Access your Raspberry Pi from anywhere

Raspberry Pi is an ideal platform for setting up personal services, like Git daemon, remote backup server, proxy server… you name it. However, while the device is small and handy it is still not so portable as a server, because conventional server installation usually includes domain name registration, setting up static IP addresses and configuring port forwarding on a router.

In this article I will show you how to utilize DDNS and UPnP technologies, so you will be able to simply plug your Raspberry Pi into an Ethernet port on (almost) any router and then securely access your own services from anywhere. The instructions are also applicable to a wireless connection (just use `wlan0` instead of `eth0` for Wi-Fi).

In a sense, the suggested approach solves the same problem as reverse SSH tunneling, but differently — there's no need for a "visible" IP address on client side / middle machine (however, the provider's IP on server side must be accessible), so it's possible to establish connection with dynamic IPs, router's NAT on server side and NAT on client side. Besides, this method is not limited to SSH and can handle other types of connections.

Dynamic DNS

The first obstacle that prevents us from accessing the device is dynamic Internet address assignment. Most Internet providers usually assign a new, arbitrary IP address (using DHCP) to a router each time you turn it on (or connect it to a provider network).
Moreover, even static (of "fixed") IP address will inevitably be different among different Internet providers (and different routers).

To handle this, we will use so called Dynamic DNS (DDNS) to create and dynamically update a mapping between a chosen domain name and an "external" IP address of our Raspberry Pi (i.e. router IP address).

You may choose any free dynamic DNS service you like. As an example, we will use FreeDNS. The usual procedure is:

- Register a new user account.
- Choose a desired domain name (like `your.domain.name`).
- Receive a URL with an unique identifier (prefer HTTPS protocol).

After the registration, it’s good idea to verify that dynamic updating works properly.

Let’s fetch the personal URL to associate our chosen domain name with an IP address of the request (in our case, an IP address of Internet router):

```
curl -ks https://your-personal-url
```

Query information for your domain name (you may need to install `dnsutils` package first):

```
nslookup your.domain.name
```

The response should contain a line like:

```
Name:    your.domain.name
Address: 123.123.123.123
```

Compare this address with an IP address printed after the following request (which displays current “external” IP address of your router):

```
curl -ks http://checkip.dyndns.org
```

If the two addresses match, then dynamic DNS works as expected.

UPnP port forwarding

The second obstacle that prevents the connection is a router’s NAT, which hides all devices in the “internal” router network (LAN) from inbound Internet connections.

To route “external” (WAN) connections to the Raspberry Pi we will employ port forwarding on a router. Because manual configuration of port forwarding rules on each router is not a portable solution (besides, routers also use DHCP to assign dynamic IP addresses to LAN devices, so a MAC-based DHCP reservation is additionally needed), we will rely on Universal Plug and Play (UPnP) protocol to dynamically configure proper port forwarding rules.

Most todays routers support both port forwarding and UPnP, so in most cases Raspberry Pi will be able to enable external access automatically.

We’ll rely on miniupnpc UPnP client, which can be installed as `miniupnpc` package via your OS package manager.

First of all, let’s ensure that our current router supports UPnP:

```
upnpc -l
```

This command should:

- enumerate all supported UPnP devices on local network,
- display their internal- and external IP addresses,
- list their current port forwarding rules.

If router supports UPnP, we can add a port forwarding (for SSH protocol):

```
upnpc -e 'SSH on Raspberry Pi' -r 22 TCP
```

After that, we may verify the result by reissuing `upnpc -l`, the output should contain something like:

```
 TCP 22->192.168.0.2:22
```

Compare this IP address with the internal IP address of the device, which can be displayed via:

```
ip -4 addr show dev eth0
```

If you need to access other services on Raspberry Pi (besides SSH), you may also configure additional port forwardings.

After both DNS and port forwarding are configured, it should be possible to establish an SSH connection to the Raspberry Pi from the Internet:

```
ssh user@your.domain.name
```

However, please keep in mind, that most routers are not able to establish "external" connections from the internal network itself, so a separate Internet connection is needed to fully test the setup.

Unattended configuration

Now, when all the subsystems are checked, we are ready to setup unattended configuration.

Create a shell script in editor:

```
sudo nano /usr/local/bin/redirect.sh
```

insert the following content (don't forget to specify your real DDNS URL there):

```
#!/bin/bash
curl -ks https://your-personal-url > /dev/null
upnpc -e 'SSH on Raspberry Pi' -r 22 TCP > /dev/null
```

Then configure Cron to periodically run this script (every 30 minutes):

```
sudo crontab -e
```

add the following line:

```
*/30 * * * * /usr/local/bin/redirect.sh
```

In this way, Raspberry Pi will automatically update both domain IP and port forwarding, so we should be able to access the device from the Internet even after provider / router / IP address change (however, it may take some time before re-configuration happens).

On-demand configuration

To avoid the delay before re-configuration, we may configure network manager to automatically run our configuration script when Ethernet cable is plugged in.

If you use Debian-based OS (like Raspbian), add `post-up` option to `/etc/network/interfaces` file:

```
auto eth0
```

```
allow-hotplug eth0
iface eth0 inet dhcp
post-up '/usr/local/bin/redirect.sh||true'
```

On Arch Linux you may use netctl's `ifplugd` for the same purpose — install `ifplugd` package, create a network profile `/etc/netctl/redirection` with the following content:

```
Interface=eth0
Connection=ethernet
IP=dhcp
ExecUpPost='/usr/local/bin/redirect.sh||true'
```

…then enable `ifplugd` service for `eth0` interface:

```
sudo systemctl enable netctl-ifplugd@eth0.service
sudo systemctl start netctl-ifplugd@eth0.service
```

Now the configuration script should be run automatically when Ethernet connection is established.

Security

Enabling global Internet access to your Raspberry Pi device is definitely convenient. However, you should keep in mind, that such access must be always complemented by adequately hardened security, namely:

- Limit root login via SSH.
- Generate a strong SSH key.
- Disable password login via SSH.
- Configure firewall rules.

* Raspberry Pi is a trademark of the Raspberry Pi Foundation

Tags: arch, ddns, dhcp, dynamic dns, ethernet, ip, linux, nat, port forwarding, raspberry pi, raspbian, router, ssh, upnp

This entry was posted on Monday, May 18th, 2015 at 11:05 pm and is filed under Linux. You can follow any responses to this entry through the RSS 2.0 feed. You can leave a response, or trackback from your own site.

25 Comments

1. *Selenith* says:
   March 17, 2016 at 10:11 am

   This is exactly what I need !

   Thank you very much 😀

2. *Addicted* says:
   March 26, 2016 at 4:02 pm

   How to change external port?

i mean i don't want to forward my SSH port at external 22 port as it's obvious

3. *Pavel* says:
March 26, 2016 at 6:18 pm

The straightforward way is to change the SSH daemon port directly – set desired `Port` value in `/etc/ssh/sshd_config`. As a bonus, this modifies SSH port in internal network as well.

If you want to change only the "external" port, you can do it like this: `upnpc -r 22 **2222** tcp` (run `upnpc` to see all command line options).

Keep in mind, that using non-standard port is considered security through obscurity and this measure is insufficient by itself. Disabling SSH password authentication is the real solution. Nevertheless, changing port can help to reduce port scanning and scripe kiddie activities.

4. *Pierre-Louis Pelsser* says:
March 29, 2016 at 5:43 pm

Hi,

Thanks for this post. I think it will help me.
However I don't really understand what the command "curl -ks https://your-personal-url > /dev/null" does and why it is needed for port forwarding . Can you explain like I'm five ?

5. *Pavel* says:
March 29, 2016 at 8:16 pm

That command is not related to port forwarding per se, it's a part of the dynamic DNS update.

After registration, dynamic DNS service gives you:
* domain name, like `yourhostname.example.com`
* update URL, like `https://example.com>/update?123456789`

The number is your unique identifier, associated with the chosen domain name. When you fetch the update URL using `curl`, DDNS service assigns the fetcher's IP address to the given domain name.

For example, if you request `https://example.com/update?123456789` from `1.2.3.4` IP address, the `yourhostname.example.com` will point to `1.2.3.4` IP address. This way, you can always find out current IP address of your device by knowing only the domain name (i.e. `yourhostname.example.com`).

The updating procedure might slightly vary depending on the DDNS service, so you need to consult service provider's instruction for exact details.

6. *Fernando* says:
August 12, 2016 at 7:42 pm

Hi Pavel,

Thanks for this! But I have a question: Could you maybe do a related post on how to do this exact procedure using NAT-PMP instead of UPnP? I'd like this to work with routers that don't support UPnP, like the Apple Airport Extreme. Maybe there's a way to determine which protocol the router supports and use one method or the other?

Thanks!

7. *Pavel* says:
August 12, 2016 at 8:08 pm

Hi Fernando,

The MiniUPnP project also provides libnatpmp library which includes natpmpc command line client.

It should be possible to install the `libnatpmp` package and then to use a command like `natpmpc -a 123 123 TCP` to add a port mapping via the NAT-PMP.

8. *Fernando* says:
August 13, 2016 at 7:55 am

Pavel,

Thanks for the info! I checked it out and was able to make it work!

Cheers,
F

9. *Igor Ganapolsky* says:
August 17, 2016 at 10:43 pm

What do you mean by `TCP 22->192.168.0.2:22`? I do not get such output.

10. *Pavel* says:
August 17, 2016 at 11:35 pm

That line is only an example, the 192.168.0.x address range is the most common private IPv4 address space. Exact LAN IP address of the device depends on your network configuration. Also keep in mind, that such a line is only a part of the complete output.

Basically, one needs to verify that there exists a mapping between a port on the WAN IP (Internet) and an address / port pair in the LAN (internal network). If your router provides a web interfaces that lists current port mappings, you can use that interface to check the mapping existence.

11. *alfian* says:
November 11, 2016 at 1:04 am

Hi Pavel!

Nice Tutorial!, thanks!, It works perfectly.

but, is there any workaround regarding this:
"However, please keep in mind, that most routers are not able to establish "external" connections from the internal network itself, so a separate Internet connection is needed to fully test the setup."

I'm both actively works from outside and inside the internal network, switching the address based on where i'am somehow defeat the purpose of making it accessible from external tough..

12. *Pavel* says:
November 20, 2016 at 12:12 am

    To access LAN devices via WAN IP you need NAT loopback.

    Many routers actually support this feature out of the box. You may also consider using OpenWRT firmware, which seems to support NAT loopback (and, in any case, it should be possible to configure that manually).

    Alternatively, you may access you device by hostname while overriding host mapping in your router to point to the internal IP (see Split DNS). In OpenWRT it's possible to use `/etc/hosts` to override the mapping.

13. *Scott Powdrill* says:
December 12, 2016 at 3:59 pm

    Hi, thanks very much for the great tutorial. You helped me an awful lot with setting up my port forwarded VPN server on the Pi. Just a quick note that tripped me up when I first followed your superb tutorial.

    upnpc -e 'SSH on Raspberry Pi' -r 22 returns an error regarding invalid arguments, as in your script this needs to be upnpc -e 'SSH on Raspberry Pi' -r 22 TCP.

    Thanks again for your help!

14. *Pavel* says:
December 12, 2016 at 6:56 pm

    Indeed… I've updated the text, thanks!

15. *Igor* says:
December 20, 2016 at 10:36 pm

    Hello Pavel,

    Thank you very much for your tutorial! It is really useful. When adding a port forwarding I got the following error message:

    AddPortMapping(22, 22, 192.168.1.62) failed with code 718 (ConflictInMappingEntry)

GetSpecificPortMappingEntry() failed with code 714 (NoSuchEntryInArray)

So, obviously the port forwarding didn't happen. Could you please give me an idea why? Thank you!

16. *ph4wks* says:
February 11, 2017 at 8:28 pm

Hi,

Sorry struggling here. In ref to this:
——————————————————-
August 12, 2016 at 8:08 pm
Hi Fernando,

The MiniUPnP project also provides libnatpmp library which includes natpmpc command line client.

It should be possible to install the libnatpmp package and then to use a command like natpmpc -a 123 123 TCP to add a port mapping via the NAT-PMP.
——————————————————-

How do I actually install libnatpmp? Can't seem to find anything on the net about this.

Many thanks.

17. *Pavel* says:
February 11, 2017 at 9:11 pm

The `natpmpc` binary can be installed using `apt install natpmp-utils` command in Debian-based distributions or using `pacman -S libnatpmp` in Arch Linux.

18. *Abhishek* says:
February 18, 2017 at 6:21 pm

Hey Pavel, that's very useful.
i need a small help of yours.
i need to make a raspberry-pi accessible over internet from anywhere, and your post gives details about that.
i need to stream live video from pi and want to access from remote location (not on same network).
i need some inputs from you.hope you will do needful.
thank you.

19. *ph4wks* says:
February 27, 2017 at 2:07 pm

Thanks Pavel,

Thats sorted it. All working great.

20. *Gilles* says:
April 21, 2017 at 4:23 pm

Thanks for the tutorial! I'm hung up on this step:

"upnpc -e 'SSH on Raspberry Pi' -r 22 TCP"

What exactly does "SSH on Raspberry Pi" mean? Do I literally enter "SSH on Raspberry Pi", or do I need to replace this text with something else?

Thanks for you help!

21. *Pavel* says:
April 21, 2017 at 5:36 pm

If you run "upnpc -h" you'll see that "-e" simply adds a description for the port mapping. That parameter is optional, so you can omit it (yet, it's nice to have an informative legend when you run "upnpc -l" or check the port forwarding rules via GUI).

22. *Scott John Powrill* says:
April 21, 2017 at 5:46 pm

Miniupnpc dosent seem to be able to forward ports on any of the routers i have tried it on. Does any one know of an alternative? Does that libnatpnp work consistantly? I dont even know if natpnp is supported on your standard router.

23. *Pavel* says:
April 21, 2017 at 7:45 pm

MiniUPnP is supposed to work with IGD (UPnP), while libnatpmp is intended for NAT-PMP.

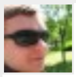
Technical specification usually tells whether a router supports either or those protocols (some routers support both). It might be required to enable corresponding protocol(s) in the router settings beforehand. You might also try to update the router firmware.

In principle, it should be possible to use upnpc and natpmpc simultaneously – to increase the likelihood of compatibility with a device.

There exists a Compatibility list for the miniupnp client (though the list is probably outdated – e.g. it seems that D-Link DIR-615 works just fine). If either upnpc or natpmpc don't work with a particular router, you may consider sending an email to Thomas Bernard (as he requested) or reporting a bug to the MiniUPnP forum.

24. *Koen* says:
June 29, 2017 at 8:12 am

Any solutions if the router does not support NAT loopback / hairpin NAT?

25.  *Pavel* says:
July 2, 2017 at 7:15 pm

NAT loopback / hairpin NAT is not required for the primary use case (i.e. for accessing a device from the Internet).

If, additionally, you need to access LAN devices on your network via WAN IP, you might consider Split DNS. Besides, you might try OpenWRT / DD-WRT alternative router firmwares (which do support NAT loopback).

Leave a Reply

Name

Mail (will not be published)

Website

Submit Comment

☐ Notify me of follow-up comments by email.

☐ Notify me of new posts by email.

- Subscribe: Entries | Comments
- **Links**

  Twitter
  GitHub
  LinkedIn

- **Categories**
  - Gadgets
  - Linux
  - Programming

- Scala
- Software

- ## Archives

  - March 2017
  - December 2015
  - June 2015
  - May 2015
  - March 2015
  - October 2013
  - January 2012
  - October 2011
  - March 2011
  - August 2010
  - August 2009
  - March 2009
  - January 2009