Bridge VLAN Table

Summary

Since RouterOS v6.41 it is possible to use a bridge to filter out VLANs in your network. To achieve this, you should use the Bridge VLAN Filtering feature. This feature should be used instead of many known VLAN misconfigurations that are most likely causing you either performance issues or connectivity issues, you can read about one of the most popular misconfigurations in the VLAN in a bridge with a physical interface section. The most important part of the bridge VLAN filtering feature is the bridge VLAN table, which specifies which VLANs are allowed on each port, but configuring it might get quite complex if you are trying to make a more advanced setup, for generic setups you should be able to configure your device using the Trunk and Access ports example, but the purpose of this guide is to provide in-depth explanation and point out some of the behavior characteristics when using bridge VLAN Filtering.

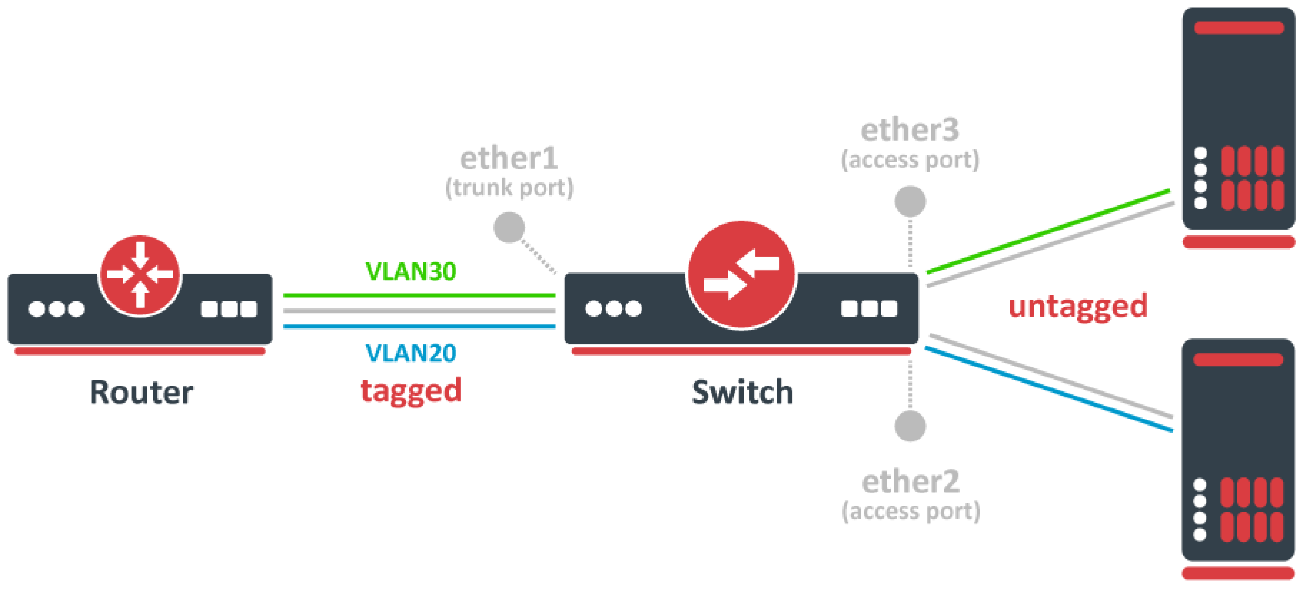
Background

Before explaining bridge VLAN filtering in-depth, you should understand a few basic concepts that are involved in bridge VLAN filtering.

- **Tagged/Untagged** - Under `/interface bridge vlan` menu, you can specify an entry that contains tagged and untagged ports. In general, tagged ports should be your trunk ports and untagged ports should be your access ports. By specifying a tagged port the bridge will always set a VLAN tag for packets that are being sent out through this port (egress). By specifying an untagged port the bridge will always remove the VLAN tag from egress packets.
- **VLAN-ids** - Under `/interface bridge vlan` menu, you can specify an entry in which certain VLANs are allowed on specific ports. The VLAN ID is checked on egress ports. If the packet contains a VLAN ID that does not exist in the bridge VLAN table for the egress port, then the packet is dropped before it gets sent out.
- **PVID** - The Port VLAN ID is used for access ports to tag all ingress traffic with a specific VLAN ID. A dynamic entry is added in the bridge VLAN table for every PVID used, the port is automatically added as an untagged port.
- **Ingress filtering** - By default, VLANs that don't exist in the bridge VLAN table are dropped before they are sent out (egress), but this property allows you to drop the packets when they are received (ingress).
- **Management access** - The bridge is supposed to simply forward packets between bridge ports and it would seem to other devices that there is simply a wire between them. With bridge VLAN filtering you can limit which packets are allowed to access the device that has the bridge configured, the most common practice is to allow access to the device only by using a very specific VLAN ID, but there are other ways you can grant access to the device. Management access is a great way to add another layer of security when accessing the device through a bridge port, this type of access is sometimes called the management port. For devices that support VLAN Filtering with hardware offloading, It is also related to the CPU port of a bridge.
- **CPU port** - Every device with a switch chip has a special purpose port called CPU port and it is used to communicate with the device's CPU. For devices that support VLAN filtering with hardware offloading, this port is the bridge interface itself. This port is mostly used to create management access but can be used for other purposes as well, for example, to route traffic between VLANs, to mark packets and apply queues.
- **frame-type** - You can filter out packets whether they have a VLAN tag or not, this is useful to add an extra layer of security for your bridge ports.
- **EtherType** - By default, a VLAN aware bridge will filter VLANs by checking the C-TAG (0x8100), all other VLAN tags are considered as untagged packets (without a VLAN tag). The selected EtherType will be used for VLAN filtering and VLAN tagging/untagging.
- **VLAN Tunnelling** - If the EtherType of the packet does not match with the EtherType configured for the bridge, then ingress packets are considered as untagged packets, this behavior gives a possibility to encapsulate VLANs into another, different VLAN. This also gives a possibility to divert specific traffic through different devices in your network.
- **Tag stacking** - If a packet has a VLAN tag that matches the EtherType, then the packet is considered as a tagged packet, but you can force another VLAN tag regardless of the packet's content. By setting `tag-stacking=yes` on a bridge port, you will add another VLAN tag with the PV ID value on top of any other tag for all ingress packets.

Trunk/Access port setup

Below you can find a very common diagram for a very typical type of setup that consists of a trunk port and multiple access ports:



This setup is very common since it gives the possibility to divide your network into multiple segments while using a single switch and maybe a single router, such a requirement is very common for companies that want to separate multiple departments. With VLANs you can use different DHCP Servers, which can give out an IP address from a different subnet based on the VLAN ID, which makes creating Firewall rules and QoS a lot easier.

In such a setup you would connect some generic devices like Desktop PCs to **ether2** and **ether3**, these can be considered as workstations and they generally only use untagged traffic (it is possible to force a VLAN tag for all traffic that is sent out a generic workstation, though it is not very common). To isolate some workstations from other workstations you must add a VLAN tag to all packets that enter **ether2** or **ether3**, but to decide what VLAN ID should the packet get is to use a concept called **Port-based VLANs**. In this concept packets get a VLAN tag with a VLAN ID based on the bridge port to which the device is connected. For example, in this setup the device on **ether2** will get a VLAN tag with **VLAN20** and the device on **ether3** will get a VLAN tag with **VLAN30**, this concept is very scalable as long as you have enough bridge ports. This should give you the understanding that traffic between the bridge and devices behind **ether2/ether3** is untagged (since there is no VLAN tag, hence the name).

When we have determined our untagged ports, we can now determine our tagged ports. Tagged ports are going to be the trunk ports (the port, that carries multiple VLANs) and usually this port is connected to a router or another switch/bridge, you can have multiple trunk ports as well. Tagged ports are always carrying packets with a VLAN tag (hence the name) and you must **ALWAYS** specify the tagged ports for each VLAN ID you want this port to forward. It is possible that a port is a tagged port for one VLAN ID and the same port is an untagged port for a different VLAN ID, but this is for a different type of setup (Hybrid port setup).

Special note must be added for the PVID property. This property should be used on access ports, but it can be used for trunk ports as well (in Hybrid port setup). By using the PVID property you are adding a new VLAN tag with a VLAN ID that is specified in the PVID to all **UNTAGGED** packets that are received on that specific bridge port. The PVID does not have any effect on tagged packets, this means that, for example, if a packet with a VLAN tag of **VLAN40** is received on **ether2** that has `PVID=20`, then the VLAN tag is **NOT** changed and forwarding will depend on the entries from the bridge VLAN table.

To configure the trunk/access port setup, you need to first create a bridge:

```
/interface bridge
add name=bridge1
```

Don't enable VLAN filtering yet as you might get locked out from the device because of the lack of management access, which is configured at the end.

Add the bridge ports and specify PVID for each access port:

```
/interface bridge port
add bridge=bridge1 interface=ether1
add bridge=bridge1 interface=ether2 pvid=20
add bridge=bridge1 interface=ether3 pvid=30
```

PVID has no effect until VLAN filtering is enabled.

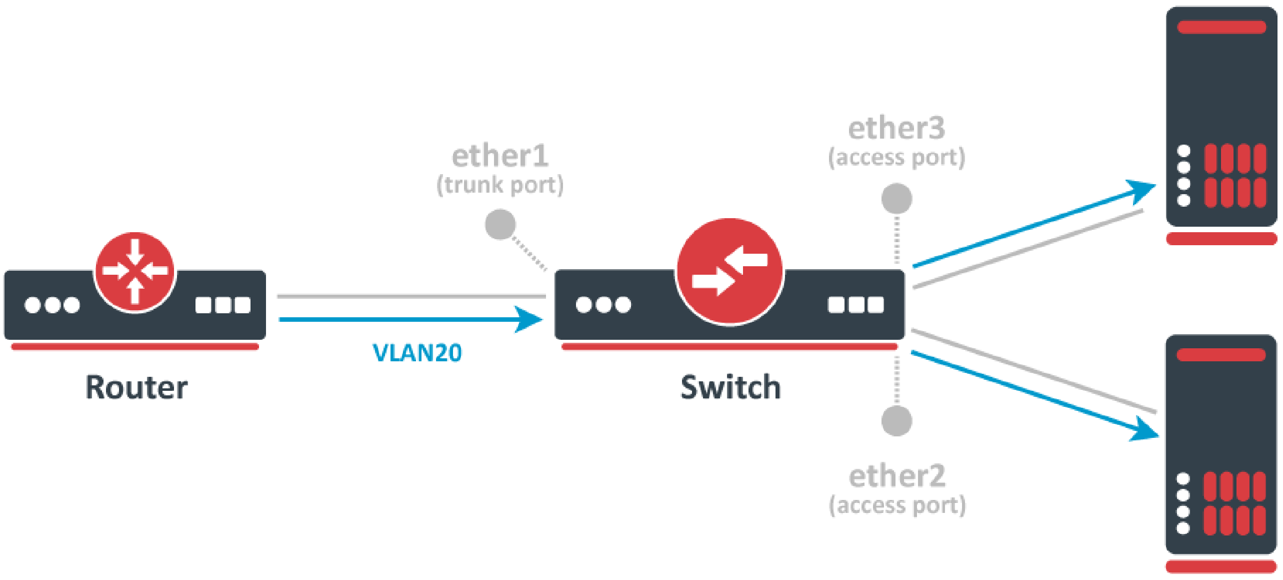
Add appropriate entries in the bridge VLAN table:

```
/interface bridge vlan
add bridge=bridge1 tagged=ether1 untagged=ether2 vlan-ids=20
add bridge=bridge1 tagged=ether1 untagged=ether3 vlan-ids=30
```

You might think that you could simplify this entry with a single entry, similar to this:

```
/interface bridge vlan
add bridge=bridge1 tagged=ether1 untagged=ether2,ether3 vlan-ids=20,30
```

Do **NOT** use multiple VLAN IDs on access ports. This will unintentionally allow both **VLAN20** and **VLAN30** on both access ports. In the example above, **ether3** is supposed to set a VLAN tag for all ingress packets to use **VLAN30** (since `PVID=30`), but this does not limit the allowed VLANs on this port when VLANs are being sent out through this port. The bridge VLAN table is responsible for deciding whether a VLAN is allowed to be sent through a specific port or not. The entry above specifies that both **VLAN20** and **VLAN30** are allowed to be sent out through **ether2** and **ether3** and on top of that the entry specifies that packets should be sent out without a VLAN tag (packets are sent out as untagged packets). As a result, you may create a packet leak from VLANs to ports that are not even supposed to receive such traffic, see the image below.



Misconfigured VLAN table allows VLAN20 to be sent through ether3, it will also allow VLAN30 through ether2

Don't use more than one VLAN ID specified in a bridge VLAN table entry for access ports, you should only specify multiple VLAN IDs for trunk ports.

It is not necessary to add a bridge port as an untagged port, because each bridge port is added as an untagged port dynamically with a VLAN ID that is specified in thePVIDproperty. This is because of a feature that automatically will add an appropriate entry in the bridge VLAN table for convenience and performance reasons, this feature does have some caveats that you must be aware of. All ports that have the samePVIDwill be added to a single entry for the appropriate VLAN ID as untagged ports, but note that the **Bridge interface** also has a VLAN ID.

For testing purposes, we are going to enable VLAN filtering, but note that it might make you lose access to the device since it does not have management access configured yet (we will configure it later). It is always recommended to configure VLAN filtering while using a serial console, though you can also configure a device through a port, that is not added to a bridge. Make sure you are using a serial console or connected through a different port (that is not in a bridge) and enable VLAN filtering:

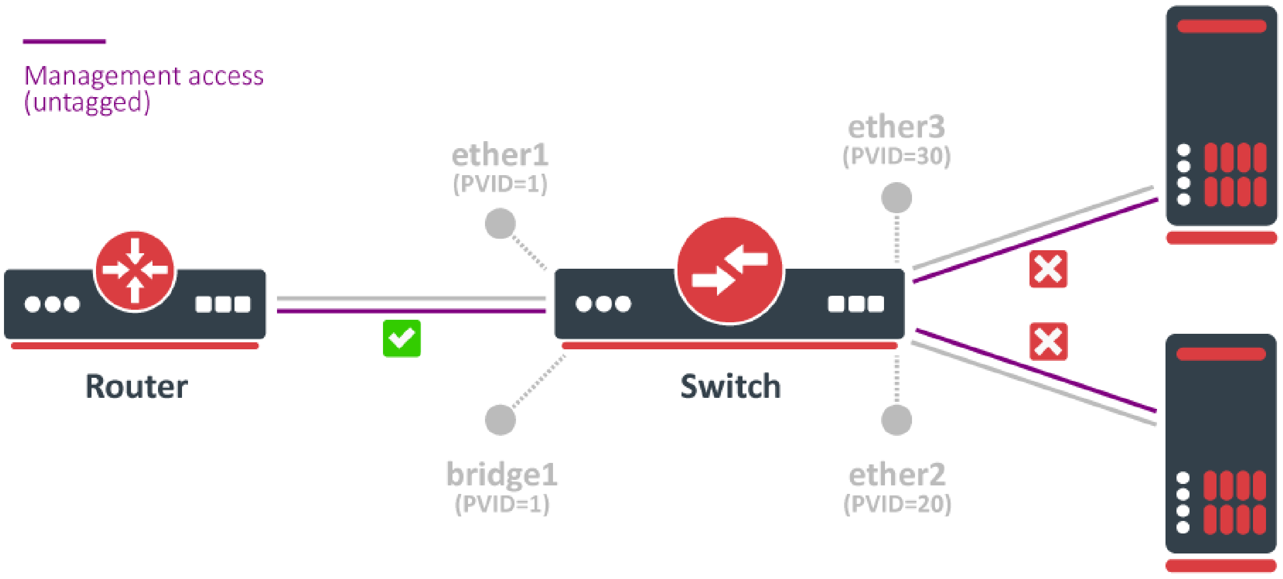
```
/interface bridge set bridge1 vlan-filtering=yes
```

⚠ You might not lose access to the device as soon as you enable VLAN filtering, but you might get disconnected since the bridge must reset itself in order for VLAN filtering to take any effect, which will force you to reconnect (this is mostly relevant when using MAC-telnet). There is a chance you might be able to access your device using untagged traffic, this scenario is described below.

If you have enabled VLAN filtering now and printed out the current VLAN table, you would see such a table:

```
[admin@MikroTik] > /interface bridge vlan print
Flags: X - disabled, D - dynamic
 #   BRIDGE                     VLAN-IDS  CURRENT-TAGGED       CURRENT-UNTAGGED
 0   bridge1                    20        ether1               ether2
 1   bridge1                    30        ether1               ether3
 2 D bridge1                    1                              bridge1
                                                               ether1
```

There is a dynamic entry added for **VLAN1** since `PVID=1` is set by default to all bridge ports (including our trunk port, **ether1**), but you should also notice that the **bridge1** interface (the CPU port) is also added dynamically. You should be aware that **bridge1** is also a bridge port and therefore might get added to the bridge VLAN table dynamically. There is a chance that you might unintentionally allow access to the device because of this feature. For example, if you have followed this guide and left **PVID=1** set for the trunk port (**ether1**) and did not change the PVID for the CPU port (**bridge1**) as well, then access through **ether1** to the device using untagged traffic is allowed, this is also visible when you print out the bridge VLAN table. This scenario is illustrated in the image below:



Unintentionally allowed management access using untagged traffic through the trunk port

ⓘ Always check the bridge VLAN table if you have not unintentionally allowed certain VLANs or untagged traffic to specific ports, especially the CPU port (bridge).

There is a simple way to prevent the bridge (CPU port) from being added as an untagged port, you can simply set the PVID on the trunk port to be different than the bridge's PVID (or change the bridge's PVID), but there is another option, which is more intuitive and recommended. Since you are expecting that the trunk port is only supposed to receive tagged traffic (in this example, it should only receive **VLAN20/VLAN30**), but no untagged traffic, then you can use ingress-filtering along with frame-type to filter out unwanted packets, but in order to fully understand the behavior of ingress filtering, we must first understand the details of management access.

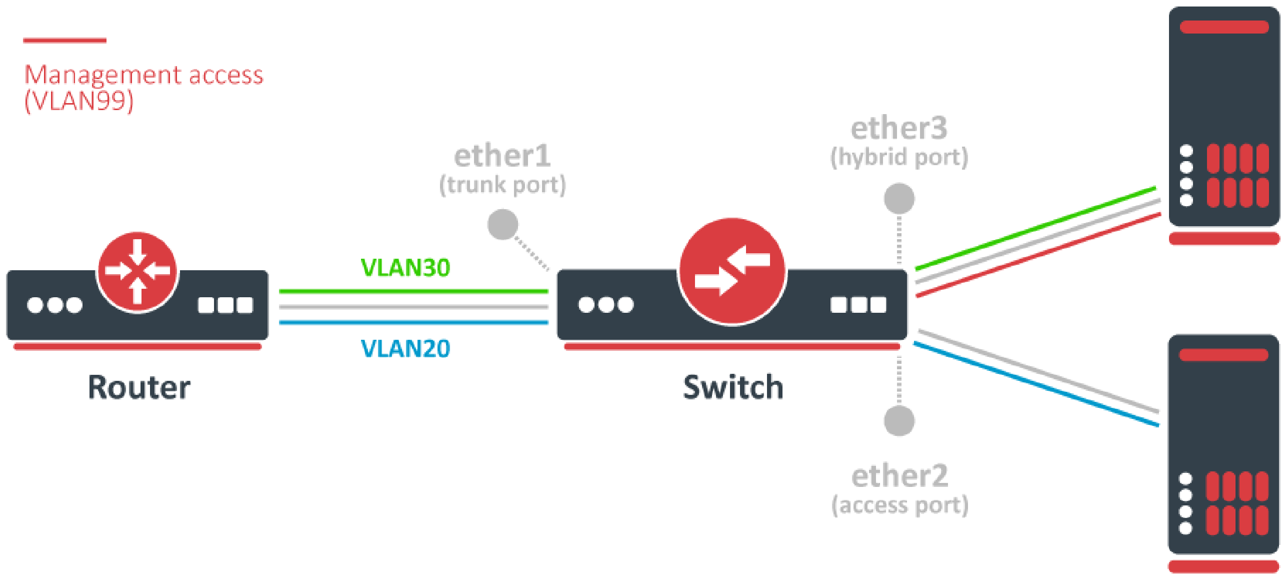
Management access is used to create a way to access a device through a bridge that has VLAN filtering enabled. You could simply allow untagged access and to do so it is fairly simple. Let us say you wanted the workstation behind **ether3** to be able to access the device, we assumed before that the workstation is a generic computer that will not use tagged packets and therefore will only send out untagged packets, this means that we should add the CPU port (**bridge1**) as an untagged interface to the bridge VLAN table, to do so, simply use the same PVID value for the **bridge1** and **ether3** ports and set both ports as untagged members for the VLAN ID. In this case, you are going to connect from **ether3** that has `PVID=30`, so you change the configuration accordingly:

```
/interface bridge set [find name=bridge1] pvid=30
/interface bridge vlan set [find vlan-ids=30] untagged=bridge1,ether3
```

You can use the feature that dynamically adds untagged ports with the same PVID value, you can simply change the PVID to match between **et her3** and **bridge1**.

Allowing access to the device using untagged traffic is not considered a good security practice, a much better way is to allow access to the device using a very specific VLAN sometimes called the management VLAN, in our case, this is going to be **VLAN99**. This adds a significant layer of security since an attacker must guess the VLAN ID that is being used for management purposes and then guess the login credentials, on top of this you can even add another layer of security by allowing access to the device using only certain IP addresses. The purpose of this guide is to provide an in-depth explanation, for that reason we are adding a level of complexity to our setup to understand some possible caveats that you must take into account. We are going to allow access from an access port using tagged traffic (illustrated in the image below). In order to allow access to the device using **VLAN99** from **ether3**, we must add a proper entry in the bridge VLAN table. Additionally, a network device connected to ether3 must support VLAN tagging.

```
/interface bridge vlan
add bridge=bridge1 tagged=bridge1,ether3 vlan-ids=99
```



Management access using tagged traffic through an access port (which makes it a hybrid port)

If PVID for ether1 and bridge1 matches (by default, it does match with 1), then access to the device is allowed using untagged traffic from ether1 because of the feature that dynamically adds untagged ports to the bridge VLAN table.

But you might notice that access using **VLAN99** does not work at this point, this is because you need a VLAN interface that listens for tagged traffic, you can simply create this interface for the appropriate VLAN ID and you can set an IP address for the interface as well:

```
/interface vlan
add interface=bridge1 name=VLAN99 vlan-id=99
/ip address
add address=192.168.99.2/24 interface=VLAN99
```

Our access port (**ether3**) at this point expects tagged and untagged traffic at the same time, such a port is called a **hybrid port**.

At this point, we can benefit from using ingress-filtering and frame-type. First, we are going to focus on frame-type, which limits the allowed packet types (tagged, untagged, both), but in order for frame-type to work properly, ingress-filtering must be enabled, otherwise it will not have any effect. In our example, where we wanted to allow access from **ether3** using tagged traffic (**VLAN99**) and at the same time allow a generic workstation to access the network, we can conclude that this port needs to allow tagged and untagged packets, but **ether1** and **ether2** is supposed to receive only specific types of packets, for this reasons we can enhance our network's security. Since **ether1** is our trunk port, it is only supposed to carry tagged packets, but **ether2** is our access port so it should not carry any tagged packets, based on these conclusions we can drop invalid packets:

```
/interface bridge port
set [find where interface=ether1] ingress-filtering=yes frame-types=admit-only-vlan-tagged
set [find where interface=ether2] ingress-filtering=yes frame-types=admit-only-untagged-and-priority-tagged
```

Let's say that you forgot to enable ingress-filtering and change the frame-type property on **ether1**, this would unintentionally add access to the device through **ether1** using untagged traffic since PVID matches for **bridge1** and **ether1**, but you are expecting only tagged traffic to be able to access the device. It is possible to drop all untagged packets that are destined to the **CPU port**:

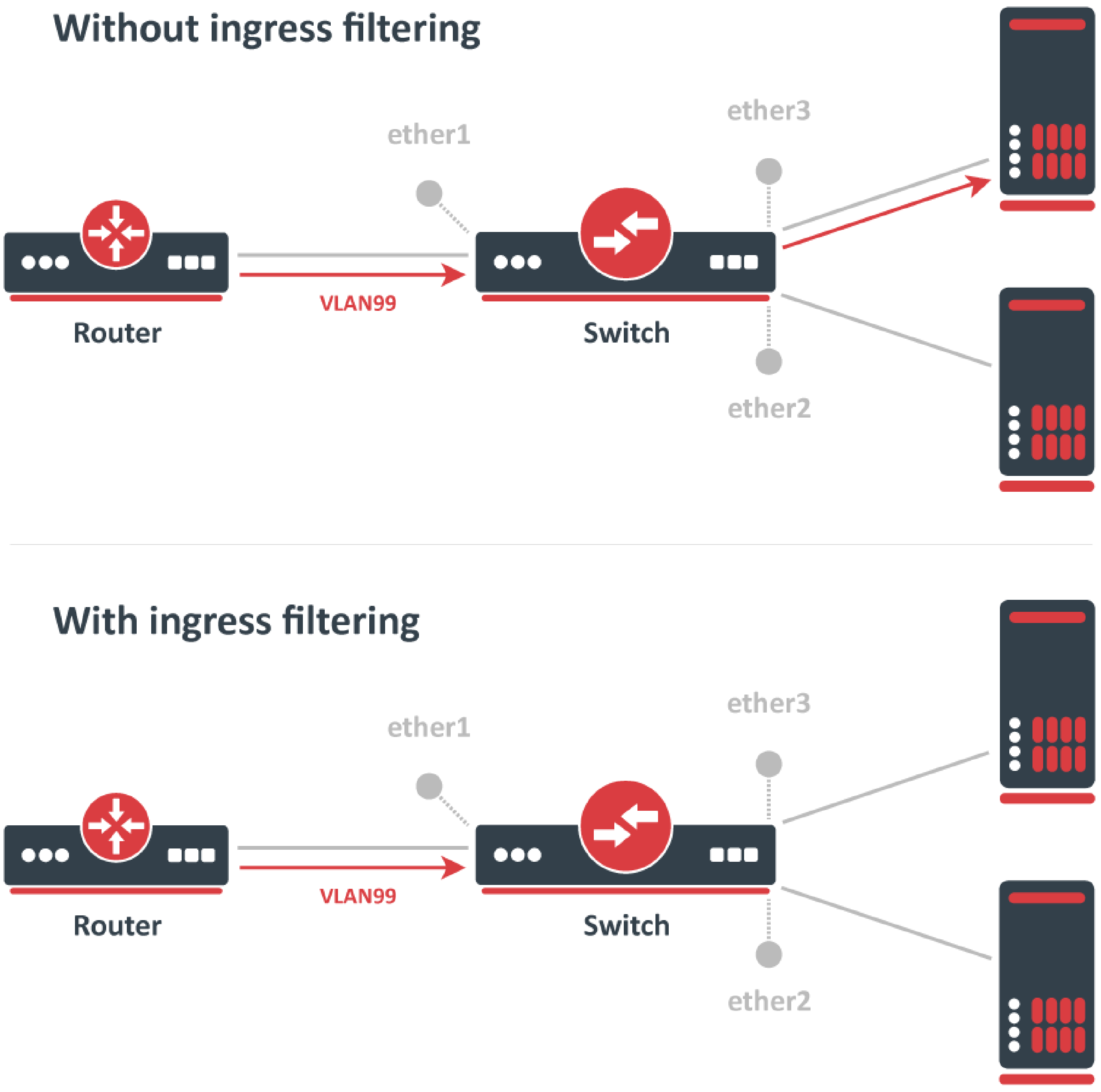
```
/interface bridge
set bridge1 frame-types=admit-only-vlan-tagged ingress-filtering=yes
```

This does not only drop untagged packets, but disables the feature that dynamically adds untagged ports to the bridge VLAN table. If you print out the current bridge VLAN table you would notice that **bridge1** is not dynamically added as an untagged port:

```
[admin@MikroTik] > /interface bridge vlan print
Flags: X - disabled, D - dynamic
 #   BRIDGE       VLAN-IDS  CURRENT-TAGGED        CURRENT-UNTAGGED
 0   bridge1      20        ether1
 1   bridge1      30        ether1                ether3
 2 D bridge1      1                               ether1
 3   bridge1      99        bridge1
                            ether3
```

When `frame-type=admit-only-vlan-tagged` is used on a port, then the port is not dynamically added as an untagged port for the PVID.

While frame-type can be used to drop a certain type of packet, the ingress-filtering can be used to filter out packets before they can be sent out. To fully understand the need for ingress filtering, consider the following scenario: **VLAN99** is allowed on **ether3** and **bridge1**, but you can still send **VLAN99** traffic from **ether1** to **ether3**, this is because the bridge VLAN table checks if a port is allowed to carry a certain VLAN only on egress ports. In our case, **ether3** is allowed to carry **VLAN99** and for this reason, it is forwarded. To prevent this you **MUST** use ingress-filtering. With ingress filtering, ingress packets are also checked, in our case the bridge VLAN table does not contain an entry that **VLAN99** is allowed on **ether1** and therefore will be dropped immediately. Of course, in our scenario without ingress filtering connection cannot be established since **VLAN99** can be forwarded only from **ether1** to **ether3**, but not from **e ther3** to **ether1**, though there are still possible attacks that can be used in such a misconfiguration (for example, ARP poisoning). The packet dropping behavior is illustrated in the image below:



Trunk/access port setup with and without ingress filtering. Ingress filtering can prevent unwanted traffic from being forwarded. Note that ether1 is not allowed to carry VLAN99 in the bridge VLAN table.

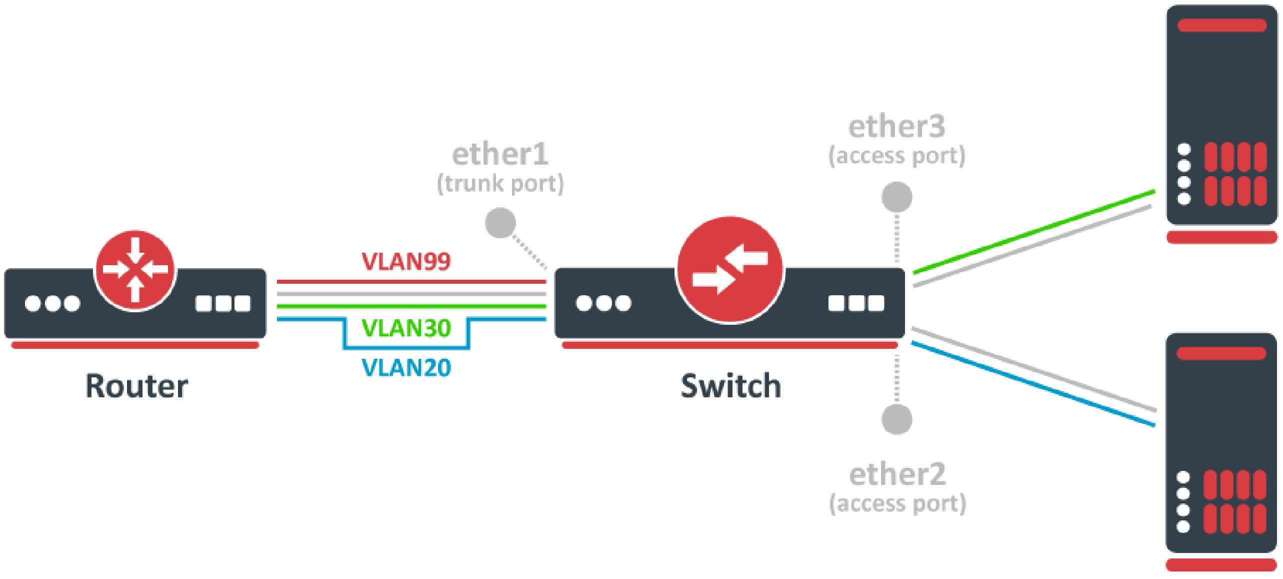
Always try to use ingress-filtering wherever it is possible, it adds a significant layer of security.

The ingress-filtering can be used on the **CPU port** (bridge) as well, this can be used to prevent some possible attack vectors and limit the allowed VLANs that can access the CPU. It is better to drop a packet on an ingress port, rather than on an egress port, this reduces the CPU load, this is quite crucial when you are using hardware offloading with bridge VLAN filtering.

The ingress-filtering property only has an effect on ingress traffic, but frame-type has an effect on egress and ingress traffic.

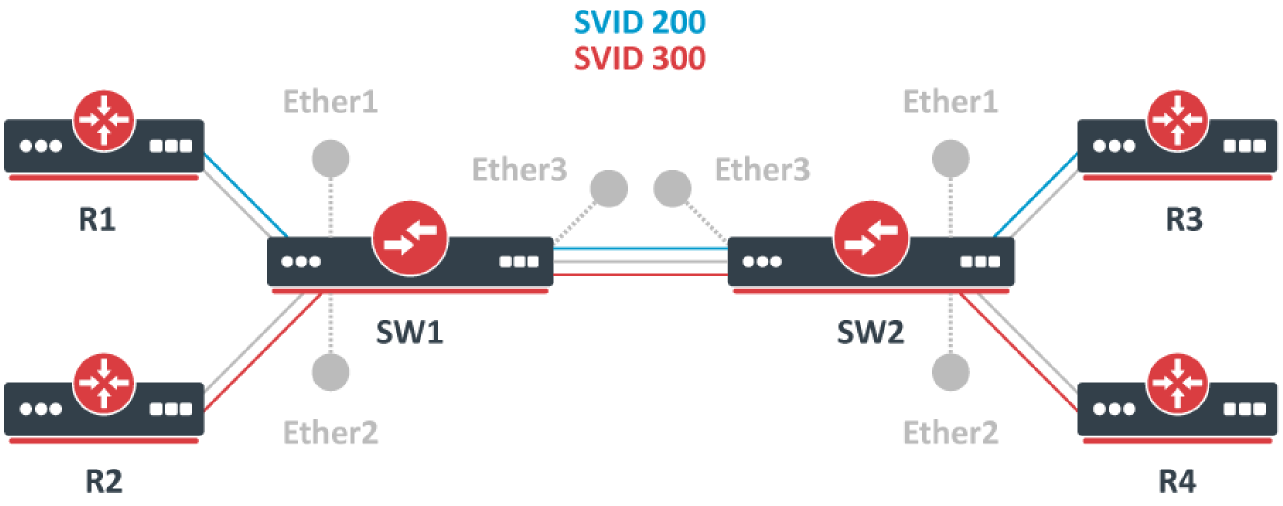
Even though you can limit the allowed VLANs and packet types on a port, it is never a good security practice to allow access to a device through access ports since an attacker could sniff packets and extract the management VLAN's ID, you should only allow access to the device from the trunk port (**ether1**) since trunk ports usually have better physical security, you should remove the previous entry and allow access to the device through the port that is connected to your router (illustrated in the image below):

```
/interface bridge vlan
add bridge=bridge1 tagged=bridge1,ether1 vlan-ids=99
```



VLAN Tunnelling setup

In some cases you might want to forward already tagged traffic through certain switches. This is a quite common setup for backbone infrastructures since it provides a possibility encapsulate traffic from, for example, your edge routers and seamlessly forward it over your backbone to another edge router. Below you can find an example of a VLAN tunneling topology:



Provider bridge topology

⚠ To fully understand how to configure VLAN tunneling properly, you should first read the Trunk/Access port setup section before proceeding any further.

Since RouterOS v6.43 there are two possible ways to achieve this, one is the standardized IEEE 802.1ad way, the other way is using **Tag stacking**, we will first review the standardized way since the same principles apply to both ways and only a couple of parameters must be changed to use the other method. The way VLAN tunneling works is that the bridge checks if the outer VLAN tag is using the same VLAN tag as specified as ether-type. If the VLAN tag matches, the packet is considered as a tagged packet, otherwise it is considered as an untagged packet.

⚠ The bridge checks only the outer tag (closest to the MAC address), any other tag is ignored anywhere in a bridge configuration. The bridge is not aware of the packet contents, even though there might be another VLAN tag, only the first VLAN tag is checked.

The ether-type property allows you to select the following EtherTypes for the VLAN tag:

- 0x88a8 - SVID, IEEE 802.1ad, Service VLAN
- 0x8100 - CVID, IEEE 802.1Q, Customer VLAN
- 0x9100 - Double tagged (not very common)

In order to properly configure bridge VLAN filtering, you must understand how does the bridge distinguish between tagged and untagged packets. Like mentioned before, the bridge will check if EtherType matches with the outer VLAN tag in the packet. For example, consider the following packet:

```
FFFFFFFFFFFF 6C3B6B7C413E 8100 6063 9999
----------------------------------------
DST-MAC = FFFFFFFFFFFF
SRC-MAC = 6C3B6B7C413E
Outer EtherType = 8100 (IEEE 802.1Q VLAN tag)
VLAN priority = 6
VLAN ID = 99 (HEX = 63)
Inner EtherType = 9999
```

Let us assume that we have set **`ether-type=0x88a8`**, in this case, the packet above is going to be considered untagged since the bridge is looking for a different VLAN tag. Lets now consider the following packet:

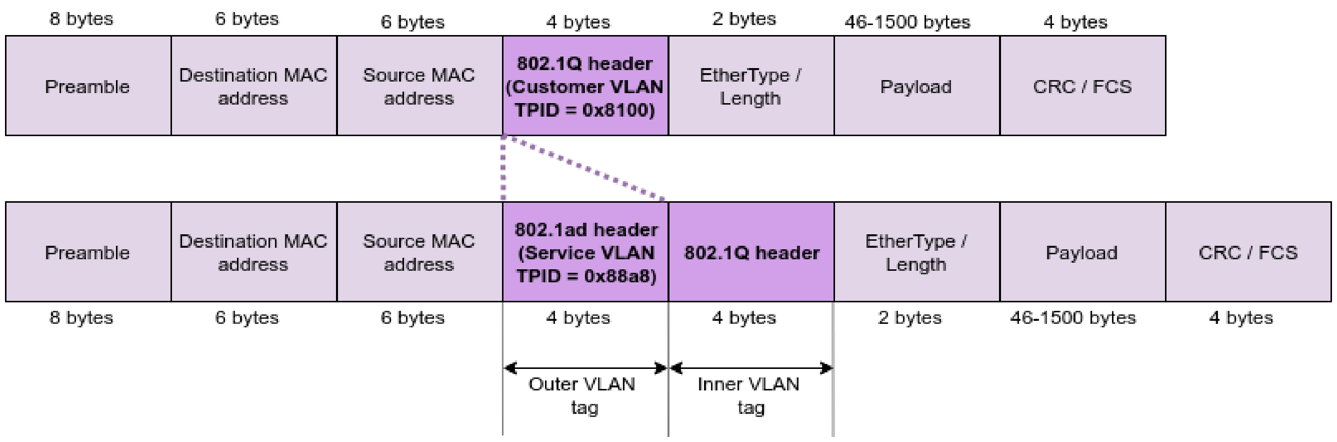
```
FFFFFFFFFFFF 6C3B6B7C413E 88A8 6063 8100 5062 9999
----------------------------------------
DST-MAC = FFFFFFFFFFFF
SRC-MAC = 6C3B6B7C413E
Outer EtherType = 88A8 (IEEE 802.1ad VLAN tag)
VLAN priority = 6
VLAN ID = 99 (HEX = 63)
Inner EtherType 1 = 8100 (IEEE 802.1Q VLAN tag)
VLAN priority = 5
VLAN ID = 98 (HEX = 62)
Innter EtherType 2 = 9999
```

This time let us assume that we have set **`ether-type=0x8100`**, in this case, the packet above is considered as untagged as well since the outer tag is using an IEEE 802.1ad VLAN tag. The same principles apply to other VLAN related functions, for example, the PVID property will add a new VLAN tag on access ports and the VLAN tag will be using the EtherType specified in ether-type.

Both **SW1** and **SW2** are using the same configuration:

```
/interface bridge
add name=bridge1 vlan-filtering=yes ether-type=0x88a8
/interface bridge port
add interface=ether1 bridge=bridge1 pvid=200
add interface=ether2 bridge=bridge1 pvid=300
add interface=ether3 bridge=bridge1
/interface bridge vlan
add bridge=bridge1 tagged=ether3 untagged=ether1 vlan-ids=200
add bridge=bridge1 tagged=ether3 untagged=ether2 vlan-ids=300
```

In this example, we are assuming that all routers are passing traffic that is using a CVID VLAN tag (the inner VLAN tag). Such traffic on switches will be considered as untagged traffic based on the principle described above. Switches will encapsulate this traffic using an SVID VLAN tag (the outer VLAN tag) and traffic between **SW1** and **SW2** is going to be considered as tagged. Before traffic reaches its destination, the switches will decapsulate the outer tag and forward the original CVID VLAN tagged frame to routers. See a packet example below:



A packet example before and after SVID VLAN encapsulation

All principles that apply to the regular trunk/access port setup using IEEE 802.1Q also apply to VLAN tunneling setups, make sure you are limiting VLANs and packet type properly using the bridge VLAN table and ingress filtering.

In case you want to create management access from, let's say, **ether3** to the device and wanted to use **VLAN99**, then you would use such commands:

```
/interface bridge vlan
add bridge=bridge1 tagged=bridge1,ether3 vlan-ids=99
/interface vlan
add interface=bridge1 name=VLAN99 use-service-tag=yes vlan-id=99
/ip address
add address=192.168.99.2/24 interface=VLAN99
```

As you may notice, the only difference is that the VLAN interface is using `use-service-tag=yes`, this sets the VLAN interface to listen to SVID (IEEE 802.1ad) VLAN tags. This will require you to use the IEEE 802.1ad VLAN tag to access the device using the management VLAN. This means that you will not be able to connect to the device using a CVID VLAN tag while using bridge VLAN filtering, the ether-type is set globally and will have an effect on all bridge VLAN filtering functions.

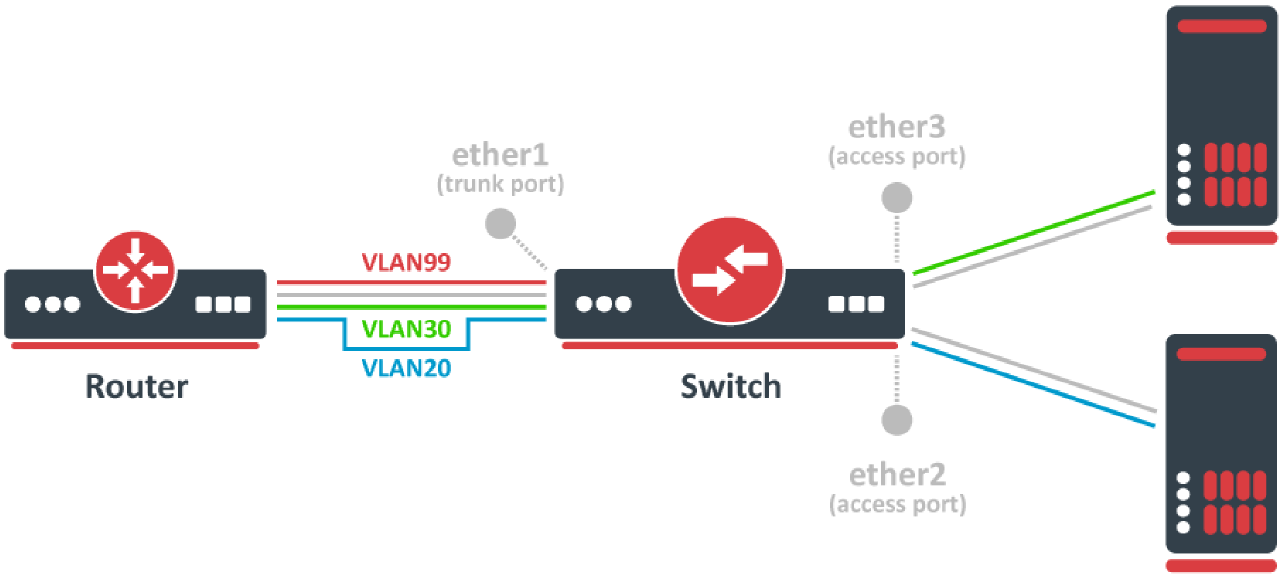
Devices with switch chip Marvell-98DX3257 (e.g. CRS354 series) do not support VLAN filtering on 1Gbps Ethernet interfaces for other VLAN types (`0x88a8` and `0x9100`).

Tag Stacking

In the VLAN Tunnelling setup, we were adding a new VLAN tag that was different from the VLAN tag, but it is possible to add a new VLAN tag regardless of the packet contents. The difference between the regular VLAN tunneling setup is that the bridge does not check if the packet is tagged or untagged, it assumes that all packets that are received on a specific port are all untagged packets and will add a new VLAN tag regardless of whether a VLAN tag is present or not, this is called **Tag Stacking** since it "stacks" VLAN tags on top of the previous tag, regardless of the VLAN tag type. This is a very common setup for networks that do not support the IEEE 802.1ad standard, but still want to encapsulate VLAN traffic into a new VLAN.

The VLAN tag that is going to be added depends on ether-type and PVID. For example, if you have `ether-type=0x8100` and `PVID=200` on a port, then the bridge will add a new IEEE 802.1Q VLAN tag right on top of any other tag (if such are present). The same VLAN filtering principles still apply, you have to determine which ports are going to be your trunk ports and mark them as a tagged port, determine your access ports and add them as untagged ports.

To explain how VLAN tagging and untagging works with tag stacking, let us use the same network topology as before:



What we want to achieve is that regardless of what is being received on **ether2** and **ether3**, a new VLAN tag will be added to encapsulate the traffic that is coming from those ports. What tag-stacking does is forces a new VLAN tag, so we can use this property to achieve our desired setup. We are going to be using the same configuration as in the Trunk/Access port setup, but with tag, stacking enabled on the access ports:

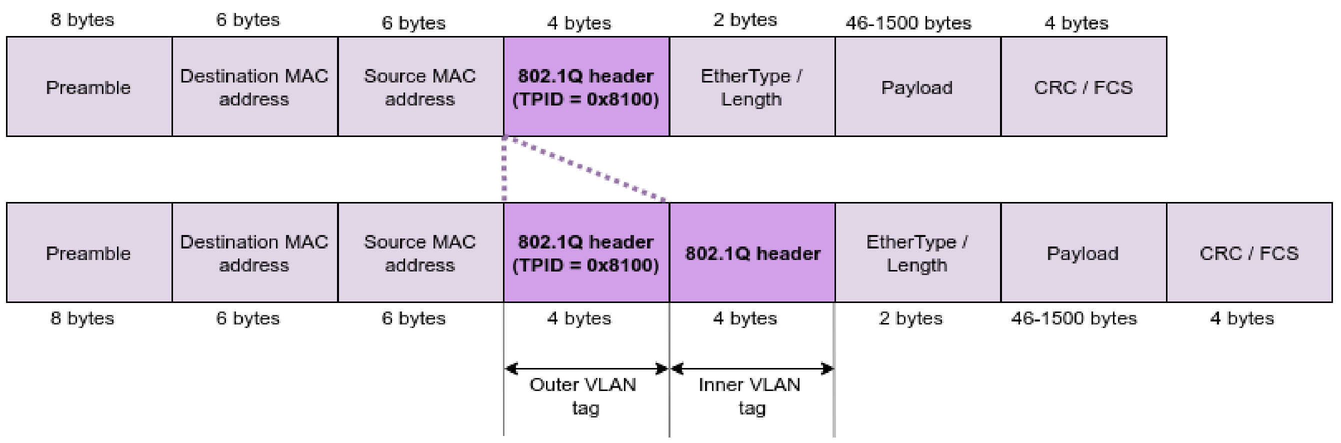
```
/interface bridge
add name=bridge1 vlan-filtering=yes ether-type=0x8100
/interface bridge port
add bridge=bridge1 interface=ether1
add bridge=bridge1 interface=ether2 tag-stacking=yes pvid=20
add bridge=bridge1 interface=ether3 tag-stacking=yes pvid=30
/interface bridge vlan
add bridge=bridge1 tagged=ether1 untagged=ether2 vlan-ids=20
add bridge=bridge1 tagged=ether1 untagged=ether3 vlan-ids=30
```

The added VLAN tag will use the specified ether-type. The selected EtherType will also be used for VLAN filtering. Only the outer tag is checked, but with tag-stacking in place, the tag checking is skipped and assumes that a new tag must be added either way.

Let us assume that the devices behind **ether2** and **ether3** are sending tagged **VLAN40** traffic. With this configuration, **ALL** packets will get encapsulated with a new VLAN tag, but you must make sure that you have added the VLAN ID from the outer tag to the bridge VLAN table. The **VLAN40** is not added to the bridge VLAN table since it is the inner tag and it is not checked, we are only concerned about the outer tag, which is either **VLAN20** or **VLAN30** depending on the port.

Similar to other setups, the bridge VLAN table is going to be used to determine if the VLAN tag needs to be removed or not. For example, **ether1** receives tagged **VLAN20** packets, the bridge checks that **ether2** is allowed to carry **VLAN20** so it is about to send it out through **ether2**, but it also checks the bridge VLAN table whether the VLAN tag should be removed and since **ether2** is marked as an untagged port, then the bridge will forward these packets from **ether1** to **ether2** without the **VLAN20** VLAN tag.

From the access port perspective, the same principles as in the Trunk/Access port setup apply. All packets that are received on **ether2** will get a new VLAN tag with the VLAN ID that is specified in PVID, in this case, a new VLAN tag will be added with **VLAN20** and this VLAN will be subjected to VLAN filtering. See a packet example below:



A packet example before and after tag stacking