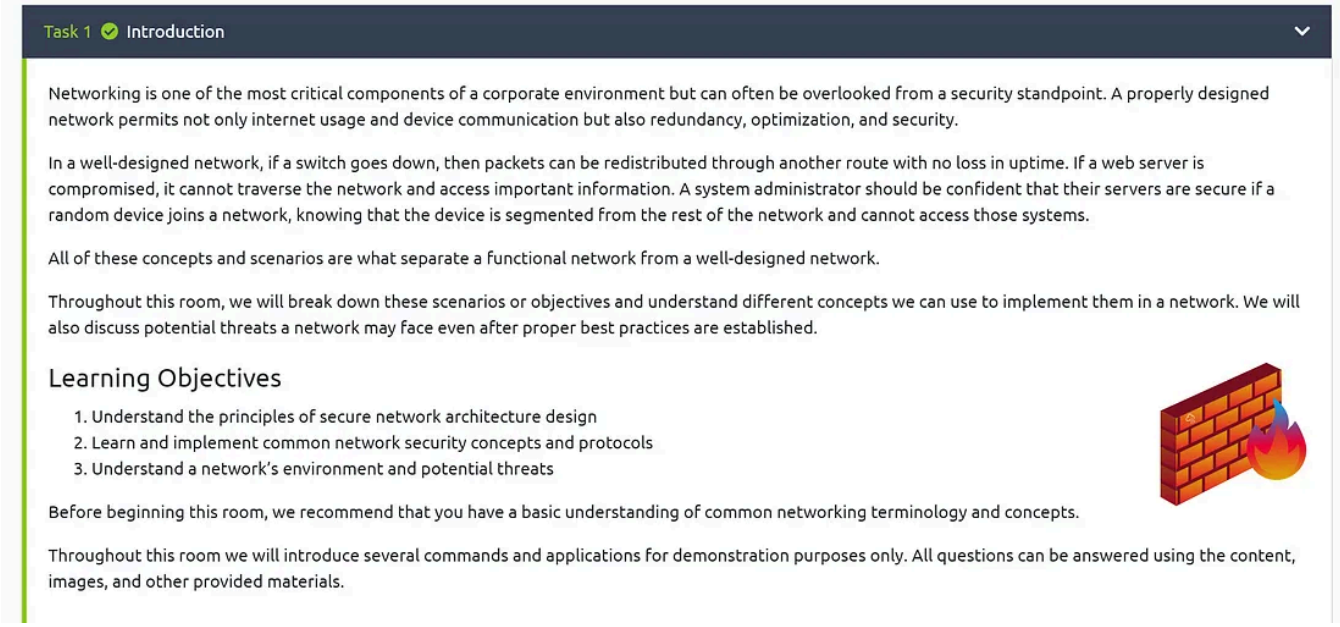
Tryhackme: Secure Network Architecture

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

Task 1: **Introduction**



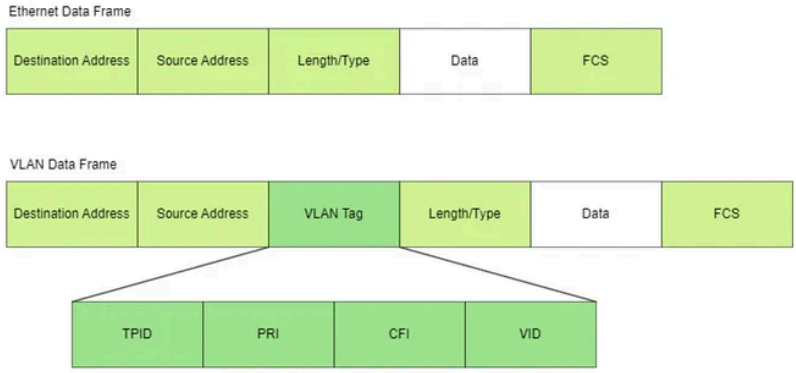
Task 2: **Network Segmentation**

The Need for Secure Segmentation

Subnets seem to solve all problems a network may face; why would we use another solution? To answer this question, let's consider a scenario where a client brings in their own device, a common practice known as BYOD (Bring Your Own Device). The client's device was infected with a Remote Access Trojan (RAT) that will attempt to traverse the network the device is connected to and exfiltrate any sensitive information. With subnetting in place, there is no restriction in place as to where the infected device could connect as long as the proper routes are in place, leaving sensitive information and servers open to the unknown device. How do we fix this problem? VLANs!

VLANs (Virtual **LAN**) are used to segment portions of a network at layer two and differentiate devices.

VLANs are configured on a switch by adding a "tag" to a frame. The **802.1q** or **dot1q** tag will designate the VLAN that the traffic originated from.



The 802.1 tag provides a standard between vendors that will always define the VLAN of a frame. For example, if our switches are Cisco and our routers are Juniper, they can send tagged frames and digest them equally because it is standardized. As a demonstration, we will show how to configure tags on the interfaces of a switch.

In this example, we will use the open-source switch: Open vSwitch

Let's first look at the default configuration of the switch.

```
$ ovs-vsctl show
```

To add a VLAN and tags to a sub-interface, we need to modify the configuration through `ovs-vsctl`

```
$ ovs-vsctl set port <interface> tag=<0-99>
```

The tag should now be listed under the sub-interface in the configuration.

```
Port eth1
        tag: 10
        Interface eth1
```

We now have our first interfaces configured to tag their frames! But what if we don't know where the VLAN traffic originates from? For example, traffic from virtualized devices or network traffic.

Tagging Unknown Traffic

The **Native VLAN** is used for any traffic that is not tagged and passes through a switch. To configure a native VLAN, we must determine what interface and tag to assign them, then set the interface as the default native VLAN. Below is an example of adding a native VLAN in Open vSwitch.

```
$ ovs-vsctl set port eth0 tag=10 vlan_mode=native-untagged
```

Now all traffic should be tagged, even with unknown origins.

Open vSwitch

To understand why virtualized environments require a new approach to switching, read the WHY-OVS document that is...

www.openvswitch.org

Routing Between VLANs

But how does a VLAN connect to the internet or access resources in other VLANS? Since they are segmented, they cannot communicate outside their tagged devices. Just as routers are used to communicate between traditional networks, routers can be used to route between VLANs.

Before modern solutions were introduced, network engineers would physically connect a switch and router separately for each VLAN present. Nowadays, that problem is solved through the **ROAS** (**R**outer **o**n **a** **S**tick) design. VLANs are configured to communicate with a router through a designated interface of a switch, known as a **switch port**. The connection between the switch and router is known as a **trunk**. VLANs are routed through the switch port, requiring only one trunk/connection between the switch and router, hence, "*on a stick.*"

Before configuring the router, we must configure a trunk on a pre-existing connection. In our demonstration lab environment, trunks are configured by default as **bridges**. Each vendor configures their trunks and switch ports differently, some even with propriety protocols; please refer to their specific documentation.

Below is an example of adding a new bridge and interface to create a trunk.

```
$ ovs-vsctl add-br br0
```

```
$ ovs-vsctl add-port br0 eth0 tag=10
```

Now, we can configure our router to route tagged traffic between VLANs. Remember, as mentioned when we introduced tags because the 802.1q tag is standardized, we only need to tell our router how to configure its switch port and what tags to accept for each interface.

Because all tagged traffic comes from a single connection, the router must be able to keep each tagged frame separate. This is accomplished using **virtual sub-interfaces**; these will act similar to physical interfaces and are commonly defined by the VLAN ID; the syntax for sub-interfaces is commonly,

```
<name>.<vlan/sub-interface id>
```

Below is an example of adding a new virtual sub-interface and configuring its corresponding addressing.

In this example, we will use the open-source router: VyOS

```
vyos@vyos-rtr# set interfaces ethernet eth0 vif 10 description 'VLAN 10'
```

```
vyos@vyos-rtr# set interfaces ethernet eth0 vif 10 address '192.168.100.1/24'
```

If all went well and was appropriately configured, we should be able to route traffic between VLANs while keeping traffic tagged and isolated!

But are they really isolated? Physically, they are isolated, but because routes exist between them, there is no security boundary, and they are not necessarily isolated. As long as a route exists between two VLANs, any device can communicate between the two.

This brings us to our following two tasks, where we will discuss designing secure VLANs and introduce the concept of zoning.

Analyzing a VLAN Configuration

Apply what you learned to analyze a VLAN configuration given the output below.

▼ *Interface Configuration Snippet (Click to view)*

```
/ # ovs-vsctl show
87c2a3ee-5374-435a-81c2-e8aafa96e3b9
    Bridge br2
        datapath_type: netdev
        Port br2
            Interface br2
                type: internal
    Bridge br1
        datapath_type: netdev
        Port br1
            Interface br1
                type: internal
    Bridge br0
        datapath_type: netdev
        Port eth9
            tag: 30
            Interface eth9
        Port br0
            Interface br0
                type: internal
        Port eth13
            tag: 30
            Interface eth13
        Port eth14
            tag: 30
            Interface eth14
        Port eth15
            tag: 30
            Interface eth15
```

```
            Interface eth14
        Port eth15
            tag: 30
            Interface eth15
        Port eth2
            tag: 10
            Interface eth2
        Port eth8
            tag: 30
            Interface eth8
        Port eth3
            tag: 20
            Interface eth3
        Port eth7
            tag: 30
            Interface eth7
        Port eth4
            tag: 20
            Interface eth4
        Port eth10
            tag: 30
            Interface eth10
        Port eth5
            tag: 20
            Interface eth5
        Port eth6
            tag: 30
            Interface eth6
        Port eth12
            tag: 30
            Interface eth12
        Port eth0
            Interface eth0
        Port eth11
            tag: 30
            Interface eth11
        Port eth1
            tag: 10
            Interface eth1
    Bridge br3
        datapath_type: netdev
```

```
        Port br3
            Interface br3
                type: internal
```

How many trunks are present in this configuration?

Answer: **4**

What is the VLAN tag ID for interface eth12?

Answer: **30**

Task 3: **Common Secure Network Architecture**

With the introduction of VLANs, there is a shift in network architecture design to include security as a key consideration. **Security**, **optimization**, and **redundancy** should all be considered when designing a network, ideally without compromising one component.

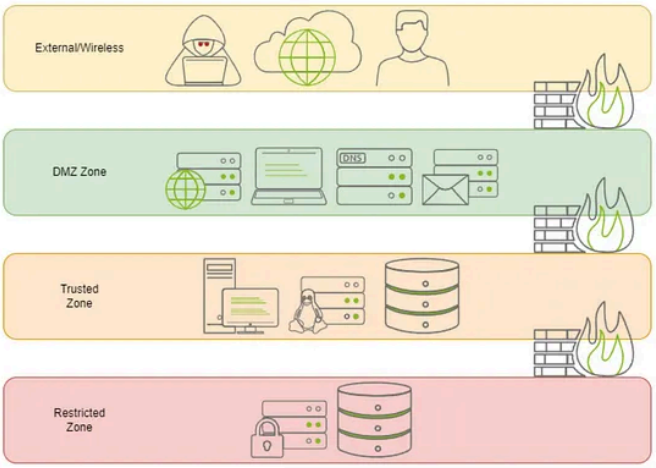
This brings us to the question, how do we properly implement VLANs as a security boundary? Security zones! **Security zones** define **what** or **who** is in a VLAN and how traffic can travel **in** and **out**.

Depending on whom you speak to, every network architect may have a different approach/opinion to the language or requirements surrounding security zones. In this task, we will immerse you in the most commonly accepted security zone standards, keeping a minimalist approach to segmentation.

Below we will present you with a table of commonly standardized zones. This is only to familiarize you with the terminology; we will cover each concept in further depth throughout this room.

Zone	Explanation	Examples
External	All devices and entities outside of our network or asset control.	Devices connecting to a web server
DMZ (demilitarized zone)	Separates untrusted networks or devices from internal resources.	BYOD, remote users/guests, public servers
Trusted	Internal networks or devices. A device may be placed in the trusted zone if there is no confidential or sensitive information.	Workstations, B2B
Restricted	Any high-risk servers or databases.	Domain controllers, client information
Management	Any devices or services dedicated to network or other device management. This zone is less commonly seen and can be grouped with the audit zone.	Virtualization management, backup servers
Audit	Any devices or services dedicated to security or monitoring. This zone is less commonly seen and can be grouped with management.	SIEM, telemetry

While security zones mostly factor in what will happen internally, it is equally important to consider how new traffic or devices will enter the network, be assigned, and interact with internal systems. Most external traffic (HTTP, mail, etc.) will stay in the DMZ, but what if a remote user needs access to an internal resource? We can easily create rules for resources a user or device can access based on MAC, IP addresses, etc. We can then enforce these rules from network security controls; in the next task, we will discuss various access controls and solutions to implement policies.



Security zones and access controls will physically direct how and where traffic goes. But how is it decided what resources users or devices have access to? Traffic rules are often governed by company security policy or compliance as equally as security controls that determine access permissions.

We've now established a system to approach designing VLANs, but how can we practically implement and enforce them? In the next task, we will cover several protocols and applications that can be used to implement and enforce VLANs.

From the above table, what zone would a user connecting to a public web server be in?

Answer: **External**

From the above table, what zone would a public web server be in?

Answer: **DMZ**

From the above table, what zone would a core domain controller be placed in?

Answer: **Restricted**

Task 4: **Network Security Policies and Controls**

Now that we have covered segmentation and secure architecture design, how do we enforce it? If there are routes between VLANs intended to be separated, how is the appropriate access restricted or granted?

Policies aid in defining how network traffic is controlled. A network traffic policy may determine how and if a router will route traffic before other routing protocols are employed. **IEEE** (Institute of Electrical and Electronics Engineers) has standardized a handful of access control and traffic policies, such as **QoS** (Quality of Service) (**802.11e**). There are still many other routing and traffic policies that are not standardized by IEEE but are commonly used by all vendors following the same objectives.

This task will focus on traffic filtering and introduce network policy concepts.

Traffic Filtering

Before formally defining what traffic filtering is, let's discuss one of the most common standards of defining traffic filtering: **ACL(s)** (Access Control List(s)).

An ACL is used as a loose standard to create a ruleset for different implementations and access control protocols. In this task, we will use ACLs within a router to decide whether to route or drop a packet based on the defined list.

An ACL contains **ACE(s)** (Access Control Entry) or rules that define a list's profile based on pre-defined criteria (source address, destination address, etc.)

Once defined, we can use an ACL for several vendor-specific implementations. For example, Cisco uses ACLs for traffic filtering, priority or custom queuing, and dynamic access control.

Formally, traffic filtering provides network security, validation, and segmentation by filtering network traffic based on pre-defined criteria.

We've now defined what traffic filtering is and how the criteria are defined. Let's look at how we can implement ACLs in traffic filtering or access control policies.

To get hands-on, we will look at the policies that VyOS offers. The VyOS access-list policy is the platform's most basic implementation of filtering. The policy will use ACLs or prefix lists to define the criteria for filtering.

Let's break down how VyOS creates an ACL and defines the policy.

Create the new access-list policy, defining the ACL number (1 - 2699)

```
set policy access-list <acl_number>
```

Set the description of the access list.

```
set policy access-list <acl_number> description <text>
```

Create a new rule (or ACE) under the ACL and define the action of the rule.

```
set policy access-list <acl_number> rule <1-65535> action <permit|deny>
```

Define criteria or parameters for the rule to enforce/match.

```
set policy access-list <acl_number> rule <1-65535> <destination|source> <any|host|inverse-mask|network>
```

At this point, we have an ACL and an ACE that is actively being enforced by the router.

Let's recap, the ACL is defined as an ACL number and is made of separate rules or ACEs that describe the behavior of the ACL. Each ACE can determine the action, destination/source, and the specific address/range to trigger the ACE.

Below is a diagram showing a valid SSH request permitted by the ACL.

```
Internet Protocol Version 4, Src: 10.10.212.209, Dst: 10.10.212.209
    Protocol: TCP (6)
    Header checksum: 0xbfdd [validation disabled]
    [Header checksum status: Unverified]
    Source: 10.10.212.209
    Destination: 10.10.212.209
Transmission Control Protocol, Src Port: 35560, Dst Port: 22, Seq: 1578, Ack: 1670, Len: 148
    Source Port: 35560
    Destination Port: 22
```

```
set policy access-list 1 rule 1 action permit
```

```
set policy access-list 1 rule 1 source 10.10.212.209
```

Below is a diagram showing an invalid SSH request denied by the ACL.

```
Internet Protocol Version 4, Src: 10.10.212.209, Dst: 10.10.212.209
    Protocol: TCP (6)
    Header checksum: 0xbfdd [validation disabled]
    [Header checksum status: Unverified]
    Source: 10.10.212.209
    Destination: 10.10.212.209
Transmission Control Protocol, Src Port: 35560, Dst Port: 22, Seq: 1578, Ack: 1670, Len: 148
    Source Port: 35560
    Destination Port: 2
```

```
set policy access-list 1 rule 1 action deny
```

```
set policy access-list 1 rule 1 source 10.10.212.209
```

If correctly implemented, the router should now drop or accept packets based on their source or destination address.

But is this the best solution? Recapping what was covered in task 3, what if we want a public web server in the DMZ and ensure that only HTTP traffic originates from it? What if specific hosts need specific protocols to be open on a server? A router can only provide a limited amount of extensibility for security.

In the next task, we will continue answering this question and look at how we may approach solutions for this problem.

Analyzing Packets and ACLs

Now that we understand the structure of an ACL and what it will look for in a packet, let's analyze a few packets and ACL policies to determine if they will be accepted or dropped.

Below is each packet and ACL policy required; answer the questions using the resources below.

▼ *Packet #1 (Click to view)*

```
Internet Protocol Version 4, Src: 10.10.212.209, Dst: 10.10.212.209
    Protocol:TCP (6)
    Header checksum: 0xbfdd [validation disabled]
    [Header checksum status: Unverified]
    Source: 10.10.212.209
    Destination: 10.10.212.209
Transmission Control Protocol, Src Port: 35560, Dst Port: 22, Seq: 1578, Ack: 1670, Len: 148
    Source Port: 35560
    Destination Port: 22
```

▼ *ACL Policy #1 (Click to view)*

```
set policy access-list 1 rule 1 action permit
```

```
set policy access-list 1 rule 1 source 255.255.255.0
```

```
set policy access-list 1 rule 1 source 10.10.212.0/24
```

▼ *Packet #2 (Click to view)*

```
Internet Protocol Version 4, Src: 10.10.212.200, Dst: 10.10.212.209
    Protocol:TCP (6)
    Header checksum: 0xbfdd [validation disabled]
    [Header checksum status: Unverified]
    Source: 10.10.212.209
    Destination: 10.10.212.209
Transmission Control Protocol, Src Port: 35560, Dst Port: 22, Seq: 1578, Ack: 1670, Len: 148
    Source Port: 35560
    Destination Port: 2
```

▼ *ACL Policy #2 (Click to view)*

```
set policy access-list 1 rule 1 action deny
```

```
set policy access-list 1 rule 1 destination 10.10.212.209
```

According to the corresponding ACL policy, will the first packet result in a drop or accept?

Answer: **accept**

According to the corresponding ACL policy, will the second packet result in a drop or accept?

Answer: **drop**

Task 5: **Zone-Pair Policies and Filtering**

View Site

Picking back up from the questions asked in the previous task, we must first understand what we are considering. Network considerations often include size, traffic, and data correlation; when considering protocols and the requirements of a zone, we need to shift our focus toward traffic and correlation.

Traffic correlation is standardized as the state of a packet, e.g. (protocol, process, direction, etc.)

We must employ a firewall to parse the state of a packet and enforce policies based on that state.

Firewalls

At the highest level, basic network firewalls are defined in two categories: **stateless** vs. **stateful**. A protocol's category is determined based on its ability to consider the state of a packet. For example, the ACLs used in the previous task would be regarded as a stateless protocol.

A stateful firewall can better correlate information in a network connection. This allows the firewall to filter based on protocols, ports, processes, or other information from a device, etc. For more information about check the Extending Your Network room.

Before configuring a firewall, we need to consider how we can apply the requirements of zones to firewall rules. How can we define different actions based on the protocol and source/destination zone? **Zone-pairs**!

Zone-Pairs

Zone-pairs are a direction-based and stateful policy that will enforce the traffic in single directions per each VLAN, hence, zone-pair. For example, DMZ → **LAN** or **LAN** → DMZ.

Each zone in a given topology must have a different zone-pair for each other in the topology and every possible direction. This approach provides the most visibility from a firewall and drastically improves the filtering capabilities.

In this task, we will use VyOS as an example of configuring a firewall for zone-pairing.

Before defining each pair's zoning policy, we must add a common name and default action for each VLAN.

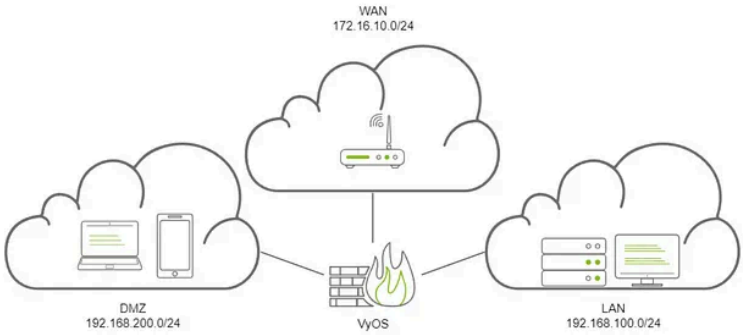
Recall from task two that VLANs are routed through a trunk and divided through sub-interfaces. The syntax is `<interface>.<VLAN #>` e.g. `eth0.30`

Below is an example of setting a default action for the DMZ and assigning the VLAN to the common zone name.

1. `set zone-policy zone dmz default-action drop`
2. `set zone-policy zone dmz interface eth0.30`

Repeat this process for each VLAN interface or zone defined in a network.

Now we can begin addressing each zone-pair direction. For this task, we will be referencing the topology below.



Each zone will have a corresponding pair to every other zone. To define each zone-pair, we will write down each possible pair and protocols or actions that we may apply to the zone-pair.

Below is an example table of each zone-pair in the above topology and what possible protocols/actions may look like.

Zone A	Zone B	Protocol	Action
LAN	WAN	ICMP	Drop
LAN	LOCAL	-	-
LAN	DMZ	-	-
WAN	LAN	ICMP	Accept
WAN	LOCAL	-	-
WAN	DMZ	-	-
LOCAL	LAN	-	-
LOCAL	WAN	-	-
LOCAL	DMZ	-	-
DMZ	LAN	HTTP	Accept
DMZ	WAN	-	-
DMZ	LOCAL	-	-

This gives us a good understanding of the expected behavior of our network and a good plan to begin configuring the firewall.

Remember: Not all traffic is IPv4! Depending on your network configuration, you may also need to configure IPv6 rules!

Because of the default action that we set, any protocols that originate on the network and are not defined will be dropped by default.

We will not go over each ruleset and zone-pair; instead, we will cover one zone-pair in each direction and test that the behavior works as expected. In this example, we will configure the LAN and WAN zone-pairs. After the example, you'll feel confident configuring a small amount of the zone-pairs on your own.

For all VyOS firewall rulesets, we must begin by defining the default action and state rules. We will not cover the intricacies of this configuration as it is not the aim of this room. Rather than repeating seven commands to create each rule, we will rely on the VyOS configuration to define each rule. Below is an example of the base VyOS ruleset. This should be the same at the top of each ruleset you create.

```
name lan-wan {
  default-action drop
  enable-default-log
  rule 1 {
    action accept
    state {
      established enable
      related enable
    }
  }
  rule 2 {
    action drop
    log enable
    state {
      invalid enable
    }
  }
}
```

Now let's add the rule for ICMP.

```
rule 100 {
    action drop # Define the action for the rule
    log enable # Enable logging to track connection attempts in VyOS
    protocol ipv4-icmp # Protocol to monitor and enforce the action on
 }
```

Now that we have our first zone-pair ruleset, let's create the rules for the opposite zone-pair direction: WAN → LAN. Below is the ruleset required to allow ICMP traffic.

```
name wan-lan {
  default-action drop
  enable-default-log
  rule 1 {
    action accept
    state {
      established enable
      related enable
    }
  }
  rule 2 {
    action drop
    log enable
    state {
      invalid enable
    }
  }
    rule 100 {
    action accept
    log enable
    protocol ipv4-icmp
 }
}
```

The zone-pair is now defined with appropriate actions and states. We can now combine the firewall ruleset with a previously configured zone.

Recall: At the beginning of this task, we configured zone policies with a corresponding interface and common name.

Below is the generic syntax for adding a zone-pair.

```
set zone-policy zone <zone A> from <zone B> firewall <name> <ruleset name>
```

Below we will set the zone-pair for both the LAN → WAN and WAN → LAN pairs.

1. `set zone-policy zone LAN from WAN firewall name lan-wan`
2. `set zone-policy zone WAN from LAN firewall name wan-lan`

We should have our first zone-pairs defined and enforced now. To test our new configuration, we can attempt to send a `ping` command in both directions to ensure the firewall is dropping or accepting our ICMP packets.

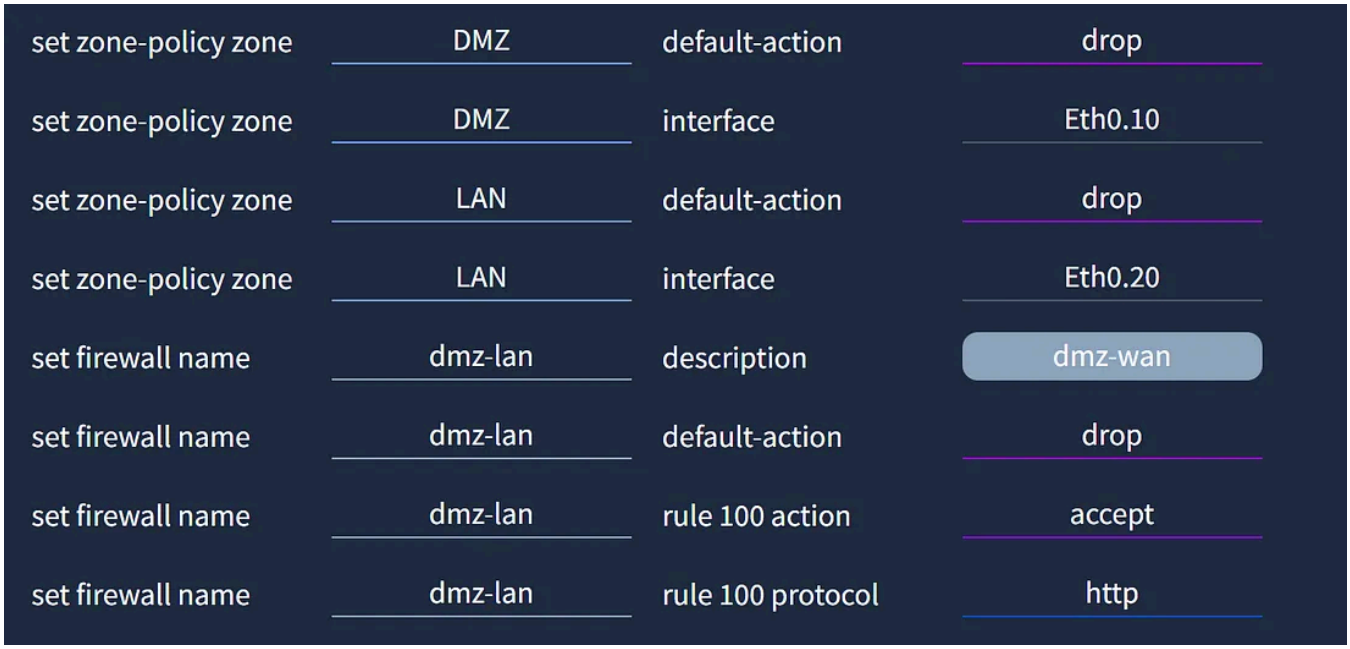
Creating a Zone-Pair from Scratch

Now that you have an understanding of how to create a basic zone-pair policy, launch the static site attached to this task by pressing the green View **Site** button. Use the table below to fill in the blanks provided in the static site.

Source	Destination	Protocol	Action	Default Action
DMZ	LAN	HTTP	Accept	Drop
DMZ	WAN	-	-	Drop
LAN	DMZ	ICMP	Accept	Drop
WAN	DMZ	-	-	Drop

Common Name	Interface	Default Action
DMZ	eth0.10	Drop
LAN	eth0.20	Drop
WAN	eth0.30	Drop

What is the flag found after filling in all blanks on the static site?



Answer: **THM{M05tly_53cure}**

Task 6: **Validating Network Traffic**

> To begin this task, let's first start with a scenario. Your organization has proper zoning and routes in place. A zone-pair between the DMZ and LAN allows an HTTPS connection. Of course, the firewall should accept these connections… How else is Susie supposed to watch Facebook or your azure updates install?
>
> In this scenario, let's say there is a threat actor that has landed an implant through phishing on a LAN machine. Assuming host defense mechanisms have failed, how can the implant be detected and monitored? If their beacon is using HTTPS through the DMZ. That will look like primarily legitimate traffic to your firewall and analysts.
>
> To solve this issue, we must use SSL/TLS inspection to intercept HTTPS connections.
>
> ### SSL/TLS Inspection
>
> SSL/TLS inspection uses an **SSL proxy** to intercept protocols, including HTTP, POP3, SMTP, or other SSL/TLS encrypted traffic. Once intercepted, the proxy will decrypt the traffic and send it to be processed by a **UTM** (Unified Threat Management) platform. UTM solutions will employ deep SSL inspection, feeding the decrypted traffic from the proxy into other UTM services, including but not limited to web filters or IPS (Intrusion Prevention System), to process the information.
>
> This solution may seem ideal, but what are the downsides? Some of you may have already noted that this requires an SSL proxy or **MitM** (Man-in-the-Middle). Even if a firewall or vendor has already implemented the solution, it will still act as a MiTM between your devices and the outside world; what if it intercepts potentially plain-text passwords? A corporation must assess the pros and cons of this solution, dependent on its calculated risk. You could allow all applications that you know are safer to prevent potential cons, but this solution will still have disadvantages. For example, an advanced threat actor could route their traffic through a cloud provider or a trusted domain.

Does SSL inspection require a man-in-the-middle proxy? (Y/N)

Answer: **Y**

What platform processes data sent from an SSL proxy?

Answer: **Unified Threat Management**

Task 7: **Addressing Common Attacks**

DHCP Snooping

Cisco defines **DHCP snooping** as "a security feature that acts like a firewall between untrusted hosts and trusted DHCP servers."

DHCP snooping was introduced to combat **rogue DHCP servers**; it will **validate** and **rate-limit** DHCP traffic as necessary. If a host is untrusted, its traffic will be filtered and rate-limited.

Although DHCP is a layer three protocol, DHCP snooping operates on the switch at layer two. The switch will store untrusted hosts with leased IP addresses in a **DHCP Binding Database**. The database is used to validate traffic and can be used by other protocols, such as dynamic ARP inspection, which we will cover later in this task.

We know how DHCP snooping will gather and store addresses, but how does it determine what to do with traffic? Below is a list of conditions the protocol will inspect to determine if a DHCP packet should be dropped.

- Any DHCP packet is received from outside of the network.
- The source MAC address and DHCP client hardware address do not match.
- A `DHCPRELEASE` or `DHCPDECLINE` packet is received on an untrusted interface that does not match an interface that the source address already has registered.
- A DHCP packet that includes a relay agent address that is not `0.0.0.0`

Although recognized by the **IEEE** in several research papers, there is no standardization of DHCP snooping. That said, it generally does not change between vendors, unlike other protocols.

Dynamic ARP Inspection

Cisco defines **ARP inspection** as "a security feature that validates **A**ddress **R**esolution **P**rotocol (**ARP**) packets in a network."

ARP inspection will **validate** and **rate-limit** ARP packets as necessary; if an ARP packet's MAC and IP address do not match, the protocol will intercept, log, and discard the packet.

ARP inspection uses the DHCP binding database filled from DHCP snooping as its list of binding IP addresses.

Let's recap; the DHCP binding database provides the expected MAC and IP address pair of untrusted hosts; ARP inspection will compare the source IP address and MAC address to the binding pair; if they are mismatched, it will drop the packet.

Below is a diagram showing a valid ARP request that matches both the binding database and request information.

```
Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: 02:c8:85:b5:5a:aa (02:c8:85:b5:5a:aa)
    Sender IP address: 10.10.0.1
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 10.10.239.154
```

```
Router# show ip dhcp snoop bind
MacAddress         IpAddress       Lease(sec) Type           VLAN Interface
------------------ --------------- ---------- -------------- ---- --------------------
02:c8:85:b5:5a:aa  10.10.0.1       23453      dhcp-snooping 10   GigabitEthernet1/1
01:02:03:04:05:06  2.2.2.2         69445      dhcp-snooping 20   GigabitEthernet2/1
```

Below is a diagram showing an invalid ARP request that does not match the binding database and requests information.

```
Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: 02:c8:85:b5:5a:aa (02:c8:85:b5:5a:aa)
    Sender IP address: 10.10.0.1
    Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
    Target IP address: 10.10.239.154
```

```
Router# show ip dhcp snoop bind
MacAddress         IpAddress       Lease(sec) Type           VLAN Interface
------------------ --------------- ---------- -------------- ---- --------------------
00:01:02:03:04:05  1.1.1.1         23453      dhcp-snooping 10   GigabitEthernet1/1
01:02:03:04:05:06  2.2.2.2         69445      dhcp-snooping 20   GigabitEthernet2/1
```

Where does DHCP snooping store leased IP addresses from untrusted hosts?

Answer: **DHCP Binding Database**

Will a switch drop or accept a DHCPRELEASE packet?

Answer: **Drop**

Does dynamic ARP inspection use the DHCP binding database? (Y/N)

Answer: **Y**

Dynamic ARP inspection will match an IP address and what other packet detail?

Answer: **MAC Address**

Task 8: **Conclusion**

In this room, we have expanded the common requirements of a well-designed network to include security standards.

We have discussed approaches and solutions for problems commonly encountered at layers two and three of the OSI model. It may be apparent that this does not solve all issues, and each environment may have different requirements for what it should monitor and block.

To continue exploring network security beyond design, complete the Network Security Protocols room. You will go beyond the layers we discussed and observe security from layers seven to three that we did not discuss in this room.

Tryhackme Walkthrough Tryhackme Writeup

Follow

Written by Daniel Schwarzentraub

116 Followers · 4 Following

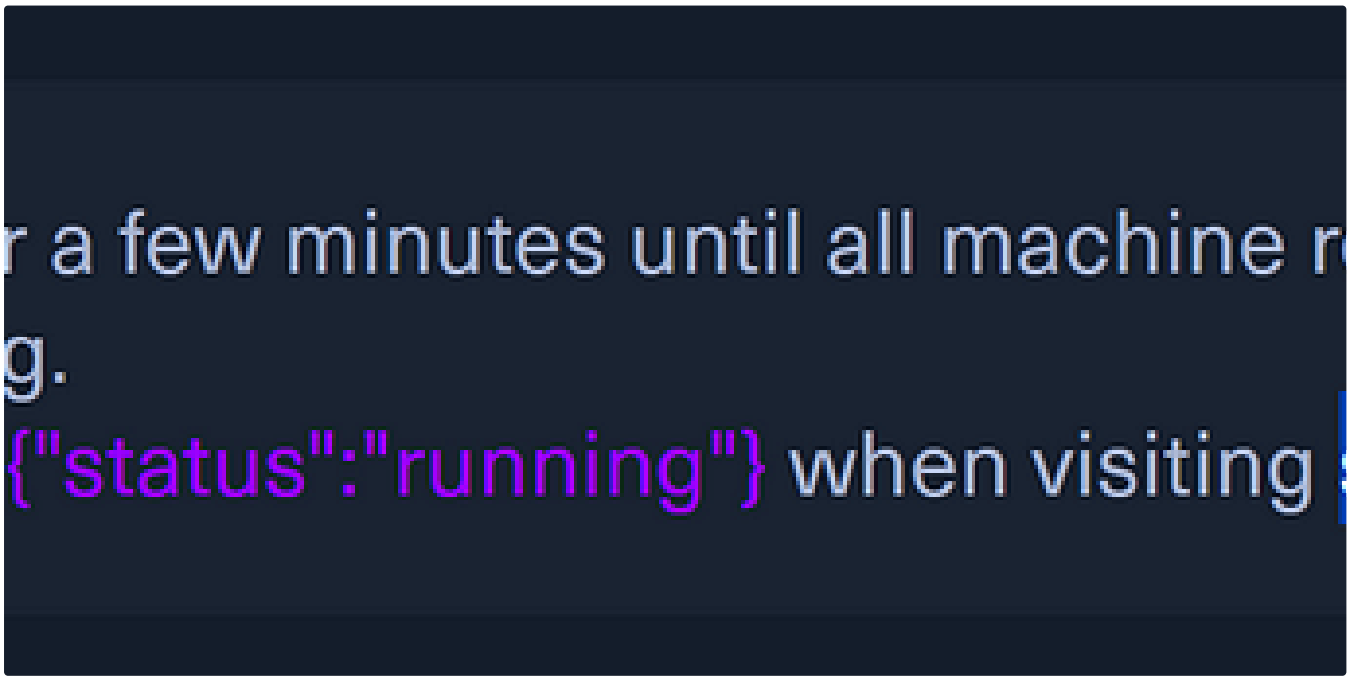
PNW_Hacker

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

s to introduce users to basic cryptography concepts such as:

n, such as AES
on, such as RSA
xchange

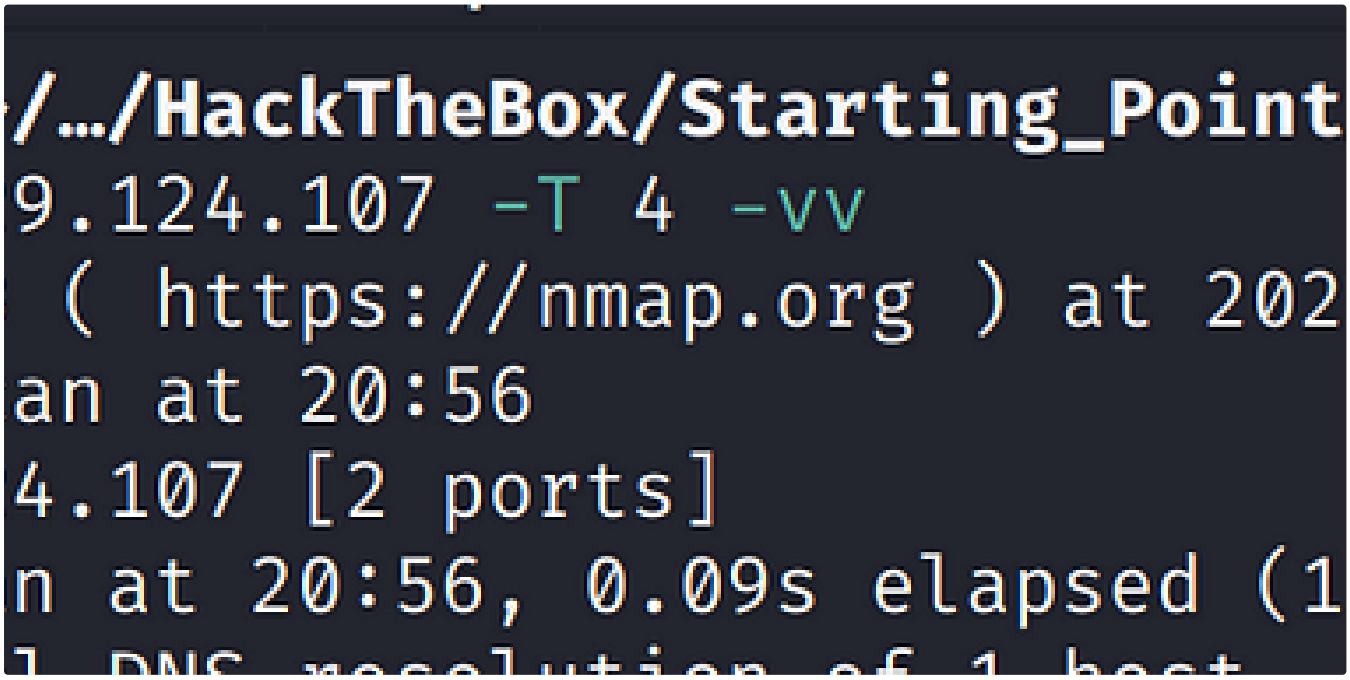
a message that no one can understand except the intended recip

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

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

Ansul Kotadia

Incident Response Process: TryHackMe Writeup

Task 1: Introduction

Nov 28, 2024 70 1

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

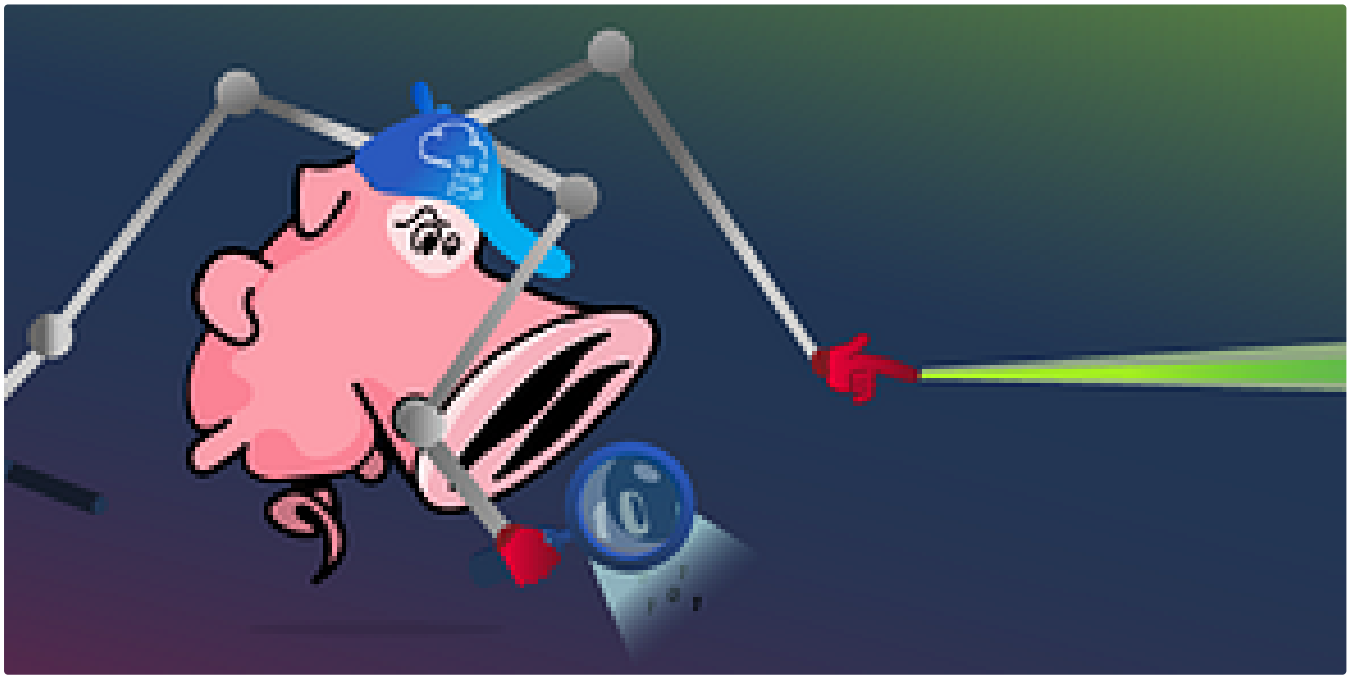
Productivity 101
20 stories · 2707 saves

In T3CH by Axoloth

TryHackMe | K8s Runtime Security | WriteUp

Secure a Kubernetes environment using in-house offerings and runtime security tools like Falco.

Sep 15, 2024 50



In T3CH by Axoloth

TryHackMe | Snort Challenge—The Basics | WriteUp

Put your snort skills into practice and write snort rules to analyse live capture network traffic

Nov 9, 2024 100

In T3CH by Axoloth

TryHackMe | Vulnerability Scanner Overview | WriteUp

Learn about vulnerability scanners and how they work in a practical scenario

Nov 23, 2024 50

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 more recommendations