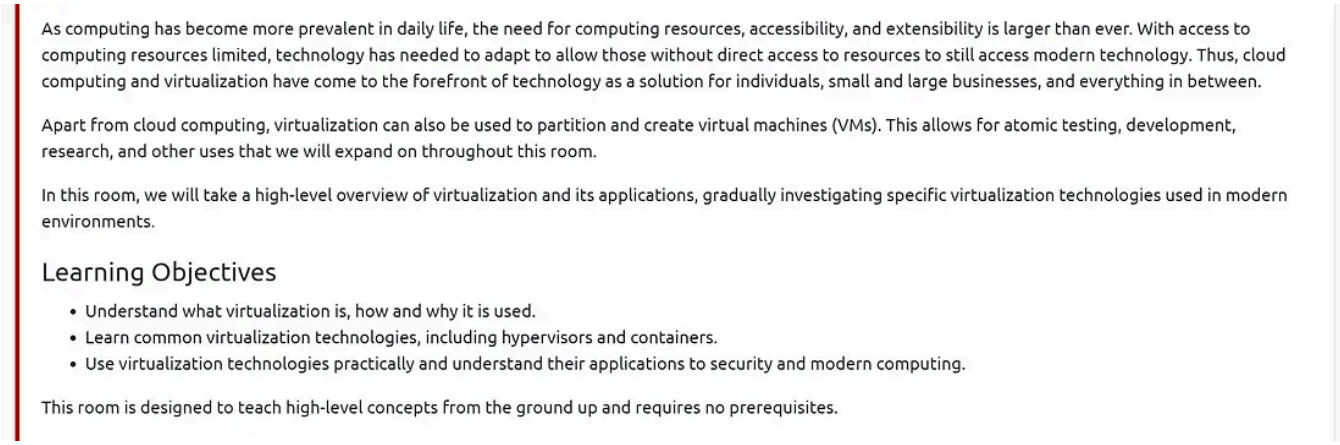
Tryhackme: Virtualization and Containers

Daniel Schwarzentraub · Follow
3 min read · Sep 28, 2023

Task 1: **Introduction**



As computing has become more prevalent in daily life, the need for computing resources, accessibility, and extensibility is larger than ever. With access to computing resources limited, technology has needed to adapt to allow those without direct access to resources to still access modern technology. Thus, cloud computing and virtualization have come to the forefront of technology as a solution for individuals, small and large businesses, and everything in between.

Apart from cloud computing, virtualization can also be used to partition and create virtual machines (VMs). This allows for atomic testing, development, research, and other uses that we will expand on throughout this room.

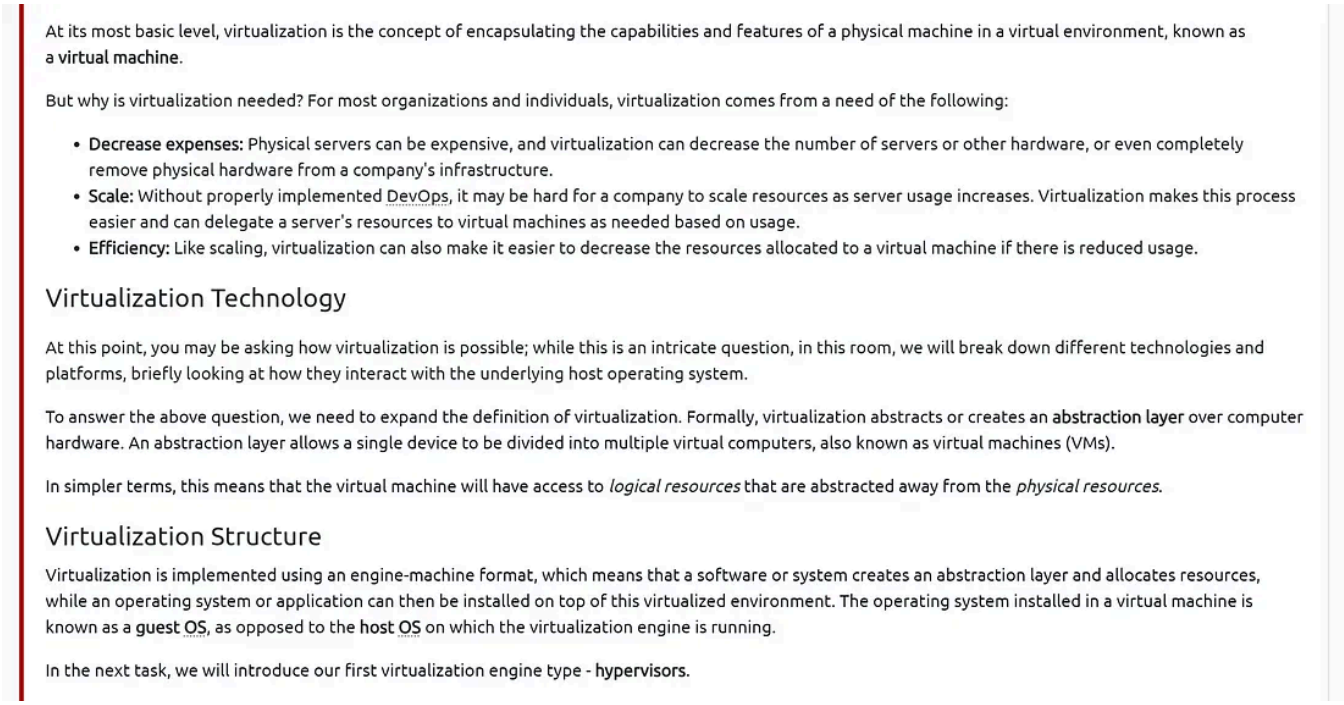
In this room, we will take a high-level overview of virtualization and its applications, gradually investigating specific virtualization technologies used in modern environments.

Learning Objectives

- Understand what virtualization is, how and why it is used.
- Learn common virtualization technologies, including hypervisors and containers.
- Use virtualization technologies practically and understand their applications to security and modern computing.

This room is designed to teach high-level concepts from the ground up and requires no prerequisites.

Task 2: **What is Virtualization**



At its most basic level, virtualization is the concept of encapsulating the capabilities and features of a physical machine in a virtual environment, known as a **virtual machine**.

But why is virtualization needed? For most organizations and individuals, virtualization comes from a need of the following:

- **Decrease expenses:** Physical servers can be expensive, and virtualization can decrease the number of servers or other hardware, or even completely remove physical hardware from a company's infrastructure.
- **Scale:** Without properly implemented DevOps, it may be hard for a company to scale resources as server usage increases. Virtualization makes this process easier and can delegate a server's resources to virtual machines as needed based on usage.
- **Efficiency:** Like scaling, virtualization can also make it easier to decrease the resources allocated to a virtual machine if there is reduced usage.

Virtualization Technology

At this point, you may be asking how virtualization is possible; while this is an intricate question, in this room, we will break down different technologies and platforms, briefly looking at how they interact with the underlying host operating system.

To answer the above question, we need to expand the definition of virtualization. Formally, virtualization abstracts or creates an **abstraction layer** over computer hardware. An abstraction layer allows a single device to be divided into multiple virtual computers, also known as virtual machines (VMs).

In simpler terms, this means that the virtual machine will have access to *logical resources* that are abstracted away from the *physical resources*.

Virtualization Structure

Virtualization is implemented using an engine-machine format, which means that a software or system creates an abstraction layer and allocates resources, while an operating system or application can then be installed on top of this virtualized environment. The operating system installed in a virtual machine is known as a **guest OS**, as opposed to the **host OS** on which the virtualization engine is running.

In the next task, we will introduce our first virtualization engine type - **hypervisors**.

Is scalability a primary benefit of virtualization? (Y/N)

Answer: **Y**

What is the operating system of a virtual machine often referred to as?

Answer: **Guest OS**

Task 3: **Hypervisors**

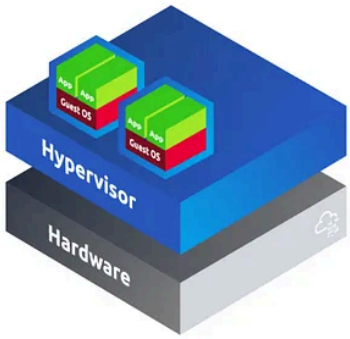
In the previous task, we introduced the concept of virtualization at a high level and briefly discussed the structure of virtualization. In this task, we will present our first type of virtualization engine: **hypervisors**.

A hypervisor provides the ability to create the abstraction layer between hardware and software. A hypervisor will also generally include some form of management application or software to provide an interface between the end user and the abstraction layer to create or load virtual machines.

Hypervisors are separated into two categories that are determined by their position relative to the hardware. They can either directly create a lightweight operating system on top of the hardware that is the hypervisor or add a hypervisor as an application on top of a pre-existing operating system.

Type 1 Hypervisors

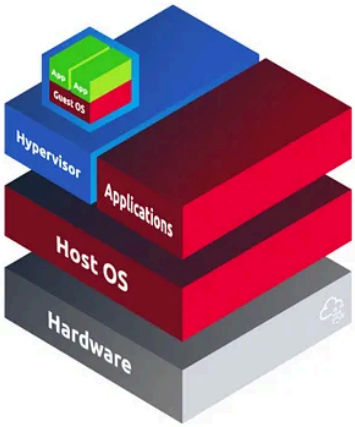
Type 1 hypervisors, also known as **bare metal hypervisors**, create an abstraction layer directly between hardware and virtual machines without a common operating system between them. Instead, the hypervisor is the operating system and is often *headless*, with only a web-based management portal remotely accessed. These hypervisors are designed for scale and to deploy a large number of virtual machines at once. They are extremely lightweight to dedicate the most resources to virtual machines. Below is a diagram of a type 1 hypervisor architecture.



Examples of type 1 hypervisors include VMware ESXi, Proxmox, VMware vSphere, Xen, and KVM.

Type 2 Hypervisors

Type 2 hypervisors, also known as **hosted hypervisors**, create an abstraction layer from a software application built on top of a pre-existing operating system. Unlike type 1 hypervisors, type 2 hypervisors are often managed directly from the application through a GUI. These hypervisors are often designed for end-users or individuals such as developers and are not strictly designed to run a large number of virtual machines for scale. Below is a diagram of a type 2 hypervisor architecture.



Examples of type 2 hypervisors include VMware Workstation, VMware Fusion, VirtualBox, Parallels, and QEMU.

What type of hypervisor is VirtualBox considered?

Answer: **Type 2**

What are type 1 hypervisors also known as?

Answer: **Bare Metal Hypervisor**

Task 4: **Containers**

Hypervisors work as expected for a large number of use cases but begin to encounter issues when scaling lightweight applications. *Microservices* give us a good example of an application architecture that encounters issues when deployed from a hypervisor. A microservice is an application structure that is broken up into smaller services that are scalable and use lightweight protocols and features. The lightweight nature of the architecture poses obvious issues to hypervisors that require a large number of virtual machines each with high resource usage.

Containers are the current solution to the issues encountered with hypervisors at scale.

What are Containers

Containers have a lot in common with virtual machines, but instead of being completely abstracted from the host operating system, containers share some properties with the host operating system. Containers have their own filesystem, a portion of computing resources (CPU, RAM), a process space, and more.

Apart from the obvious benefits of being lightweight, containers are also *portable* and *robust* because they are not completely abstracted.

Container engines are our second type of virtualization. As virtual machines use a hypervisor to create an abstraction layer for virtualization, containers use a container engine to create an abstraction layer using logical resources.

In the next task, we will introduce our first container engine - **Docker**.

Are containers completely abstracted from the host operating system? (Y/N)

Answer: **N**

Task 5: **Docker**

If someone is familiar with containers, Docker is likely the first name that comes to mind. Docker is a container platform and engine that is used to run Docker "images" as containers.

Start Machine

Each Docker image is built of a base image, such as Alpine or Ubuntu, that is specifically built for use in containers and is lightweight. To build a Docker image, a Dockerfile must be created, which defines the base image for a container and any commands to be run.

For more information about Docker, check out the Intro to Docker room.

Running and Interacting with a Docker Container

Docker Hub is a remote repository for Docker images, similar to GitHub - a remote repository for Git. Using Docker Hub, we can pull Docker images created by others or push our own.

```
docker pull <user>/<image>
```

Alternatively, a container image can be automatically pulled when running the container for the first time. Once a container is pulled for the first time, it will be cached locally, and Docker will look for it locally before attempting to download it.

```
docker run <user>/<image>
```

Once the image is started, we can verify that the Docker engine is running the container by listing the processes running in Docker using the below command.

```
docker ps
```

From the above command, you may notice that the container will be assigned a random identifier, IP address, and network interface.

Hands-On Application

To get hands-on with containers and Docker, we will deploy a Flask web server in a Docker container. To access the terminal, deploy the virtual machine attached to this task task by pressing the green **Start Machine** button. Please allow the machine 3 - 5 minutes to deploy. The machine will start in split view; if the machine does not appear, you can click the blue **Show Split View** button located at the top right of this room. Alternatively, you can access the machine with SSH using the credentials provided below.

Machine IP: `MACHINE_IP` Username: `thmuser` Password: `TryHackMe!`

Before starting the docker container, we must first introduce two new flags that must be used with the `run` command to allow the container to detach from the current terminal (`-d`) and expose ports externally (`-p`). Below is the required syntax to start the example Flask app in a Docker container, exposing the webserver to port 5000.

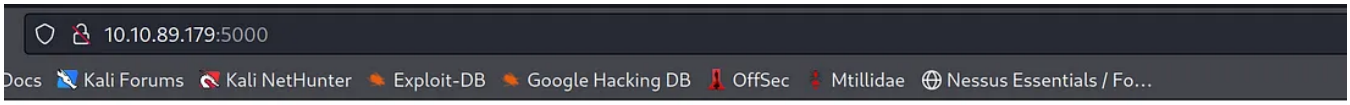
```
docker run -p 5000:5000 -d cryillic/thm_example_app
```

To verify that the web server is running in the Docker container, we can scan the container or access the designated port in a web browser.

Note: Certain special characters may not be visible on the provided VM. When doing a copy-and-paste, it will still copy all characters.

What flag is obtained at MACHINE_IP:5000 after running the container?

We will start off with docker image to see what the docker container name is



THM{this_is_running_in_docker}

Answer: **THM{this_is_running_in_docker}**

Task 6: **Kubernetes**

Through the use of hypervisors and containers, most problems associated with traditional computing are resolved, such as *cost* and *efficiency*. This still leaves the question, what if we need a faster and more scalable solution? That is, as load or other criteria changes, the resources or the number of instances allocated to the application or service increase or decrease on the fly as needed.

Start Machine

Kubernetes, also shortened to "**K8s**," is one such solution known as an **orchestration platform**. An orchestration platform aims to integrate into other products, such as Docker, and extend their capabilities or "synchronize" them with other products or applications.

Kubernetes relies on these traditional virtualization models like hypervisors and containers and extends their uses, features, and capabilities.

These capabilities and features include the following:

- **Horizontal scaling**: Unlike traditional "vertical" scaling, "horizontal" scaling refers to adding devices or machines to handle increased workload, rather than adding logic resources such as CPU or RAM.
- **Extensibility**: Clusters can be modified dynamically without affecting containers outside of the intended group.
- **Self-healing**: K8s can automatically restart, replace, reschedule, and kill containers that are not properly functioning based on user-defined health checks.
- **Automated rollouts and rollbacks**: K8s can progressively roll out changes to containers. As changes are made, it will monitor the application's health and decide whether to continue the rollout or rollback. This ensures the constant uptime of your cluster even if some containers fail.

Hands-On Application

To get hands-on with K8s, we've given you access to a pre-deployed cluster configured with *kubectl*. Because Kubernetes provides most of the basic automation by default, we want you to explore the differences between K8s and a traditional virtualization platform on your own. That being said, Kubernetes is a large platform with a pervasive list of features. We encourage you to continue investigating DevOps and the features of K8s, as we will only provide you with a basic set of questions to get started exploring a Kubernetes cluster.

To access the cluster, deploy the virtual machine attached to this task task by pressing the green **Start Machine** button. Please allow the machine 3 - 5 minutes to deploy. The machine will start in split view; if the machine does not appear, you can click the blue **Show Split View** button located at the top right of this room. Alternatively, you can access the machine with SSH using the credentials provided below.

Machine IP: `MACHINE_IP` Username: `thmuser` Password: `TryHackMe!`

To aid you in using kubectl, you can use the online reference found here. We encourage you to use the hints to guide you as you answer questions.

kubectl Cheat Sheet

This page contains a list of commonly used kubectl commands and flags. Note: These instructions are for Kubernetes...

kubernetes.io

Before proceeding, ensure all clusters are started by running minikube start.

```
$ minikube start
😄  minikube v1.30.1 on Ubuntu 20.04
✨  Using the docker driver based on existing profile

❗  The requested memory allocation of 1955MiB does not leave room for system overhead (total system memory: 1955MiB). You may face stability is
sues.
💡  Suggestion: Start minikube with less memory allocated: 'minikube start --memory=1955mb'

👍  Starting control plane node minikube in cluster minikube
🚜  Pulling base image ...
🔄  Restarting existing docker container for "minikube" ...
!   This container is having trouble accessing https://registry.k8s.io
💡  To pull new external images, you may need to configure a proxy: https://minikube.sigs.k8s.io/docs/reference/networking/proxy/
🐳  Preparing Kubernetes v1.26.3 on Docker 23.0.2 ...
🔗  Configuring bridge CNI (Container Networking Interface) ...
🔎  Verifying Kubernetes components...
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟  Enabled addons: default-storageclass, storage-provisioner
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
$
```

How many pods are running on the provided cluster?

```
$ kubectl get pods -A
NAMESPACE     NAME                               READY   STATUS    RESTARTS        AGE
default       hello-tryhackme-f79f667fd-lh87b    1/1     Running   1 (114s ago)    127d
kube-system   coredns-787d4945fb-wqxqr           1/1     Running   1 (114s ago)    127d
kube-system   etcd-minikube                      1/1     Running   1 (114s ago)    127d
kube-system   kube-apiserver-minikube            1/1     Running   1 (114s ago)    127d
kube-system   kube-controller-manager-minikube   1/1     Running   1 (114s ago)    127d
kube-system   kube-proxy-sqwx6                   1/1     Running   1 (114s ago)    127d
kube-system   kube-scheduler-minikube            1/1     Running   1 (114s ago)    127d
kube-system   storage-provisioner                1/1     Running   3 (14s ago)     127d
$
```

Answer: **1**

How many system pods are running on the provided cluster?

Answer: **7**

What is the pod name on the provided cluster?

Answer: **hello-tryhackme-f79f667fd-lh87b**

What is the deployment name on the provided cluster?

```
$ kubectl get deployment
NAME              READY   UP-TO-DATE   AVAILABLE   AGE
hello-tryhackme   1/1     1            1           127d
$
```

Answer: **hello-tryhackme**

What port is exposed by the service in question 5?

```
$ kubectl get services
NAME              TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
hello-tryhackme   NodePort    10.109.213.246   <none>        8080:31449/TCP   127d
kubernetes        ClusterIP   10.96.0.1        <none>        443/TCP          127d
$
```

Answer: **8080**

How many replica sets are deployed on the provided cluster?

ReplicaSet

A ReplicaSet's purpose is to maintain a stable set of replica Pods running at any given time. Usually, you define a...

kubernetes.io

Answer: **1**

What is the replica set name on the provided cluster?

Answer: **hello-tryhackme-f79f667fd**

What command would be used to delete the deployment from question 5?

How to Use Kubectl Delete Deployment (With Examples)?

Learn how to use the "kubectl delete deployment" command to delete Deployments from both default and specific...

kodekloud.com

Delete Deployment(s) in a Specific Namespace

Until now, we have seen how to delete a single Deployment, multiple Deployments, and all Deployments from the "default" Namespace.

What if we want to delete Deployments from specified Namespaces? We can do so using the same commands we studied in the previous sections, with just one simple addition. We need to specify the Namespace we want to delete the Deployment(s) from using the "-n" or "--namespace" flag.

For example, to delete the "alpine-dev" Deployment inside the "development" Namespace, run the following command:

```
kubectl delete deployment alpine-dev -n=development
```

Answer: **kubectl delete deployment hello-tryhackme**

Task 7: **Conclusion**

> In this room, we've briefly overviewed the current virtualization technologies, the problems they solve, and how they are implemented. You may still be asking, how is virtualization used every day?
>
> Throughout this room, we have looked at virtualization in the context of microservices, but virtualization goes far beyond that.
>
> Virtual machines are portable; they can be stored, moved, and redeployed in the same state. With portability also comes ease of deployment; templates and "golden images" can be used to create an image or snapshot of a VM so it can be redeployed multiple times.
>
> Because virtual machines can be easily redeployed, they make a perfect environment for development and testing. If an unknown exception occurs or a machine breaks, you can always revert to a previous snapshot or the image from which you started.
>
> These are only a handful of the applications of virtualization, and we encourage you to explore these solutions in light of your challenges or opportunities and see how they could benefit you.

Tryhackme Walkthrough Tryhackme Writeup

Follow

Written by Daniel Schwarzentraub

116 Followers · 4 Following

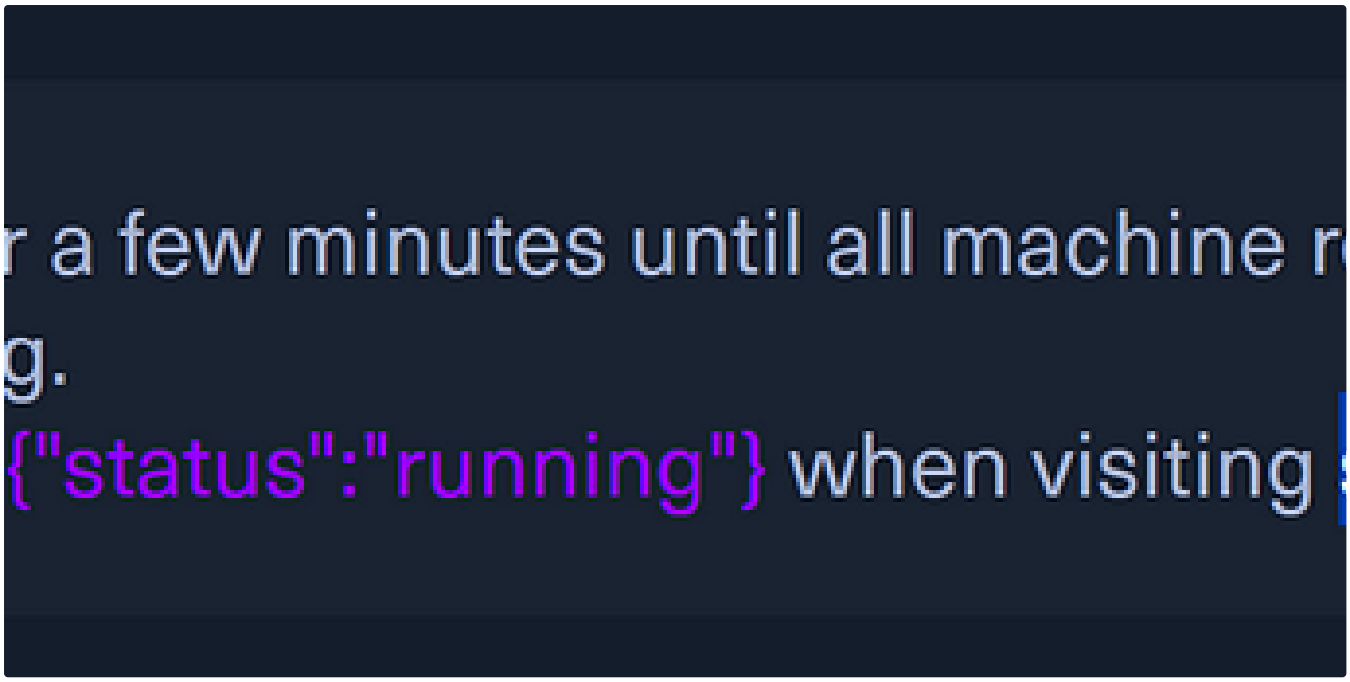
PNW_Hacker

No responses yet

What are your thoughts?

Respond

More from Daniel Schwarzentraub

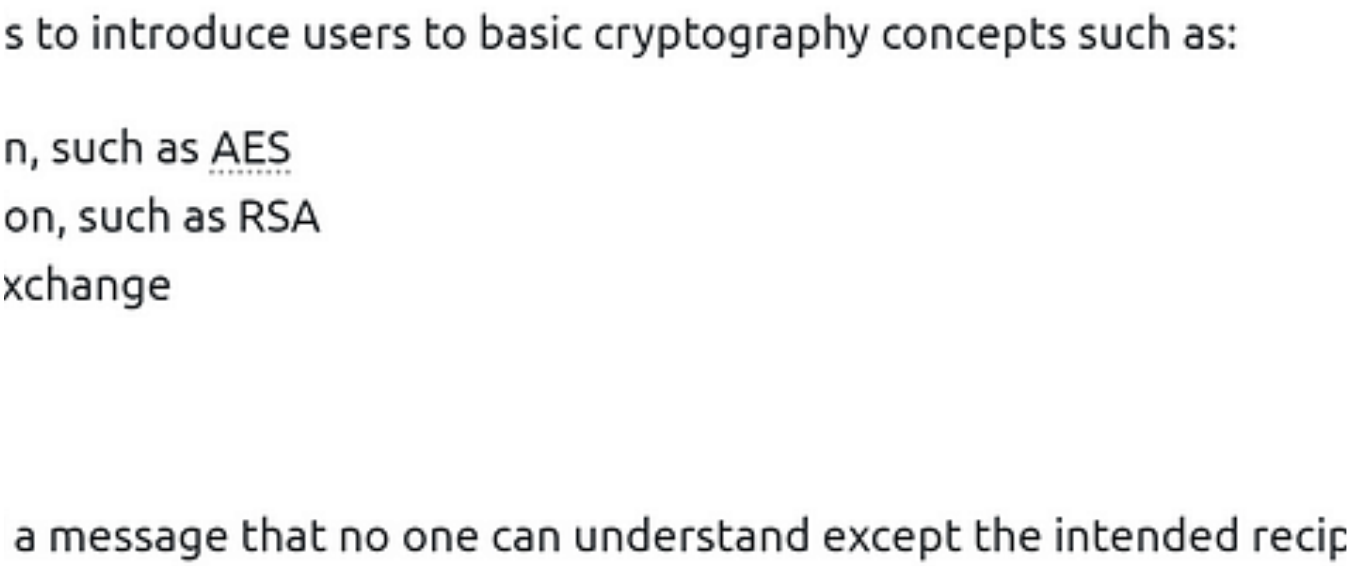


Daniel Schwarzentraub

HTB—Tier 1 Starting Point: Three

HTB—Tier 1 Starting Point: Three

Jul 20, 2023 4 2

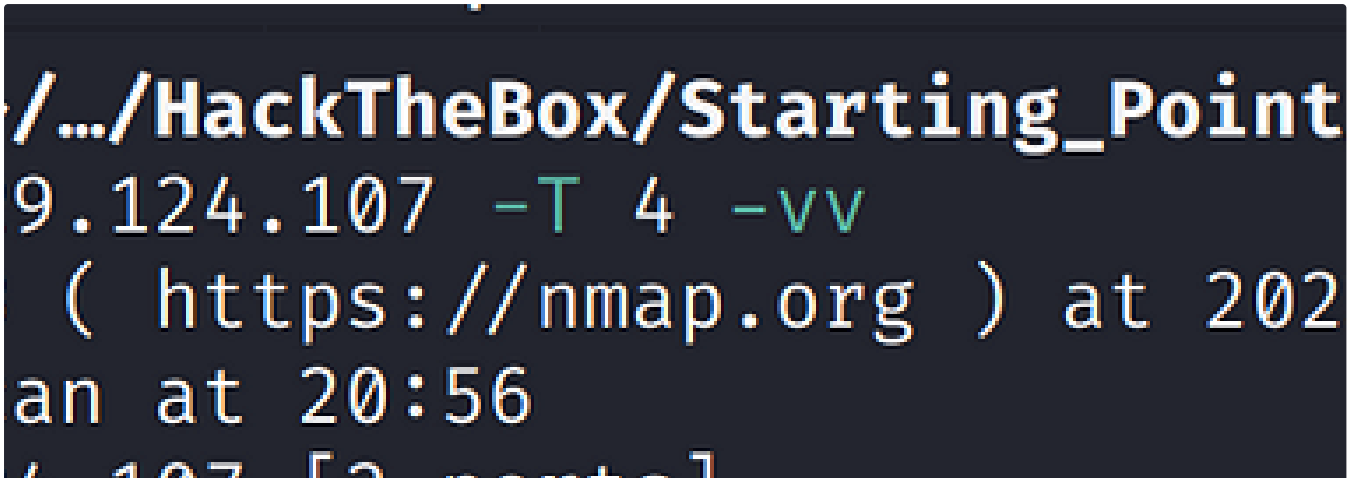


Daniel Schwarzentraub

Tryhackme: Introduction to Cryptography

Tryhackme: Introduction to Cryptography

Sep 26, 2023 2



Daniel Schwarzentraub

HTB—Tier 2 Starting Point: Archetype

HTB—Tier 2 Starting Point: Archetype

Jul 21, 2023

erative that we understand and can protect against common attacks.

mon techniques used by attackers to target people online. It will also teach some of the best wa

Daniel Schwarzentraub

Tryhackme Free Walk-through Room: Common Attacks

Tryhackme Free Walk-through Room: Common Attacks

Sep 13, 2024

See all from Daniel Schwarzentraub

Recommended from Medium

In T3CH by Axoloth

TryHackMe | Training Impact on Teams | WriteUp

Discover the impact of training on teams and organisations

Nov 5, 2024 60

IritT

Nmap—TryHackMe Insights &Walkthrough

An in depth look at scanning with Nmap, a powerful network scanning tool.

Dec 23, 2024

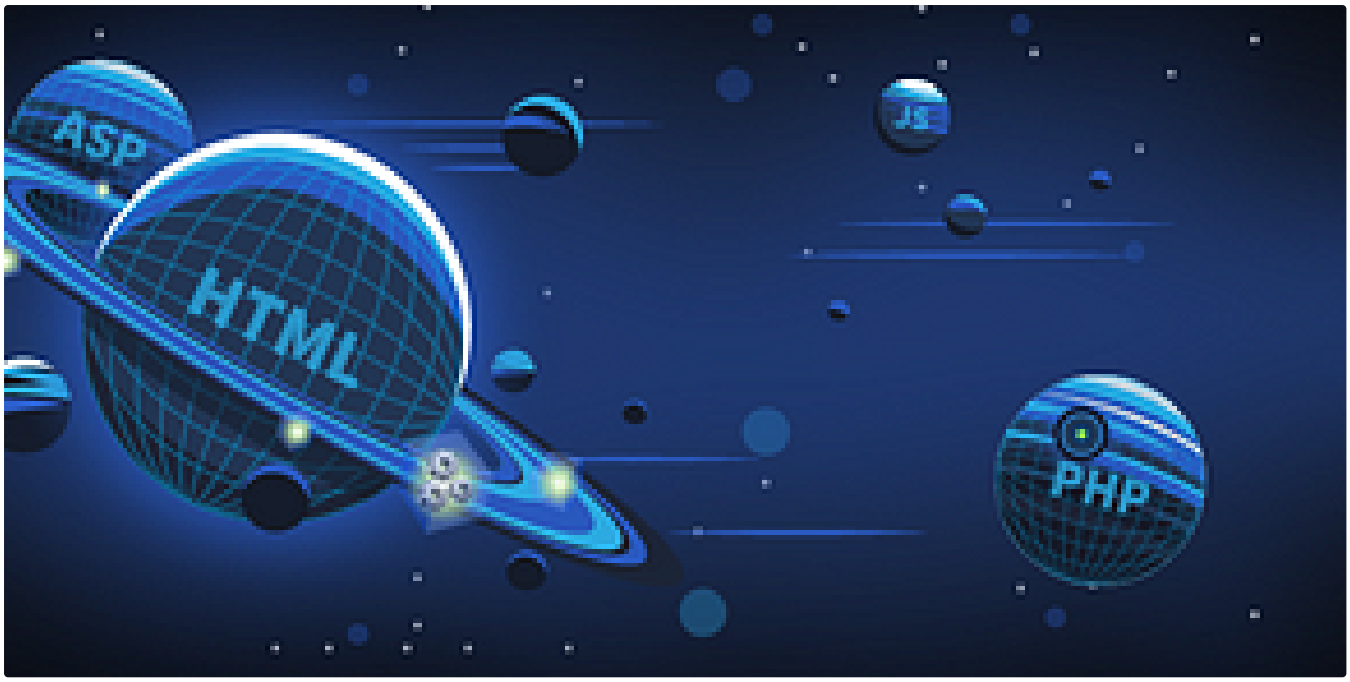
Lists

Staff picks
796 stories · 1560 saves

Stories to Help You Level-Up at Work
19 stories · 912 saves

Self-Improvement 101
20 stories · 3196 saves

Productivity 101
20 stories · 2707 saves

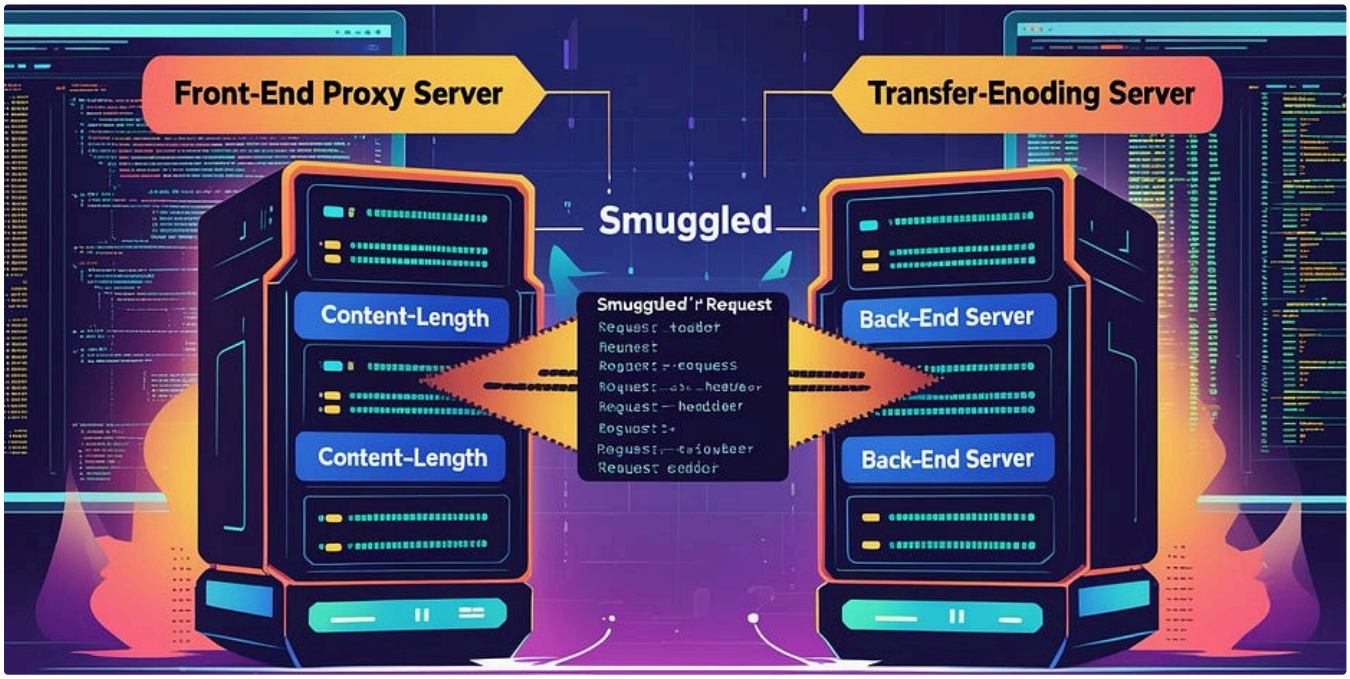


In T3CH by Axoloth

TryHackMe | Web Application Basics | WriteUp

Learn the basics of web applications: HTTP, URLs, request methods, response codes, and headers

Oct 26, 2024 56



CyferNest Sec

HTTP Request Smuggling | TryHackMe Walkthrough

TASK 2: Modern Infrastructure

4d ago 1 1

erative that we understand and can protect against common attacks.

mon techniques used by attackers to target people online. It will also teach some of the best wa

Daniel Schwarzentraub

Tryhackme Free Walk-through Room: Common Attacks

Tryhackme Free Walk-through Room: Common Attacks

Sep 13, 2024

Ansul Kotadia

Incident Response Process: TryHackMe Writeup

Task 1: Introduction

Nov 28, 2024 70 1