man-da.de
metropolitan area network darmstadt.gmbh

RANCID on Speed

Salvation for Network Operators

Marcus Stögbauer
ms@man-da.de

Summary

* Introduction to RANCID
* Tools for our everyday work
* Routine work

Preface

* Might be a bit boring if you already know RANCID

* Or if you are waiting for the social event to start

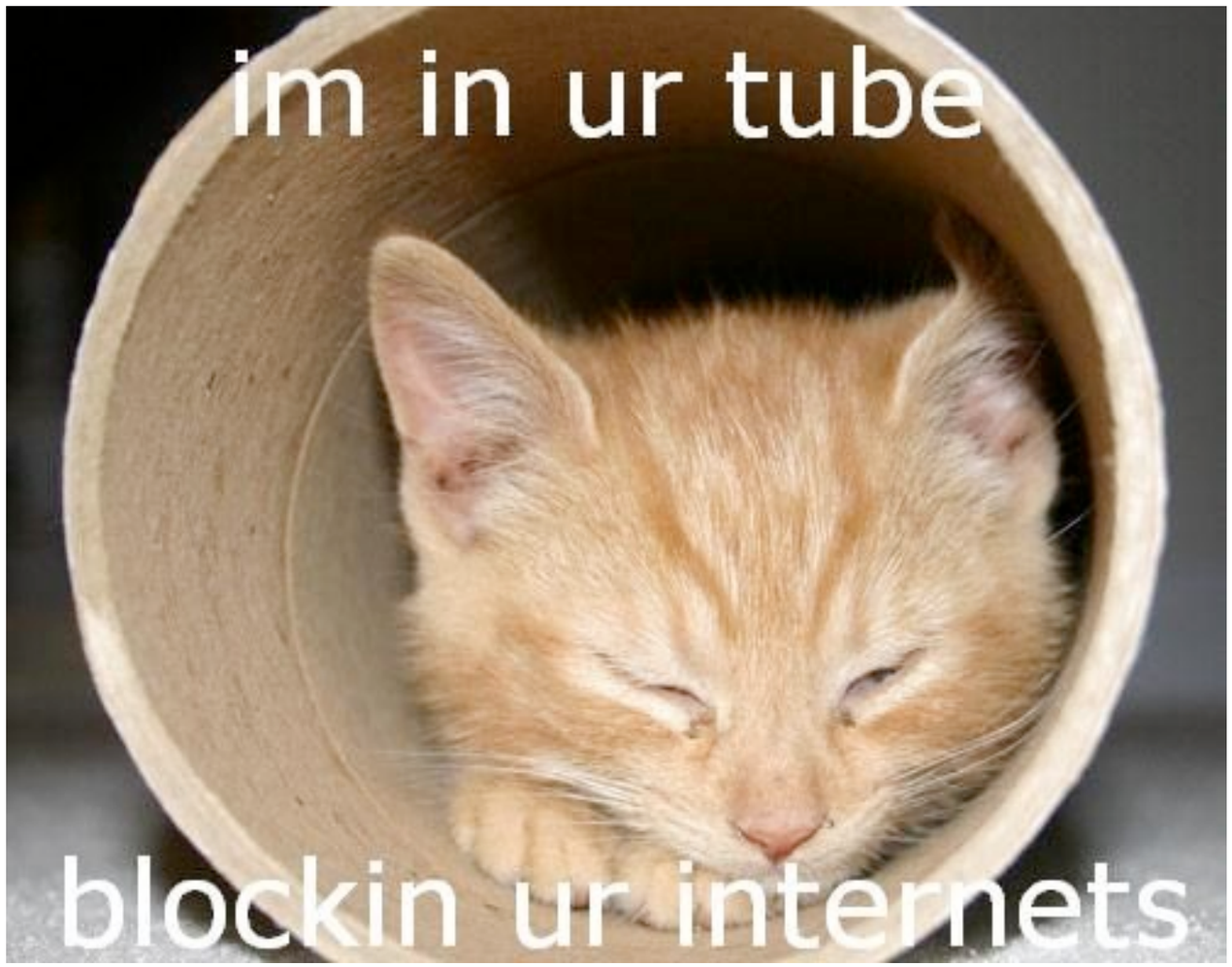
* Some basic information about RANCID needed to understand the presentation

* As a compromise: included cute pictures of cats throughout the presentation

Part 1 - Introduction to RANCID

Introduction to RANCID

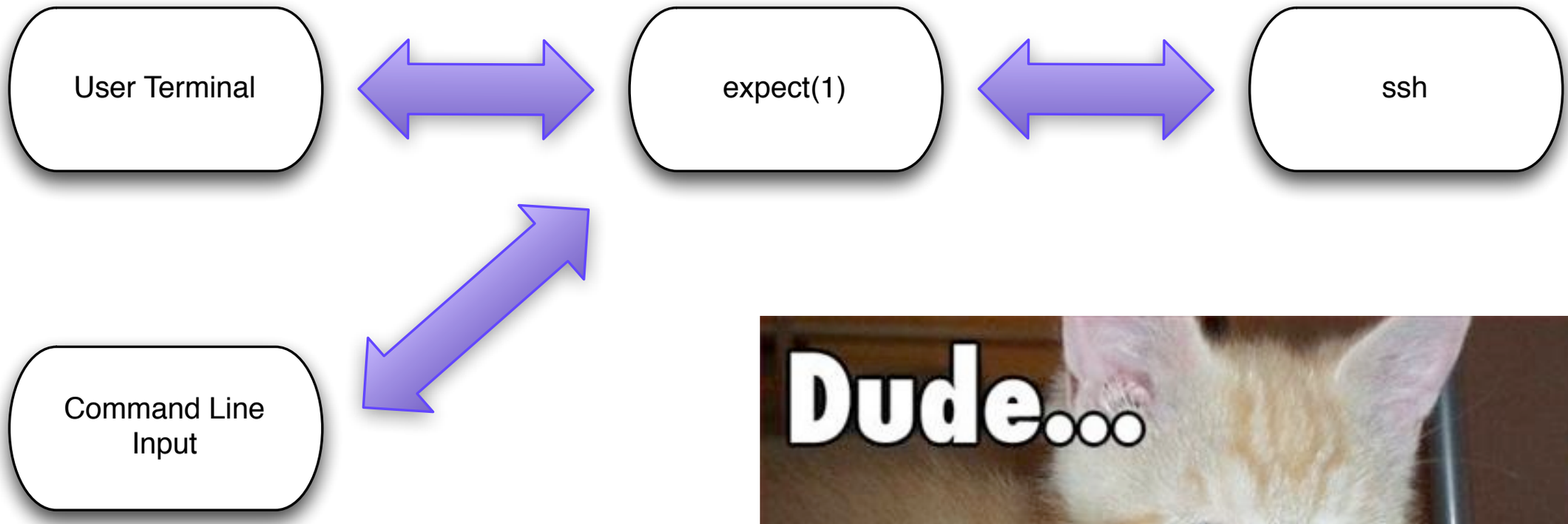
* RANCID - ***R***eally ***A***wesome ***N***ew ***C***isco conf***I***g ***D***iffer
* www.shrubbery.net/rancid/



Introduction to RANCID

* Combination of TCL/expect, Perl and Shell Scripts
* expect(1) spawns ssh/ telnet/rsh and sends commands to the network device
* expect(1) scripts for many platforms, e.g.
  * Cisco: `clogin`
  * Juniper: `jlogin`
  * Foundry: `flogin`

Introduction to RANCID



Login example

```
$ clogin core1.f.test.man-da.net
```
← expect(1) script

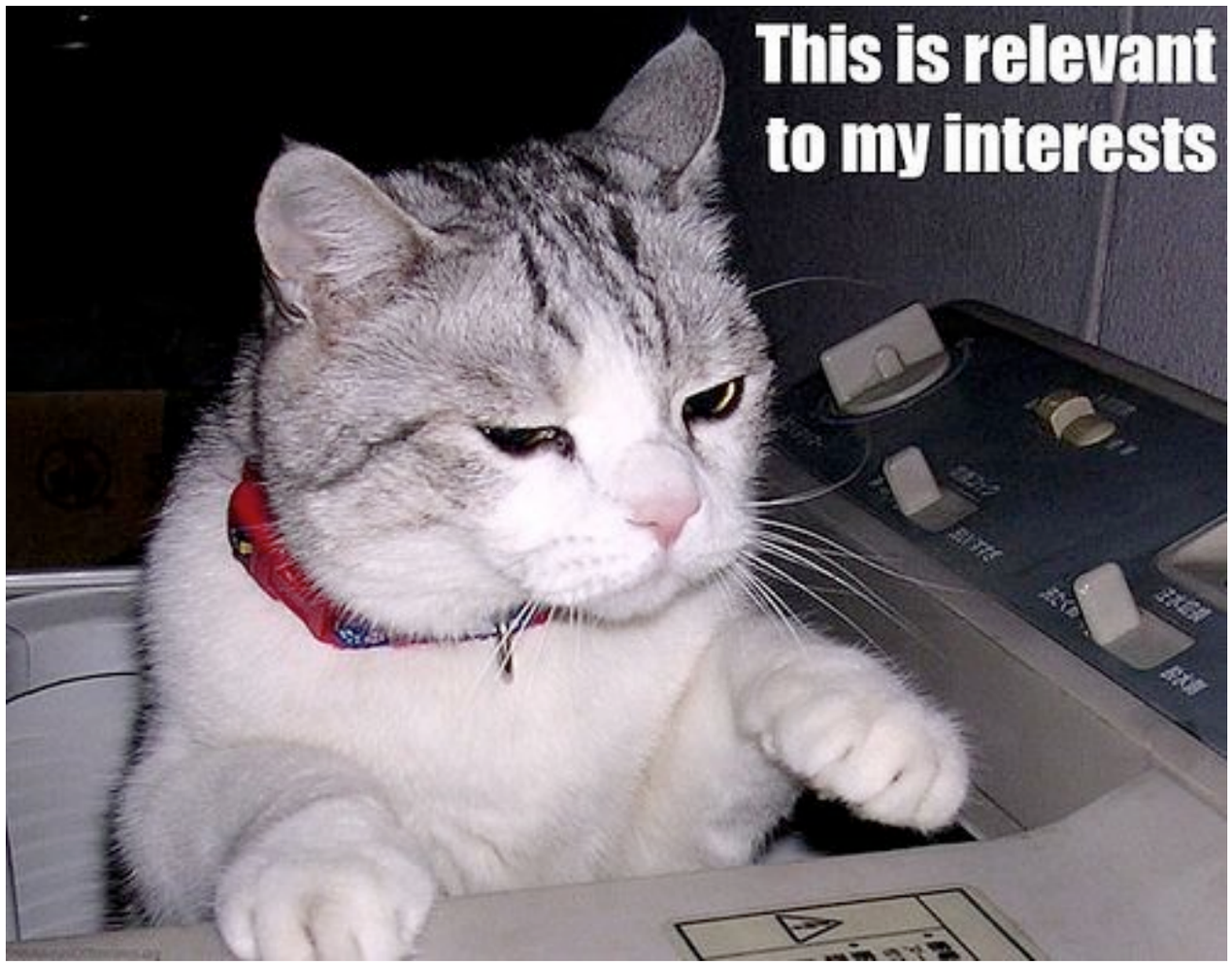
```
spawn ssh -c 3des -x -l lysis
core1.f.test.man-da.net
```
← spawning ssh

```
core1.f.test (ASR1002):
Password:
```
← submitting password

```
core1.f.test#
```
← and we're in

Introduction to RANCID

* Perl/Shell scripts use expect script to collect information from network devices
  * running-config
  * inventory
  * sensors
  * flash contents



Introduction to RANCID

* Reformatting and saving device configuration and information to disk
* mail diff against previous version to admin
* commit to SVN/CVS

Minimal configuration

rancid.conf:

```
TERM=network;export TERM
umask 027
TMPDIR=/tmp; export TMPDIR
BASEDIR=/Users/lysis/share/rancid; export BASEDIR
PATH=/opt/local/libexec/rancid:/opt/local/bin:/usr/bin: \
     /usr/sbin:/bin:/usr/local/bin:/usr/bin; export PATH
CVSROOT=$BASEDIR/SVN; export CVSROOT
LOGDIR=$BASEDIR/logs; export LOGDIR
RCSSYS=svn; export RCSSYS
OLDTIME=4; export OLDTIME
LIST_OF_GROUPS="darmstadt frankfurt wiesbaden test"
```

Minimal configuration

.cloginrc:

```
add method *.man-da.net ssh
add method sw1.sm.test.man-da.net telnet

add autoenable *.man-da.net 1

add user * admin
add password * adminPassword enablePassword
```

Minimal configuration

router.db:

```
core1.da.test.man-da.net:cisco:up
core2.da.test.man-da.net:juniper:up
sw1.sm.test.man-da.net:cisco:up
sw1.tiz.test.man-da.net:force10:up
core2.f.test.man-da.net:cisco:down
```

Introduction to RANCID

* configuration and inventory of network device stored on disk
* in SVN/CVS repository
* every saved version accessible

Customer: "ZOMG my interwebs are broken since your mainten-ance last night !1!!"

- Find the diff in your mailbox and see if something important is missing, OR
- Get a diff from the SVN/CVS repository and check the differences

You: “3 years ago I used to have a Juniper configuration for $feature which I migrated to Cisco. How did I do this in JunOS again?”

- `svn cat -r ‘{2007-11-04}’ rt-core.man-da.net`

Introduction to RANCID

* Additional advantages:
  * log into your network devices without entering a password:
    * `clogin core1.f.test.man-da.net`
    * `jlogin core2.da.test.man-da.net`
* A lot of vendors supported. See www.shrubbery.net/rancid/#manpages

Security concerns

You have to put the user and enable password in a plaintext file.

* Workaround for configuration history:
  * create user for RANCID purposes only
  * use TACACS to allow only commands RANCID needs

Security concerns

You have to put the user and enable password in a plaintext file.

* Workaround for admin purposes:
  * put password in encrypted partition/file on USB stick
  * only mount when you are on your workstation

Part 2 - Tools for our everyday work

Level 1 - make logging in more comfortable

* Don’t like using a mouse for logging into all your network devices?
* Don’t like typing long names like core1.fra1.ix.f.man-da.net?
* Can’t remember if core1.fra1.ix.f.man-da.net is Juniper, Cisco or Brocade?

- `~/bin/l <hostname> [options]`

~/bin/l

* greps search on router.db for hostname
* extracts first match and router type
* calls appropriate RANCID login expect script

~/bin/l

```
lysis@sparkles:~$ l core2.da
spawn ssh -c 3des -x -l lysis core2.da.test.man-da.net
core2.da.test
lysis@core2.da.test.man-da.net's password:
--- JUNOS 10.0R4.7 built 2010-08-22 03:07:19 UTC
lysis@core2.da.test>
```

```
lysis@sparkles:~$ l core1.f.t
spawn ssh -c 3des -x -l lysis core1.f.test.man-da.net

core1.f.test (ASR1002):
Password:

core1.f.test#
```

~/bin/l

* additional usage: execute commands on the network device

```
lysis@sparkles:~$ l core1.f.t 'show ver | i ^Cisco;show bgp ipv4 u su | i version'
spawn ssh -c 3des -x -l lysis core1.f.test.man-da.net

core1.f.test (ASR1002):
Password:

core1.f.test#
core1.f.test#terminal length 0
core1.f.test#show ver | i ^Cisco
Cisco IOS Software, IOS-XE Software (PPC_LINUX_IOSD-ADVIPSERVICESK9-M), Version
15.0(1)S, RELEASE SOFTWARE (fc1)
Cisco IOS-XE software, Copyright (c) 2005-2010 by cisco Systems, Inc.
core1.f.test#show bgp ipv4 u su | i version
BGP table version is 37, main routing table version 37
core1.f.test#exit
```

Expert Level - using ~/bin/l while scripting

* extract serial numbers of all Juniper routers

```
$ for i in $(cat router.db  | grep :juniper | cut -d ":" -f 1); do
    l $i show chassis hardware | grep ^Chassis
  done

  Chassis                             JNAAAAAAAAAAA      MX240
  Chassis                             J1337              M7i
  Chassis                             JN00B              M7i
```

Serial numbers might be fake

Sage level

"Interface Gi0/8 on device sw1.lw.tu.da.man-da.net is flapping according to syslog. I wonder what's connected there."

Goal: Quickly find the interface description for Gi0/8.

Boring:

* Log into device the old-fashioned way and typing all the commands interactively

Sage level

* Option 1 - mildly invigorating:

```
$ l sw1.lw show int descr | grep
0/8

GigabitEthernet 0/8 YES up    up
t[srv] ll[eth0 lysis-test] c[MANDA]
```

* Disadvantage: spawns ssh

Sage Level - Using config parsers

Remember: You have the configuration on your disk

* parse previously saved config files and output interface descriptions; way faster

```
$ interfaces.py sw1.lw | grep 0/8

GigabitEthernet 0/8: t[srv] ll[eth0
lysis-test] c[MANDA]
```

Sage Level, advanced - rcat.py

* Easy access to latest saved configuration:

```
$ rcat.py <part of hostname>
[optional: section from config]
```

* Like ~/bin/l matches the hostname against router.db
* Optionally prints only the given section

Sage Level, advanced - rcat.py

$ rcat.py core1.da int lo

```
interface Loopback0
 ip address 82.195.95.2
255.255.255.255
 ipv6 address
2001:41B8:FFFF::2/128
```

Sage Level, advanced - rcat.py

```
$ rcat.py core2.da int lo
 unit 0 {
    family inet {
       filter {
          input local-access-v4
       }
       address 82.195.95.13/32
    }
    family inet6 {
       filter {
          input local-access-v6
       }
       address 2001:41b8:f::13/128
    }
}
```

Überadmin Level

Goal: Rollout new standard configuration to all devices

Steps:

* collect all router hostnames from router.db
* parse a config file and match regexp against router hostnames or vendor type
* generate config per router
* use RANCID to push config to router

Überadmin Level - Example

* You need to update a MAC ACL on all your switches
* You have Force10 and Cisco Switches in your network
* Obviously both vendors have a different configuration syntax

Überadmin - write configuration for devices

```
=vendor:cisco
!cisco
conf t
mac access-list extended notebooks
 no permit host 0011.2222.3333 any
 permit host 0011.3333.4444 any
 no deny   any any
 deny   any any
!
=exit

=vendor:force10
!force10
conf t
mac access-list extended notebooks
 no seq 5 permit host 00:11:22:22:33:33
 seq 5 permit host 00:11:33:33:44:44 any
=exit
```

Überadmin - write configuration for devices

* filter can also be a regular expression matching the hostname, for example:

```
=core1.(da|f).*

...

=exit
```

Überadmin - generate config

```
$ genconfig.py text/mac-acl-change

$ ls -1 generated/

gsw1.an.f.man-da.net
gsw1.cchh.da.man-da.net
gsw1.dolivo.da.man-da.net
gsw1.fh.di.man-da.net
gsw1.hmwk.wi.man-da.net
gsw1.igd.da.man-da.net
...
```

Überadmin - generate config

```
$ cat generated/gsw1.an.f.man-da.net

!force10
conf t
mac access-list extended notebooks
 no seq 5 permit host 00:11:22:22:33:33
 seq 5 permit host 00:11:33:33:44:44 any
```

Überadmin - generate config

```
$ cat generated/gsw1.cchh.da.man-da.net

!cisco
conf t
mac access-list extended notebooks
 no permit host 0011.2222.3333 any
 permit host 0011.3333.4444 any
 no deny   any any
 deny   any any
!
```

Apply configuration to devices

```
$ cd generated
$ for i in *; do
    l $i -f $i
  done
```

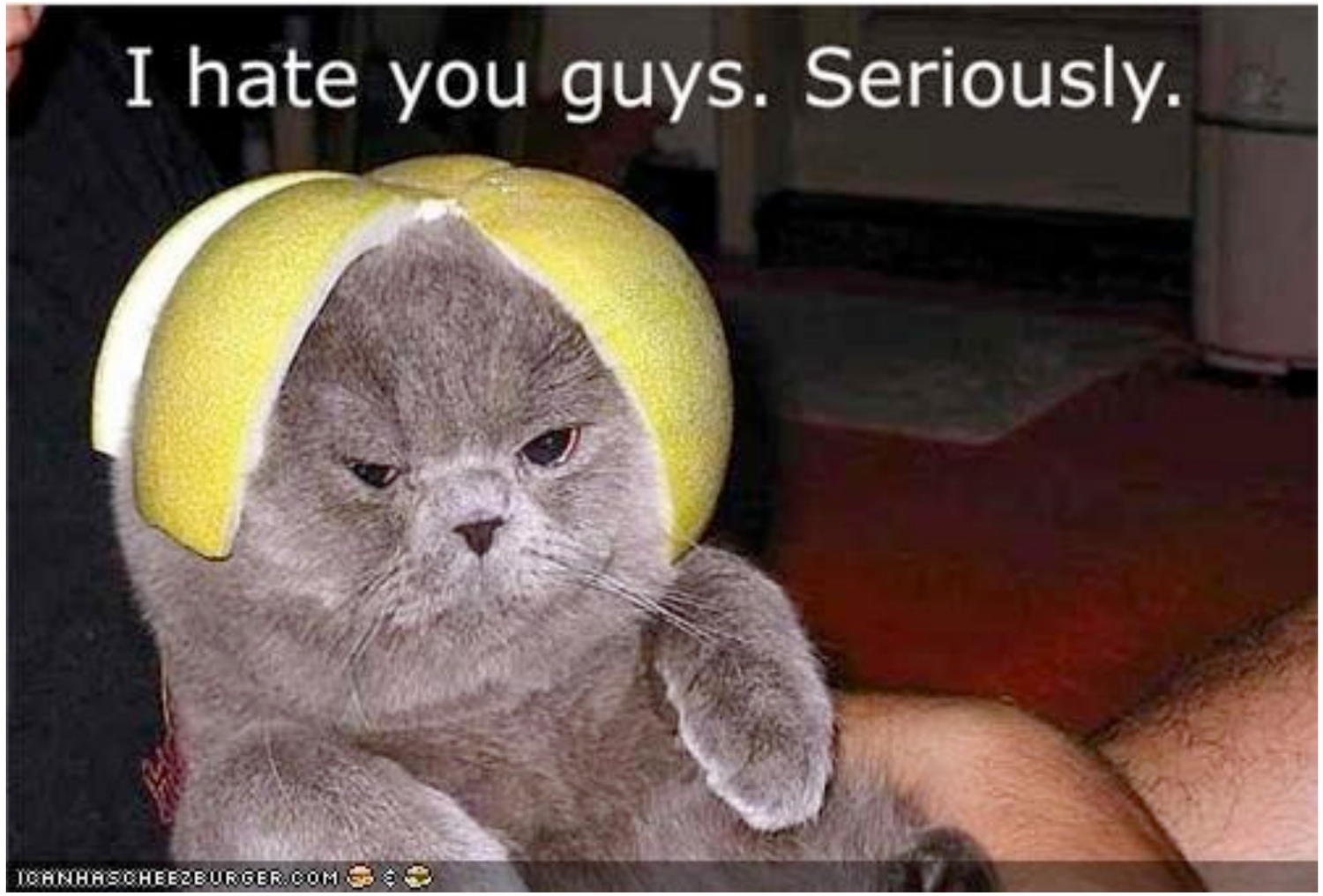
Part 3 - Routine tasks

Generating PTRs

* In any ISP environment you configure new router (sub-)interfaces on a daily basis
* IPv4 and IPv6
* Do you always add a PTR for those newly configured IP addresses?

Generating PTRs

* Do you update your DNS if you swap a router and the interface name changes?
* Are you sure all your PTRs are correct?



Generating PTRs

`compareifip.py`

* finds IP addresses for all interfaces (v4 and v6) and all devices
* generates PTR for interface
  * e.g.: Gi0/0/0.400 on core1.f.test.man-da.net will become ge-0-0-0-400.core1.f.test.man-da.net
* compares generated PTR with DNS data
* prints differences sorted by zone

PTRs currently in DNS

82.195.67.22: ge-0-0-0-334.core1.f.test.man-da.net.

2001:41b8:ff:a::10: ge-0-0-0-334.core1.f.test.man-da.net.

Output compareifip

```
lysis@sparkles:~$ compareifip.py
Reverse Delegation for 67.195.82.in-addr.arpa:
222 IN PTR ge-2-0-3-334.core2.da.test.man-da.net.

Reverse Delegation for f.f.0.0.8.b.1.4.1.0.0.2.ip6.arpa:
0.1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.a.0.0.0 IN PTR
ge-2-0-3-334.core2.da.test.man-da.net.
```

Conclusion

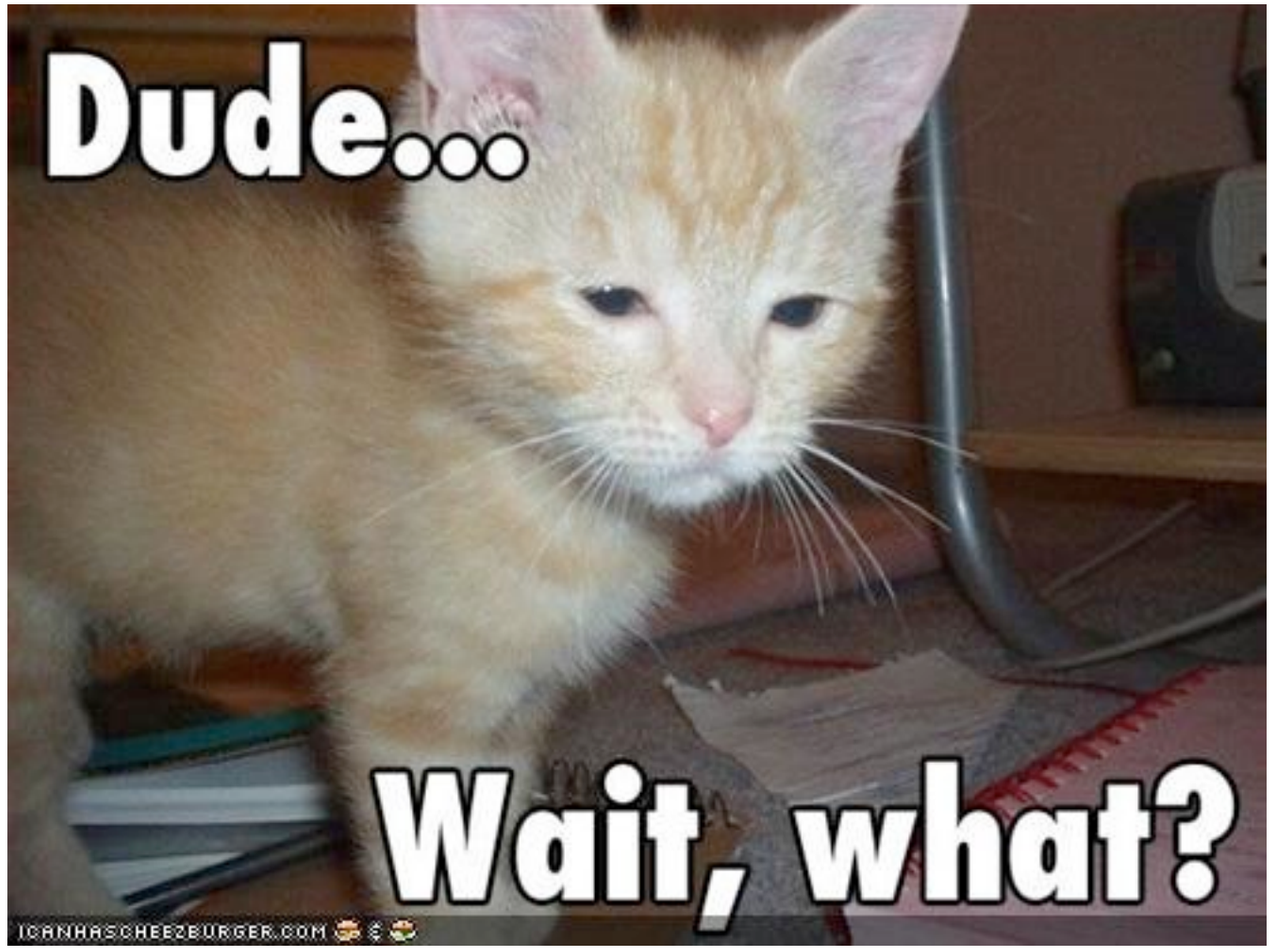
Getting the scripts

* www.man-da.de/software/
* Collection of Python scripts and libraries
* Includes functions to parse Juniper and Cisco configuration
* Starting point for adding new functionality

Getting the scripts

* Need ideas?
  * generate interface descriptions from database
  * generate configuration templates, e.g. for customer interfaces and check all customer interfaces against those templates

Questions?



One more thing …