Tor: The Onion Router Project

November 1, 2012

Disclaimer: huge swaths of this talk are lovingly ripped off of Roger Dingledine's 2005 What The Hack presentation and his follow-on talks at 24C3, 25C3, and 28C3, as well as Nick Matthewson's LEET'11 talk.
`https://www.torproject.org/docs/documentation`

Outline

Introduction

What is Tor?
Uses For Anonymity
Anonymity Against Who and What?
Adversary Threat Model

Introduction
What is Tor?

- Jargon: Tor is a "second generation" onion router.
    - To be explained later.
- Slightly less jargon: Tor aims to protect users against *traffic analysis*.
- Tor is a mechanism for building "anonymous" connections to services on the Internet.

Introduction
Uses For Anonymity: Censorship Resistance

- Want to dodge attempted censorship.
  - The Great Firewall
  - "Arab Spring" examples
- So we want to be able to hide who we are from
  - Local adversaries
    - Want a truly general purpose link
  - The remote service itself!
    - We're "just some guy"

Introduction

Uses For Anonymity: More Censorship Resistance

- The dual problem: *we* have information that we want to *publish* which
  - is culturally taboo
  - would get us in trouble with the authorities
- Here, we want to publish exactly controlled information
  - Before, we were consuming information anonymously.
  - We want only a handle, or a pseudonym, or a strong cryptographic identity, but *not* our real identity, associated with the publication.

Introduction
Uses For Anonymity: More Censorship Resistance

- This *is not just theoretic*.
- Notable examples include political blogging.
  - Yahoo has cooperated with Chinese authorities!
- Whistleblowers likely want to be anonymous.

Introduction
Uses For Anonymity: (Semi)private Information

- Selective disclosure of information.
- Roger's old examples:
  - Forums / chat rooms for abuse survivors
  - Look up information about disease without revealing who's asking.
- Roger's new example (DEF CON 2007):
  - Big, Burly Biker secretly has love of looking at pictures of cute kittens on the Internet.
  - Doesn't want his Biker Buddies to find out.

Introduction
Uses For Anonymity: (Semi)private Information

A few kinds of services we want to guard against (Mathewson, LEET'11):

- "Indifferent" services ("Not *my* problem").
- "Incompetent" services who might lose the logs (see: AOL).
- "Hostile" services who might sell logs (see: everybody).

Of course, services might also be *coerced* into revealing their logs or contents.

Introduction
Uses For Anonymity: The man

- Wait, *The man* wants anonymity?
- It's very handy to adopt new identities!
  - Have to shed the old one first.
- Corporations
  - Hide supplier/client relationships or patterns.
  - Google and Bing are interested in poking at each other without revealing who they are.
  - Check resume sites to see if employees are unhappy?
- Law Enforcement
  - Covert surveillance & honeypot operations.
  - Wants you to have it too: anonymous tips!

Introduction
Uses For Anonymity: The Man

- Wait, *The Man* (note cap M) wants anonymity too?
- Intelligence gathering without revealing identity
  - "The DoD wants to know..."
- International relations
  - Hide extent of communication between parties.
- Elections & voting!
  - These only work if The Man *grants* some weak form of authenticated anonymity.
  - Even Congresscritters want to vote anonymously.

Introduction
Uses For Anonymity: Criminals

- Yep, them too, for obvious reasons.
- But they already had it.
  - More resources available for it.
  - More willingness to learn tools to get it.
  - Can instead just outright steal credentials, phones, computers (or network links), . . .

Introduction
Anonymity Against Who and What?

- What does it mean to be anonymous?
- It means we want to make it computationally infeasable to identify us, even partially.
- Who might identify us?
  - The people we're talking to.
  - People watching the network between us.
- Anonymity means you know nothing about me, except what I choose to give you
  - When web browsing, a service learns that *somebody* requested a URL.

Introduction
Anonymity Against Who and What?

- Jargon: Authenticated anonymity makes sense.
- If I can make anonymous connections, I can authenticate myself to a service using a pseudonym.
- As long as *all* my connections to the service are anonymous, the service has no idea who I "really" am.

Introduction
Anonymity Against Who and What?

- You can't be anonymous by yourself!
  - (But you can be *private* by yourself.)
- If JHU ran a proxy for JHU students,
  - Servers don't know who connected
  - But they know that it was a student at JHU.
- To be anonymous, you have to carry traffic for others.
  - The others have to believe that you aren't out to get them.
- To be secure, the network needs to be diverse.

Introduction
Anonymity Against Who and What?

- Tor is mostly interested in the *connection*'s anonymity, not with the data that goes over it.
    - There are protocols for anonymous messaging
        - Now if only I could send you a message.
    - Contrawise, clients can be dumb and send clear text.
        - If you want *private* bits, don't be dumb.
- We'll talk about *data anonymity* later.

Introduction
Adversary Threat Model

- Technical jargon:
  - Probabilistic Polynomial Time (PPT)
  - Either logarithmic or polynomial space
  - Bounded ability to compromise nodes
- Realistically:
  - Can only watch a subset of the network traffic.
  - $P \neq NP$: can't invert RSA, DHKEX, RC5, etc.
  - Can't screw "too much" with the network
    - e.g. can't DoS the whole thing at once.
    - can't have owned the entire thing.

Introduction
Adversary Threat Model

- Important to note that adversary is attacking the *connection*'s anonymity.
  - The data that goes over the link is assumed to be sufficiently clean or encrypted etc.
  - This is actually a real problem in the Tor world, but we'll talk about it later.
- Several choices of threat model (big space).
  - Is it realistic to assume that the adversary can observe 1 node? 1000 nodes? The entire network?
  - How willing is the adversary to attack nodes?
  - Just *how much* computing power does the adversary have up her sleeve?

Older Mechanisms for Anonymity
Anonymizing Proxy: Server

- One server (to rule them all)
  - Accepts connections from clients.
  - Makes connections to services on behalf of the clients.
  - In real time (i.e. without delay)
- Good:
  - Hides client location from servers.
  - Works even for interactive connections
  - Very easy to set up.

Older Mechanisms for Anonymity
Anonymizing Proxy: Server

- Bad:
    - Hard to find out about proxies!
        - "Hey buddy, wanna buy a proxy? I know a good one..."
    - High load on proxy (can't be alone).
    - Single point of failure
    - Single point of compromise for a large # of clients.
        - Threat model: adversary unable or unwilling to compromise proxy.
        - Assume the proxy is trustworthy!

Older Mechanisms for Anonymity
Anonymizing Proxy: Chains

- Clients can route through multiple proxies.
- May help eliminate the single point of failure
  - For example, rotate proxies over time.
- Single point of identity compromise, still:
  - First proxy in any particular connection
  - But it's only $1/n$.
- Deciedly less easy to set up, but workable.
  - Even harder to find out about N proxies.
- But: remember this idea!

Older Mechanisms for Anonymity
Basic Chaum-type Mix Network: Servers

- Network of N servers, called "mixes". Each server ...
  - ... publishes a public key, PK_s.
  - ... permutes messages randomly before sending.
  - (That is, it holds on to messages for an arbitrary amount of time)
- Threat model: panopticon mostly-passive PPT adversary
  - can and will record all traffic on the entire network
  - can't DoS the entire network
  - can't invert public key cryptography

Older Mechanisms for Anonymity
Basic Chaum-type Mix Network: Clients

- Alice wants to send Bob M:
  - (Simplification: assume that Alice and Bob know each other and they found their addresses out of band.)
  - Select $n \approx 3$ servers from the network, s_i.
  - Compute a multiply encrypted message:

$$M' = E_{PK_{s_0}}\left[s_1, E_{PK_{s_1}}\left[s_2, \cdots \left[\text{Bob}, E_{PK_{\text{Bob}}}\left[M\right]\right]\right]\right]$$

  - Send this message to s_0.

Older Mechanisms for Anonymity
Basic Chaum-type Mix Network: Message Passing

- Now what?
- s_0 decrypts M' and gets

$$\left[s_1, E_{PK_{s_1}}\left[s_2, \cdots \left[s_n, E_{PK_{s_n}}\left[\text{Bob}, M\right]\right]\right]\right]$$

- So it (eventually) sends this message to s_1.
- So long as *one* of s_i are behaving, the mix works fine.
- Because the mixes shuffle messages, it's impossible to know which of its outputs corresponds to which input.

Older Mechanisms for Anonymity
Basic Chaum-type Mix Network: Uniformity

- What about metadata attacks?
  - Suppose there's only one message of 124 bytes in the entire mix?
  - What if there's no traffic on the net for longer than the mixes are willing to delay messages?
- Remember! You can't be anonymous by yourself!
  - Constant message sizes for the entire network!
  - (Fixed-rate) cover traffic between servers (ick!) or tolerate potentially infinite delays (also ick!).

Older Mechanisms for Anonymity
Basic Chaum-type Mix Network: Problems

- Doesn't work for realtime operations
  - Definitionally, mixes delay messages.
  - All that public key cryptography is really slow.
- What happens if a server fails?
- Need for uniformity of servers and messages.
- Server discovery is "unspecified"
  - Same basic problem as the proxy and proxy chains.
  - (In fairness, most implementations specify a way)

Onion Routing

Onion Routing
The Tor Network: Servers

- A combination of approaches we've seen before:
  - Mix-net like: N servers, each with published public key.
  - Proxy-like: Servers make real-time connections for clients.
  - Chain-like: Servers contact other servers as clients.
- With some new stuff tossed in
  - Use symmetric cryptography when possible (fast!)
  - Specify the One True Way to find the network.
    - Corollary: fine tuning of server properties.
- Tor aims to be a *real-time* anonymizing system
  - Nodes are more just-forward than store-and-forward.
  - This makes it useful for both bulk transfer and web browsing.

Onion Routing
The Tor Network: Basics

- Vocabulary:
  - A "onion router connection" ("orconn") is a connection to a Tor server.
  - A "circuit" is a chain of Tor servers, each connected to the next.
  - A "stream" is a flow of data over a circuit.
- Tor circuits use fixed-length cells, making traffic analysis a little harder.
  - How much harder is an open research question.

Onion Routing
The Tor Network: Onion Proxies

- The tor software can be run in a number of configurations.
- The basic no-nothing mode is as a SOCKS *proxy*.
  - So any SOCKS aware application can take advantage of Tor.
- The Onion Proxies manage the client's anonymity.
  - Runs on the client machine (or within the client's network).
  - It's paranoid, just like you want it to be.
  - It's growing more paranoid as the network develops.

Onion Routing
Circuit Building Basics: Basic idea

- Suppose the clients know a large directory of servers.
- For Alice to talk to Bob,
  1. Alice finds a server willing to talk to Bob, s^*.
     - This is called the "exit node"
  2. Alice selects some random nodes: s_i.
  3. Alice connects to one of these nodes, s_0, directly.
     - This is called the "entry node"
  4. Alice tells s_0 to connect to s_1.
  5. ...
  6. Alice tells s_{n-1} to connect to s^*.
  7. Alice tells s^* to connect to Bob.
- Let's look at this in more detail.
- We'll talk about that directory of servers later.

Onion Routing
Circuit Building Basics: Contacting the entry

- Because Alice has a server directory,
  - She knows the address *and public key* of each
- So Alice connects to the entry node
  - Using the address from the directory.
- Upon connecting, Alice demands that the node prove its identity.
  - Alice does not prove her identity.
  - This is a one-sided authenticated key exchange.

Onion Routing
Circuit Building Basics: Contacting the entry

- As a side effect of identity verification, Alice and the entry node derive a shared session secret.
  - Jargon: Tor uses TLS (SSL); the usual key exchange and shared secret derivation protocol is "authenticated Diffie-Hellman."
- Now Alice and the entry node communicate using a fast symmetric cypher for as long as they're connected.
  - This is just like https.

Onion Routing
Circuit Building Basics: Extending the Circuit

- For each intermediate hop that Alice selected,
    - Alice tells the current end of the circuit to make a connection to the next hop.
    - This repeats until the circuit reaches the exit or fails.
- These connection requests include *both* the address and the public key of the next hop.
    - This handles servers and clients having slightly different views of the network.
- Upon failure, Alice starts all over
    - Trying to build from the last node that didn't fail would allow an adversary to game where Alice's connections could go!

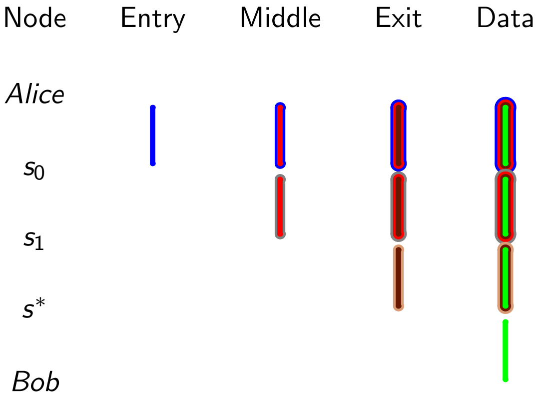
Onion Routing
Circuit Building Basics: Extending the Circuit

- Alice demands the next hop prove its identity over the circuit so far (and derives a session key).
- The previous hop and next hop strongly authenticate each other and may use this connection to carry other circuits if they believe the results.
- Jargon: Router-router connections are also encrypted with symmetric cyphers to avoid acting like decryption oracles.

Onion Routing
Circuit Building Basics: The End Is Here

- Once Alice's circuit has reached the exit,
  - Alice asks the exit to connect to Bob.
- Alice may put any number of streams on each circuit, and may leave idle circuits around.
  - Building circuits is expensive.
  - Dramatically speeds up things like web browsing.
  - Usually use one circuit per remote (Bob) address.

Onion Routing
Circuit Building Basics: Pretty Pictures



Onion Routing
Directory Authorities

- How do clients get the directories of servers?
- Simple: from directory servers.
  - Run by the project and volunteers.
  - Public keys come with the Tor source.
- Servers register themselves with all the directories
  - (they know about)
- Directories periodically get together and derive a consensus.
- Each authority signs the consensus.
- Consensuses are dated and have staged expiration.
- (Far too complex for this talk)

Onion Routing
Directory Authorities: Notes

- Publications are signed, so have routers mirror them!
- Can pretty easily find a cheating authority in this scheme.
  - Untested, though, as it hasn't been seen.
- Tor clients do not need to contact the directories directly.
  - After bootstrapping
  - Can ask around for caches
  - Only have to go to the directory if we've been gone so long that *none* of the routers we knew about respond.

Onion Routing
Exit Nodes And Middlemen

- A simple extension to the directory allows servers to publish arbitrary key/value pairs.
- One such knob is used for servers to specify their "exit policy"
    - A list of IP addresses $\otimes$ TCP ports that this node is willing to route outwards.
    - Useful to keep abuse down.
- Nodes that are unwilling to exit traffic are called "middlemen"
    - Generally quite hard to abuse middlemen.
    - From time to time, I have run one and had no complaints.

Onion Routing
Guard Nodes

- Here's a cute attack.
- Eve wants to try to see what Alice is doing over Tor.
- Eve runs (at least) two Tor routers.
- Eve waits until Alice picks her two nodes as entry and exit nodes.
- Statistical timing correlation can pretty well identify which packets flowing through both nodes are Alice's.

Onion Routing
Guard Nodes

- Tor clients solve this problem by having only a slowly rotating set of "entry guards" that it uses for all circuits.
- If Eve is in Alice's entry guard set, Alice is owned.
- BUT! If Eve isn't in Alice's entry guard set, she'll be waiting a very long time.
- Therefore, this simple countermeasure increases the resource requirements for Eve's attacks.
- Presumably beyond practicality
  - No follow-on paper saying that entry guards don't work.

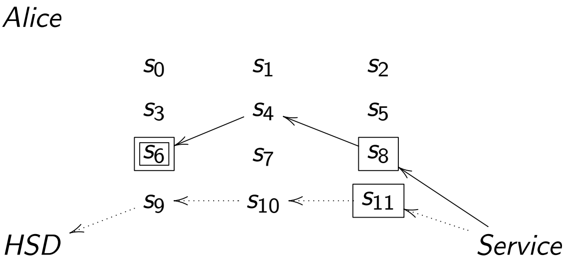
Advanced Features of Tor
Hidden Services: Anonymity for Servers, Too!

- Suppose we're not interested in using other services, but want to *host* services without people knowing where the servers are.
- We'll need:
    - Some kind of collision-avoiding, random naming scheme.
    - Another kind of directory server.
- Tor uses public key fingerprints as the name of hidden services, e.g. `http://eqt5g4fuenphqinx.onion/`.
- There are a few "hidden service directory servers" on the Tor net.

Advanced Features of Tor
Hidden Services: Anonymity for Servers, Too! : Mechanism

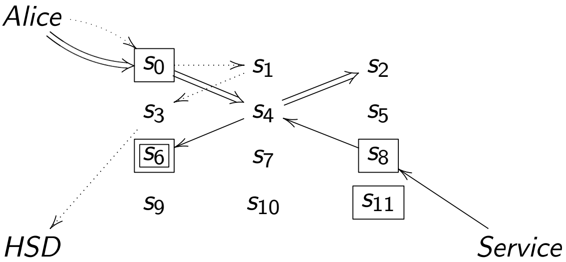
- A server
  - will pick "a few" nodes in the net as "introduction point" and build circuits to them.
  - also builds a circuit to the hidden service directories and registers itself and its introduction circuits there.
  - And waits...
- A client
  - Asks the hidden service directories (over Tor) for the introduction points.
  - picks a "rendezvous point" and builds a circuit there.
  - builds a circuit to an intro point and tells the server that it would like a connection at the rendezvous point.
  - Waits for the server to connect to it at the rendezvous

Advanced Features of Tor
Hidden Services: Anonymity for Servers, Too! : Pictures



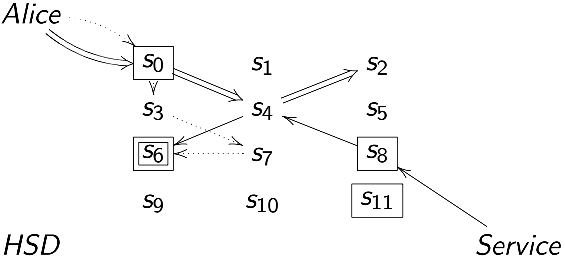
- Server picks entry guards (single box) and introduction point (double box).
- Server builds circuit to intro. point and registers.

Advanced Features of Tor
Hidden Services: Anonymity for Servers, Too!: Pictures



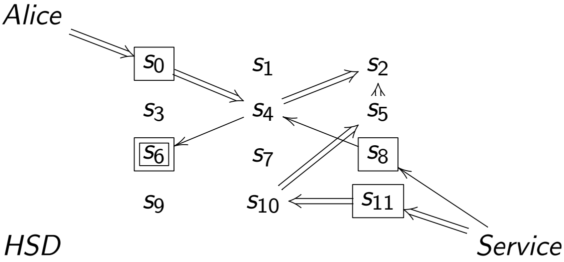
- Alice asks for service's introduction point.
- Alice also builds a rendezvous circuit and gives the end some unique token X.

Advanced Features of Tor
Hidden Services: Anonymity for Servers, Too!: Pictures



- Alice connects to the introduction point and says to the service "Tell Service (by public key) (s_2, X)"

Advanced Features of Tor
Hidden Services: Anonymity for Servers, Too!: Pictures

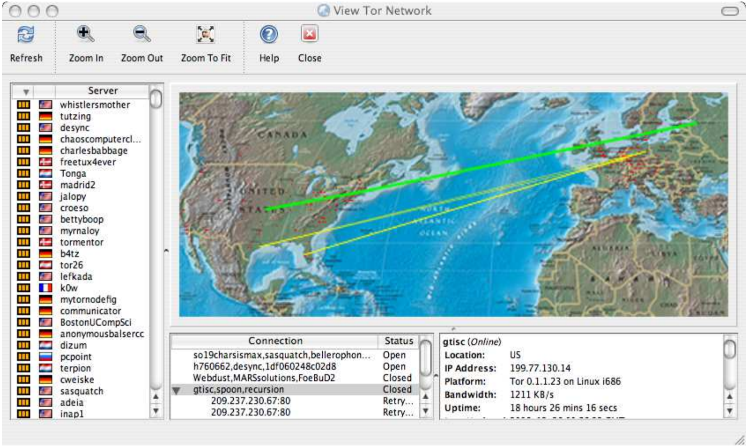


- The service builds a circuit to s_2 as requested and uses X to connect the two circuits.

Advanced Features of Tor
Tor Controllers

- If you want to experiment, having to get your hands dirty in Tor's sensitive code is probably not your idea of fun.
- Tor defines a simple, text protocol which allows other programs to see what it's doing and have some say in how it behaves.
- Controllers can range from mere observers to completely replacing the circuit selection algorithms.

Advanced Features of Tor
Tor Controllers: Vidalia



(Image credit: Softpedia)

Advanced Features of Tor
Tor Controllers: Vidalia

- Arm: Command-line UI.
- Torflow controller: Python tools for measuring Tor.
  - Includes Snakes On A Tor project for testing exits for fiddling with SSL.
- TorCtl: Python library for controllers.
- Tor::Controller: ruby module for controllers.
- jtorctl{,2}: Java

You get the idea.

Open Problems, Tradeoffs, and Gotchyas
DNS Attacks

- We wanted to hide to where we're connecting.
- But some applications are dumb:
  - `gethostbyname()` then `connect()`
  - *Even when running with a SOCKS proxy*
- Whoops, our DNS caches know where we're asking about.
  - So does anybody watching us (*that* was easy!)
- Tor will do DNS resolution for you using some exit.
- Tor warns when it's given an IP address
  - Check the controller or log messages
- Please use SOCKS4A (the "A" is for "Address") or SOCKS5.

Open Problems, Tradeoffs, and Gotchyas
TCPv4 Only

- Tor can only handle TCPv4 connections, at the moment.
  - TCPv6 is being investigated
- In fact, it uses TCP connections between routers.
- This raises a few problems:
  - Not all applications use TCP.
  - Circuits may fail with some of your data on them (cue Major Tom).
- There is thought and the beginnnings of a “modular transport” layer for Tor.

Open Problems, Tradeoffs, and Gotchyas
Need for Protocol Scrubbing Proxies

- Everybody's favorite protocol, HTTP, leaks information to the server like crazy.
  - Headers: Cookies, Referer, User-Agent, X-Proxied-For, Languages, ...
  - Java, (in)ActiveX, Javascript, Flash, ...
  - iframes can try to see "around" Tor if your setup is buggy
- So if you're going to browse the web, you need a good HTTP "sanitizer"
  - Fortunately, privoxy (nee "Internet Junk-Buster") exists.
  - Also polipo, for the more adventerous.

Open Problems, Tradeoffs, and Gotchyas

- It's worse: need such a santizer for *every application protocol* that runs over Tor.
  - SMTP HELO/EHLO include your host name.
  - FTP PORT/EPRT commands (anybody remember these?) include your IP.
  - BitTorrent (etc.) can tell the tracker your IP.
  - IRC DCCP messages can include IP addresses.
  - ...
- HTTPS/SSL/TLS/... prevents sanitizing proxies from running client-side!

Open Problems, Tradeoffs, and Gotchyas
Exit Nodes Are Watching You Masturbate

- If your bits aren't private (and http isn't), the exit nodes can watch your traffic.
    - So don't read private LJ entries over Tor.
    - More impressively, don't emulate the Sweedish embassy and send passwords in the clear.
- More dramatic: they can *modify* your traffic in flight.
    - Replace all images with `hello.jpg`?
- If your bits *are trying to be* private, exit nodes are men in the middle and are free to attack your key exchange.
    - Check those SSL certs *carefully*!
- There is some effort to spot-check exits in Tor.

Open Problems, Tradeoffs, and Gotchyas
Abuse

- (John Gabriel's Greater Internet Fuckwad Theory).
- There are people using Tor to abuse sites.
  - So most exit nodes get IP-banned quickly.
  - (It's a dumb solution, but)
- Germany for a while seemed not to understand the idea of "I let other people use my connection"
  - Apparently checking public lists of exit nodes is too hard for some people.
- May be easier, ironically, for a large institution to run an exit node.

Open Problems, Tradeoffs, and Gotchyas
Blocking Tor But Doing It Right

- Freenode, for example, may block exit nodes, but runs two Tor hidden services:
    - A truly anonymous one which goes up and down with abuse.
    - An authenticated (by password and GPG email) one which stays up all the time.

Open Problems, Tradeoffs, and Gotchyas
What if It's Illegal To Use Tor?

- Directories imply that you, and your worst enemy, both know the full list of Tor routers.
- Leaking through Tor's cleverness is *the fact that you're using Tor!*
- So the government runs a router or watches the network or ...
- And when you connect, the KGB kicks down your door.
  - Less evilly, The Man just blackholes the routers

Open Problems, Tradeoffs, and Gotchyas
What if It's Illegal To Use Tor?: Not A Hypothetical

Specific attacks on Tor:

- April 2006, Thailand: DNS filter Tor's webpage.
- 2006 Smartfilter/Websense (and Cisco): block URLs with /tor/. . . (broke unencrypted directory fetches; no longer a problem)
- 2007-2009, Iran: deploys Websense.
- 2009, Iran: throttles all SSL everywhere (caught Tor because TLS handshake looks like FF+Apache)
- 2009, Tunisia: Smartfilter; only 80 and maybe 443, if they like you.

Open Problems, Tradeoffs, and Gotchyas

What if It's Illegal To Use Tor?: Not A Hypothetical

Specific attacks on Tor, continued (or: Enter China):

- 2009, China: Blocks public relays, enumerates 1/3rd of bridges.
- 2010, China: Whoops, make that 2/3rds of bridges.
- 2011, Iran: Blocks Tor by DPI for SSL DH parameter.
- 2011, Syria: DPI Tor SSL.
- 2011, Iran: DPI SSL certificate lifetime
- 2011, China: Actively probes SSL endpoints for Tor protocol!
    - That puts a crimp on Tor's style!
    - Still possible to connect, using obfuscation.
    - Obfuscation still "research" despite being deployed.

Open Problems, Tradeoffs, and Gotchyas
Connecting to Tor privately : Bridges

- A second tier of routers ("bridges") that *do not* publish themselves in the general directory.
  - To be used as entry, not exit, nodes.
- Instead, they (may) publish themselves in special bridge directories that do not reveal the whole list to anybody.
  - Send `get bridges` to `bridges@bridges.torproject.org` from a gmail account.
  - Visit `https://bridges.torproject.org/`
- This makes the adversary's life much harder (though. . .)
- (Still concerns about accidentally picking an adversary-controlled bridge.)

Open Problems, Tradeoffs, and Gotchyas
Connecting to Tor privately : Bridges

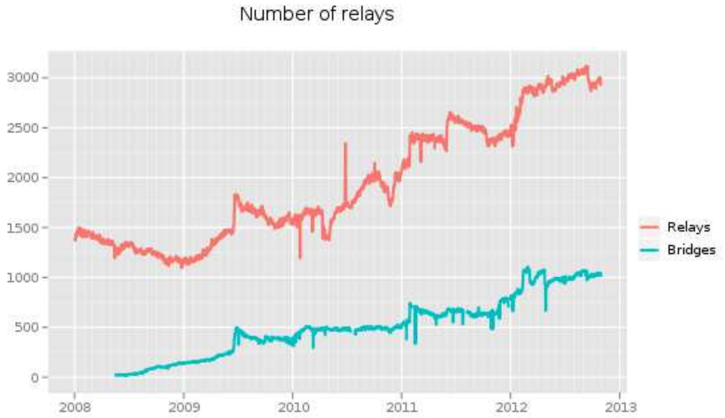
- Bridges are "relatively" new
- There's still plenty of work to be done here...
  - More and harder-to-enumerate distribution channels.
  - Metrics for detecting when and where bridges are blocked.
- As mentioned earlier, "modular transport" and "obfuscation"

Network Statistics

- Network running, without downtime, since October 2003.
- Across major revisions of the protocol.
- No directory servers compromised.
- Whitehats seem to be winning at the moment
  - No evidence of unmasking users except papers which resulted in bug fixes.
  - (That's just what they want you to think?)
- Up to 400K users/day these days?

Network Statistics

- Recent years have seen more emphasis on measuring the network for performance issues (several papers, even).
- `https://metrics.torproject.org/`



Links To More
https://www.torproject.org

- Tor Browser Bundle
- Source tree via git
- Anonbib, maintained by Roger Dingledine, is full of awesome:
  - `http://freehaven.net/anonbib/topic.html`