Network iPXE dnsmasq Examples PXE BOOT

From richud.com

Contents

Primers / info

It is extremely easy to get into a ravel with this, it isn't helped by some rather dodgy documentation, I hope this makes it a bit clearer.

Dnsmasq

- A "#" or "!" negates a match
- dnsmasq is acting on a 192.168.2.x subnet in these examples
- dhcp-match=ipxe,175

  This sets a 'variable' to be 'ipxe' (whatever the text is before the comma), if option 175 is set in the DHCP request. (175 returns text string 'Etherboot', after original project name)

  ipxe/gpxe have this set if they are doing the DHCP request.

  The ethernet card's pxe boot code wont have this set.

  This lets them be differentiated by the DHCP server, so it can do different things.

- xxxx=net:#ipxe,yyyy

  This is a rule match condition, the setting of xxxx to a value yyyy after the

comma only takes place if the 'variable' 'ipxe' IS NOT set (# = not).

In this case xxxx gets set to yyyy if it was a dhcp request from the ethernet card, but wouldnt if it was from ipxe.

- xxxx=net:ipxe,yyyy

  this is the opposite and only sets xxxx to yyyy if it matches, i.e. a request came from ipxe with 175 set

- To enable dnsmasq logging to syslog append this to /etc/dnsmasq.conf

  log-queries
  log-dhcp

- Certain settings only work together,

  These 3 all work together, when acting as a true DHCP server
  ```
  dhcp-range=192.168.2.100,192.168.2.200
  dhcp-boot=
  dhcp-option-force=
  ```

  Wheras these two work together, (dhcp-option-force= doesn't work when using 'proxy') and (pxe-service= essentially takes over the 'function' of what dhcp-boot does.)
  ```
  dhcp-range=192.168.2.0,proxy
  pxe-service=
  ```

- Note dhcp-boot you need to use the full name of the next file to load (blah.0) wheras with pxe-service you omit the .0 (just blah)

pxelinux

DHCP options 209/210 are use to configure pxelinux. These can either come forcibly set in a ipxe script or via the DHCP server normally.

- 209 = pxelinux config path, "pxelinux.cfg/default" . This relative to the 'root'
- 210 = pxelinux 'root', "http://xxx.xxx.xx.xx:81/" . This is prefixed with HTTP (in this case on port 81) , so pxelinux will use instead of slow TFTP . It can also use FTP.

dnsmasq.conf real DHCP server

This acts as a normal DHCP server, passing out dhcp options 209/210 to pxelinux

How it works

- Client boots and net card does a DHCP Discover, it will get a DHCP offer from dnsmasq
- Client does a DHCP Request, which dnsmasq will ACK, telling it to get undionly.kkpxe.0 (as no option 175 "Etherboot").
- Client then gets undionly.kkpxe.0 via TFTP.
- ipxe then does a DHCP Discover/offer/request/ack and dnsmasq should respond this time telling it to load pxelinux.0 via TFTP (now option 175 is set, so it no longer tries to pass unidonly.kpxe.0)
- Passes DHCP options 209/210 passed, which pxelinux later looks at for the config path and document root.
- pxelinux.0 loads via TFP.
- pxelinux then takes over boot process, all data now going via HTTP.

```
enable-tftp
dhcp-match=ipxe,175
dhcp-range=192.168.2.100,192.168.2.200
tftp-root=/tmp/memstick/host0_part1/
dhcp-boot=net:#ipxe,undionly.kkpxe.0
dhcp-boot=pxelinux.0
dhcp-option-force=209,pxelinux.cfg/default
dhcp-option-force=210,http://xxx.xxx.xx.xx:81/
```

- Note if running this so its competing with a 'normal' DHCP server , although it should accept the first PXE boot request from a machine, the second one from iPXE seems to fail as iPXE ACK's the real dhcp server, not the pxe one (dnsmasq). (Needs looking into)

dnsmasq.conf proxy DHCP server

This relies on an existing DHCP server for normal network settings, IP etc. and dnsmasq just deals with extra PXE stuff (altbootservice)

How it works

- The client DHCP Discovers, to which real dhcp server and dnsmasq both reply to.
- The 'real' DHCP server then gets ACK'd by client and gets network settings.
- The dnsmasq server gets a 'altserviceboot' request on port 4011 via UDP from client
- Dnsmasq tells client the filename to get, undionly.kkpxe.0
- Client requests undionly.kkpxe.0 via TFTP
- undionly.kkpxe.0 boots and runs ipxe
- ipxe sees the internal script and executes it, ignoring anything else. (this sets pxelinux DHCP options and then loads pxelinux.0, see below)

```
dhcp-match=ipxe,175
enable-tftp
```

```
dhcp-range=192.168.2.0,proxy
tftp-root=/pxe/
pxe-service=net:#ipxe,x86PC, "splashtop by richud.com", undionly.kkpxe
```

ipxe config file

undionly.kkpxe.0 is embedded at compile time with script (below) & pxelinux.0

How it works

- Uses cached dhcp info
- DHCP option 209/210 are overidden to configure pxelinux to boot from location specified, via HTTP in this example
- Boots embedded pxelinux.0

```
#!ipxe
set use-cached 1
dhcp
set 209:string pxelinux.cfg/default
set 210:string http://xxx.xxx.xx.xx:81/
boot pxelinux.0
```

dnsmasq.conf proxy DHCP and PHP

Same as above for initial bits (DHCP/altbootservice, Please read first)

- Initially option 175 is empty, pxe-service rule returns true, and undionly.kkpxe.0 will be sent.

  dhcp-boot rule returns false and is ignored. [however it would be ignored anyway in proxy mode]

- Another DHCP Discover is then generated from ipxe (not sure what triggers this?)
- dnsmasq sees DCHP request option 175 is set to ipxe, dhcp-boot rule evaluates true and returns the dhcp-boot value,
- ipxe interprets as a "chain to a config file", which is ipxe.php?xxx and passes the MAC and UUID variables it does with variable substitution on the fly.

```
dhcp-match=ipxe,175
dhcp-boot=net:ipxe,http://xxx.xxx.xx.xx/ipxe.php?mac=${mac}&uuid=${uuid}
enable-tftp
dhcp-range=192.168.2.0,proxy
tftp-root=/pxe/
pxe-service=net:#ipxe,x86PC, "splashtop by richud.com", undionly.kkpxe
```

- Note you can't replace pxe-service with dhcp-boot=net:#ipxe,undionly.kkpxe.0 as you may imagine, as its in proxy

mode and using altbootservice via UDP/4011 at this point.
- Note not setting/setting incorrectly the ipxe rule match in pxe-service will make ipxe go in a loop and keep booting undionly.kkpxe.0
- Note Any script/files built into the initially selected undionly.kkpxe will override everything.

ipxe.php generating ipxe config file

Example php generated config file

e.g. UUID=0017001E-8C00-00AD-5087-90E6BA2F3423

This would tell iPXE to chain load pxelinux from http://xxx.xxx.xx.xx:81 /pxelinux.0, with the config file to look at being http://xxx.xxx.xx.xx:81 /pxelinux.cfg/0017001E-8C00-00AD-5087-90E6BA2F3423.cfg

If it fails to find this chain load to another config file from ipxe.org, namely http://boot.ipxe.org/demo/boot.php (which in turn chain loads a small test linux system)

If that then fails drop to the iPXE shell.

foreach turns any vars passed into varibles of the same name. These can be used to determine what to dish out next in iPXE script.

```
<?PHP
foreach($_GET as $k=>$v) $$k=$v;
echo "#!ipxe
set 209:string pxelinux.cfg/$uuid.cfg
set 210:string http://xxx.xxx.xx.xx:81/
chain \${210:string}pxelinux.0 ||
chain http://boot.ipxe.org/demo/boot.php ||
shell
";
?>
```

- Note The text "#!ipxe" must appear on the FIRST LINE of the response, else it will fail.
- Note escape the $ as it will get literal interpretation by PHP otherwise

For more info, another example is here , which is used for a rescue mode of a server (for example). Using PHP is a great way to extend functionality as it can be instantly changed, unlike embedded scripts or dhcp config files that need services restarting.

Category: Network
