NAME

uruk-rc – uruk resource file, defining access policy

SYNOPSIS

`/etc/uruk/rc`

DESCRIPTION

rc is a shell script snippet, sourced in **uruk** by /bin/sh.

rc lists IP addresses, allowed to use services.

EXAMPLES

default

The simplest valid *rc* file is the empty file. This *rc* file blocks all TCP and UDP connection attempts to services on our host: this is the default behaviour.

simplest

The simplest *rc* file which does allow traffic to our services looks like e.g.:

```
interfaces=eth0

ips_eth0=default
ip_eth0_default=192.168.26.27
net_eth0_default=192.168.0.0/16

ip6_eth0_default=2001:db8::1/64
net6_eth0_default=2001:db8::/32

services_eth0_default_tcp=local
ports_eth0_default_tcp_local="0:65535"
sources_eth0_default_tcp_local="0.0.0.0/0 ::/0"

services_eth0_default_udp=local
ports_eth0_default_udp_local="0:65535"
sources_eth0_default_udp_local="0.0.0.0/0"
```

This *rc* file allows all IPv4 and IPv6 UDP and TCP traffic from publicly routable IPs to eth0's IP.

realistic

If you'd like to block traffic on wlan0 and allow traffic to ssh on your wired interface, and don't like to explicitly set your IPs in *rc*:

```
# list of interfaces you'd like uruk to protect
interfaces=eth0 wlan0

# set variables ip{,6}_eth0_default and net{,6}_eth0_default
. /lib/uruk/init/autodetect-ips

# names for eth0's 2 IPv4 addresses
ips_eth0="default dhcp"

# allow access to our sshd on eth0's primary IP on tcp port 443
# from anywhere
services_eth0_default_tcp=ssh
ports_eth0_default_tcp_ssh=443
sources_eth0_default_tcp_ssh="0.0.0.0/0 ::/0"

# we get a static IPv4 via dhcp
ip_eth0_dhcp=10.0.0.3
net_eth0_dhcp=10./8

services_eth0_dhcp_tcp=http
ports_eth0_dhcp_tcp_http=http
sources_eth0_dhcp_tcp_http=$net_eth0_dhcp

# we leave services_wlan0_default_{tcp,udp} unset: don't allow any
# incoming connections on wlan0's default IP
```

autodetect-ips

The script autodetect-ips --as used in the previous example-- looks for files /etc/sysconfig/network-scripts/ifcfg-* (commonly found at e.g. Red Hat and Fedora systems) and /etc/network/interfaces (as found at e.g. Debian and Ubuntu systems), and, for each interface *nic*, and each found IPv4 and IPv6 address and network, sets variables `ip_`*nic*`_default`, `ip6_`*nic*`_default`, `net_`*nic*`_default` and `net6_`*nic*`_default` . Then it calls ip(8) and adds any other found *nic*, `ip` and `net` triplets (for IPv4 and, for IPv6, only addresses in scope "global").

The script autodetect-ips is useful if you'd like to share your *rc* file among different hosts.

another example

For an even more reasonable *rc* file, look at the well-commented example *rc* file in `/usr/share/doc/uruk/examples/rc`.

IPv4 AND IPv6

You can mix IPv4 and IPv6-addresses in sources_*. E.g.:

```
ips_eth0='default private'
ip_eth0_default=1.2.3.4
ip6_eth0_default=

services_eth0_default_tcp='mail local'

sources_eth0_default_tcp_mail='10.0.0.0/24 192.0.32.0/24 192.168.6.26'
sources_eth0_default_tcp_local='192.0.32.0/24 svejk.example.com 2001:db8::/32'

ports_eth0_default_tcp_mail=smtp
ports_eth0_default_tcp_local='ssh ftp'
```

If svejk.example.com has both an IPv4 PTR record in DNS, as well as an IPv6 PTR record, connection attempts from svejk to the ssh and ftp TCP ports are allowed, via both IPv4 and IPv6.

Uruk used to require variables sources6_* to be set to support ip6tables. Since uruk version 20140319 (The Alfama Release), this is no longer needed; setting sources_* suffices. To be precise, the semantics since uruk version 20140319 is: 1) If both sources_* and sources6_* are defined (even if they're just empty), each is used for its respective address family. (This ensures backwards compatibility.) 2) If sources6_* is undefined, sources_* is used for both v4 and v6. 3) In either case, v4 literals in v6 context and v6 literals in v4 context are silently (!) ignored.

HOOKS

Uruk offers hooks for inserting your own code between iptables invocations. Examples will show the usefulness of these hooks.

allowing broadcasts
In *rc*, there is:

```
rc_b=$etcdir/bootp
```

while the file `bootp` reads

```
iptables -A INPUT -m state --state NEW -i eth0 \
  --protocol udp --destination-port bootps -j ACCEPT
```

. This enables one to add rules for packets with broadcast addresses in their destination. (Uruk has no support for this in its regular *rc*.)

allowing non-matching returntraffic
In *rc* there is:

```
rc_d=$etcdir/dns
```

while the file `dns` reads

```
 for source in 10.5.0.27 10.56.0.40
 do
   $iptables -A INPUT -i eth0 --protocol udp \
     --source "$source" --source-port domain \
     --destination "$ip_eth0" \
     --destination-port 30000: -j ACCEPT
 done
```

This allows one to allow (return)traffic, disregarding the state. (Uruk has no support for this in its regular *rc*.)

allowing NAT
In *rc* there is:

```
rc_a=${etcdir}/nat
```

while the file `nat` reads

```
  $iptables -t nat -A POSTROUTING \
    --out-interface eth0 -j SNAT \
    --to-source $ip_eth0
```

This allows Network Address Translation. However, beware! Like all extensive use of hooks, this will break the **uruk-save** script. If you make sure your active iptables rules are wiped, and invoke **uruk** manually to load new rules, you're safe. Using the init script with its default settings is safe too.

allowing IPv6 tunneling
In *rc* there is:

```
rc_b=${etcdir}/proto_41
```

while the file `proto_41` reads

```
$iptables -A INPUT -i ppp0 --protocol 41 --destination $ip_ppp0 -j ACCEPT
```

This allows IP protocol 41, typically used for this kind of tunneling.

allowing any traffic on an interface
In *rc* there is:

```
interfaces_unprotect="lo eth2"
```

This allows any traffic on `eth2` (and on `lo`, the default), including any ICMP packets and packets from any source address.

using multiple hooks at one entry point in the main uruk process
In case rc_a, rc_b, ... , or rc_i does not have a file as its value, but a directory, all files matching "$rc_x"/*.rc will get sourced. This helps configuration management in complex situations involving lots of uruk configuration files for lots of hosts.

See the section "THE GORY DETAILS: uruk INTERNALS" in **uruk(8)** (or the **uruk** source) to find out which hook (there are hooks rc_a, rc_b, ... , rc_i) to use.

NETWORK INTERFACES WITH MULTIPLE IP ADDRESSES

Uruk supports situations where a network interface has more than one IP address attached. Variables `ips_`*nic* and `bcasts_`*nic* are used for this.

If `ips_`*nic* is set, e.g. like

```
ips_eth0="ip0 ip1 ip2"
```

we assume multiple (three in this example) IPs are assigned to `eth0`. If this variable is not set only one IP is supported on `eth0`.

In multiple-IP mode, IP addresses are listed as e.g.

```
ip_eth0_ip0="137.56.247.16"
```

(If you're used to the Linux ifconfig(8) output, you could use the name `ip` for `eth0`, and `ip0` for `eth0:0`.) The *ports*, *services* and *sources* variables look like e.g.

```
services_eth0_ip2_tcp=local
ports_eth0_ip2_tcp_local=smtp
sources_eth0_ip2_tcp_local=$localnet
```

and, similarly,

```
net_eth0_ip1=192.168.0.0/16
```

Furthermore, for dropping broadcast packets, specify e.g.

```
bcasts_eth0="ip0 ip2"     # yes, possibly a subset of ips_eth0
bcast_eth0_ip0="10.0.0.255"
bcast_eth0_ip2="10.0.255.255"
```

As an additional feature, if you have multiple IP addresses that all need to get the same rules, you can assign them to a single name:

```
ip_eth0_ip0="137.56.247.16 137.56.247.17 137.56.247.18"
```

LOGGING AND DEBUGGING

Uruk has support for logging network packets, and for debugging the uruk script.

Logging

By default, uruk logs denied packets. This is adjustable using the *loglevel* variable. The settings are:

- "zero": be silent; do not log any packet. *rc* file features `loglevel=10`.
- "low": log denied packets, which are targeted at one of our IPs. *rc* file features `loglevel=30`.
- "medium": log denied non-broadcast packets. This is the default: *loglevel* is unset or *rc* file features `loglevel=50`.
- "fascist": log all packets. *rc* file features `loglevel=90`.

Debugging

To debug the **uruk** script, invoke uruk as

```
sh −x /sbin/uruk
```

this shows what is done, along with executing it. (Like an uruk '-v' option.)

If you'd rather prefer not to execute, but just watch what would've been done, invoke uruk as

```
URUK_IPTABLES='echo iptables' URUK_IP6TABLES='echo ip6tables' uruk
```

(Like an uruk '-n' option.) If you have this statement set, you can run **uruk** under a non-privileged user account.

If you'd like to test a new *rc* file before installing it, run something like:

```
URUK_CONFIG=/path/to/new/uruk/rc/file uruk
```

Of course, all these tweaks can be combined.

VARIABLES

The uruk script honors the following variables in *rc* files:

- "version" Uruk version compatibility of this *rc* file
- "loglevel"
- "iptables" Full pathname of iptables executable.
- "ip6tables" Full pathname of ip6tables executable.
- "interfaces" List of network interfaces.

More variables are available. For now, you'll have to take a look at the example *rc* file in `/usr/share/doc/uruk/examples/rc` for more details.

ENVIRONMENT VARIABLES

See **uruk(8)** for a list of honored environment variables.

FILES

`/etc/uruk/rc`

SEE ALSO

A well-commented example *rc* file is in `/usr/share/doc/uruk/examples/rc`. And see **uruk(8)**, **uruk-save(8)**.

COPYRIGHT

Copyright (C) 2005, 2007, 2008, 2010, 2011, 2012, 2013 Joost van Baal-IliÄ <joostvb-uruk@mdcc.cx>

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see http://www.gnu.org/licenses/.

AUTHOR

Joost van Baal-IliÄ <joostvb-uruk@mdcc.cx>