CS790 - Privacy Enhancing Technologies
Assignment 3 - Tor Circuits

Shantanu Welling

April 2025

Contents

Task 1 - Tor Circuit Creation

1. Uncommented `ControlPort 9051` and `CookieAuthentication 1` lines in `torrc` file
2. `ControlPort 9051`: This line enables the Tor ControlPort on port 9051, allowing external applications (such as scripts or controllers) to communicate with the Tor daemon and control it programmatically. Through this port, commands can be issued to build circuits, obtain circuit status, and perform other advanced operations.
3. `CookieAuthentication 1`: This enables secure cookie based authentication for the control port using a cookie file automatically generated by Tor. Any client wishing to interact with the ControlPort must read and present this authentication cookie (usually located at `/run/tor/control.authcookie` or `/var/run/tor/control.authcookie`). This avoids the need to hardcode a password and improves security.
4. `tor -f /usr/local/etc/tor/torrc` to run tor with the updated config file
5. `query(url)`: This function sends an HTTP request to a given URL using `pycurl` through the SOCKS5 proxy at `localhost:9050`, which is the default Tor proxy port. It is used to test if the built circuit is functional.
6. `get_path(nodes)`: Randomly selects and returns a valid 4-hop circuit composed of:
   - One **Guard** node (`"Guard"` in flags),
   - Two **Middle** relays (no specific role),
   - One **Exit** node (`"Exit"` in flags).

   Only relays with `"Valid"`, `"Stable"`, and `"Running"` flags are considered, and those with `"BadExit"` are excluded.

7. `controller = Controller.from_port(port=9051)`: Establishes a connection to the Tor controller port (enabled in `torrc` using `ControlPort 9051` and `CookieAuthentication 1`).

8. `controller.authenticate()`: Authenticates with the control port using the Tor-generated cookie (required due to `CookieAuthentication 1`).

9. `controller.get_network_statuses()`: Fetches all known relays and their status from the Tor consensus for selection in the path-building step.

10. `controller.new_circuit(path, await_build=True)`: Establishes a custom new 4-hop Tor circuit from the output of `get_path` with the selected relay fingerprints, waiting until the circuit is fully constructed.

11. `controller.set_conf("__LeaveStreamsUnattached", "1")`: Prevents Tor from automatically attaching streams to circuits, allowing manual stream attachment.

12. `controller.add_event_listener(attach_stream, EventType.STREAM)`: Registers a callback function `attach_stream()` that listens for new streams and manually attaches them to the custom circuit using `controller.attach_stream()`.

13. `query("https://www.example.com")`: Sends a request through the built circuit. The stream generated by this request is intercepted and manually attached to the intended circuit, ensuring it follows the selected path.

14. `controller.remove_event_listener()` and `reset_conf()`: Cleans up the controller's state after the request is made by removing the event listener callback function and resetting all configuration to the default state.

```
(.myvenv) root@vbox:/home/vboxuser/Desktop# python3 tor_circuit.py
[*] Fetching relay fingerprints...
[+] Building 4-hop circuit: ['5F46DDAAFCCBC9EF7503365608EF1C6A49DC5250', '66DCBBB971CA555F86BB2B1D17C00C1828368736', '8CD3F8019FAFAC6
D18C56E51DD143D8794E08D4D', 'A1E76D835876DF5BC5D59FD8817715667116A3EB']
[-] Failed to build circuit: Circuit failed to be created: CHANNEL_CLOSED
[-] Retrying...
[+] Building 4-hop circuit: ['D9E4F7FA740152EBD98C3DE7525F488E7CA859FA', 'D5FC829FE684B1FC3496C8FB502E7D19E928DF13', '8AD04B12C1664D7
C8063674A35C03B7703E9D8AE', '9C61A0D830BDC2E1378F0E7ED7C8C8E06B312827']
[+] Built circuit with ID: 41

[+] Circuit path:
[+] 1) Guard:   ledevin                 50.7.115.67
[+] 2) Middle1: Unnamed                 77.220.107.195
[+] 3) Middle2: Unnamed                 185.193.126.236
[+] 4) Exit:    CCCStuttgartBer         185.220.101.77

[*] Using SOCKS proxy to send request through custom circuit...
[DEBUG] Attaching stream 41 to circuit 41
[DEBUG] Stream 41 - Status: NEW | Target: www.example.com:443 | Circuit: 41
STREAM 41 NEW 0 www.example.com:443 SOURCE_ADDR=127.0.0.1:48692 PURPOSE=USER CLIENT_PROTOCOL=SOCKS5 NYM_EPOCH=0 SESSION_GROUP=-4 ISO_
FIELDS=SOCKS_USERNAME,SOCKS_PASSWORD,CLIENTADDR,SESSION_GROUP,NYM_EPOCH
[DEBUG] Stream 41 - Status: CONTROLLER_WAIT | Target: www.example.com:443 | Circuit: 41
STREAM 41 CONTROLLER_WAIT 0 www.example.com:443 CLIENT_PROTOCOL=SOCKS5 NYM_EPOCH=0 SESSION_GROUP=-4 ISO_FIELDS=SOCKS_USERNAME,SOCKS_P
ASSWORD,CLIENTADDR,SESSION_GROUP,NYM_EPOCH
[DEBUG] Stream 41 - Status: SENTCONNECT | Target: www.example.com:443 | Circuit: 41
STREAM 41 SENTCONNECT 41 www.example.com:443 CLIENT_PROTOCOL=SOCKS5 NYM_EPOCH=0 SESSION_GROUP=-4 ISO_FIELDS=SOCKS_USERNAME,SOCKS_PASS
WORD,CLIENTADDR,SESSION_GROUP,NYM_EPOCH
[DEBUG] Stream 41 - Status: REMAP | Target: 95.100.248.139:443 | Circuit: 41
STREAM 41 REMAP 41 95.100.248.139:443 SOURCE=EXIT CLIENT_PROTOCOL=SOCKS5 NYM_EPOCH=0 SESSION_GROUP=-4 ISO_FIELDS=SOCKS_USERNAME,SOCKS
_PASSWORD,CLIENTADDR,SESSION_GROUP,NYM_EPOCH
[DEBUG] Stream 41 - Status: SUCCEEDED | Target: 95.100.248.139:443 | Circuit: 41
STREAM 41 SUCCEEDED 41 95.100.248.139:443 CLIENT_PROTOCOL=SOCKS5 NYM_EPOCH=0 SESSION_GROUP=-4 ISO_FIELDS=SOCKS_USERNAME,SOCKS_PASSWOR
D,CLIENTADDR,SESSION_GROUP,NYM_EPOCH
```

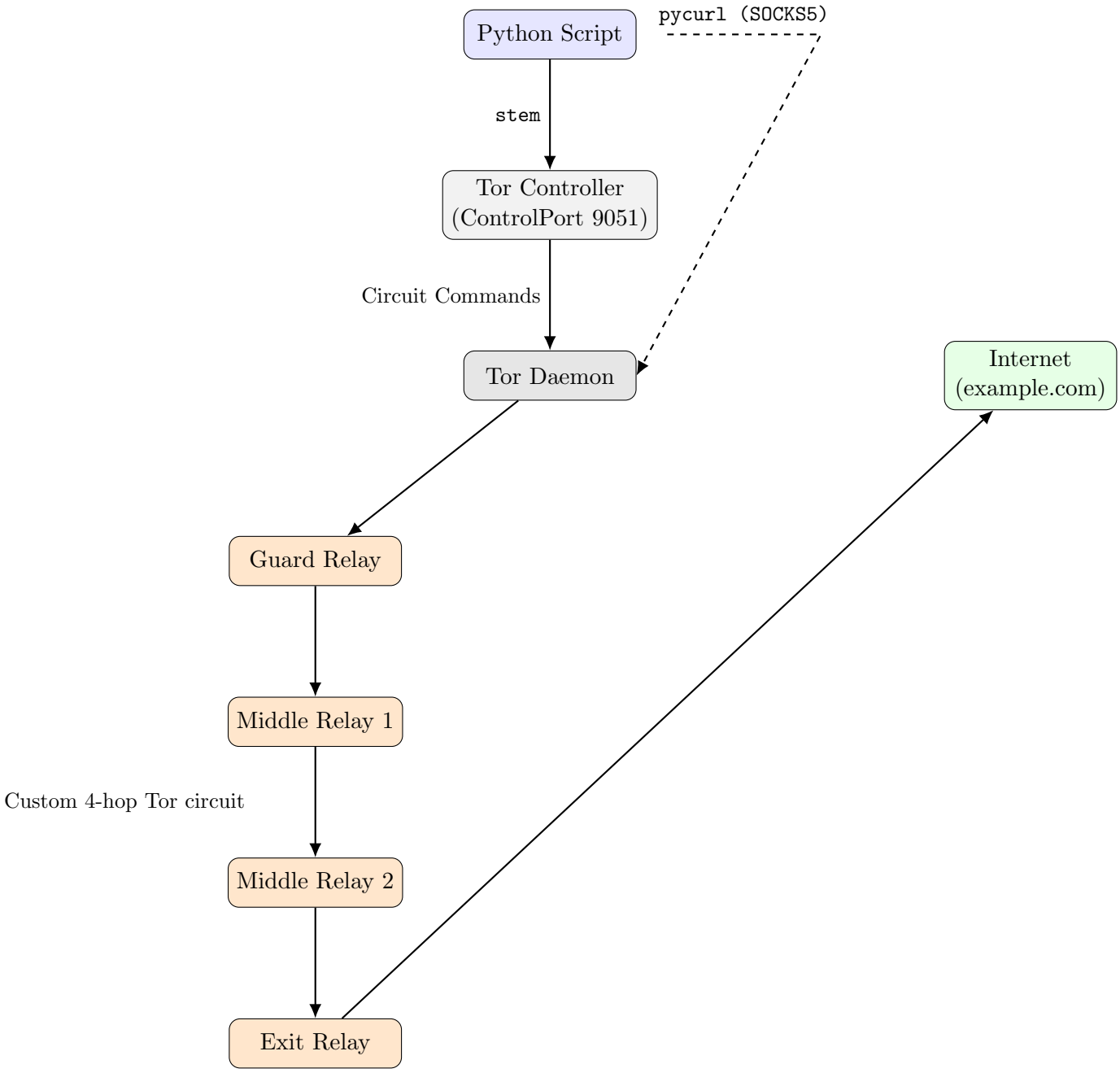
Screenshot showing the connection to the appropriate website is going over the Tor circuit

```
[DEBUG] Stream 41 - Status: CONTROLLER_WAIT | Target: www.example.com:443 | Circuit: 41
STREAM 41 CONTROLLER_WAIT 0 www.example.com:443 CLIENT_PROTOCOL=SOCKS5 NYM_EPOCH=0 SESSION_GROUP=-4 ISO_FIELDS=SOCKS_USERNAME,SOCKS_P
ASSWORD,CLIENTADDR,SESSION_GROUP,NYM_EPOCH
[DEBUG] Stream 41 - Status: SENTCONNECT | Target: www.example.com:443 | Circuit: 41
STREAM 41 SENTCONNECT 41 www.example.com:443 CLIENT_PROTOCOL=SOCKS5 NYM_EPOCH=0 SESSION_GROUP=-4 ISO_FIELDS=SOCKS_USERNAME,SOCKS_PASS
WORD,CLIENTADDR,SESSION_GROUP,NYM_EPOCH
[DEBUG] Stream 41 - Status: REMAP | Target: 95.100.248.139:443 | Circuit: 41
STREAM 41 REMAP 41 95.100.248.139:443 SOURCE=EXIT CLIENT_PROTOCOL=SOCKS5 NYM_EPOCH=0 SESSION_GROUP=-4 ISO_FIELDS=SOCKS_USERNAME,SOCKS
_PASSWORD,CLIENTADDR,SESSION_GROUP,NYM_EPOCH
[DEBUG] Stream 41 - Status: SUCCEEDED | Target: 95.100.248.139:443 | Circuit: 41
STREAM 41 SUCCEEDED 41 95.100.248.139:443 CLIENT_PROTOCOL=SOCKS5 NYM_EPOCH=0 SESSION_GROUP=-4 ISO_FIELDS=SOCKS_USERNAME,SOCKS_PASSWOR
D,CLIENTADDR,SESSION_GROUP,NYM_EPOCH
[+] Response Status Code: 200
[+] Response: <!doctype html>
<html>
<head>
    <title>Example Domain</title>

    <meta charset="utf-8" />
    <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style type="text/css">
    body {
        background-color: #f0f0f2;
        margin: 0;
        padding: 0;
        font-family: -apple-system, system-ui, BlinkMacSystemFont, "Segoe UI", "Open Sans", "Helvetica Neue", Helvetica, Arial, sans-
serif;

[DEBUG] Stream 41 - Status: CLOSED | Target: 95.100.248.139:443 | Circuit: 41
STREAM 41 CLOSED 41 95.100.248.139:443 REASON=DONE CLIENT_PROTOCOL=SOCKS5 NYM_EPOCH=0 SESSION_GROUP=-4 ISO_FIELDS=SOCKS_USERNAME,SOCK
S_PASSWORD,CLIENTADDR,SESSION_GROUP,NYM_EPOCH
(.myvenv) root@vbox:/home/vboxuser/Desktop#
```

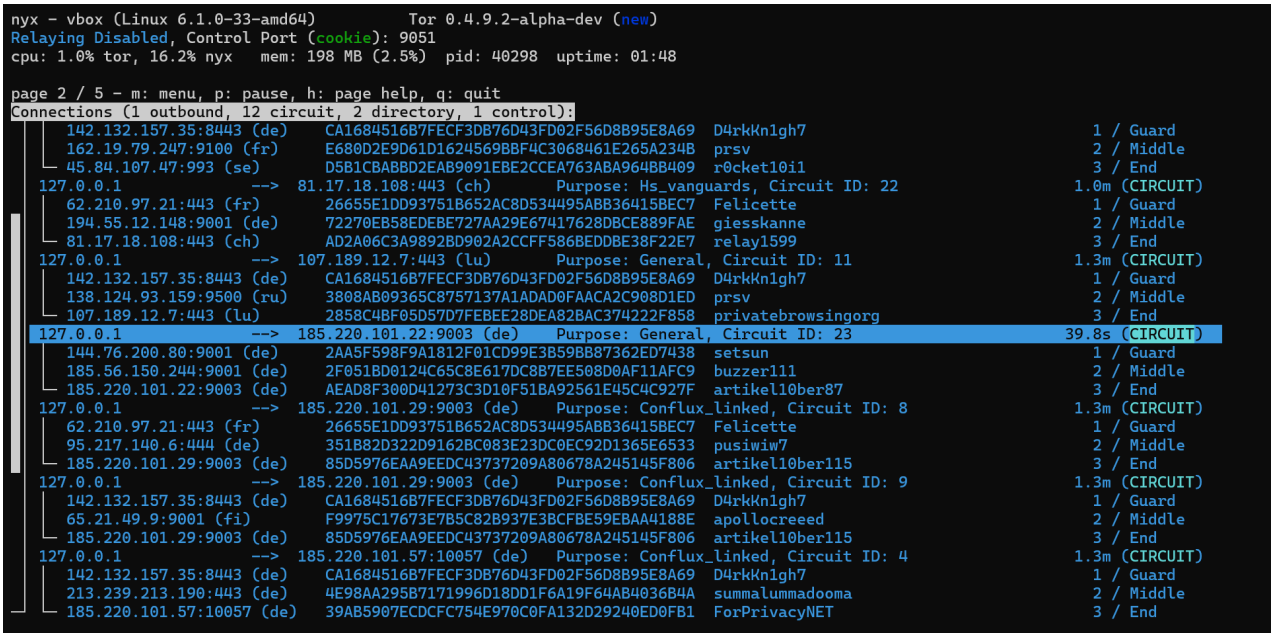
Screenshot showing the connection to the appropriate website is going over the Tor circuit

```
(see https://nyx.torproject.org/nyxrc.sample for its options).ev (new)
Relaying Disabled, Control Port (cookie): 9051
cpu: 0.2% tor, 1.1% nyx    mem: 84 MB (1.1%)   pid: 36981  uptime: 50:23

page 2 / 5 - m: menu, p: pause, h: page help, q: quit
Connections (6 outbound, 9 circuit, 2 directory, 1 control):
  127.0.0.1             -->  185.220.101.44:10044 (de)   Purpose: Conflux_linked, Circuit ID: 23                4.8m (CIRCUIT)
  |  145.239.41.102:9100 (fi)    3A04AC8969E55DF51C8D11C49ABF18A0CC847FC0  Unnamed                                  1 / Guard
  |  45.141.215.28:9000 (pl)     9C788AA15E1878734117CDBC1A4A369C307CE8AF3  Aramis                                   2 / Middle
  |_ 185.220.101.44:10044 (de)   4A169C0A14E41F647D009EC49D28A3D11629DAF0  ForPrivacyNET                            3 / End
  127.0.0.1             -->  185.220.101.44:10044 (de)   Purpose: Conflux_linked, Circuit ID: 24                4.8m (CIRCUIT)
  |  142.132.157.35:8443 (de)    CA1684516B7FECF3DB76D43FD02F56D8B95E8A69  D4rkKn1gh7                              1 / Guard
  |  185.242.225.24:39109 (gb)   970F61B6E5DCF3976DD182F657B4835FA8A8B585  hyberion                               2 / Middle
  |_ 185.220.101.44:10044 (de)   4A169C0A14E41F647D009EC49D28A3D11629DAF0  ForPrivacyNET                            3 / End
  127.0.0.1             -->  185.220.101.77:9100 (de)    Purpose: General, Circuit ID: 41                      34.8s (CIRCUIT)
  |  50.7.115.67:9001 (us)       D9E4F7FA740152EBD98C3DE7525F488E7CA859FA  ledevin                                 1 / Guard
  |  77.220.107.195:9001 (at)    D5FC829FE684B1FC3496C8FB502E7D19E928DF13  Unnamed                                 2 / Middle
  |  185.193.126.236:9001 (se)   8AD04B12C1664D7C8063674A35C03B7703E9D8AE  Unnamed                                 3 / Middle
  |_ 185.220.101.77:9100 (de)    9C61A0D830BDC2E1378F0E7ED7C8C8E06B312827  CCCStuttgartBer                          4 / End
  127.0.0.1             -->  188.68.36.28:443 (de)       Purpose: General, Circuit ID: 27                       4.8m (CIRCUIT)
  |  142.132.157.35:8443 (de)    CA1684516B7FECF3DB76D43FD02F56D8B95E8A69  D4rkKn1gh7                              1 / Guard
  |  77.221.157.237:443 (ru)     B9C5243087CF69B6A57A8318B312B79F19D6D594  cozybeardev                            2 / Middle
  |_ 188.68.36.28:443 (de)       E8018E2C62E982758DBC173E3FEE26C917B353B1  artikel5ev42                           3 / End
  127.0.0.1             -->  193.189.100.201:443 (se)    Purpose: General, Circuit ID: 33                       3.0m (CIRCUIT)
  |  45.94.31.22:110 (de)        633DF3F3CBC6BF9C58D91D5028EB76933A666C0D  RDPdotSH                                1 / Guard
  |  51.210.179.144:9200 (fr)    E9F1C405585300BC196EA7E9C021B9307734AB552  prsv                                   2 / Middle
  |  217.182.79.225:9998 (pl)    97B326AB73BD5B484F89E976EED9FD9B807CD39F  torkowo3                               3 / Middle
  |_ 193.189.100.201:443 (se)    31D391F4720DE896C598B72369AF880ED086D614  TORKeFFORG8                            4 / End
  10.0.2.15:47696       -->  142.132.157.35:8443 (de)    CA1684516B7FECF3DB76D43FD02F56D8B95E8A69  D4rkKn1gh7  +50.2m (DIRECTORY)
  10.0.2.15:52084       -->  145.239.41.102:9100 (fi)    3A04AC8969E55DF51C8D11C49ABF18A0CC847FC0  Unnamed     +50.2m (DIRECTORY)
  127.0.0.1:34952 (??)        -->  127.0.0.1:9051         nyx (36986)                                             +50.2m (CONTROL)
```

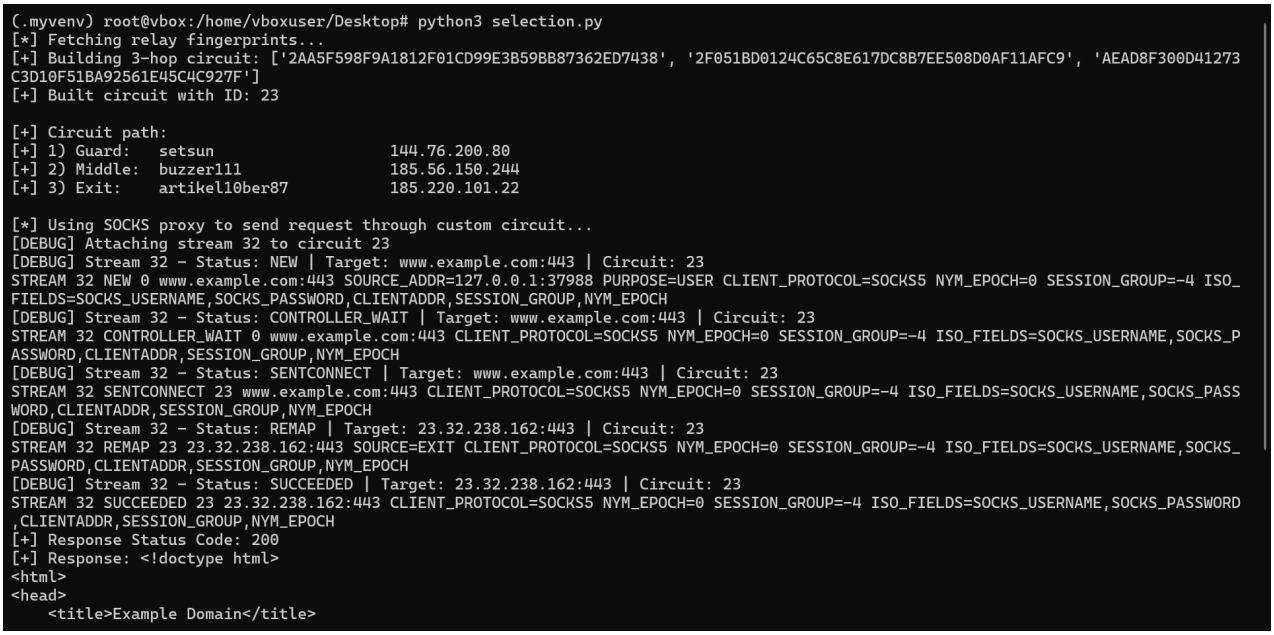
Output of `nyx`. Notice Control Port and cookie authentication enabled. Notice **Circuit ID: 41**


Python Script
pycurl (SOCKS5)
stem
Tor Controller
(ControlPort 9051)
Circuit Commands
Tor Daemon
Internet
(example.com)
Guard Relay
Middle Relay 1
Custom 4-hop Tor circuit
Middle Relay 2
Exit Relay

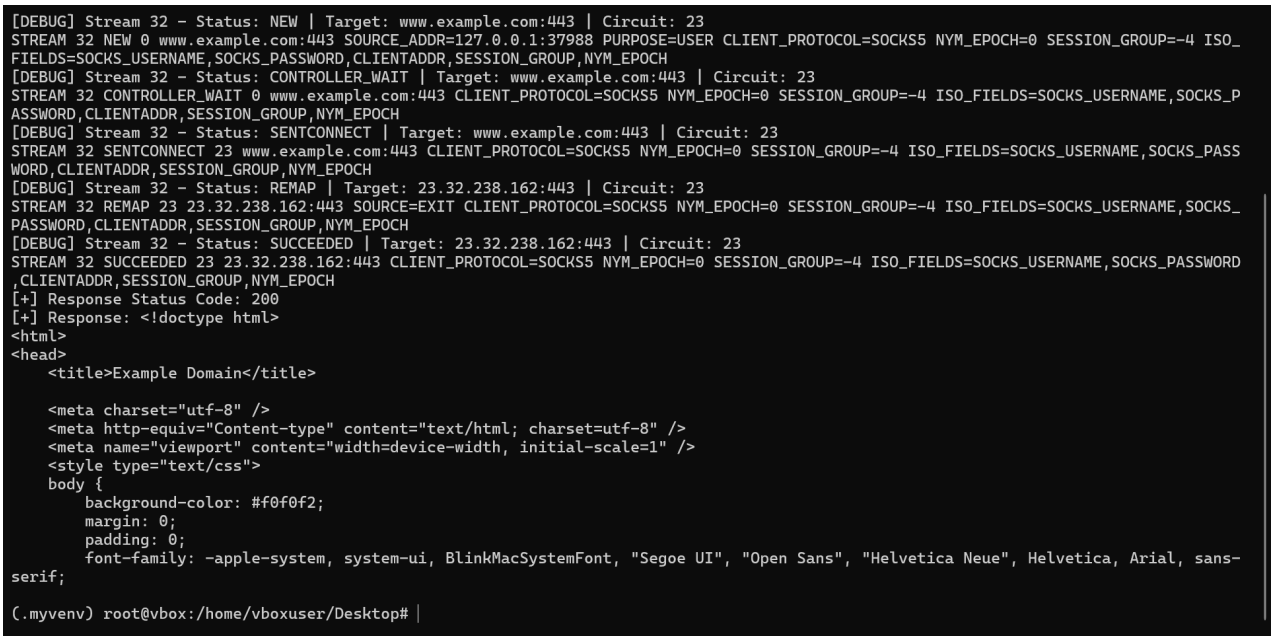
Task 2 - Tor Relay Selection Simulator



Output of `nyx`. Notice **Circuit ID: 23**



Screenshot showing the circuit creation simulation and connection to the appropriate website is going over the Tor circuit



Screenshot showing the connection to the appropriate website is going over the Tor circuit

- **Circuit Construction Order:**
  - Exit node (hop 3) is selected first.
  - Then, the Guard node (hop 1) is selected.
  - Finally, the Middle node (hop 2) is selected.
- **Constraints on node selection:**
  - All nodes must be `Running`, `Valid`, `Stable`, and must not have the `StaleDesc` flag.
  - The same node must not appear more than once in a path.
  - Nodes in the same family (as declared in the descriptor) are not allowed in the same circuit.
  - No two nodes may be within the same /16 subnet.
  - The first node must be a Guard.
  - For the Exit node, nodes flagged with `BadExit` are excluded.
  - The implementation assumes a non-`Fast` circuit; however, converting to a `Fast` circuit is a one-line change.
- **Relay Weighting:**
  - Each node is selected via weighted random sampling proportional to:

$$\text{Weight}(\text{node}) = w \cdot B$$

    where B is the measured bandwidth of the node and w is a consensus-published weight depending on node type and its position.
  - These weights are from the consensus' `bandwidth-weights` line and are denoted:
    * `Wgg, Wgm, Wgd` – Guard position weights
    * `Wmg, Wmm, Wme, Wmd` – Middle position weights

  - `Weg, Wem, Wee, Wed` – Exit position weights
- If a node has both `Guard` and `Exit` flags, we linearly interpolate its weight using:

$$\text{Effective Weight} = F \cdot W_{pf} \cdot B + (1 - F) \cdot W_{pn} \cdot B$$

  where F is a bias factor, W_{pf} is the weight if flagged, W_{pn} if not.
- If any weights are malformed or missing, default value of 10000 is used.

- **Selection Heuristics:**
  - **Exit Node:** Must be flagged as `Exit` and not `BadExit`. Preference for high-bandwidth and optionally also `Guard`-flagged.
  - **Guard Node:** Must be `Guard`-flagged and disjoint in /16 subnet and family from the Exit node. Nodes flagged as both `Guard+Exit` receive adjusted weighting.
  - **Middle Node:** Must be disjoint in /16 subnet and family from both Exit and Guard. Weight depends on which flags (`Guard`, `Exit`, both, or none) the node has.

- **Implementation Details:**
  - I parsed the latest network consensus from local `cached-consensus` file using `NetworkStatusDocumentV3` to obtain relay descriptors. From these, I extracted and stored the bandwidth weights.
  - The path selection logic was customized to prioritize relays with higher consensus bandwidth weights while satisfying the above mentioned constraints.
  - Circuit construction, attaching stream (callback), adding event listener and sending web request is similar to Task 1. The only thing different here is how the circuit nodes are selected.
  - Robust error handling was implemented: the script retries circuit construction if an extension fails due to relay downtime, unreachable ports, or outdated descriptors.
  - In addition to uncommenting the lines in `torrc` file from Task 1, I also added the line `UseMicrodescriptors 0` which tells the Tor client not to use microdescriptors, and instead to fetch and use full server descriptors. These full descriptors have the family information available for each relay, which isn't the case with microdescriptors. So, in order to incorporate the family logic, we need full server descriptors and therefore we need this line in `torrc`.
  - I used `controller.get_server_descriptors()` to fetch the full server descriptors, from which I extracted and stored the fingerprint → family mapping for each relay. I used this mapping to check whether the family sets of relays are disjoint while selecting a circuit path.

- **Extensibility:**
  - By modifying one line, this implementation can:
    - Enforce the `Fast` flag.
    - Allow or disallow `BadExit` nodes.
    - Disable the `Stable` requirement.

References

- Tor Path Spec
- Stem Router Status Entries
- Stem Controller
- Server Descriptors
- Microdescriptors
- Network Status Document Parsing
- Bandwidth File