System Administration

Software

Some tips

- Update your firmware where you can - UEFI (confusingly called BIOS by most manufacturers, able to download from your motherboard's manufacturer's website), possibly your drive, soundcard from your headphones,...
- Learn to recognize what an executable is and do not run any that you do not trust, then you have no need for an AV as long as you keep the software up to date (Chocolatey is a huge help with managing updates)

Wireguard

Example Client config

```
[Interface]
PrivateKey = <censored>
Address = 10.200.200.2/32
DNS = 8.8.8.8

[Peer]
PublicKey = aHcw4mjbI0md5VwQSJovvASLs0bkd0Dkwa1Ma4y6yW0=
AllowedIPs = 0.0.0.0/0
Endpoint = sc1.rys.pw:51820
```

Example Server config

```
[Interface]
Address = 10.200.200.1/24
```

```
SaveConfig = true
PostUp = iptables -A FORWARD -i %i -j ACCEPT; iptables -t nat -A POSTROUTING -o ens2 -j MASQUERADE
PostDown = iptables -D FORWARD -i %i -j ACCEPT; iptables -t nat -D POSTROUTING -o ens2 -j MASQUERADE
ListenPort = 51820
PrivateKey = {{ wireguard_private_key }}

[Peer]
PublicKey = 766tBrNv7iinsbd9wMP3yM2ksnIprdT9mgfM9VtFcRM=
AllowedIPs = 10.200.200.0/24

[Peer]
PublicKey = SoFi/vC8IOhBYEMqnFzzuz9umlgDKoo3yytbuIMvizg=
AllowedIPs = 10.200.200.0/24
```

GRUB

GRUB on Arch wiki

Get grub to ignore bad devices and install properly:

blockdev --flushbufs /dev/sde && blockdev --flushbufs /dev/sda && grub-mkdevicemap -n

SSH

[[SSH]]

Wine

Dependencies: wine wine_gecko wine-mono winetricks

Wine is used to run Windows only executables on other operating systems. More information about Wine

You can have multiple Windows "installations" which are called prefixes. In fact it is suggested that you use a new prefix for each application you use.

~/.wine is the default wineprefix (a.k.a. "configuration directory" or "bottle"). You can change which prefix Wine uses by changing the WINEPREFIX environment variable (outside Wine). To do this, run the following in a terminal:

export WINEPREFIX=~/.wine-new wine winecfg

Alternatively, you can specify the wine prefix in each command, e.g.

WINEPREFIX=path_to_wineprefix wine winecfg

you can create a new 32 bit wineprefix using the WINEARCH environment variable(note: you can also export WINEARCH). In a terminal, type:

WINEARCH=win32 WINEPREFIX=~/.wine32bit winecfg

Do not use an existing directory for the new wineprefix: Wine must create it.

Once a 32 bit wineprefix is created, you no longer have to specify WINEARCH in the command line to use it, as the architecture of an existing wineprefix cannot be changed.

You can use `wine64` instead of `wine` to force 64bit.

There are three Direct3D backends for Wine. Which one you use depends on what features your card supports.

wined3d - The D3D backend included with Wine upstream. It is a translation layer that converts Direct3D calls to OpenGL and then sends them to your OpenGL GPU driver. Usable on all GPUs, but has the worst performance.

wined3d with CSMT - A multi threaded, more optimized version of wined3d. It has the same support as wined3d but is much faster. It still incurs a high CPU overhead but if your CPU is good it can help give you better FPS.

Gallium Nine - A native D3D9 implementation that skips the OpenGL translation entirely, requires less CPU overhead, but requires you use a GPU driver which has the GPU side support built in, which are all the Gallium mesa drivers (radeonsi, r600g, nouveau). Nouveau is the open source nVidia driver, but it lacks performance due to reclocking issues and it does not support the GTX10 series because they haven't released signed binaries to support it.

winecfg - > Drives > Autodetect - binds your home folder

Make AppDB reports, it helps the community!

How to install SVP on Arch Linux to play interpolated movies

Dependencies: qt5-3d, mpv-git(AUR), svp(AUR)

Pre-requisites(optional): proprietary GPU drivers already installed.

Download and install mpv-git from AUR

Set up mpv socket - `cat > ~/.config/mpv/mpv.conf << EOF`
`input-ipc-server=/tmp/mpvsocket  # Receives input from SVP`
`hr-seek-framedrop=no  # Fixes audio desync`
`resume-playback=no  # Not compatible with SVP`
`EOF`

Note: There's currently a small bug in SVP causing video stuttering - go to SVP control panel > Utilities > Application settings; and play with the number of "threads" which are set to 0 by default. Setting it to 15 fixed the stuttering issues for me.

That's it, running movies through mpv while having SVP manager turned on will play them smoothly!

Additionally you can install SMplayer, because MPV alone has almost no GUI and relies heavily on CLI commands.

Dependencies: smplayer

Launch SMplayer > open Preferences > Advanced > Options for MPlayer/mpv and add this to Options `--input-ipc-server=/tmp/mpvsocket`

rsync

Packages: rsync

Needs to be installed on both computers.

Using rsync over SSH and custom port:

rsync -avz -e "ssh -p PORT" path/to/folder/or/file domain.com:/copy/to/folder

-z flag for compression, -r flag for recursive, but that is already implied with -a, which preserves file permissions and such. (-a equals -rlptgoD (no -H,-A,-X))

Use destructive syncing – “rsync --del” – This will delete any items on the destination that are not present on the source.

Virtualization

Hypervisors

Xen

QEMU

KVM

Hyper-V

Virtualization under QEMU/KVM

virt-manager - start the interface. Make sure to do so after you're already connected to the internet, else it might use the wrong interface and you'll have no internet connectivity on the VMs.

virt-manager --no-fork - virtmanager will let you type passwords in the terminal instead of openssh-askpass or something like that

Create a new Virtual Machine using an .ISO image and default settings.

Now you should have a working BIOS VM. To create a UEFI one make sure to check customize install and select UEFI for firmware when creating a new VM.

Bi-directional copy pasting and drag-n-dropping files to a Windows KVM is possible by simply installing spice-guest-tools on the KVM(default virt-manager setup uses Spice for display, so it works out of the box)

To enlarge .qcow2 image, use command **qemu-img resize *ubuntu-server.qcow2 +5G*** Remember it'll end up as unallocated space

---

Using LXC/LXD containers

https://wiki.archlinux.org/index.php/LXD

Virtualization under VirtualBox

Packages: virtualbox linux-headers virtualbox-host-dkms

GPU Passthrough

https://www.youtube.com/watch?v=37D2bRsthfI

http://blog.wikichoon.com/2014/07/enabling-hyper-v-enlightenments-with-kvm.html

More stuff

Webserver in current folder

You can instantly create a webserver hosting contents of the folder you're currently in via python:

python -m http.server 8080

Tmux

Tmux is a terminal multiplexer, meaning you can SSH somewhere, run tmux there and disconnect without killing whatever you were running, or just have multiple terminal tabs without actually launching more terminals.

Full cheatsheet: http://hyperpolyglot.org/multiplexers

Tip: tmux running a session but list-sessions doesn't show it? This might help **killall -s SIGUSR1 tmux**

If you're running nested tmux sessions, explanation and tips how to do it efficiently. (CTRL+B twice to get into the second level session, thrice to get into third level etc)

Command to detach all other sessions(in case the window is small and other session is blocking resizing): attach -d

Basic usage:

tmux - start new tmux session

tmux ls - list active sessions

tmux a -t *sessionName* - attach to specific session

tmux kill-session -t *sessionName* - kill specific session

Inside of tmux:

CTRL+B d - detach session

CTRL+B % - split current pane vertically

CTRL+B " - split current pane horizontally

CTRL+B ARROW_KEY - move between panes

CTRL+B+ARROW_KEY - resize current pane

CTRL+B z - toggle current pane fullscreen state

CTRL+B x - kill current pane

CTRL+B c - create a new window

CTRL+B n - next window

CTRL+B p - previous window

Apache

Packages: apache php php-apache(why?) nghttp2

Sources:Arch wiki

Configuration files are located in the folder */etc/httpd/conf* , the main configuration file is *httpd.conf*

sudo systemctl enable --now httpd - Enable and start the httpd service, you should now be able to access the Apache server via localhost:80

PHP7 >

in **httpd.conf**

comment **#LoadModule mpm_event_module modules/mod_mpm_event.so**

uncomment **LoadModule mpm_prefork_module modules/mod_mpm_prefork.so**

place **LoadModule php7_module modules/libphp7.so** at the end of the LoadModule list

and **Include conf/extra/php7_module.conf** at the end of the Include list

sudo systemctl restart httpd

Notes:

DocumentRoot in the config sets the folder for the website, default is /srv/http/

nginx + PHP

Packages: php nginx-mainline php-fpm openssl

systemctl enable --now php-fpm

sudo nano /etc/nginx/nginx.conf - Example config of the server blocks

```
server {
      listen       0.0.0.0:80; # listen on IPv4
      listen       [::]:80 # listen on IPv6
      server_name  *.rys.pw rys.pw; #Redirect all port 80 requests to HTTPS(443)
      return 301 https://$host$request_uri;
}
server {
       listen       0.0.0.0:443 ssl http2; #listen for TLS IPv4 connections and enable HTTP2
       listen       [::]:443 ssl http2; #listen for TLS IPv6 connections and enable HTTP2
       server_name  rys.pw;
       root /usr/share/webapps/mediawiki;
       index index.php;
       location ~ \.php$ { # serve .php files via php-fpm
            fastcgi_pass   unix:/run/php-fpm/php-fpm.sock;
```

```
            fastcgi_index  index.php;
            include        fastcgi.conf;
            }
        location / {
            index  index.html index.htm index.php;
            }

}
server { #forward traffic going to proxy.rys.pw to another server - useful if you need more servers running.
        listen       0.0.0.0:443 ssl http2; #listen for TLS IPv4 connections and enable HTTP2
        listen       [::]:443 ssl http2; #listen for TLS IPv6 connections and enable HTTP2
        server_name  proxy.rys.pw;
        location / {
            proxy_pass         http://10.0.0.10:443/;
            proxy_redirect     default;
            proxy_set_header   X-Real-IP  $remote_addr;
            proxy_set_header   Host       $host;
            proxy_set_header   X-Forwarded-Proto https;
            proxy_set_header   X-Forwarded-Ssl on;
        }
}
```

You can check if your config is valid via **nginx -t**, and then reload the server config via **nginx -s reload**, instead of restarting the daemon.

systemctl enable --now nginx - enable and start nginx

TLS(used to be SSL)

https://cipherli.st/

https://raymii.org/s/tutorials/Strong_SSL_Security_On_nginx.html

https://www.ssllabs.com/ssltest/analyze.html?d=rys.pw

https://securityheaders.io/?q=https%3A%2F%2Frys.pw%2F

https://observatory.mozilla.org/analyze.html?host=rys.pw

TLS 1.0 being deprecated 30th June 2018[1]

All versions of nginx as of 1.4.4 rely on OpenSSL for input parameters to Diffie-Hellman (DH). Unfortunately, this means that Ephemeral Diffie-Hellman (DHE) will use OpenSSL's defaults, which include a 1024-bit key for the key-exchange.

cd /etc/ssl/certs && sudo openssl dhparam -out dhparam.pem 4096 - This takes time depending on your single core performance as it's not multithreated.(few mins on i7-4790K, 42~ mins on Raspberry Pi 3B) You can use 2048 but it's weaker, create the stronger file at a later date if you just want to get it running for now.

sudo nano /etc/nginx/nginx.conf - place these outside of the server blocks so it applies to all servers.

Hardening

```
ssl_dhparam /etc/ssl/certs/dhparam.pem;
ssl_protocols TLSv1.2 TLSv1.3; # Keep in mind this will break software that is way past it's end of life.
ssl_prefer_server_ciphers on;
ssl_ciphers "EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH";
ssl_ecdh_curve secp384r1; # Requires nginx >= 1.1.0
ssl_session_cache shared:SSL:10m;
ssl_session_tickets off; # Requires nginx >= 1.5.9
ssl_stapling on; # Requires nginx >= 1.3.7
ssl_stapling_verify on; # Requires nginx => 1.3.7
#resolver $DNS-IP-1 $DNS-IP-2 valid=300s; # I do not understand those so I disabled them
#resolver_timeout 5s;
#RESOLVERS: if you don't specify any, nginx will resolve HTTP upstream server hostnames when starting up, and will never attempt to re-resolve them. This is a problem if later the IP addresses of these upstream servers change. But if you define resolvers in nginx.conf, it will honor the TTL of DNS records, and re-resolve the hostnames periodically.
#Make sure you correctly respond to this or the issue is fixed before defining the resolver.
http://blog.zorinaq.com/nginx-resolver-vulns/
add_header Strict-Transport-Security "max-age=63072000; includeSubdomains; preload"; # You can add your domain to Chromium's source code for automatic preloading https://hstspreload.org/?domain=rys.pw
add_header X-Frame-Options DENY;
add_header X-Content-Type-Options nosniff;
add_header X-XSS-Protection "1; mode=block";
#add_header Content-Security-Policy "default-src 'self';"; # if you require no scripting.. likely not the case.
add_header Content-Security-Policy "default-src 'self'; script-src 'self'; img-src 'self' data:; style-src 'self' 'unsafe-inline'; font-src 'self' data:; child-src 'self'; connect-src 'self' https://apis.google.com; object-src 'none' ";
# READ THIS - http://lollyrock.com/articles/content-security-policy/
```

SSL certs - you'll need to use letsencrypt to get these

```
ssl_certificate /etc/letsencrypt/live/rys.pw/fullchain.pem;
ssl_certificate_key /etc/letsencrypt/live/rys.pw/privkey.pem;
```

Add a block that redirects all HTTP requests to HTTPS

```
server {
listen 80;
listen [::]:80;
server_name rys.pw;
return 301 https://$host$request_uri;
}
```

Additionally use `listen 443 ssl http2;` `listen [::]:443 ssl http2;` in every other server block to force TLS and support HTTP2 protocol.

MariaDB

Packages: mariadb

sudo mysql_install_db --user=mysql --basedir=/usr --datadir=/var/lib/mysql

sudo nano /etc/php/php.ini - uncomment `extension=mysqli.so`

sudo systemctl restart php-fpm

sudo systemctl enable --now mysqld

**sudo /usr/bin/mysql_secure_installation **

Backup:

```
mysqldump --single-transaction --flush-logs --master-data=2 --all-databases -u root -p | gzip >
all_databases.sql.gz
```

Restore:

```
gunzip all_databases.sql.gz | mysql -u root -p
```

mysqldump --defaults-file=/path-to-file/SQLcreds.txt --all-databases > my_db.sql

nano SQLcreds.txt

```
[mysqldump]
user=mysqluser
password=secret
```

sudo chown root:root SQLcreds.txt

sudo chmod 700 SQLcreds.txt

PhpMyAdmin

Packages: phpmyadmin php-mcrypt

sudo nano /etc/nginx/nginx.conf - add a whole new server block for phpmyadmin

```
server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;
    server_name phpmyadmin.localhost;
    root /usr/share/webapps/phpMyAdmin;
    index index.php;
    location ~ \.php$ {
        try_files $uri =404;
        fastcgi_pass unix:/run/php-fpm/php-fpm.sock;
        fastcgi_index index.php;
        include fastcgi.conf;
    }
}
```

Murmur

Packages: murmur

Port: TCP/UDP 64738

Config: /etc/murmur.ini

Setting valid TLS(SSL) certificate:

uncomment and set these two lines in the config

```
sslCert=/etc/letsencrypt/live/rys.pw/fullchain.pem
sslKey=/etc/letsencrypt/live/rys.pw/privkey.pem
```

Let's Encrypt

Packages: certbot certbot-apache/certbot-nginx

https://letsencrypt.org/getting-started/

sudo systemctl stop nginx - Stop your webserver. In case of apache you want to stop httpd

sudo certbot certonly --standalone -d rys.pw -d www.rys.pw -d phpmyadmin.rys.pw -d tickets.rys.pw -d pihole.rys.pw -d mumble.rys.pw -d esp8266.rys.pw -d cloud.rys.pw -d paste.rys.pw --email email@example.com --rsa-key-size 4096 --agree-tos

sudo systemctl start nginx

To non-interactively renew *all* of your certificates, run **certbot renew --rsa-key-size 4096 **.

Postfix

?Final setup - TODO - postfix+dovecot+roundcube+postfixadmin?

Order of importance of records:

SPF > DKIM > DMARC

https://wiki.archlinux.org/index.php/postfix

Packages: postfix #dovecot roundcubemail postfixadmin php-imap

First set up DNS records. I will be using rys.pw, so I set MX record of @ pointed to rys.pw, which is in turn pointed at my VPS.

systemctl enable --now postfix

This will likely land in your spam folder. echo "Message" | mailx -s "important mail" yourmail@gmail.com

Edit /etc/postfix/main.cf

myhostname = rys.pw

postfix reload

Now you should be able to resend the test email and see it came from your domain.

Edit /etc/postfix/aliases

root: c0rn3j

change to your user account, reading email as root is bad

postalias /etc/postfix/aliases

For later changes run `newaliases`

Now you should be able to read mail coming from the internet(only for users that exist on the system) and the services on the box.

less /var/mail/c0rn3j

Access point (WIP)

Packages: hostapd dnsmasq

https://w1.fi/cgit/hostap/plain/hostapd/hostapd.conf

https://wiki.gentoo.org/wiki/Hostapd

https://frillip.com/using-your-raspberry-pi-3-as-a-wifi-access-point-with-hostapd/

sudo nano /etc/hostapd/hostapd.conf

```
ssid=myWifi # SSID of the network
wpa_passphrase=MySuperSecurewifi123 # password for the network
interface=wlan0 # Interface it'll run on
auth_algs=1 # 1=wpa, 2=wep, 3=both
channel=6 # Channel it'll broadcast on
driver=nl80211
hw_mode=g # 2.4GHz, 'a' for 5GHz
rsn_pairwise=CCMP
wpa=2 # WPA2 only
wpa_key_mgmt=WPA-PSK
#In addition to these, RPi3 seems to require those
ieee80211n=1 # nothing would work without this
#wmm_enabled=1 # QoS support
#ht_capab=[HT40][SHORT-GI-20][DSSS_CCK-40] #I did not actually need this
```

sudo nano /etc/sysctl.conf # is this an outdated way to set ipv4 forward on a systemd distro?

```
net.ipv4.ip_forward = 1
```

sudo sysctl -p

sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE

sudo iptables -A FORWARD -i eth0 -o wlan0 -m state --state RELATED,ESTABLISHED -j ACCEPT

sudo iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT

sudo sh -c "iptables-save > /etc/iptables.rules"

sudo iptables-restore /etc/iptables.rules # This needs to be executed after boot

sudo nano /ect/dnsmasq

port = 0

sudo ip link set wlan0 up

sudo ip addr add 192.168.2.1/24 broadcast 192.168.2.255 dev wlan0

sudo ip route add default via 192.168.0.1

Samba(file sharing)

https://wiki.archlinux.org/index.php/samba

Packages: samba

sudo cp /etc/samba/smb.conf.default /etc/samba/smb.conf - copy the default config file to the default config path

sudo systemctl enable --now smb

sudo nano /etc/samba/smb.conf

```
workgroup = WORKGROUP #change to WORKGROUP so it's the same as default windows WG.


valid users = %S # - add this to [homes] to allow users login to their home directories(?)
```

Example block

```
[dolphin]
comment = dolphin ISOs
path = /mnt/3tbRED/DOLPHIN ISOs   # SAMBA DOESN'T NEED ESCAPE SEQUENCES FOR SPACES AND SUCH
read only = yes
valid users = c0rn3j
```

```
Samba requires a Linux user account - you may use an existing user account or create a new one.
Although the user name is shared with Linux system, Samba uses a password separate from that of the Linux user accounts.
```

sudo smbpasswd -a *c0rn3j* - change samba password of the user

testparm -s - will show you the current config

sudo smbstatus - list connections to the shares on the server

sudo systemctl restart smb - restart samba service to apply new config

Now on the client side...

```
smbclient -L //192.168.1.10 -U% - list public shares on a server

sudo mount //192.168.1.10/homes /mnt/dolphin/ -o user=c0rn3j - example: Mount the home of user c0rn3j to /mnt/dolphin/
```

Mounting every time is tedious though, let's add an entry to fstab to mount it on boot. First we'll need to store the credentials safely though.

sudo nano /mnt/credentials

```
username=c0rn3j
password=supersafepassword
```

sudo chmod 600 /mnt/credentials - secure it so it's not readable by anyone but root or owner.

sudo nano /etc/fstab - and add this line at the bottom

```
//192.168.1.10/dolphin /mnt/dolphin cifs auto,x-systemd.automount,_netdev,credentials=/mnt/credentials 0 0
```

mount -v - list all mountpoints

mount -t *cifs* - list mountpoints by fs

GPG Encryption

-c specifies to encrypt symmetrically(symmetrical is harder to crack than asymmetrical), defaults to AES-128 which should be secure enough for now and the near future. AES-256 seems to be noted as 30-40% slower, so if you don't mind taking that performance hit feel free to use that instead(but I do suggest reading why you'd want to do that first as AES-128 is possibly enough for you).

Encryption with a password and AES-256:

gpg --batch --cipher-algo AES256 --passphrase *password* -c *file*

Decryption with a password:

gpg --batch --passphrase *password* -o *file* -d *file.gpg*

If you are not going to be using an automatic script for encryption/decryption, you can simply omit *--passphrase password* and you will be asked to enter it manually.

The above example is not secure because any user can execute **ps aux** and see the whole command, including the password.

Now let's do it better!

nano password.txt - write your super secret password there

sudo chown root:root password.txt

sudo chmod 700 password.txt

Encryption with a password in a restricted file:

sudo gpg --batch --passphrase-file *password.txt* -c *file*

Decryption with a password in a restricted file:

sudo gpg --batch --passphrase-file *password.txt* -o *file* -d *file.gpg*

LUKS

Check if your password is correct and list slots:

cryptsetup luksOpen --test-passphrase --verbose /dev/sda

Add a key file for automatic unlocking via /etc/crypttab:

cryptsetup luksAddKey /dev/nvme1n1p1 /etc/adatapass

Ansible

Encrypt a file:

ansible-vault encrypt --vault-id C0rn3j/configs@~/C0rn3j_configs-vaultpass.txt id_ed25519

Encrypt a string for use in playbooks/templates:

ansible-vault encrypt_string --vault-id C0rn3j/configs@~/C0rn3j_configs-vaultpass.txt 'supersecretpassword' --name 'bree_matomo_db_password'

Revision #5
Created 28 June 2021 19:49:30 by C0rn3j
Updated 12 January 2024 08:16:58 by C0rn3j