DNS

Edit New Page

Michael Bikovitsky edited this page on Oct 21, 2016 · 34 revisions

Index

Description

DNS primarily uses User Datagram Protocol (UDP) on port number 53 to serve the requests, TCP (Transmission Control Protocol) is only in usage if data size exceeds >512 bytes or for special tasks, some resolvers implementing an TCP for all queries option (mostly optional by default).

DNS is known as not to be secure anymore for several reasons. Like most "secure" communications protocols, it is susceptible to undetectable public-key substitution MITM-attacks an populate examples was the Apple iMessages security problem.

The main problem is the protocol insecurity by DNS and X.509. See also Certificate Transparency.

The problems are:

- MITM (Man-In-The-Middle)
- DNS-based censorship circumvention
- Domain theft's ("seizures")
- Certificate revocation
- DOS (Denial-Of-Service) attacks
- DRDoS (UDP prot. based attacks)
- Logging on exit notes
- DNS cache poisoning
- Other exploits that are used for e.g. phishing
- DNS hijacking
- Pre-defined DNS resolvers (hardcoded)

▾ Pages 28

Find a Page...

- Home
- Advertisements blocking
- Android kernel traffic
- Apps leak private user data during boot
- BusyBox
- Contributors
- CustomScripts
- DNS
- Error codes
- FAQ
- Google Play services and other special cases
- HOWTO blocking WhatsApp
- HOWTO Compile AFWall
- HOWTO Compiling busybox
- HOWTO Compiling iptables

Show 13 more pages...

Clone this wiki locally

https://github.com/ukanth/a

Clone in Desktop

- Typosquatting
- Zone File Compromise / Zone Information Leakage/DNS Footprinting
- DNS Amplification Attack
- DNS Vulnerabilities in Shared Host Environments
- DNS Client flooding

Blocking the DNS (Port 53) isn't possible (without problems) since this is necessary on Android/Windows/Linux/Mac OS or any other OS, but we simply can use secure and proofed alternatives. - Which is more or less complicated and depending on your knowledge about how to change that. But don't worry we are here to explain all stuff in easy steps!

There are alternatives like:

- Choosing a logging free DNS resolver e.g OpenNIC
- DNSSEC / DNS-based Authentication of Named Entities (DANE)
- OpenDNS (or other providers which claiming to not log/censorship anything)
- DNSCrypt / *TCPCrypt* (DNSCrypt will be a part of the final CM 12.1)
- TACK / HPKP implementations within the apps

But even with such popular alternatives there are several problems, e.g. DNSSEC suffering from miscellaneous leaks mentioned over here or here + it doesn't prevent MITM attacks.

DNS under Android

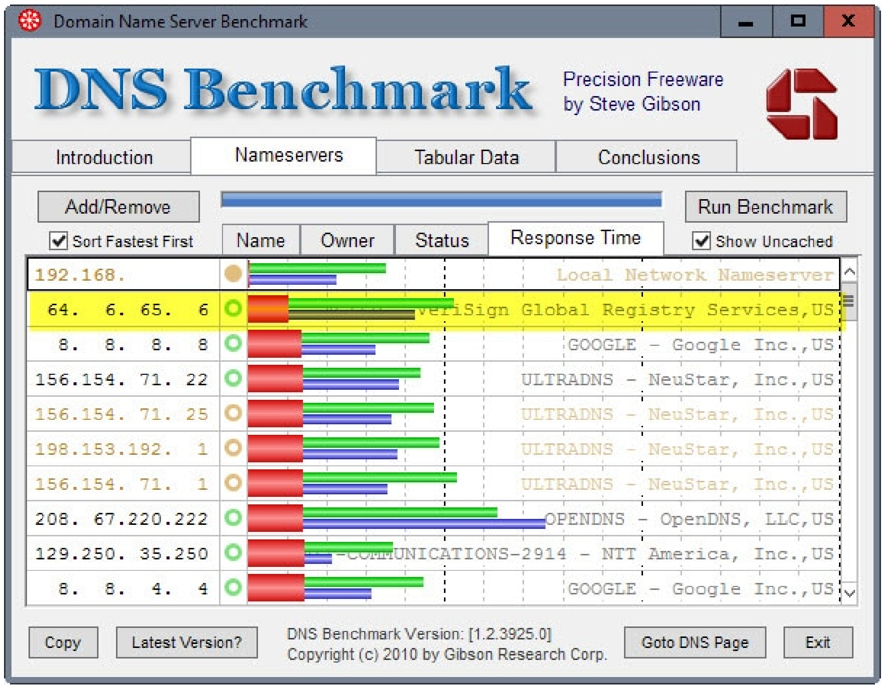
Android use a a dynamic DNS (DDNS) by default, which updates the DNS every time when the IP address changes ISP <-> wifi.

By default the Google DNS server (since 2009) is set (8.8.8.8/8.8.4.4), currently the DNS servers gets overridden after each reboot even with setprop. Googles public DNS supports DNSSEC validation since 2013 by default unless you do not want this (via opt-out), which means that this server is secured and nothing speaks against using it (except the anti-Google paranoia of course).

// IP change or reconnection (connectivity changes -> RIL via e.g. ndc resolver setifdns rmnet0 x.x.x.x y.y.y.y).

Benchmark

An small and fast benchmark utility is available over here (also runs under Wine).



```
192.168.                  Local Network Nameserver
 64.  6. 65.  6           VeriSign Global Registry Services,US
  8.  8.  8.  8           GOOGLE - Google Inc.,US
156.154. 71. 22           ULTRADNS - NeuStar, Inc.,US
156.154. 71. 25           ULTRADNS - NeuStar, Inc.,US
198.153.192.  1           ULTRADNS - NeuStar, Inc.,US
156.154. 71.  1           ULTRADNS - NeuStar, Inc.,US
208. 67.220.222           OPENDNS - OpenDNS, LLC,US
129.250. 35.250           -COMMUNICATIONS-2914 - NTT America, Inc.,US
  8.  8.  4.  4           GOOGLE - Google Inc.,US
```

DNS Benchmark Version: [1.2.3925.0]
Copyright (c) 2010 by Gibson Research Corp.

How can I gather DNS (A/AAA/...) requests?

~~All AOSP based ROMs coming with TCPdump as binary included, so we can just use this to show what's going on behind, there are several Tutorials and documents available to handling TCPdump. If this is to complicated for you, you can just grab AdAway (needs root) and use there own TCPDump/dnsmasq/libpcap interface to list all requests - it also provides an interface to add them to your hosts or to an separate white-/blacklist.~~

// Android use a kind of BIND (which includes "dig"). `dig github.com | grep "Query time"` // See, #37668 // Workaround, just use external apps like DNS Lookup (it use nslookup)

resolv.conf

The configuration file for DNS resolvers is `/etc/resolv.conf` take a look at the man page.

Allows a maximum of three nameserver lines. In order to overcome this limitation, you can use a locally caching nameserver like dnsmasq (see below). Changes made to /etc/resolv.conf take effect immediately (needs confirmation?).

dnsmasq

Dnsmasq provides services as a DNS forwarder cacher and a DHCP server. As a Domain Name Server (DNS) it can cache DNS queries to improve connection speeds to previously visited sites, and as a DHCP server dnsmasq can be used to provide internal IP addresses and routes to computers on a LAN. A list for discussion about the dnsmasq DNS and DHCP server, configuration, bugs and development can be found here.

❗ dnsmasq only allows three nameservers as a workaround we can create `resolv.dnsmasq.conf` and add this list into our `/etc/dnsmasq.conf` -> *resolv-file=/etc/resolv.dnsmasq.conf*

// listen-address=127.0.0.1 needs to be set for local DNS cache ... because DNSCrypt isn't an DNS Cache (so we use dnsmasq) // Empty example dnsmasq // start it via service dhcpsrv /system/bin/logwrapper /system/bin/dnsmasq -k -C /system/etc/dnsmasq.conf // Behind a firewall (like AFWall+) try: iptables -A INPUT -i $iface -p udp -s 0.0.0.0 --sport 68 -d 255.255.255.255 --dport 67 -j ACCEPT if the firewall precenting dnsmasq from working as LAN DHCP server

DNSCrypt

DNSCrypt encrypts and authenticates DNS traffic between user and DNS resolver, the latest flashable .zip for Android is available over here. When DNSCrypt is enabled, *dnscrypt-proxy* accepts incoming requests on `127.0.0.1:53` to one chosen DNS resolver. Compatible resolver names are visible under `/etc/dnscrypt-proxy/dnscrypt-resolvers.csv` (always latest dnscrypt-resolvers.csv).

// Unbound, dnsmasq (Android default) or pdnsd are working with DNSCrypt but may needs configuration changes to work proper together

```
// Ensure that DNSCrypt is running
dnscrypt-proxy enable
dnscrypt-proxy disable
ps w | grep dnscrypt-proxy
```

```
// An example could be looks like this, the .zip package normally contains an init.d but
this is just in case
# DNSCrypt copy & paste example config for init.d

RESOLVER_NAME=dnscrypt.org-fr

dnscrypt-proxy \
  --resolvers-list=/system/etc/dnscrypt-proxy/dnscrypt-resolvers.csv \
  --resolver-name=<INSERT-YOUR-DNS-RESOLVER-NAME-HER> \
  --provider-key=<OPTIONAL-ONLY-NESSARY-IF YOU-ARE-NOT-USING-ONE-FROM-the-resolvers.csv-
list!> \
  --provider-name=<OPTIONAL-ONLY-NESSARY-IF YOU-ARE-NOT-USING-ONE-FROM-the-resolvers.csv-
list!>  \
  --resolver-address=<OPTIONAL-IF-THIS-is-different!> \
  --max-active-requests=100 \
  --edns-payload-size=1252\
  --test=3600 && \
dnscrypt-proxy \
  --daemonize \
  --loglevel=3
  --resolver-name="$RESOLVER_NAME" \
  --resolvers-list=/system/etc/dnscrypt-proxy/dnscrypt-resolvers.csv
```

// All other command-line switches can be found on the official homepage and of course within the documents.

Currently DNSCrypt is far away from been stable and daily usage, since there are several troubles. If you're not use special cases like tethering/paring or other features you can us it, but as long the problems are not fixed it can't be recommend for the mass.

How do I know if my applications are leaking DNS?

The following systems or apps are suffering from DNS leakages:

- Windows 8 and 8.1, fix
- Windows 10 [no fix available]
- OpenVPN (you need some scripts to disable all DNS in external interfaces)
- All Android versions below 3.x

There are several ways, the most easiest way is to visit some web pages that automatically detect what is your current DNS, like:

- DNS Leak Test | dnsleaktest.com
- IP Leak Test | ipleak.net

If your Browser shows a wrong DNS according to what your own settings telling you, this usually means something is wrong or maybe compromised. On Firefox below 39.0.2 try to disable the internal DNS system.

- On Firefox Mobile (about:config): *network.dns.disablePrefetch* needs to be set to *true*.

On the OS level:

- Use 3rd Party Local DNS Servers/Resolvers, here.
- Apply Windows Tweak and Registry Hacks, here - on non servers max 4 hours is enough.
- Apply MacOS Tweaks, here.
- Configure Firewall as *Fail-safe* to prevent leaks, see here how
- Generally use secure alternatives + use a online browser check
- Use always the latest software to ensure possible bugs and security holes are fixed tools like sumo
- Use a secure, user/privacy friendly search engine like DuckDuckGo, Disconnect or Ixquick. Even better would be an decentralized search like YaCy, FAROO or any other based on P2P/...
- Verify no external addons/software/app leaks something (speaking about external metadata and protocol headers from going in .plaintxt out!)

On Tor:

- Read the *Advanced Tor usage* FAQ section, the two important ones are this and this

Sometimes these messages may be false alarms. To find out, you should run a packet sniffer on your network interface. The basic command to do this is `tcpdump -pni eth0 'port domain'`.

If you are using an VPN this also can "fix" the DNS problem, but sometimes even this isn't enough, especially on Android and OpenVPN, some older versions and provider still suffering from this issue, a workaround can be found over here.

Another possible problem is that you ISP mitm and manipulate the DNS traffic (mostly due censorship or to spoof)! There are only a few methods to bypass this:

- Use TOR + setup it (simply use the setup wizard if you don't know how)
- Use a SSH tunnel
- Choose an VPN which doesn't censorship
- Use JonDo (the proxy) [but browser is also good]
- Use DNSCrypt or httpsdnsd (HTTPSDNS daemon is already running if you use JonDo)
- For general implementation info about DNS Transport over TCP take a look at here

There are also several tips, tricks and guides directly with a lot of examples over the official Tor Wiki page, see here & here. Remember that the given tricks on this pages are optimized for TOR/I2P, so you may need to adjust some example configuration given from there.

DNS problems

❗ Since Android 6.1.x you only can change the DNS for tether device and nothing else, everything else get ignored, no matter what you set or which app you're use ❗

An easy method to look at opened or closed threads is to search via `is:issue is:open dns` / `is:issue is:closed dns` which shows the important threads (if the topic/thread content was correct labaled), alternative just click on the follow links (or copy/paste the issue number in the search e.g. `https://github.com/ukanth/afwall/issues/`.

Already reported DNS related topics:

- #377 DNS (port 53) is blocked for Wifi tethering

- #344 Wi-Fi-Hotspot not working while AFWall+ is enabled
- #326 Add mDNS support for local networks
- #318 Show host names in log view
- #257 USB/Bluetooth Tethering fails on CM11 mako
- #209 Bluetooth tethering borked as well
- #206 DNS Requests fail
- #178 Tethering
- #18 UDP 53 bypass because logging & whitelisting are enabled
- On AFWall+ in whitelisting mode root/DNS+DHCP options needs to be checked to allow DNS traffic (netd runs as root that why you also need root checked too)

AOSP based reported problems:

- #159406 Local DNS names are resolved to external IP
- #178654 DNS query for type A record is not sent to DNS IPV4 server
- #511506 Investigate horrible DNS performance on Android when running dualstack
- Android 5 broke tethering (DNS REFUSED)
- #183489 Consult dnsmasq when hotspot is enabled in order to resolve local hostnames
- #79504 DNS (local) resolution on Android Lollipop
- #177536 Android 5.0 VPN: DNS not being updated in system properties
- 152819 Android ignores DHCP option 26 (MTU)
- .. and others

Important: Please always use the search function here on AFWall's/AOSP issue tracker to search already known existent problems to avoid duplicate threads.

Resolver commands

Android 4.0.4

http://androidxref.com/4.0.4/xref/system/netd/CommandListener.cpp#778

- `ndc resolver setdefaultif <iface>`
- `ndc resolver setifdns <iface> <dns1> <dns2> ...`
- `ndc resolver flushdefaultif`
- `ndc resolver flushif <iface>`

Android 4.1.1

http://androidxref.com/4.1.1/xref/system/netd/CommandListener.cpp#803

- `ndc resolver setdefaultif <iface>`
- `ndc resolver setifdns <iface> <dns1> <dns2> ...`
- `ndc resolver flushdefaultif`
- `ndc resolver flushif <iface>`

Android 4.2_r1

http://androidxref.com/4.2_r1/xref/system/netd/CommandListener.cpp#873

- `ndc resolver setdefaultif <iface>`
- `ndc resolver setifdns <iface> <dns1> <dns2> ...`

- `ndc resolver flushdefaultif`
- `ndc resolver flushif <iface>`

Android 4.3_r2.1

http://androidxref.com/4.3_r2.1/xref/system/netd/CommandListener.cpp#770

- `ndc resolver setdefaultif <iface>`
- `ndc resolver setifdns <iface> <domains> <dns1> <dns2> ...`
- `ndc resolver flushdefaultif`
- `ndc resolver flushif <iface>`
- `ndc resolver setifaceforpid <iface> <pid>`
- `ndc resolver clearifaceforpid <pid>`

Android 4.4_r1

http://androidxref.com/4.4_r1/xref/system/netd/CommandListener.cpp#941

- `ndc resolver setdefaultif <iface>`
- `ndc resolver setifdns <iface> <domains> <dns1> <dns2> ...`
- `ndc resolver flushdefaultif`
- `ndc resolver flushif <iface>`
- `ndc resolver setifaceforpid <iface> <pid>`
- `ndc resolver clearifaceforpid <pid>`
- `ndc resolver setifaceforuid <iface> <l> <h>`
- `ndc resolver clearifaceforuid <l> <h>`
- `ndc resolver clearifacemapping`

Android 4.4.2_r1

http://androidxref.com/4.4.2_r1/xref/system/netd/CommandListener.cpp#942

- `ndc resolver setdefaultif <iface>`
- `ndc resolver setifdns <iface> <domains> <dns1> <dns2> ...`
- `ndc resolver flushdefaultif`
- `ndc resolver flushif <iface>`
- `ndc resolver setifaceforpid <iface> <pid>`
- `ndc resolver clearifaceforpid <pid>`
- `ndc resolver setifaceforuidrange <iface> <l> <h>`
- `ndc resolver clearifaceforuidrange <l> <h>`
- `ndc resolver clearifacemapping`

Android 4.4.3_r1.1

http://androidxref.com/4.4.3_r1.1/xref/system/netd/CommandListener.cpp#940

- `ndc resolver setdefaultif <iface>`
- `ndc resolver setifdns <iface> <domains> <dns1> <dns2> ...`
- `ndc resolver flushdefaultif`
- `ndc resolver flushif <iface>`
- `ndc resolver setifaceforpid <iface> <pid>`
- `ndc resolver clearifaceforpid <pid>`

- `ndc resolver setifaceforuidrange <iface> <l> <h>`
- `ndc resolver clearifaceforuidrange <if> <l> <h>`
- `ndc resolver clearifacemapping`

Android 4.4.4_r1

http://androidxref.com/4.4.4_r1/xref/system/netd/CommandListener.cpp#940

- `ndc resolver setdefaultif <iface>`
- `ndc resolver setifdns <iface> <domains> <dns1> <dns2> ...`
- `ndc resolver flushdefaultif`
- `ndc resolver flushif <iface>`
- `ndc resolver setifaceforpid <iface> <pid>`
- `ndc resolver clearifaceforpid <pid>`
- `ndc resolver setifaceforuid <iface> <l> <h>`
- `ndc resolver clearifaceforuid <if> <l> <h>`
- `ndc resolver clearifacemapping`

Android 5.0.0_r2

http://androidxref.com/5.0.0_r2/xref/system/netd/server/CommandListener.cpp#776

- `ndc resolver setnetdns <netId> <domains> <dns1> <dns2> ...`
- `ndc resolver flushnet <netId>`

Android 5.1.0_r1

http://androidxref.com/5.1.0_r1/xref/system/netd/server/CommandListener.cpp#791

- `ndc resolver setnetdns <netId> <domains> <dns1> <dns2> ...`
- `ndc resolver clearnetdns <netId>`
- `ndc resolver flushnet <netId>`

Android 6.0.0_r1

http://androidxref.com/6.0.0_r1/xref/system/netd/server/CommandListener.cpp#831

- `ndc resolver setnetdns <netId> <domains> <dns1> <dns2> ...`
- `ndc resolver clearnetdns <netId>`
- `ndc resolver flushnet <netId>`

Master tree (latest version 01.16.2016 checked)

https://android.googlesource.com/platform/system/netd/+/master/server/CommandListener.cpp#805

- `ndc resolver setnetdns <netId> <domains> <dns1> <dns2> ...`
- `ndc resolver clearnetdns <netId>`
- `ndc resolver flushnet <netId>`

Changing default DNS

On Android < 4.3 we can use the command `getprop | grep dns` to know all the DNS properties being used. This command requires BusyBox!

'rmnet0' is the interface name for the 3G connection. net.rmnet0.dns1 and net.rmnet0.dns2 are the properties to be changed to point to OpenDNS server (the settings are still present in CM/AOSP code). Since, these properties are changed after the connection is established, net.dns1 and net.dns2 also have to be changed. Execute these commands as root user: `setprop net.rmnet0.dns1 208.67.222.222.` `setprop net.rmnet0.dns2 208.67.220.220`. `setprop net.dns1 208.67.222.222`. `setprop net.dns2 208.67.220.220`.

Remember, the settings will be applicable only for the current session! You will have to repeat it when you are re-connecting to the network.

Android system chooses the DNS servers using the script located at */system/etc/dhcpcd/dhcpcd-hooks/20-dns.conf*

```
20-dns.conf

To change the DNS servers, use the command: 'setprop property name'

setprop net.dns1=208.67.222.222
setprop net.dns2=208.67.220.220
setprop net.eth0.dns1=208.67.222.222
setprop net.eth0.dns2=208.67.220.220
setprop net.rmnet0.dns1=208.67.222.222
setprop net.rmnet0.dns2=208.67.220.220
setprop dhcp.tiwlan0.dns1=208.67.222.222
setprop dhcp.tiwlan0.dns2=208.67.220.220
setprop net.ppp0.dns1=208.67.222.222
setprop net.ppp0.dns2=208.67.220.220
setprop net.pdpbr1.dns1=208.67.222.222
setprop net.pdpbr1.dns2=208.67.220.220
```

Or via init.d script (won't survive connectivity changes):

```
#!/system/bin/sh
setprop net.dns1 208.67.222.222
setprop net.dns2 208.67.220.220

# Optional
setprop dhcp.tiwlan0.dns1 208.67.222.222
setprop dhcp.tiwlan0.dns2 208.67.220.220
setprop net.ppp0.dns1 208.67.222.222
setprop net.ppp0.dns2 208.67.220.220
setprop net.rmnet0.dns1 208.67.222.222
setprop net.rmnet0.dns2 208.67.220.220
setprop net.pdpbr1.dns1 208.67.222.222
setprop net.pdpbr1.dns2 208.67.220.220
```

To check against it (on e.g. wlan) we use: `tcpdump -ns0 -i wlan0 'port 53'`

DNS check tool is a secure proof if DNS is working or not, alternative you can use nslookup via command line. Please remember that there are some problems generally with the DNS security protocol, there are several known attacks, like DOS, Cache poisoning, ghost domain names & others. For more information take a look over here

If there is no setprop you can write the values before the `unset_dns_props()` begins. Here is an example 20-dns.conf file. You can get the dns information by using the `getprop | grep dns` command but this will only work for Android < 4.3 devices.

The `getprop` or `setprop` method **does not work on Android versions >4.4+** anymore. Those values, when changed, get simply ignored by the *netd* daemon. It's necessary to communicate directly to the daemon via the `/dev/socket/netd socket`. In Android it's now present a tool called `ndc` which does exactly this job.

On 4.3 or 4.4 KitKat (#su):

```
ndc resolver setifdns eth0 "" 208.67.222.222 208.67.220.220 192.168.1.1
ndc resolver setdefaultif eth0
```

Or via AFWall+ custom script:

```
$IPTABLES -t nat -A OUTPUT -p tcp --dport 53 -j DNAT --to-destination 208.67.222.222:53 || t
$IPTABLES -t nat -A OUTPUT -p udp --dport 53 -j DNAT --to-destination 208.67.222.222:53 || t
```

Or via init.d:

```
#!/system/bin/sh
# File without file extension

#IP6TABLES=/system/bin/ip6tables
IPTABLES=/system/bin/iptables

# Maybe need to change $IPTABLES to iptables (if there are troubles applying them)
$IPTABLES -t nat -A OUTPUT -p tcp --dport 53 -j DNAT --to-destination 208.67.222.222:53
$IPTABLES -t nat -A OUTPUT -p udp --dport 53 -j DNAT --to-destination 208.67.222.222:53
```

If you still like external apps, you should take a look at Override DNS [tested, working on 4.4.4/5.x] which does more or less the same. That may solve some problems on Android 4.4/Lollipop/M but there is no guarantee, some ROMs may handle it different.

Commands to check if DNS is working

The following may are necessary to indicate if all is working (dhcp/nameserver/dnsmasq,...), may needs to be changed for your interfaces you want to check: cellular, tethered, ...

- Grep the current DNS resolver/settings, reads them via: `adb shell getprop | grep dns`
- The actual DNS servers used are the ones listed in the output of: `adb shell dumpsys connectivity` or `adb shell dumpsys connectivity | grep DnsAddresses`
- ~~Via nslookup~~ `nslookup google.com`
- See the current dhcp info `cat /system/etc/dhcpcd/dhcpcd.conf`
- List the tethered dns configuration `adb logcat | egrep '(TetherController|dnsmasq)'`
- Check routing via `ip ru` + `ip route ls`
- On Bluetooth tethering `ip addr show bt-pan (optional)`
- TCPDump check state and export them: `tcpdump -ni any -s0 -U -w /sdcard/icmp4.pcap icmp4` or `adb shell /data/tcpdump -ni wlan0 "icmp6 or port 67 or port 68"`

Browser

The normal mobile browser (even stock yes) working out-of-the-box without need any manual adjustment to work with the default DNS.

But some browsers are hardcode them (so that it isn't possible to overriding them) and on some systems it may needs configurations changes to get theme proper working e.g. if you changed the DNS system widely.

Working without any manual adjustments:

- Dolphin
- Naked Browser
- Firefox (supports DANE via addon)
- TOR (Orweb), see this article ~~deprecated~~
- ....

Needs changes in the settings:

- Chrome -> Clean DNS cache .... (doesn't support DANE (official), addon is available)
- Firefox -> for Tor/Orbot .... (disable internal via network.dnsCacheExpiration;0)
- ...

Apps to change the default DNS

The following apps are success tested on all systems to work:

- OverrideDNS (paid)
- DNS Server (free)
- ... add more, remember they must work on all systems (even 5.1.1/M)

Useful links

- Latest AOSP netd version (source code) | Android.GoogleSource.com
- List common DNS issues (read-only) | Google Issue Tracker
- DNS (local) resolution on Android Lollipop #79504 | Android Open Source Project - Issue Tracker
- Android 5 broke tethering (DNS REFUSED) #82545 | Android Open Source Project - Issue Tracker
- NsdManager | Android Developers.com
- Microsoft KB99686 – Enabling IP Routing | Support.Microsoft.com
- Net_DNS2 v1.2.5 DANE TLSA Support | netdns2.com
- DNSSEC-Tools | dnssec-tools.org
- Unbound | unbound.net (name server)
- OpenNIC nameserver list + nearest nameserver
- Extension Mechanisms for DNS (EDNS0) | Wikipedia
- DNS Reply Size Test Server | DNS-OARC.net
- Unofficial Android dnsmasq FAQ | GitHub
- Official dnsmasq binary and docs | thekelleys
- Reverse Whois Lookup | ViewDNS.info
- Google Caching Overview
- Linux workstation security checklist | GitHub (lfit)

Todo:

- ~~I won't explain router side configuration, just use OpenWRT or any other mod (documented very well)~~
- ~~Complete the missing parts~~
- ~~Android 5.x doesn't support WPAD,.... but Android 6 does~~
- ~~Link all DNS related stuff in this thread (e.g. from the FAQ)~~
- ~~Add several workarounds since newer systems ignoring the etc/resolver.conf or dhcpcd/dhcpcd-hooks/20-dns.conf files~~, explained with given links

- ~~Add AFWall+ workarounds via custom scripts or separate tips~~
- Possible explain why Android 4.4.+/5+ wants to call the RIL with RIL_REQUEST_SETUP_DATA_CALL
- Explain the broken MTU (saw this on so many roms) and list this bug (see ConnectivityService.cpp), since Android 5.1.x always seems to use 1500 regardless of what value has dhcp daemon set in option 26 (call interface_mtu).
- How can I gather DNS (A/AAA/...) requests? -> must be re-written (high-prio)
- Add example output how it should looks like, really? (low-prio)
- Add traceroute, whois, and dig commands to work with (low-prio)
- On Android 5.x DNS problems try to disable IPv6, since Android 5.0.1/5.x doesn't like DHCPv6
- Is there a decentralized search for Android as app available (similar to yacy)? Mail me if there is one. (low-prio) + add them in the article
- Android auto config compatibility (not yet), e.g. test network fails
- DNSCrypt, dnsmasq & resolv.conf should be mentioned + example configuration should be given (I need confirmation if that works or not)
- nslookup shouldn't be used (outdated)