1. Intro

Djedefre is a tool for documenting your network.

Djedefre consists of 3 parts:

- the database
- scan scripts
- the GUI

The scan scripts fill the database. The GUI displays the network in the database and allows the user to change certain values. The database is created when the GUI is first started.

Some time ago, there was a tool called Cheops, that automated network discovery and gave a nice drawing of the network. Cheops is now abandonware. Djedefre was pharaoh after Cheops. The main difference between Cheops and Djedefre is, that Djedefre assumes that the network is managed. This means that there is access to at least some servers, routers and switches.

2. Using Djedefre

2.1 Installing

From "https://github.com/ljmdullaart/djedefre" download at least

- djedefre.pl
- the directory scan_scripts
- the directory images

Make sure that the following Perl modules are installed:

- Tk -Tk::JBrowseEntry
- File::Slurp
- File::Slurper
- File::HomeDir
- DBD/SQLite.pm

Make sure that the following programs are installed:

- ipcalc
- grepcidr
- fping

2.2 First start

After installing or unpacking,

- Move to the directory where Djedefre is installed.
- Make sure that there is a directory database and create it if it is not there
- Create the database with perl djedefre_create_db.pl
- start the GUI with ./dedefre.pl &

You will be greeted with a cartouche depicting the hieroglyphs for Djedefre with a number of buttons above it.

The buttons give access to functionality:

- "Listings" provides several lists (servers, subnets, interfaces)
- "Manage pages" allows you to create pages with network drawing and determine what should be on those pages
- "Layer 2 input" provides a way to add input for the layer-2 connections
- "View page" has a selection list of pages that can be viewed.

At this point, when you select the page "top", you will get an empty drawing.

2.3 Start scanning

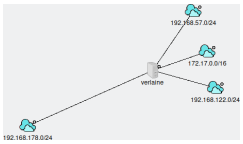
To get some information into the database, scan-scripts are used. These scan-scripts can be found in the directory `scan_scripts.`

The best start would be to scan the local system. Launch the script `scan_local_system.sh` in the `scan_scripts` directory in a separate `xterm` window with:

```
bash scan-scripts/scan_local_system.sh database/djedefre.pl
```

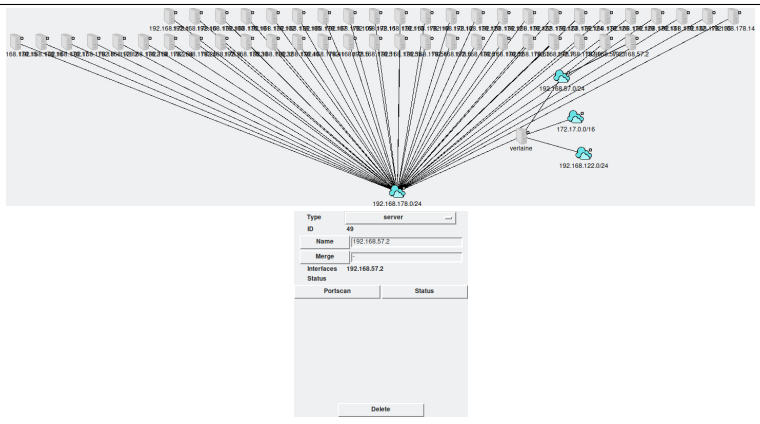
When the scan is finished, the plot page "top" will show the local system and the subnets that it is connected to.

With the left mouse button, you can pick-up the server and the subnets and place them anywhere on the canvas that you like. When a server or subnet is selected, an info-window opens on the right displaying some information about the selected item.

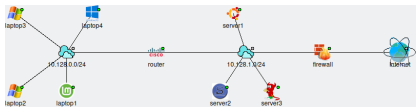


As there are now subnets in the database, a scan of the subnets is possible. Run the script `scan_subnet.sh` with:

```
bash scan-scripts/scan_subnet.sh database/djedefre.pl
```



When you have scanned the network, placed all devices and subnets, set the correct type and name, your network may look something like this:



2.4 Scanning the network

You may have noticed that the scan of subnets did not provide a complete list of all the servers on the network. This is because the subnet scan uses `fping` to scan the subnet. Typically, for example, Windows 10 does not reply to a ping, so no Windows 10 machines will show up. An ARP scan however will detect those machines if they're on the same layer 2 segment.

There are a number of scan scripts available. These scripts try to find out what the network lay-out is, what the systems are. Although the scripts can run in any sequence, the fastest discovery seems to be:

- local_system
- subnet
- arp
- access
- type
- server
- remote_system
- cisco
- dns
- vbox
- internet
- database_integrity
- status

The scan scripts are in the directory `scan_scripts` and they are called `scan_xxxxx.sh` with `xxxxx` meaning the name of the scan.

2.4.1 local_system

Adds the local system to the network. Also adds the interfaces and the subnets that the system is connected to.

2.4.2 subnet

Scans all the known subnets for servers. Uses `sudo fping` for the scan.

2.4.3 access

Scan all the known systems for ssh access. It must be password-less login. Three users are used:

- current user: should be the default
- `root` : should be available only is a very closed environment, because it is unsafe. Some IoT things require this.
- `admin` : same remark as `root` But some devices need this, for example older Qnap NASses.

2.4.4 arp

The ARP table is read from every device that has ssh access. Interfaces that have an IP address and where the MAC-id is not 00:00:00:00:00:00 are added. For interfaces without host, a server is created.

2.4.5 type
Tries to determine the type of servers. This works reasonably well for servers that have ssh access.

2.4.6 server
For the servers that have ssh access, try to appropriate all the interfaces.

Also, if the host name is still the IP address, rename the server to its host name.

2.4.7 remote_system
For all the servers that have ssh access, add the networks that they are connected to.

2.4.8 cisco
Cisco IOS devices do not have a standard shell. The Cisco scan scans devices that have the type "cisco" and does a "server" scan and a "remote_system" scan on these devices.

2.4.9 dns
For all interfaces, set the host name to the host name that DNS has given it. Also, for all servers where the name is an IP address, set the host name to the corresponding name in DNS.

2.4.10 vbox
Determine for all VirtualBox managers which machines are running on it.

2.4.11 status
Determines the status of the servers. In its simple form, it tests whether the server responds to a ping.

If under `scan_scripts` there is a script`status_`*name*`.sh`, where *name* is the name of the server, then that script is used to determine the status of the server. If the script has an exit code 0, then the server is up.

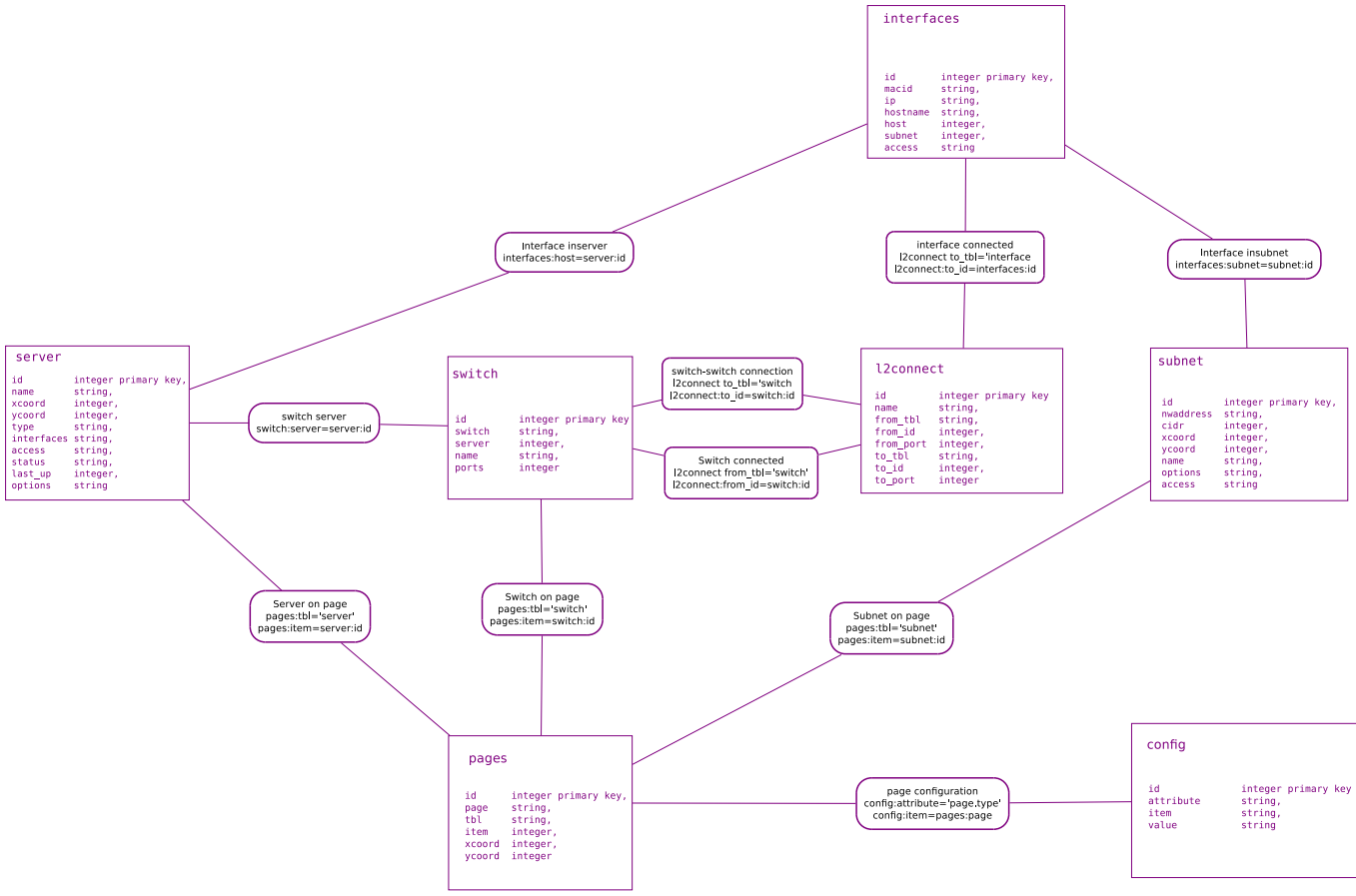
2.4.12 internet
Uses `traceroute` to determine where the break-out to the Internet is. The last IP address that is in the database table "interfaces" is considered as belonging to the server that has the interface to the Internet.

3. Details

3.1 The database

The main component of Djedefre is the database. The database is updated by the scan scripts and is used by the GUI to show the network.

The conceptual model (MCD) of the database is shown below.



3.1.1 Subnet

Name	Type	Description
id	integer primary key,	Primary key for the subnet
nwaddress	string,	Network address
cidr	integer,	CIDR, the number of bits in the network
xcoord	integer,	the x-coordinate in the drawing
ycoord	integer,	the y-coordinate in the drawing
name	string,	the name that the subnet has been given
options	string,	future use
access	string	future use

3.1.2 Interfaces

Name	Type	Description
id	integer primary key,	Primary key for the interface
macid	string,	MAC-id if the MAC-id is known
ip	string,	IP address of the interface
hostname	string,	hostname that resolves to the IP
host	integer,	the id of the host that this interface belongs to
subnet	integer,	the subnet where the interface is connected
access	string	the way that access can be gained to the server via this interface (in general: ssh + uid)

3.1.3 Server

A server is a host on the network. It may be an actual server, it may be a network component like a router or even a client. The properties of a server are:

Name	Type	Description
id	integer primary key,	Primary key for the server
name	string,	name of the server
xcoord	integer,	x-coordinate on the drawing
ycoord	integer,	y-coordinate on the drawing
type	string,	type of the server.
interfaces	string,	future use
access	string,	future use
status	string,	Result of the last status scan
last_up	integer,	Last time the server was up
options	string	Other optional fields. For example: a virtual machine can have a reference to the host it runs on

4. The source code

If I want to change anything, I find myself first digging through the source code before I can even begin to add new functionality. This chapter should help finding my way through the code easier.

4.1 TK frame hiërarchy

Most os the actual work in the frames is done in the frame `subframe` which is destroyed and recreated, depending on what is displayed. The `subframe` is set within the `main_frame` because sometimes, destroying and re-creating the `main_frame` causes resizing of the main window and/or flickering.

- main_window
  - button_frame
    - unnamed button "List netork"
    - unnamed button "Plot network"
    - unnamed button "Options"
    - unnamed button "exit"
  - unnamed label with textvariable $Message
  - main_frame
    - subframe: is destroyed and recreated, depending on what to display
      - On start-up:
        - unnamed label "Dedefre"
        - Photo image with djedefre.gif
      - As network-plot:
        - nw_info_frame
        - nw_button_frame
        - nw_frame
      - As Options:
        - Label "Options"
        - opt_scan_frame
      - As List network:
        - list_main_frame

4.1.1 On start-up

The subframe contains:

- unnamed label "Dedefre"
- Photo image with djedefre.gif

Not much interaction is possible.

CONTENTS