Port knocking

All available public IP addresses are constantly being port scanned by bots and services like shodan.io and anyone can use this information to perform brute-force attacks and execute any known exploits. Port knocking is a cost-effective way to defend against this by not exposing any ports and simply listening to connection attempts - if the correct sequence of port connection attempts is made, the client is considered safe and added to a list of secured address list that bypass the WAN firewall rules.

Setup example

We are assuming you have already set up a firewall that drops all connection attempts from the WAN port, so you will need to add additional rules before that.
First, create a firewall rule that listens on a given port and adds the connected source IP to an address list - this is the first knock.

```
/ip/firewall/filter add action=add-src-to-address-list address-list=888 address-list-timeout=30s chain=input
dst-port=888 in-interface-list=WAN protocol=tcp
```

Then add a rule that does the same on another port, but only approves IPs that are already in the first list. You can repeat this step as many times as you like.

```
/ip/firewall/filter add action=add-src-to-address-list address-list=555 address-list-timeout=30s chain=input
dst-port=555 in-interface-list=WAN protocol=tcp src-address-list=888
```

Finally, the last knock will be added to an IP list that is trusted and any input is accepted.

```
/ip/firewall/filter add action=add-src-to-address-list address-list=secured address-list-timeout=30m
chain=input dst-port=222 in-interface-list=WAN protocol=tcp src-address-list=555
/ip/firewall/filter add action=accept chain=input in-interface-list=WAN src-address-list=secured
```

Knock to gain access

To access the board from WAN, a port-knocking client could be used, but a simple bash one-liner with nmap can do the job.

```
for x in 888,555,222; do nmap -p $x -Pn xx.xx.xx.xx; done
```

Blacklists

Unless you are using a lot of knocks, a simple port scan could accidentally trigger the correct ports in the correct order, so it is advisable to add a blacklist as well.

At the very top of your firewall stack add a drop rule for the blacklist.

```
/ip/firewall/filter add action=drop chain=input disabled=yes in-interface-list=WAN src-address-list=blacklist
```

Then add suspicious IPs to the blacklist.

Bad ports - ones that will never be used by a trusted user and hence have a high timeout penalty.

```
/ip/firewall/filter add action=add-src-to-address-list address-list=blacklist address-list-timeout=1000m
chain=input disabled=yes dst-port=666 in-interface-list=WAN protocol=tcp
```

Ports that slow down the port scanning process significantly to the point where it is pointless, but will never lock out a real user for too long. This could include every single port apart from the 'knock' ports, the key is that the source IP is not already in the secure list and hence those ports can be used after a successful knock.

```
/ip/firewall/filter add action=add-src-to-address-list address-list=blacklist address-list-timeout=1m
chain=input disabled=yes dst-port=21,22,23,8291,10000-60000 in-interface-list=WAN protocol=tcp src-address-
list=!secured
```

⚠ Blacklist rules from this section are added **disabled=yes** in order to avoid locking out the user. Enable the filter rules, once the alternative access available or use <Safe Mode>

Use a passphrase for each knock

You could go even further by sending a passphrase with each knock.

> **Warning**
>
> Layer7 rules are very resource-intensive. Do not use it unless you know what you are doing.

Then create a layer7 regex check that can be requested on the knock rule.

/ip firewall layer7-protocol add name=pass regexp="^passphrase/$"
/ip firewall filter
add action=add-src-to-address-list address-list=888 address-list-timeout=30s chain=input dst-port=888 in-interface-list=WAN protocol=udp layer7-protocol=pass

⚠ For additional security layer see the Bruteforse prevention article: Bruteforce prevention