GTK+ Frontend and Client Mode Improvements for arm

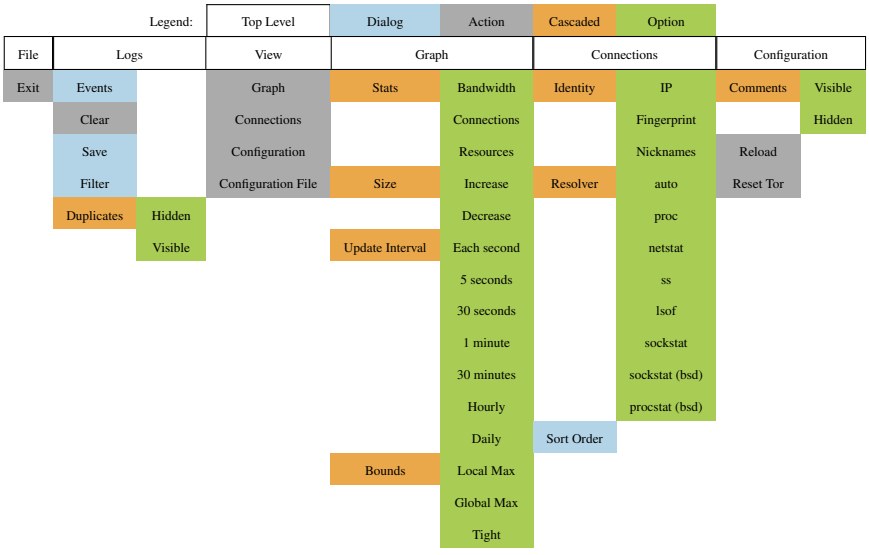
Kamran Riaz Khan
<krkhan@inspirated.com>

April 8, 2011

What project would you like to work on? Use our ideas lists as a starting point or make up your own idea. Your proposal should include high-level descriptions of what you're going to do, with more details about the parts you expect to be tricky. Your proposal should also try to break down the project into tasks of a fairly fine granularity, and convince us you have a plan for finishing it.

In a nutshell I want to increase arm's appeal for general public. In order to achieve this I plan on a two pronged approach:

- Use menus for curses interface. The following hierarchy can be used to make all arm features accessible:



- Implement a windowed interface in GTK+.
  - The current curses based UI can be converted into something similar to this:

(Click on thumbnails for larger versions.)

  - Care shall be taken to separate interface from backend logic as much as possible so that it would be easy to port features specific to GTK+ back to curses and vice versa. Following methods can be used to port the panels:
    * Graphs can be implemented via Cairo drawing surfaces. Event listeners would be registered for `TorCtl`'s `PostEventListener`:
      · Bandwidth can be graphed by listening to `bandwidth_event()`s.
      · Connections can be graphed by processing `getConnections()` on a tor resolver.
      · Resources can be graphed by processing a resource tracker returned by `sysTools`'s `getResourceTracker()`.
    * Connections can be implemented by filling TreeViews with appropriate IP/fingerprints in circut change event handlers.
    * Configuration options can be documented using `torConfig`'s `getConfigSummary()` and `getConfigDescription`.
  - The interface design would need to go through few iterations before we settle on something that works according to community feedback.

- Implement client mode use cases:
  - Display more information in header about Tor startup information like HTTP/SSL/Socks proxies, DNS listener and their ports. For example, Polipo configuration can be scanned to determine the port its listening on which can be subsequently scanned along with the default Polipo port to determine if it's running. Also, the `torTools`' `Controller` shall be used to fetch Tor configuration parameters such as the DNS listening port. This shall provide the user with a nice overview of how applications are supposed to connect to Tor and whether or not some misconfiguration is blocking connections on the ports Tor or relevant proxies are listening on.
  - Allow users to request new identity. The `NEWNYM` signal shall be sent to `Connection` in `TorCtl` for the request. Since Tor ignores frequent `NEWNYM`s the option shall be disabled after each request. Vidalia achieves this by disabling the new identity action for `MIN_NEWIDENTITY_INTERVAL` after each trigger. `MIN_NEWIDENTITY_INTERVAL` is `#defined` in `MainWindow.cpp` to be 10 seconds long.

– Allow path selection using `PathSupport`. A separate dialog can be used to input values for the following options:
  * A `SelectionManager`shall be used to provide support for:

`pathlen`	Number of hops in each circuit
`use_all_exits`	Whether a router can appear more than once in a path

  * Similarly, `GeoIPConfig`can be used to specify:

`unique_countries`	Whether a country can appear more than once in a path
`continent_crossings`	Maximum number of continets traversed in a single path.
`ocean_crossings`	Maximum number of oceans traversed in a single path
`pathlen`	Number of hops in each circuit
`entry_country`	A single country for entry.
`middle_country`	A single country in middle of the path.
`exit_country`	A single country where path would terminate.
`excludes`	A list of countries which should be avoided in the path.

– Implement a simple configuration interface for setting up Tor in bridge or relay mode. Prompt the user for contact info, relay port, bandwidth limit and in case of relayes, exit policies. In case Tor's up and running, `tor-Tools` can again be used for altering configuration. In case it isn't, the `torrc` could be manipulated directly for setting up the appropriate mode. As per Damian's suggestion in comments the wizard/dialog would be disabled in case other controllers are detected running on the system.

  If our network interface has a private IP, find out whether we can use UPnP to address NAT issues. If we can't, guide user through a wizard to set up port forwarding on their router as well as configuring the computer's firewall by referring them to the Port Forward Wiki. For UPnP environments, follow this procedure:
  * Set up a UDP port `X` for listening on local machine and notify the user that `X` needs to be opened. This is critical since if our local firewall is dropping incoming packets for `X` we shall not be able to communicate with the gateway device (e.g. router).
  * Bind to `X` and multicast a Simple Service Discovery Protocol request to `239.255.255.250:1900`. Any gateway device on LAN shall respond to the SSDP `discover` message by unicasting a response in our direction at port `X`.
  * Receive the response from gateway device on port `X` and control the port mapping through the control URL specified in services.

  The major hinderance in making this "zero-configuration" is user's local firewall. Here is an example scenario for UPnP usage:
  * User opens port 1900 for UPnP communication and port 54321 for his bridge/relay.
  * arm communicates with the router and receives SSDP responses on port 1900, sets up port-forwarding for port 54321.

  The steps for firewall configuration *can* be reduced by operating the bridge/relay over the same port as the one we use for UPnP but that would be a bad idea since we won't be able to listen for future events from the gateway (apart from the inherent confusion it's going to generate for users).

Estimated Timeline

- April 25-May 22 *Community Bonding Period*: Familiarize myself with the Tor architecture, read up on path spec, understand the issues faced by bridge and relay operators. Inspect Vidalia and TorK. Dig into arm code and understand TorCtl internals and usage. Discuss ideas with the Tor community.
- May 23-May 30 *(Week 1)*: Implement menus and new identity requests in the curses interface to get familiar with arm codebase. Start connecting the PyGTK backend with arm `utils`.
- May 30-June 20 *(Week 2-4)*: Implement banwidth monitors, connections information and configuration editor.
- June 20-June 27 *(Week 5)*: Get community feedback and tweak the UI as necessary.
- June 27-July 11 *(Weeks 7-8)*: Implement proxy and DNS information gathering and subsequent port scanning required to check proxies' status.
- July 11-July 15 *(Week 8)*: Prepare code for midterm evaluation by reviewing the progress with atagar.
- July 15 *(Mid-term evaluation)*: Submit code for mid-term evaluation.
- July 18-July 25 *(Weeks 9-10)*: Implement path selection and UPnP port forwarding.
- July 25-August 8 *(Weeks 10-11)*: Implement bridge and relay setup configuration panels (á la Vidalia).
- August 8-August 15 *(Week 12, Suggested 'pencils down' date)*: Comment the code thoroughly, write user documentation and review the project state.
- August 15-August 22 *(Week 13, Firm 'pencils down' date)*: Polish the code and documentation and fill up final evaluations.

Point us to a code sample: something good and clean to demonstrate that you know what you're doing, ideally from an existing project.

(Most of my coding work is available at code.inspirated.com.)

- Google Summer of Code 2010:
  - Successfully completed the Ubuntu project: "Bug Triaging Improvements for Launchpad/Arsenal".
  - Implemented the following features for Launchpad:
    - Attachment Search (Text searches for attachments in bugs of a particular project).
    - Automatic Upstreamer (Forward bugs to remote trackers).
    - Bug Matchmaking (Search remote trackers for entries similar to a given bug).
    - Automatic Patcher (Generate Debian packages automatically patched using attachments of a bug).
  - Technologies used: Python, OAuth, REST, XML-RPC, Web scraping, Debian patch systems, Zope interfaces
    Links: Code Summary, Blog Archive
- Open Source Development:
  - Facebook Friends Graph: An application for creating a connected graph of the user's Facebook contacts for highlighting interesting patterns created by mutual friends.
    Technologies used: Python, GTK+, Graphviz, POSIX threads
    Links: Launchpad Project Page, Blog Archive
  - Bookmark Undertaker: An application which imports Firefox's bookmarks, updates their status and exports them back to a compatible HTML file.
    Technologies used: Python, GTK+, POSIX threads
    Links: Launchpad Project Page, Blog Archive
  - Pidgin Countdown: A plugin for the popular IM client which allows status messages to "countdown" towards/after a specific point in time.
    Technologies used: C, GTK+
    Links: Launchpad Project Page, Blog Archive
  - GSmolt: A GTK+ frontend for Smolt hardware profiler.
    Technologies used: Python, GTK+
    Links: GitHub Project Page, Blog Archive
  - Pidgin Countdown: A plugin for the popular IM client which allows status messages to "countdown" towards/after a specific point in time.
    Technologies used: C, GTK+
    Links: Launchpad Project Page, Blog Archive
  - Inbox Stats: A set of PyS60 scripts written to generate graphical statistics for Symbian mobile phones' received text messages.
    Technologies used: PyS60, Symbian threads
    Links: Blog Archive

  - CSV Auto-Responder: An application for Symbian phones which scans incoming SMS messages for queries and then replies back with results of data look-up from a CSV file.
    Technologies used: PyS60, Symbian threads
    Links: Blog Archive
  - Emu8086 Hardware Interrupt Editor & Generator: A set of utilities for manipulating the Interrupt Vector Table of Emu8086 and generating hardware interrupts while the emulator is running.
    Technologies used: Python, GTK+
    Links: Blog Archive
  - Worked on fixing bugs in various open-source projects.
  - Wrote various articles related to programming.

- Miscellaneous:
  - Progressed through the qualification round for Google Code Jam 2008.
  - Designed an XHTML compliant, resolution and browser independent website for my blog.

Why do you want to work with The Tor Project / EFF in particular?

I grew up in Saudi Arabia. Spending 15 years as an expatriate in a country where free speech is virtually non-existent etched in my mind the importance of privacy and anonymity. My beliefs were further shaped by Orwellian writings, specifically Nineteen Eighty-Four. It's intriguing as well as unnerving how the dystopian world of that masterpiece has already been realized by today's governments.

The interest in privacy ultimately lead to my introduction to network security and cryptography. I was suspended from my university for a semester for cracking their databases through publicly available methods like ARP spoofing and SSL stripping. Similarly, my graduation project is on implementing shared RSA key generation in Apache/SSL. However, cryptography isn't particularly my strong side and as such I would still want to help the activists' ecosystem through other avenues. Tor has already saved me from trouble on a plethora of occasions and I would like to contribute something back to the project and more importantly, the community.

Tell us about your experiences in free software development environments. We especially want to hear examples of how you have collaborated with others rather than just working on a project by yourself.

My most significant experience of free software development took place last year when I participated in Google Summer of Code 2010 as a student for the Ubuntu organization. During the course of the project I had to collaborate frequently with my mentor. In addition to this I had to get my code thoroughly reviewed by core Launchpad developers in order to get it merged upstream. Some of my changes were accepted, some were accepted after I wrote unit tests for them while some were rejected on grounds of being too resource intensive for Launchpad production servers. The whole experience taught me valuable lessons on development in collaborative environments.

Will you be working full-time on the project for the summer, or will you have other commitments too (a second job, classes, etc)? If you won't be available full-time, please explain, and list timing if you know them for other major deadlines (e.g. exams). Having other activities isn't a deal-breaker, but we don't want to be surprised.

This is going to be my last semester in university unless I flunk a particular course again. In which case I'll have to attend a 3-credit hours' class during the summers which should not be a problem for me to manage in parallel with GSoC.

Will your project need more work and/or maintenance after the summer ends? What are the chances you will stick around and help out with that and other related projects?

The project shall need further testing, packaging and improvements and I plan to get packages for the new GUI accepted in repositories of main distributions after the summer. I already have packaging experience with RPMs and Debian packages. At any rate I shall be in active development of ARM at least through 2011. After that I may have to shift focus on a job prospect but I'll still continue to oversee the major developments on GTK+ frontend.

What is your ideal approach to keeping everybody informed of your progress, problems, and questions over the course of the project? Said another way, how much of a "manager" will you need your mentor to be?

I am always available at IRC which I check at least once per day so any message there should reach me within 24-hours. On the other hand I have email configured on my cell phone so unless I'm dozing off I shall always read any email sent to me within a few minutes. The progress of my project shall be documented over at my blog which I have been regularly updating for more than 4 years now. Damian and I shall stay in touch mostly through IRC which is how I collaborated with my mentor last year as well.

What school are you attending? What year are you, and what's your major/degree/focus? If you're part of a research group, which one?

I'm currently attending National University of Computer & Emerging Sciences in Pakistan. I'm enrolled for a degree of BS (Electrical Engineering with majors in Telecom) and am expected to graduate this summer. My interests include open-source development, scripting languages, networking, security, assembly and low-level system programming.

How can we contact you to ask you further questions? Google doesn't share your contact details with us automatically, so you should include that in your application. In addition, what's your IRC nickname? Interacting with us on IRC will help us get to know you, and help you get to know our community.

I can be reached via email at krkhan@inspirated.com. For IRC communication I can be found idling on Freenode and OFTC with the nickname `krkhan`.