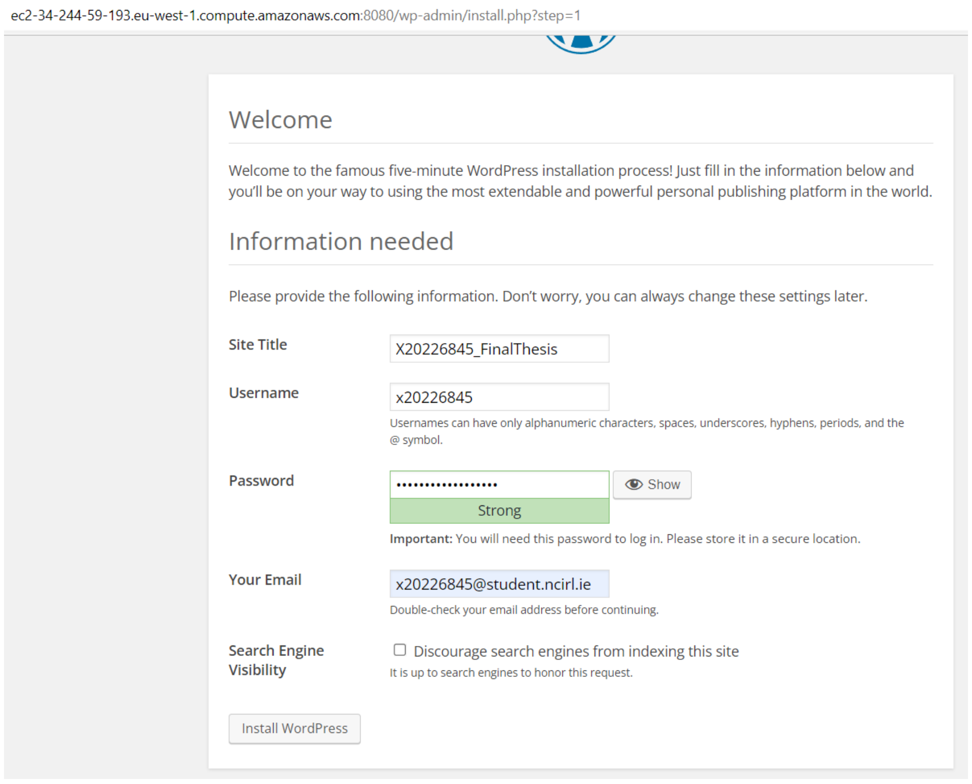
Figure 5: Docker Container for Clair and WordPress Application

6 Evaluation

This section represents the effects of putting the suggested Framework into practice through various case studies and experiments.All of the container image samples used for study were obtained from Docker hub [6], a service specialising in hosting private Docker repositories and official/unofficial images, serving as the largest repository and community for container images in the world. At first, using Clair Scanner, all pre-installed images are scanned to find all the possible exposures through Clair's local CVE database. To facilitate the assessment, the vulnerabilities detected have been classified into category High, Medium and low vulnerabilities of container images. Next and most crucial stage of evaluation, the custom AppArmor profile coded will be attached to the following:

- Case 1- Nginx standalone Docker image.
- Case 2- WordPress Application DockerCompose, where application wraps Apache PHP and mysql into a multi-docker container solution.

[6]https://hub.docker.com/


ec2-34-244-59-193.eu-west-1.compute.amazonaws.com:8080/wp-admin/install.php?step=1

Welcome

Welcome to the famous five-minute WordPress installation process! Just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.

Information needed

Please provide the following information. Don't worry, you can always change these settings later.

Site Title: X20226845_FinalThesis

Username: x20226845

Usernames can have only alphanumeric characters, spaces, underscores, hyphens, periods, and the @ symbol.

Password: •••••••••••••••••• Show

Strong

Important: You will need this password to log in. Please store it in a secure location.

Your Email: x20226845@student.ncirl.ie

Double-check your email address before continuing.

Search Engine Visibility: ☐ Discourage search engines from indexing this site

It is up to search engines to honor this request.

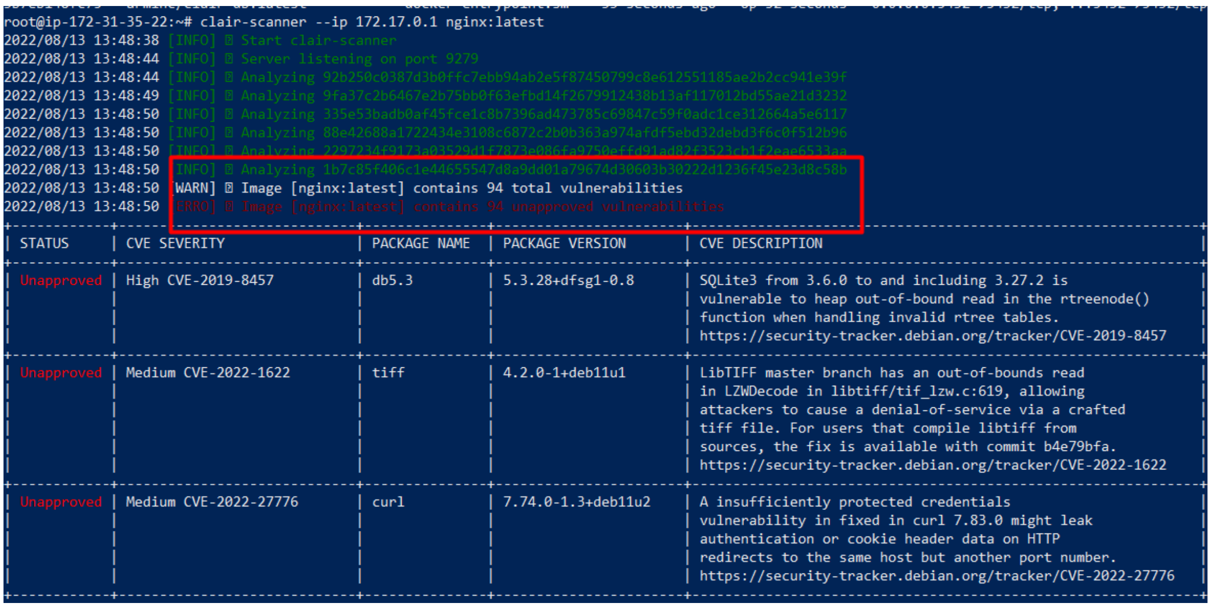
Figure 6: WordPress Application Installation on AWS host

6.1 Experiment / Case Study 1

For the first case example standalone Nginx docker image pulled from docker hub is considered where we a custom AppArmor profile for Nginx secure container solution is designed based on one of the many vulnerability's being detected.With the aim to protect host from container, the AppArmor profile is designed in a manner to limit network capabilities, mounting of resources and restricting shell operation for this experiment purpose.Fig 7 represents the Clair scanner analysis running as local analyzing different layers of Docker image. The vulnerability report has been saved using -r option for later references.

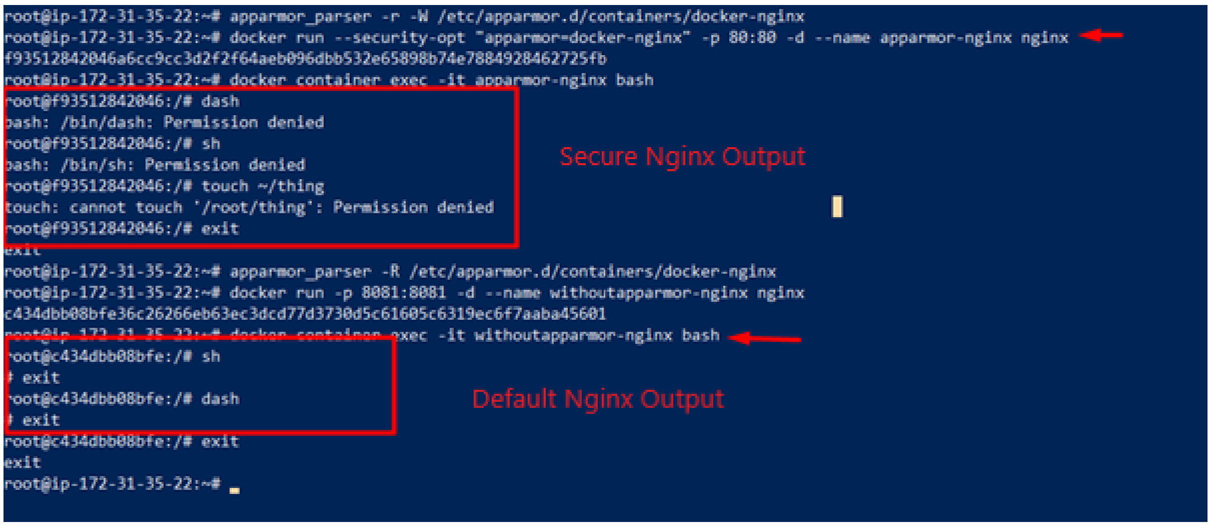
Based on vulnerability assessment, considering Non-OS package vulnerability such as shell access which can result in a Man-In-Middle attack[7], a custom AppArmor profile is been created. Figure 8 represents the activation of the AppArmor profile into OS through the AppArmor parser command line and running an AppArmor-nginx secure container. The figure further depicts that even if the user can launch a bash session into the container is still not allowed to perform the host-related command like dash, touch and get permission denied in comparison to a non-secure Nginx container where no restrictions are to be seen. Therefore, through this evaluation, this is to be confirmed that AppArmor profiling provides secure capabilities that can be customised for any process as per the requirement. Be it a docker daemon, application functionality or non-os capabilities, with a modified version of AppArmor, the implementer can restrict access exclusively to any resources/binaries from outside intrusion such as network attack, denying file system mounting, etc.

[7]https://security-tracker.debian.org/tracker/CVE-2022-32208



```
root@ip-172-31-35-22:~# clair-scanner --ip 172.17.0.1 nginx:latest
2022/08/13 13:48:38 [INFO] □ Start clair-scanner
2022/08/13 13:48:44 [INFO] □ Server listening on port 9279
2022/08/13 13:48:44 [INFO] □ Analyzing 92b250c0387d3b0ffc7ebb94ab2e5f87450799c8e612551185ae2b2cc941e39f
2022/08/13 13:48:49 [INFO] □ Analyzing 9fa37c2b6467e2b75bb0f63efbd14f2679912438b13af117012bd55ae21d3232
2022/08/13 13:48:50 [INFO] □ Analyzing 335e53badb0af45fce1c8b7396ad473785c69847c59f0adc1ce312664a5e6117
2022/08/13 13:48:50 [INFO] □ Analyzing 88e42688a1722434e3108c6872c2b0b363a974afdf5ebd32debd3f6c0f512b96
2022/08/13 13:48:50 [INFO] □ Analyzing 2297234f9173a03529d1f7873e086fa9750effd91ad82f3523cb1f2eae6533aa
2022/08/13 13:48:50 INFO] □ Analyzing 1b7c85f406c1e44655547d8a9dd01a79674d30603b30222d1236f45e23d8c58b
2022/08/13 13:48:50 WARN] □ Image [nginx:latest] contains 94 total vulnerabilities
2022/08/13 13:48:50 ERRO] □ Image [nginx:latest] contains 94 unapproved vulnerabilities
+------------+-----------------------------+----------------+------------------------+-------------------------------------------------------------+
| STATUS     | CVE SEVERITY                | PACKAGE NAME   | PACKAGE VERSION        | CVE DESCRIPTION                                             |
+------------+-----------------------------+----------------+------------------------+-------------------------------------------------------------+
| Unapproved | High CVE-2019-8457          | db5.3          | 5.3.28+dfsg1-0.8       | SQLite3 from 3.6.0 to and including 3.27.2 is               |
|            |                             |                |                        | vulnerable to heap out-of-bound read in the rtreenode()     |
|            |                             |                |                        | function when handling invalid rtree tables.                |
|            |                             |                |                        | https://security-tracker.debian.org/tracker/CVE-2019-8457   |
+------------+-----------------------------+----------------+------------------------+-------------------------------------------------------------+
| Unapproved | Medium CVE-2022-1622        | tiff           | 4.2.0-1+deb11u1        | LibTIFF master branch has an out-of-bounds read             |
|            |                             |                |                        | in LZWDecode in libtiff/tif_lzw.c:619, allowing             |
|            |                             |                |                        | attackers to cause a denial-of-service via a crafted        |
|            |                             |                |                        | tiff file. For users that compile libtiff from              |
|            |                             |                |                        | sources, the fix is available with commit b4e79bfa.         |
|            |                             |                |                        | https://security-tracker.debian.org/tracker/CVE-2022-1622   |
+------------+-----------------------------+----------------+------------------------+-------------------------------------------------------------+
| Unapproved | Medium CVE-2022-27776       | curl           | 7.74.0-1.3+deb11u2     | A insufficiently protected credentials                      |
|            |                             |                |                        | vulnerability in fixed in curl 7.83.0 might leak            |
|            |                             |                |                        | authentication or cookie header data on HTTP                |
|            |                             |                |                        | redirects to the same host but another port number.         |
|            |                             |                |                        | https://security-tracker.debian.org/tracker/CVE-2022-27776  |
+------------+-----------------------------+----------------+------------------------+-------------------------------------------------------------+
```

Figure 7: Nginx Vulnerabilities Report

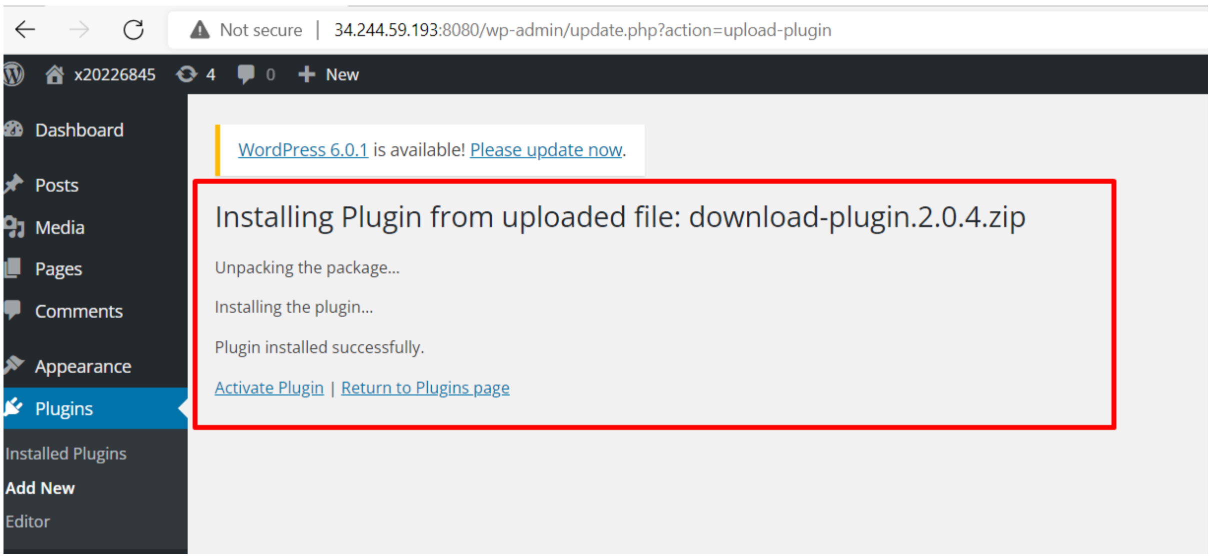


```
root@ip-172-31-35-22:~# apparmor_parser -r -W /etc/apparmor.d/containers/docker-nginx
root@ip-172-31-35-22:~# docker run --security-opt "apparmor=docker-nginx" -p 80:80 -d --name apparmor-nginx nginx
f93512842046a6cc9cc3d2f2f64aeb096dbb532e65898b74e7884928462725fb
root@ip-172-31-35-22:~# docker container exec -it apparmor-nginx bash
root@f93512842046:/# dash
bash: /bin/dash: Permission denied
root@f93512842046:/# sh
bash: /bin/sh: Permission denied
root@f93512842046:/# touch ~/thing
touch: cannot touch '/root/thing': Permission denied
root@f93512842046:/# exit
exit
root@ip-172-31-35-22:~# apparmor_parser -R /etc/apparmor.d/containers/docker-nginx
root@ip-172-31-35-22:~# docker run -p 8081:8081 -d --name withoutapparmor-nginx nginx
c434dbb08bfe36c26266eb63ec3dcd77d3730d5c61605c6319ec6f7aaba45601
root@ip-172-31-35-22:~# docker container exec -it withoutapparmor-nginx bash
root@c434dbb08bfe:/# sh
# exit
root@c434dbb08bfe:/# dash
# exit
root@c434dbb08bfe:/# exit
exit
root@ip-172-31-35-22:~# _
```

Figure 8: AppArmor-Nginx vs Defaul Nginx container solution

6.2 Experiment / Case Study 2

This assessment serves as a significant case study experiment for the defense in depth approach. In this phase, audience can discover how AppArmor may safeguard a Docker containerized Application even when seccomp and Capabilities of underlying host are ineffective. After the WordPress application is up through docker-compose following implementation step 5, login to WordPress application and upon clicking on install new Plugin users can notice that default-docker will not restrict the plugin installation, which resonates the plugins vulnerability and before mentioned famous Panama Papers incident [8].Fig 9 shows the successful installation of plugin without any restriction even if Docker-Default capabilities were enabled but has implied no restrictions.



Not secure | 34.244.59.193:8080/wp-admin/update.php?action=upload-plugin

WordPress 6.0.1 is available! Please update now.

Installing Plugin from uploaded file: download-plugin.2.0.4.zip

Unpacking the package...

Installing the plugin...

Plugin installed successfully.

Activate Plugin | Return to Plugins page

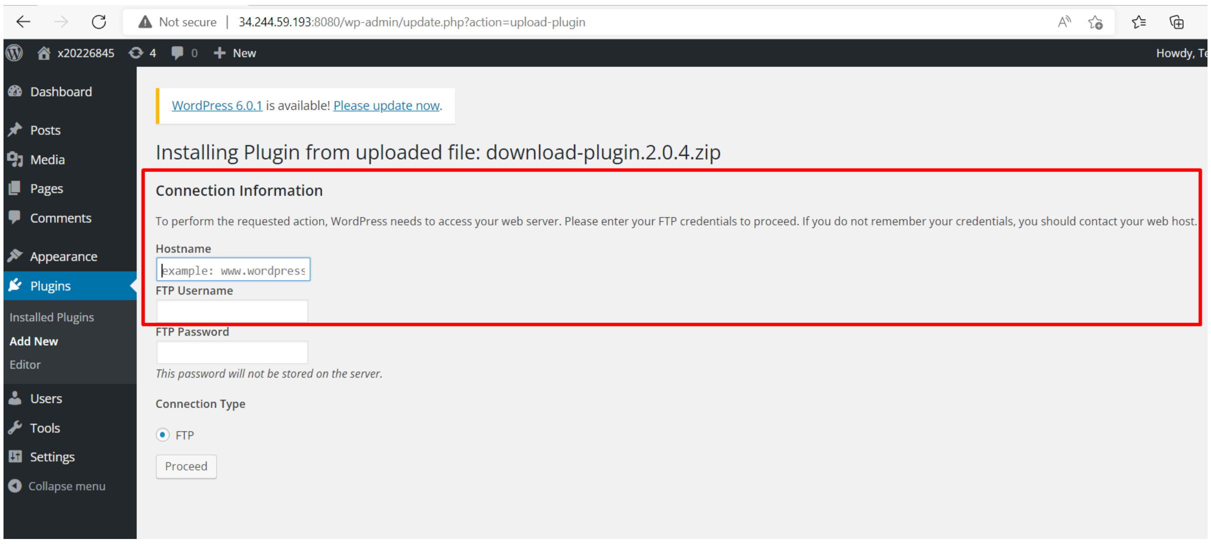
Figure 9: WordPress Plugin installaed withou any restrictions

Whereas in second step, upon adding wparmor profile to Application there will be restrcition imposed on plugin upload by adding it to Deny rule. Bringing the app up users can verify that it prompts to upload via FTP Fig 10 (WordPress prompts to upload through FTP when the standard way for doing so fails; this is evidence that the AppArmor profile has been successful).

6.3 Discussion

The case experiment evaluation in this study demonstrates that the default docker security measures are insufficient to estimate the risks. Docker alone could not shield the containerised application from different nefarious attempts, even with the isolation of Linux capabilities and namespaces. Therefore, more advanced mechanisms are needed, such as AppArmor profiles, which can limit other access, such as network access and OS kernel access, depending on the application's needs. This study primarily examines two cases: container to host protection and application host protection, and observed that the suggested framework provides a solid foundation for static and dynamic threat detection.

[8]https://www.infoworld.com/article/3053654/sloppy-patching-insecure-plugins-made-panama-papers-leak-possible.html


Not secure | 34.244.59.193:8080/wp-admin/update.php?action=upload-plugin

WordPress 6.0.1 is available! Please update now.

Installing Plugin from uploaded file: download-plugin.2.0.4.zip

Connection Information

To perform the requested action, WordPress needs to access your web server. Please enter your FTP credentials to proceed. If you do not remember your credentials, you should contact your web host.

Hostname

example: www.wordpress

FTP Username

FTP Password

This password will not be stored on the server.

Connection Type

● FTP

Figure 10: WordPress Aplication Plugin Failed with AppArmor restriction

However, considering real-world scenarios and many y 0day kernel vulnerabilities in the underground world, there is the possibility that the framework may not be enough. Therefore, along with DevOpsSec and automation, it is essential to defend the entire SDLC from the beginning. During the proposal implementation, we identified some gaps in the accuracy of the vulnerability detection tool. We considered that 100% effectiveness could not be achieved by simply using one static vulnerability tool. Also, with the paradigm shift to one-click deployment, It can be suggested that the existing research can be improved by automating the process.

7 Conclusion and Future Work

Cybersecurity comes down to three basic principles - Detect, Protect and Respond. Though many uses cases in past have illustrated security measures for the container, it is believed that its a continuous process and should be periodically reviewed as the dynamics of IT are changing quickly. Although the microservices-based approach is gaining traction, the only barrier to widespread adoption of containers is its security concern, where it is argued that containerisation is not a protective technology. In this paper, an analysis has been conducted to measure the operational behaviour of Docker containers to provide an enhanced defence mechanism using AppArmor Profile and Clair assessment tool. Based on the findings, it can be concluded that the AppArmor profile utilises the virtue of security capabilities which can be tailored per the program profile and for the overall effectiveness of the proposed framework. Clair scanning tool has been proven beneficial for detecting OS/non-OS package vulnerabilities. The suggested strategy effectively ensures the security and dependable performance of containerised applications. With this approach, developers can easily detect the security threats and can thus rectify the images for more secure content.
Future research extending this study may focus on runtime container security and automated vulnerability scanner. Such research can result in a more thorough study of secure frameworks and tool alternatives for a unified container security platform.

References

Bashari Rad, B., Bhatti, H. and Ahmadi, M. (2017). An introduction to docker and analysis of its performance, *IJCSNS International Journal of Computer Science and Network Security* **173**: 8.

Bui, T. (2015). Analysis of docker security.

Cernica, I. and Popescu, N. (2018). Wordpress honeypot module, *2018 IEEE 16th International Conference on Embedded and Ubiquitous Computing (EUC)*, pp. 9–13.

Cernica, I., Popescu, N. and ţigănoaia, B. (2019). Security evaluation of wordpress backup plugins, *2019 22nd International Conference on Control Systems and Computer Science (CSCS)*, pp. 312–316.

Chelladhurai, J., Chelliah, P. R. and Kumar, S. A. (2016). Securing docker containers from denial of service (dos) attacks, *2016 IEEE International Conference on Services Computing (SCC)*, pp. 856–859.

Chen, J., Feng, Z., Wen, J.-Y., Liu, B. and Sha, L. (2019). A container-based dos attack-resilient control framework for real-time uav systems, *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1222–1227.

Combe, T., Martin, A. and Di Pietro, R. (2016). To docker or not to docker: A security perspective, *IEEE Cloud Computing* **3**(5): 54–62.

Jain, V., Singh, B., Khenwar, M. and Sharma, M. (2021). Static vulnerability analysis of docker images, *IOP Conference Series: Materials Science and Engineering* **1131**: 012018.

Javed, O. and Toor, S. (2021). *An Evaluation of Container Security Vulnerability Detection Tools*, Association for Computing Machinery, New York, NY, USA, p. 95–101.
URL: *https://doi.org/10.1145/3481646.3481661*

Jian, Z. and Chen, L. (2017). A defense method against docker escape attack, *Proceedings of the 2017 International Conference on Cryptography, Security and Privacy*, ICCSP '17, Association for Computing Machinery, New York, NY, USA, p. 142–146.
URL: *https://doi.org/10.1145/3058060.3058085*

Koskinen, T., Ihantola, P. and Karavirta, V. (2012). Quality of wordpress plug-ins: An overview of security and user ratings, *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing*, pp. 834–837.

Loukidis-Andreou, F., Giannakopoulos, I., Doka, K. and Koziris, N. (2018). Docker-sec: A fully automated container security enhancement mechanism, *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pp. 1561–1564.

Luo, Y., Luo, W., Sun, X., Shen, Q., Ruan, A. and Wu, Z. (2016). Whispers between the containers: High-capacity covert channel attacks in docker, *2016 IEEE Trustcom/BigDataSE/ISPA*, pp. 630–637.

Martin, A., Raponi, S., Combe, T. and Di Pietro, R. (2018). Docker ecosystem – vulnerability analysis, *Computer Communications* **122**: 30–43.
URL: *https://www.sciencedirect.com/science/article/pii/S0140366417300956*

Mattetti, M., Shulman-Peleg, A., Allouche, Y., Corradi, A., Dolev, S. and Foschini, L. (2015). Securing the infrastructure and the workloads of linux containers, *2015 IEEE Conference on Communications and Network Security (CNS)*, pp. 559–567.

MP, A. R., Kumar, A., Pai, S. J. and Gopal, A. (2016). Enhancing security of docker using linux hardening techniques, *2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*, pp. 94–99.

Shu, R., Gu, X. and Enck, W. (2017). A study of security vulnerabilities on docker hub, *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*, CODASPY '17, Association for Computing Machinery, New York, NY, USA, p. 269–280.
URL: *https://doi.org/10.1145/3029806.3029832*

Sprabery, R., Evchenko, K., Raj, A., Bobba, R. B., Mohan, S. and Campbell, R. (2018). Scheduling, isolation, and cache allocation: A side-channel defense, *2018 IEEE International Conference on Cloud Engineering (IC2E)*, pp. 34–40. CORE2021 Rank:Not Available.

Sultan, S., Ahmad, I. and Dimitriou, T. (2019). Container security: Issues, challenges, and the road ahead, *IEEE Access* **7**: 52976–52996.

Vaucher, S., Pires, R., Felber, P., Pasin, M., Schiavoni, V. and Fetzer, C. (2018). Sgx-aware container orchestration for heterogeneous clusters, *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pp. 730–741. CORE2021 Rank: A.