
BOOTSTRAPPING TIME ON OPENBSD

Otto Moerbeek otto@openbsd.org

OPENBSD

- **BSD derivatie, focus on security**
 - **Many techniques, e.g. privilege separated daemons**
 - **Sane defaults**
 - **If a service is enabled out of the box, there are extra requirements**
 - **Useful for a very large fraction of users**
 - **Even more focus on security, including architecture and implementation**
-

OPENBSD INITIAL TIME AS IT USED TO BE

- **Get time from (battery backed) Real Time Clock**
 - **If that fails: read time from root filesystem last mounted field**
 - **Consequence: initial time is either mostly correct or behind**
 - **When OpenNTPd starts, set time based on NTP but only if -s is used, which is not default**
-

GOALS: A BETTER TIME BOOTSTRAP

- **Do not fully trust NTP replies necessarily**
 - **Get correct time on boot with a high level of trust**
 - **Do not *rely* on battery backed up RTC being available**
 - **Think cheap boards or old machines where battery ran out**
 - **Time based validations complicate matters, but make it work with a DNSSEC enabled resolver running on the same machine**
-

NTP PROTOCOL

- **Quite old, RFC 958 from 1985, latest RFC 5905 from 2010 (plus some more recent followup RFCs)**
 - **Follows design principles which are also found in DNS**
 - **Can be secured with shared keys**
 - **RFC 8915 defines NTS, Network Time Security that includes a key establishment protocol**
 - **Simple variant, RFC 4330, concerned with client role. This is mostly what OpenBSD's ntpd does**
-

OPENBSD'S IMPLEMENTATION

- **Privilege separated**
 - **Process handling network I/O**
 - **Process adjusting time**
 - **Process doing (asynchronous) DNS requests**
 - **Processes handling constraints**
 - **All with minimal permissions (pledged) and minimal access to file system**
-

SAFETY MEASURES

- **Initially no cryptographic measures: shared keys not ideal and NTS complex, not widely used**
 - **Basic spoof protection: expect the server to answer with a cookie we sent earlier**
 - **Re-use (misuse?) a field for that**
-

TRANSMIT TIME AS COOKIE

```
/*  
 * Send out a random 64-bit number as our transmit time. The NTP  
 * server will copy said number into the originate field on the  
 * response that it sends us. This is totally legal per the SNTP spec.  
 *  
 * The impact of this is two fold: we no longer send out the current  
 * system time for the world to see (which may aid an attacker), and  
 * it gives us a (not very secure) way of knowing that we're not  
 * getting spoofed by an attacker that can't capture our traffic  
 * but can spoof packets from the NTP server we're communicating with.  
 *  
 * Save the real transmit timestamp locally.  
 */
```

```
p->query.msg.xmttime.int_partl = arc4random();  
p->query.msg.xmttime.fractionl = arc4random();
```

BUILDING TRUST

- **Actually outside of scope of SNTP**
 - **“Full” NTP peer selection is quite complex, OpenNTPd uses a simple approach**
 - **Poll several servers**
 - **Filter peers that are unreliable in replying or replied with bad cookie**
 - **Select “median” time**
-

CONSTRAINTS

- **Extra measure**
 - **Independent of NTP protocol: different protocol, different code, different time source**
 - **Ask a few HTTPS servers for time**
 - **It's already in the reply header!**
 - **Low resolution, but used to filter out bad NTP replies**
-

HTTPS CERTIFICATE CHECK

- **Time dependent!**
 - **Use time in reply header to validate certificate time validity**
 - **This is a bit weird, requires a certificate valid at the time the server is telling us**
 - **Talking to multiple widely used https servers strengthens this check at least a bit**
 - **More on this later**
-

DNS DEPENDENCY

- **NTP servers and constraint sources specified by IP or name**
 - **So we have to resolve names, typically using DNS**
 - **DNS resolver on other host: assume it has the right time for DNSSEC validation**
 - **Hardest case: resolver on same host with DNSSEC validation enabled: bootstrap issue**
-

DNSSEC

- **DNSSEC signatures have a validity period**
 - **DNS resolver must check these**
 - **Luckily, a client can signal to skip the DNSSEC validation**
 - **CD flag: Check Disabled**
 - **No API for that! :-)**
-

ADD API

```
=====
RCS file: /cvs/src/include/resolv.h,v
retrieving revision 1.21
retrieving revision 1.22
diff -u -r1.21 -r1.22
--- src/include/resolv.h 2016/09/12 19:35:31 1.21
+++ src/include/resolv.h 2019/01/14 06:23:06 1.22
@@ -1,4 +1,4 @@
-/* $OpenBSD: resolv.h,v 1.21 2016/09/12 19:35:31 guenther Exp $ */
+/* $OpenBSD: resolv.h,v 1.22 2019/01/14 06:23:06 otto Exp $ */

/*
 * Copyright (C) 1995, 1996, 1997, and 1998 WIDE Project.
@@ -190,6 +190,7 @@
#define RES_USE_EDNS0 0x40000000 /* use EDNS0 */
/* DNSSEC extensions: use higher bit to avoid conflict with ISC use */
#define RES_USE_DNSSEC 0x20000000 /* use DNSSEC using OK bit in OPT */
+#define RES_USE_CD 0x10000000 /* set Checking Disabled flag */

#define RES_DEFAULT (RES_RECURSE | RES_DEFNAMES | RES_DNSRCH)
```

USE CD BIT WHEN RELEVANT

```
int
host_dns(const char *s, int synced, struct ntp_addr **hn)
{
    int error, save_opts;

    log_debug("trying to resolve %s", s);
    error = host_dns1(s, hn, 0);
    if (!synced && error <= 0) {
        log_debug("no luck, trying to resolve %s without checking", s);
        save_opts = _res.options;
        _res.options |= RES_USE_CD;
        error = host_dns1(s, hn, 1);
        _res.options = save_opts;
    }
    log_debug("resolve %s done: %d", s, error);
    return error;
}
```

OPENBSD INITIAL TIME REVAMPED

- **Get time from RTC. If that fails: read time from root filesystem last mounted field**
 - **Consequence: initial time is either mostly correct or behind**
 - **When OpenNTPd starts, it gets constraints and will set (bump) time based on NTP data if**
 - **Time shift is moving forward compared to initial time**
 - **Constraints are set and met (or *trusted* NTP peers are configured)**
 - **Time shift is “large” (> 1 minute)**
 - **Otherwise, and after initial set, do a gradual adjust, speeding the clock up or slowing it down**
-

ONE MORE TIME

- **When synced: re-resolve and refetch constraints**
 - **With no Checking Disabled DNS fallback**
 - **With standard check of certificate chain**
-

STATE SINCE A FEW RELEASES

- **ntpd enabled by default**
- **you can be pretty sure that time is set based on trusted sources if you have net**
- **default config uses assorted NTP servers and assorted constraints sources**

NTPD DEFAULT ON

```
# $OpenBSD: ntpd.conf,v 1.16 2019/11/06 19:04:12 deraadt Exp $  
#
```

```
# See ntpd.conf(5) and /etc/examples/ntpd.conf
```

```
servers pool.ntp.org  
server time.cloudflare.com  
sensor *
```

```
constraint from "9.9.9.9"  
constraint from "2620:fe::fe"  
constraints from "www.google.com"
```

```
# quad9 v4 without DNS  
# quad9 v6 without DNS  
# intentionally not 8.8.8.8
```

QUESTIONS?

- **Thanks to: Henning Brauer, Reyk Floeter, Alexander Guy and others**
 - **You can reach me on otto@openbsd.org**
 - **Mastodon: [@otto@bsd.network](https://bsd.network/@otto)**
-