Как собрать chatbot'а для Tox

Shroom [OmskLUG]*

25 января 2015 г.

В этом документе собраны некоторые подводные камни, обнаруженные лично автором при сборке чатбота для tox на Debian GNU/Linux 7 (Wheezy) для архитектуры amd64. Приведены способы обхода обнаруженных ошибок.

Подготовка

Для начала нам нужно установить некоторое количество пакетов с библиотеками и заголовочными файлами и вообще со средствами разработки (если таковых в системе ещё нет). Начнём с инструментария разработчика:

```
apt-get install build-essential libtool autotools-dev \
automake checkinstall check git yasm
```

Для удовлетворения сборочных зависимостей и для разных дополнительных плюшек типа аудио/видео, возможности запуска bootstrap-демона и хостинга ноды, ncurses-интерфейса поставим ещё несколько пакетов:

```
apt-get install libopus-dev libvpx-dev pkg-config \
libconfig-dev ncurses-dev
```

В принципе, эти рекомендации указаны на странице, посвящённой сборке **toxcore** (`https://github.com/irungentoo/toxcore/blob/master/INSTALL.`

`md`). А дальше пойдут некоторые особенности... Кстати, я не рекомендую проходить данный квест, одновременно читая этот текст, поскольку стиль изложения материала довольно свободный и допускает возврат на несколько шагов назад и повтор действий с возможным улучшением стратегии. В общем, информация дана в чисто ознакомительных целях и ни в коей мере не является полноценным HowTo и, тем более, пошаговой инструкцией.

Итак, начнём, так сказать, с базовых вещей. Для шифрования нужна библиотека **NaCl** (старая и плохо портируемая, но быстрая) или **sodium** (новая, хорошо переносимая, с совместимым API, но несоколько медленнее). При этом существуют косяки как с первой, так и со второй библиотекой. Косяк с первой: нет объектников *randombytes.o* и *cpucycles.o*, которые требуются для сборки **toxcore**, но которые не являются частью **NaCl** [http://comments.gmane.org/gmane.network.curvecp/81]. Косяк со второй — её тупо нет в репах для Wheezy (появилась только в Jessie), поэтому нужно сливать сырцы для Jessie и собирать[1]. Соответственно, на старых системах (или если хочется «скорости»):

```
apt-get install libnacl-dev
```

А вообще **NaCl** лучше собирать из исходников, чтобы было откуда выцарапать *randombytes.[c|o]*.

На новых системах (а также на Ubuntu, Mint, Arch, Gentoo и прочих, где всё новьё можно найти в стандартных репах/ppa/портах и пр.):

```
apt-get install libsodium-dev libsodium13
```

На старых же системах, где **libsodium** придётся брать в виде исходиков для Jessie, нужно дополнительно поставить **pkg-config** и **dh-autoreconf**, которые указаны в зависимостях для сборки. Впрочем, первый из них мыуже поставили в самом начале, поэтому остаётся только второй[2].

[1] Естественно, для этого нужно добавить соответствующий репозиторий в `/etc/apt/sources.list`, настроить pinning, чтобы ненароком не обновить систему и т.п. Хотя можно, конечно, скачать нужный пакет прямо с сайта Debian.

[2] Я лично ограничился сборкой с использованием **NaCl**, поэтому не могу сказать, какие глюки можно словить с **sodium**. Однако, смею предположить, что как минимум отсутствие *randombytes.o* породит похожие эффекты.

Сборка NaCl

Предположим, мы собрались использовать **NaCl**.

```
apt-get source nacl-tools
```

Из этого получатся пакеты для «-tools» и для «-dev». Здесь в зависимостях для сборки присутствует **docbook-to-man**, поэтому ставим и его тоже.

Сборка пакета: ничего страшного. Как обычно,

```
dpkg-buildpackage -b -us -uc
```

Но перед этим нужно в файлике *debian/rules* закомментировать строчку вида

```
rm -f $(CURDIR)/build/$(SHORTHOSTNAME)/lib/*.o
```

чтобы иметь возможность упаковать вместе с основной библиотекой и нужные нам объектники (ну или просто чтобы найти их после сборки). На самом деле по зрелом размышлении можно понять, что **сборка пакета** по сути **не нужна**. Можно ограничиться готовым пакетом из репы. Однако, при этом в любом случае нужно в верхнем каталоге дерева исходников **libnacl** выполнить магический скрипт `./do`, который, собственно, и занимается сборкой либы. И уже после этого можно выдернуть и куда-нибудь сложить *cpucycles.o* и *randombytes.o*. Этот вариант будет даже более правильным с точки зрения поддержания чистоты пакетов.

Сборка ToxCore

Со стандартными либами покончено. можно сливать с гитхаба исподники **toxcore**:

```
git clone git://github.com/irungentoo/toxcore.git
```

После этого нужно перейти в каталог *toxcore* и фигачить далее по тексту руководства [`https://github.com/irungentoo/toxcore/blob/master/INSTALL.md`]. Почти. Потому что, как обычно, появятся небольшие нюансы. Начнётся всё просто замечательно:

```
autoreconf -i
```

отработает без ошибок, поскольку здесь ломаться в принципе нечему. Зато следующий шаг

```
./configure --enable-nacl --enable-log --enable-ntox \
--enable-daemon --disable-testing
```

(в данном случае вся фишка в `--enable-nacl`; всё остальное — по желанию) закончится ошибкой. Нам скажут, что либа **NaCl** якобы не найдена, и предложат явно указать путь к ней. Хорошо, сделаем то, что просят, и даже больше, предполагая, что те самые *cpucycles.o* и *randombytes.o* лежат в одном из каталогов, известных линкеру (например, в `/usr/lib` рядом с *libnacl.a*):

```
./configure --enable-nacl --enable-log --enable-ntox \
--enable-daemon --disable-testing --with-nacl-libs=/usr/lib \
--with-nacl-headers=/usr/include/nacl
```

После того, как конфигуратор удовлетворится этой подсказкой, можно делать

```
make
```

Однако, сборка тоже не пройдёт с первого раза. Это, вероятно, связано как раз с ошибочным мнением разрабов **toxcore** о том, что *randombytes.o* принадлежит **libnacl**. Но это метафизика, а нам нужно что-нибудь уже собрать. Короче, процесс останавливается на линковке *tox-bootstrapd* с воплями о том, что в нескольких модулях найдены *«undefined reference to 'randombytes'»*. Хорошо, предоставим костыль в виде

```
make LIBS=''/usr/lib/randombytes.o''
```

после чего чудесным образом[3] дособерётся демон *tox-bootstrapd*[4]. После этого параноики могут проверить качество сборки, запустив

```
make check
```

(кстати, все тесты должны завершиться успешно), а все остальные могут сразу же приступить к сборке deb-пакета (ну или любого другого пакета на выбор) и его установке. Например, так:

```
sudo checkinstall -D make install
```

По умолчанию либы и бинарники будут установлены в `/usr/local`, поскольку мы не меняли эти пути в параметрах конфигуратора. Собственно, никто не мешает при вызове `./configure` добавить параметр `--prefix=/usr`. Однако, это не является критичным для сборки бота.

[3] Ну то есть сразу после ошибки запускаем «костыль» — и всё. Не нужно ничего нигде чистить, менять, копаться в файлах и др.пр.

[4] Естественно, если нет необходимости в этом функционале (а её в подавляющем большинстве случаев нет) и не был указан параметр `--enable-daemon` при вызове `./configure`, то всё соберётся без проблем.

Сборка ToxBot

Теперь, собственно, то, ради чего затевался весь этот сыр-бор. Сборка чат-бота.

Перед тем, как что-то собирать, это что-то нужно стянуть с гитхаба:
`git clone git://github.com/JFreegman/ToxBot.git`

Далее всё тривиально. Почти, опять же... Если взглянуть на руководство по установке [`https://github.com/JFreegman/ToxBot`], может возникнуть ложное чувство уверенности в своих силах и полного контроля над происходящим. В самом-то деле, что уж, так сложно единственную команду `make` запустить что ли?! Однако, в реальности всё не так, как на самом деле... Наверняка сборка загнётся опять же на этапе линковки, и линкер будет орать, что нашёл *«undefined reference to 'clock_gettime'»* где-то в недрах **toxcore**. В принципе, он прав, потому что эта функция находится в библиотеке **librt**, а про неё линкеру никто ничего не рассказал. Поэтому придётся лезть в *Makefile* и менять строчку
`LDFLAGS = $(shell pkg-config --libs $(LIBS))`
на почти такую же, только с упоминанием нужной библиотеки:
`LDFLAGS = -lrt $(shell pkg-config --libs $(LIBS))`

После этого всё должно пройти без ошибок. Результат сборки — свежий бинарник *toxbot*, который не требует настройки и заводится с полпинка. Дальнейшие вопросы вроде «что делать, чтобы он признал хозяина?» и «как пробиться через заботливо закрытый файрвол?» автор статьи оставляет в качестве упражнения для читателя. Поверьте, это абсолютно несложно и даже описано в документации.

Спасибо за внимание!