```
// START_DEFINITION
/**
 * Fingerprint Tor authoritative directories enacting the directory protocol.
 */
fingerprint('anonymizer/tor/node/authority') = $tor_authority
  and ($tor_directory or preappid(/anonymizer\/tor\/directory/));
// END_DEFINITION

// START_DEFINITION
/*
Global Variable for Tor foreign directory servers. Searching for potential Tor
clients connecting to the Tor foreign directory servers on ports 80 and 443.
*/

$tor_foreign_directory_ip = ip('193.23.244.244' or '194.109.206.212' or
'86.59.21.38' or '213.115.239.118' or '212.112.245.170') and port ('80' or
'443');
// END_DEFINITION

// START_DEFINITION
/*
this variable contains the 3 Tor directory servers hosted in FVEY countries.
Please do not update this variable with non-FVEY IPs. These are held in a
separate variable called $tor_foreign_directory_ip. Goal is to find potential
Tor clients connecting to the Tor directory servers.
*/
$tor_fvey_directory_ip = ip('128.31.0.39' or '216.224.124.114' or
'208.83.223.34') and port ('80' or '443');
// END_DEFINITION


// START_DEFINITION
requires grammar version 5
/**
 * Identify clients accessing Tor bridge information.
 */
fingerprint('anonymizer/tor/bridge/tls') =
ssl_x509_subject('bridges.torproject.org') or
ssl_dns_name('bridges.torproject.org');

/**
 * Database Tor bridge information extracted from confirmation emails.
 */
fingerprint('anonymizer/tor/bridge/email') =
email_address('bridges@torproject.org')
  and email_body('https://bridges.torproject.org/' : c++
  extractors: {{
    bridges[] = /bridge\s([0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}):?([0-9]{2,4}?[^0-9])/;
  }}
  init: {{
    xks::undefine_name("anonymizer/tor/torbridges/emailconfirmation");
  }}
  main: {{
    static const std::string SCHEMA_OLD = "tor_bridges";
```

```
    static const std::string SCHEMA_NEW = "tor_routers";
    static const std::string FLAGS = "Bridge";
    if (bridges) {
      for (size_t i=0; i < bridges.size(); ++i) {
        std::string address = bridges[i][0] + ":" + bridges[i][1];
        DB[SCHEMA_OLD]["tor_bridge"] = address;
        DB.apply();
        DB[SCHEMA_NEW]["tor_ip"] = bridges[i][0];
        DB[SCHEMA_NEW]["tor_port_or"] = bridges[i][1];
        DB[SCHEMA_NEW]["tor_flags"] = FLAGS;
        DB.apply();
      }
      xks::fire_fingerprint("anonymizer/tor/directory/bridge");
    }
    return true;
  }});
// END_DEFINITION


// START_DEFINITION
/*
The fingerprint identifies sessions visiting the Tor Project website from
non-fvey countries.
*/
fingerprint('anonymizer/tor/torpoject_visit')=http_host('www.torproject.org')
and not(xff_cc('US' OR 'GB' OR 'CA' OR 'AU' OR 'NZ'));
// END_DEFINITION


// START_DEFINITION
/*
These variables define terms and websites relating to the TAILs (The Amnesic
Incognito Live System) software program, a comsec mechanism advocated by
extremists on extremist forums.
*/

$TAILS_terms=word('tails' or 'Amnesiac Incognito Live System') and word('linux'
or ' USB ' or ' CD ' or 'secure desktop' or ' IRC ' or 'truecrypt' or ' tor ');
$TAILS_websites=('tails.boum.org/') or ('linuxjournal.com/content/linux*');
// END_DEFINITION

// START_DEFINITION
/*
This fingerprint identifies users searching for the TAILs (The Amnesic
Incognito Live System) software program, viewing documents relating to TAILs,
or viewing websites that detail TAILs.
*/
fingerprint('ct_mo/TAILS')=
fingerprint('documents/comsec/tails_doc') or web_search($TAILS_terms) or
url($TAILS_websites) or html_title($TAILS_websites);
// END_DEFINITION


// START_DEFINITION
```

```
requires grammar version 5
/**
 * Aggregate Tor hidden service addresses seen in raw traffic.
 */
mapreduce::plugin('anonymizer/tor/plugin/onion') =
  immediate_keyword(/(?:([a-z]+):\/\/){0,1}([a-z2-7]{16})\.onion(?::(\d+)){0,1}/c : c++
    includes: {{
      #include <boost/lexical_cast.hpp>
    }}
    proto: {{
      message onion_t {
        required string address = 1;
        optional string scheme = 2;
        optional string port = 3;
      }
    }}
    mapper<onion_t>: {{
      static const std::string prefix = "anonymizer/tor/hiddenservice/address/";

      onion_t onion;
      size_t matches = cur_args()->matches.size();
      for (size_t pos=0; pos < matches; ++pos) {
        const std::string &value = match(pos);
        if (value.size() == 16)
          onion.set_address(value);
        else if(!onion.has_scheme())
          onion.set_scheme(value);
        else
          onion.set_port(value);
      }

      if (!onion.has_address())
        return false;

      MAPPER.map(onion.address(), onion);
      xks::fire_fingerprint(prefix + onion.address());
      return true;
    }}
    reducer<onion_t>: {{
      for (values_t::const_iterator iter = VALUES.begin();
          iter != VALUES.end();
          ++iter) {
        DB["tor_onion_survey"]["onion_address"] = iter->address() + ".onion";
        if (iter->has_scheme())
          DB["tor_onion_survey"]["onion_scheme"] = iter->scheme();
        if (iter->has_port())
          DB["tor_onion_survey"]["onion_port"] = iter->port();
        DB["tor_onion_survey"]["onion_count"] = boost::lexical_cast<std::string>(TOTAL_VALUE_COUNT);
        DB.apply();
        DB.clear();
      }
      return true;
    }});
```

```
/**
 * Placeholder fingerprint for Tor hidden service addresses.
 * Real fingerpritns will be fired by the plugins
 *   'anonymizer/tor/plugin/onion/*'
 */
fingerprint('anonymizer/tor/hiddenservice/address') = nil;
// END_DEFINITION


// START_DEFINITION
appid('anonymizer/mailer/mixminion', 3.0, viewer=$ascii_viewer) =
        http_host('mixminion') or
        ip('128.31.0.34');
// END_DEFINITION
```