Open Source Porting [Tor's core]

deepankar-tyagi

Organization: Portable Native Client

Abstract: Currently Tor browser bundle is one of the most used approach to bypass censorship, be anonymous etc, which is not a seamless user experience. In this project the tor's core will be ported to nacl platform, which can allow the end user to gain the benefits of tor network in much more seamless and dynamic manner.

Table of Contents

Section One

01. To be Ported

Tor's Core [https://www.torproject.org]

---

02. AIM

To bypass censorship in dynamic manner.

What do I mean by dynamic ?

Not all the traffic goes through the tor proxy ,rather the app/extension figures out when the user comes across the blocked page and that url is re-routed through the tor proxy, remote dns is of primary importance here.This also holds if some (embedded)elements in the page is blocked and not the full url.

However it is possible to route entire traffic through tor.

---

03. Motivation

- The **initial idea** came after hearing a talk by tor creators, during the presentation they point out that barring some totalitarian regimes, the people who impose censorship do not go after the the people who seek "prohibited content" [they do go after people who create "prohibited content"], they just want to impose their supremacy.It just so happens that in most of the scenarios something famous such as facebook and google services are blocked for everyone because a particular person posted something bad.
- **the most recent motivation (and guidance)** came from Mr Bradley Nelson. After I tinkered with existing naclport [unzip](section 2,part 2) to get more clearer understanding of project.I posted that I was afraid of overpromising if I went ahead with this project, it was only after he said that this goal is achievable[Discussion thread], I went ahead with renewed energy , and I am happy to report that code and dependency analysis (First Report) of tor is complete,the result of the same is posted at the end of this document (section 2 , part 03).

---

04. My understanding of this project.

As per my understanding, this project involves following main task:

- Removing any kind of assembly code,
- Removing any dynamic linking of libraries and porting any dependent library if they are not ported already.
- Modifying and adapting their build system such as makefile etc according to requirement of nacl project in such a manner that it is easily usable and extensible.
- Removing/replacing any unsupported methods/functions from the code or writing wrappers for those methods/functions.
- It might also require fixing limitations in nacl_io's POSIX emulation and or fixing bugs in it.(Thank You Mr. Bradley for pointing this out)

---

05. What I will contribute by the end of this project?

The main focus will be to port the tor's core and libevent, making sure they both run successfully and in useful manner.

---

06. Time Schedule [How much time I plan to devote, any other plans, commitments etc]

For some duration of gsoc coding period I will have other commitments.The time schedule will be as follows

- May 15-28 : semester exams (so will be unable to work on may 25-28).
- May 29 - August 03 : Most active period, with avg 10 hour of work including weekends.
- August 03 - August 28 : My seventh semester will start on August 03, however this semester will be much lighter as compared to current one, I am sure I can manage to squeeze in 6 hours on weekdays, and 10-14 hours on weekends.

07. Deliverables/Milestones/Timeline

Current - August 15

- Turn skill level in following from "familiar" to "proficient" > GNU build system, Bash scripting, vim.
- Practice more with pthread, as I perceive I will be dealing with them quite a lot when taking care of fork(), as well as practice on sockets, mutexes etc also.
- Keep analyzing the code of tor and libevent further.

August 29 - July 03 [Midterm]

- Successful completion of modification of build system, build scripts etc, of both tor and libevent (and any further dependency which I might encounter in the process), Ideally the build must complete without any hassles by now.
- Please note, the compiled binaries may not be much useful at this point, that will be handled in later half.

July 03 - August 28

By this time tor binaries must be working in useful manner.

In the most basic implementation :

1. User downloads the pre-compiled tor binary in nacl DevEnv [curl]
2. Runs it in bash [./tor]
3. Installs extension like this : https://chrome.google.com/webstore/detail/proxy-switchyomega/padekgcemlokbadohgkifijomclgjgif
4. Configures it(the extension) to use socks proxy, with host : localhost && port : 9050

Extended work

In the end if time permits ,focus will be to package the tor binaries in the more user friendly manner.

[*Big thanks to Mr Bradley Nelson for clearing the doubts and suggesting the ways to achieve what I intended.*]

- **First idea** was to make the chrome "app" and give the user interface where he/she can enter the url and it will display the requested webpage embedded ,(since in the context of app we have tcp/ip and other stuff exposed, it really shouldn't be a problem) .But **this idea is not a good one**, because it forces the user to leave the main tab where user intended to browse the page initially, which completely defeats the purpose of seamless integration.
- **Second idea** is to use the Chrome proxy api. Socks proxy (with remoteDNS) will be running (in background, the chrome app) and via the extension we can configure the client to listen to the designated port .SInce extensions aren't allowed to use the sockets api, I would have to **provide app and extension separately**. where chrome app will run the run the tor binary and the extension will used to reroute the traffic dynamically using Chrome::Proxy API.Chrome messaging can be used to tightly couple the two making sure that both app and extension are installed in the browser.
- **Third and preferred idea** is **only provide tor binaries packaged as Chrome app** and and leave it to the user to configure their proxy settings, such as using any existing extension which allows to

configure proxy.

---

08. Preferred means to submit progress report and suggested frequency of the same.

- I plan to write a **blog** (with max frequency of per week and minimum frequency of per 2 weeks) and send the synopsis of same blog post to mentors if required.
- **Regarding questions** I plan to post them to Native client Discuss group, however I think it will be better If I can also have the option to contact the mentor directly.

---

09. Academic and Other details.

University : NIIT University, Rajasthan, India

Degree : Bachelor of Technology [B.Tech]

Major : Computer Science and Engineering.

For any more details about my previous work experience, academic details etc please visit : https://codebuff.net/resume

Other Details :

Linkedin : https://www.linkedin.com/in/codebuff

Github : https://github.com/codebuff

Section Two

---

01. Previous related experience.

[for details, videos , screenshots please visit the link provided]

- Cross compilation, analyzing and modifying the existing codebase, dependency analysis, porting etc.
  - Experiments with Android Kernel : https://codebuff.net/experiments/details/android_kernel
- Computer Networks, sockets etc :
  - DIrect Transfer (java based) : https://codebuff.net/past_projects/details/direct_transfer
- Chrome Extension [for extended work] :
  - Motify : https://codebuff.net/past_projects/details/motify

Info about all other projects can be found here : https://codebuff.net/

---

02. What I have built / tinkered with till now ? (in order to understand porting process better).

In this (sort of) of nacl port I utilized already ported unzip as base (modified the main(),makefile, buildscript)

such that first argument has to be a pseudo pass ie -gsoc-dt , if the the user provides correct argument then argc, argv are rearranged and unzip is called in normal manner

You can see the diff here.

Primary goal was to get more familiar with the environment.

NACL SDK

1. **Download** zip file containing build script and package info from here.
2. **Extract** this zip into "naclports/src/ports" directory of your naclports
3. From root directory of naclport type this, **make gsoc-dt** (I compiled it against pepper_canary and on linux-64 bit)
4. If everything goes fine then you should see a binary in publish directory "naclport/src/out/publish/gsoc-dt" with name of this kind gsoc-dt_{NACL-ARCH} (in my case it is gsoc-dt_x86_64.nexe)

Testing on nacl-dev-env extension:

You can see the screenshots here.

1. **Create a directory** (lets say work/gsoc/demo)
2. In that directory **download** the compiled binary (currently only linux 64 bit version is available) by typing this command
3. **curl -k https://codebuff.net/gsoc/nacl/dev-env/gsoc-dt-compiled.zip -o gsoc-dt.zip** (without -k curl refuses to download)
4. Unzip the file > **unzip ./gsoc-dt.zip**
5. First lets try giving wrong or no (pseudo) pass
6. **./gsoc*.nexe or ./gsoc*.nexe -tysg** (it will inform you accordingly).
7. Next lets try correct psuedo pass ie -gsoc-dt
8. **./gsoc*.nexe -gsoc-dt** (in this case unzip message will displayed asking for a file)
9. **./gsoc*.nexe -gsoc-dt ./demo.zip** (demo.zip was in the downloaded zip file which we just extracted)

---

03. Code and build system analysis of Tor, A Report

Main and unported dependency of Tor is libevent, other details are as follows.

Syscalls count

Calls	Tor	Libevent
Fork()	**6**	**5**
exec()	**0**	**1**

During a typical build of tor on a linux system, following are queried :

working C99 mid-block declaration syntax

working C99 designated initializers

library containing pow... -lm

library containing pthread_create... -lpthread

sha1sum

openssl

grep that handles long lines and -e

libevent

zlib

python

ranlib

sed

gawk

flexible array members

Header files

- sys/types.h,sys/stat.h,sys/fcntl.h,sys/fcntl.h,sys/fcntl.h,sys/stat.h,sys/time.h,sys/time.h,sys/types.h,sys/eventfd.h,sys/file.h,sys/ioctl.h,sys/limits.h,sys/mman.h,sys/param.h,sys/prctl.h,sys/resource.h,sys/select.h,sys/socket.h,sys/sysctl.h,sys/syslimits.h,sys/un.h,sys/wait.h,sys/param.h,sys/ucontext.h
- stdlib.h,string.h,memory.h,strings.h,inttypes.h,stdint.h,unistd.h,libscrypt.h,assert.h,errno.h,fcntl.h,signal.h,string.h,unistd.h,arpa/inet.h,execinfo.h,grp.h,ifaddrs.h,inttypes.h,limits.h,malloc.h,malloc_np.h,netdb.h,pwd.h,stdint.h,syslog.h,utime.h
- netinet/in.h,netinet/in6.h
- machine/limits.h
- net/if.h
- linux/types.h,linux/if.h,linux/netfilter_ipv4.h,linux/netfilter_ipv6/ip6_tables.h

Whether following flags are allowed

Compiler flags

-g

-fstack-protector-all

-Wstack-protector

-fwrapv

--param ssp-buffer-size=1

-fPIE

-fomit-frame-pointer

-fasynchronous-unwind-tables

Linker flags

-pie

-z relro -z now

-rdynamic

For libevent the requirement were more or less the same, with few differences in header files requirement.