Firewalld, libnftables, and json, oh my

Eric Garver
egarver@redhat.com
eric@garver.life
Freenode: erig
Github: erig0
libnftables

- Library for interacting with nftables
- >= 0.9.0
- man libnftables
- Initially started by Eric Leblond
- Recent work and upstreaming by Phil Sutter
- functions
  - nft_run_cmd_from_buffer()
  - nft_run_cmd_from_filename()
libnftables JSON

- man libnftables-json
- Build time: --with-json (libjansson)
- Functions
  - nft_ctx_output_set_flags(ctx, ... | NFT_CTX_OUTPUT_JSON)
libnftables JSON example

Add a table

```json
{
   "add": {
      "chain": {
         "family": "inet",
         "hook": "prerouting",
         "name": "raw_PREROUTING",
         "prio": -290,
         "table": "firewalld",
         "type": "filter"
      }
   }
}
```
libnftables JSON example

Add a reject rule

```json
{
    "add": {
        "rule": {
            "chain": "filter_FORWARD",
            "expr": [{"reject": {"expr": "admin-prohibited",
                                "type": "icmpx"}}],
            "family": "inet",
            "table": "firewalld"}
    }
}
```
JSON CLI

- Restore a ruleset
  - `nft -j -f .../rules.json`
- Save a ruleset (no string quotation issues!)
  - `nft -j list ruleset > .../rules.json`
python-nftables

- Thin python wrapper for libnftables (ctypes)
- Converts python data structures to/from JSON
- Build time: --enable-python
- py/nftables.py
firewalld

- abstraction over nftables/iptables/ipset
- has an nftables backend since 0.6.0 (July 2018)
- libnftables (python-nftables) support not yet upstream (soon!)
  - Thanks Phil/Pablo for cache fixes!
- Sends very large JSON blobs to python-nftables
firewalld example

Python side

```python
def add_rule(proto, port):
    TABLE_NAME = "iptables"  # Change to your table name
    target = "allow"  # Change to your target
    return {
        "add": {
            "rule":
                {
                    "family": "inet",
                    "table": TABLE_NAME,
                    "chain": "%s_%s_allow" % (table, target),
                    "expr": [{
                        "match": {
                            "payload": {
                                "protocol": proto,
                                "field": "dport"
                            }
                        },
                        "op": "==",
                        "right": self._port_fragment(port)
                    },
                    {
                        "accept": None
                    }]
                }
        }
    }
```
firewalld example

Generated JSON passed to libnftables

```json
{"add": {
    "rule": {
        "chain": "filter_IN_public_allow",
        "expr": [{"match": {"left": {"payload": {"field": "dport",
                                        "protocol": "tcp"}}},
                  "op": "+",
                  "right": 22}],
        "accept": null},
        "family": "inet",
        "table": "firewalld"
    }
}
```
The reply (--echo, --handle)

```json
{"add": {
"rule": {
"chain": "filter_IN_public_allow",
"expr": [{"match": {"left": {"payload": {"field": "dport",
"protocol": "tcp"}}},
"op": "==",
"right": 22}},
{"accept": null}],
"family": "inet",
"table": "firewalld",
"handle": 1234
}
```
Using libnftables JSON: Insights

- Only use >=0.9.1
- Output is the same as input, but with handles inserted
  - Including delete/flush/etc
- No more double quoting strings or escaping due to shell
  - “‘foobar’” → “foobar”
  - \;
  - “priority -123”
Using libnftables JSON: oddities

- Some keys are different than CLI
  - hook “prio” vs “priority”
- “ct helper” not {“ct”: {“key”: “helper”}}
- Sometimes significantly more verbose than CLI
  - “ip daddr”
    vs
    {“payload”: {“protocol”: “ip”, “field”: “daddr”}}
  - “1.2.3.4/24”
    vs
    {“prefix”: {“addr”: “1.2.3.4”, “len”: 24}}
Using libnftables JSON: wish list

- Cookies for matching sent JSON with reply
  - e.g. `{“add”: {“rule”: {“cookie”: 123456, ... }}}`
  - workaround is to use a comment, but that affects the ruleset (i.e. `nft list ruleset`)
- Further define JSON schema