Note on Software Construction and Reliability for Privately Signed Root Zones

Paul Vixie, TISF

November, 2016

Seoul, KR

Odd use of “git”

- Three coordinators, who are peers wrt system ops/config
- We have a three-way shared read-write “git” repo
- At TISF, we fetch from the file system, to avoid key management:
  - ```
    git remote -v
    origin  /home/yeticonf/dm (fetch)
    origin  /home/yeticonf/dm (push)
    ```
- At TISF, we push to all three coordinators:
  - ```
    git remote -v
    origin  yeticonf@yeti-conf.dns-lab.net:dm (fetch)
    origin  yeticonf@yeti-conf.dns-lab.net:dm (push)
    origin  yeticonf@yeti-dns.tisf.net:dm (push)
    origin  yeticonf@yeti-repository.wide.ad.jp:dm (push)
    ```

Limitations and Lessons of using “git” this way

- Won’t support a large set of project members, or high update rates
- Difficult to set up, verify, monitor, and debug
- Can require out-of-band notification of changes
- Inadequately protects the KSK and ZSKs
- In all ways, unsuitable for Internet-scale production work
- Can work for science and enterprise networks

“To Hell With It, Let’s Just Put It In Cron”

- ```
  crontab -l
  SHELL=/bin/sh
  MAILTO=vixie@tisf.net
  40 * * * * cd ~/work/yeti-dm && sh scripts/cronrun-often.sh
  ```
- One of the coordinators runs at :00, one at :20, one at :40
- Most hours, there is no new IANA zone, so, no work to be done
- `cronrun-often.sh` only produces output on work (or failures)
  ```

The Cron Job

- grep '^[#$]' work/yeti-dm/scripts/cronrun-often.sh

```
#!/bin/sh
# first, fetch the iana zone from f-root, and fetch yaml config
from yeticonf
# second, remake the conf-include file (allow-transfer, also-
notify)
# third, create the yeti zone based on the iana zone, and sign it
# fourth and finally, reload and reconfig as needed
```

- if dnssec-signzone -Q -R -o . -x yeti-root.dns $keys \

```
                > dnssec-signzone.out 2>&1
        then
                rndc -s yeti-dm reload . 2>&1 \
                        | grep -v 'zone reload up-to-date‘
        else
                cat dnssec-signzone.out
                exit 1
        fi
```

The YAML File

```
- name: ns-yeti.bondis.org
  public_ip: 2a02:2810:0:405::250
- name: yeti-ns.ix.ru
  notify_addr:
  - 2001:6d0:fffc:4000::2
  - 2001:6d0:6d06::53
  public_ip: 2001:6d0:6d06::53
  transfer_net:
  - 2001:6d0:fffc:4000::2
  - 2001:6d0:6d06::53
```

The conf-include File

- allow-transfer {
  …
  2a02:cdc5:9715:0:185:5:203:53;
  …
  2001:6d0:fffc:4000::2; 2001:6d0:6d06::53;
  …
  also-notify {
  …
  2a02:cdc5:9715:0:185:5:203:53;
  …
  2001:6d0:fffc:4000::2; 2001:6d0:6d06::53;
  …

Lessons So Far

- Need to automate the creation of new MZSK's
- Need to crypt and sign the KSK and MZSK's at rest
- Need to automate the RFC 5011 timer-based key management
- Need external reviewers and users – publication would add sunshine
- Could be used to create locally signed root zones for local production
- Perl isn't dead (Net::DNS is particularly useful for this application)
- ISC BIND9 server and utilities are ideally suited to this task
- Bourne shell isn't dead, and makes for easy-to-review mainline