Privacy
Privacy in Computing

Location privacy
Anonymous web surfing
Data loss prevention
Data mining privacy
LOCATION Privacy
Location Privacy

- Pervasive use of GPS-enabled mobile devices allows location tracking
  - E.g., Foursquare lets users "check in" to a place when they're there,
    - tell friends where they are and track the history of where they've been
  - Over-sharing may create privacy risk; others may ask:
    - Did you see an AIDS counselor?
    - Did you go to an anti-war rally on Tuesday?
  - http://pleaserobme.com/
    - a website that shows a list of people who are currently not at home
Raising awareness about over-sharing

Check out our guest blog post on the CDT website.

Check your own Twitter timeline for checkins

Are you curious if people can see your checkins?
Enter your Twitter username and find out.

Your Twitter username

Find!

More Info
Home
Why

Made Possible By
Foursquare
Twitter
@boyvanamstel
@frankkoeneveld
GPS Location

- Service providers want your GPS for location-based services
- How about sharing distance?

Triangulation: 3 distances measures $\rightarrow$ exact location

Distance from a third known point C reveals that you must be at location 2
GPS Location

- Service providers want your GPS for location-based services
- How about sharing distance?

![Diagram showing triangulation using three points to determine location.](image)

Distance from a third known point C reveals that you must be at location 2.
A more Concrete Example (My Early Work)

- 50M active users
- Real-time traffic update using millions of users’ locations

User reported events
- Accidents, police trap, etc.
- Alert users of nearby events

Social features
- See nearby users on the map
- Say “hi”/msg nearby users
Threat of Sybil Devices

- Sybil devices
  - Software scripts emulating as real devices
  - Allowing a single user to control many devices

- In the context of Waze (popular navigation app)
  - Creating a large number of Sybil devices with low costs
  - Attacks: injecting fake events, user location tracking
  - Generalizable to other mobile communities
Creating Sybil Devices

- Naïve approach: mobile emulators
  - Not scalable: ~10 emulators per PC

- Our way: emulate a mobile client using scripts
  - Server communicates with client via limited APIs
  - Mimic API calls to replace full client

We can create 10,000 Sybil devices on a single PC
Attack #1: Polluting Waze Database

- Fake road-side events.
  - Any type of event at any location
  - Potentially affect 1 billion Google Maps users

- Fake traffic hotspots
  - Simulate cars driving slowly
  - Large groups of Sybil devices to overwhelm normal users' data

Users are re-routed
**Attack #2: User Location Tracking**

- Follow (stalk) any Waze user in real-time
  - Waze marks nearby users on the map

- Pinpoint to **exact GPS location**
  - Specific hotels, gas stations, etc.

- Remain **invisible**
  - Move in and out quickly

- Track users in the **background**
  - Waze uploads GPS in the background

- Track users across days
  - Use creation time as GUID
A Tracking Example

Experiments

- Two highly challenging tracking scenarios
  - High-way 101 (high traveling speed)
  - LA downtown (high nearby user density)
- 40+ mins driving, both cases were successful
Web Privacy
Cookies based tracking of your web behaviors

- Service providers want your information
  - For selling customer demographics to marketers
  - How much private data is worth?

- A user usually uses the same username (e.g., email address) for many different services

- Cookie-based tracking and cookie syncing
Spyware

- Spyware – code designed to spy on a user, collecting data

- KaZaa (a P2P file-sharing app) bundled with Altnet program

- If agreed, Altnet share your unused computing power/storage

- But it is difficult to control what Altnet is sharing

- E.g., tax returns of users found on the Internet 2006
Anonymous Web Surfing

- Slides credits to Vitaly Shmatikov
Anonymous Surfing

- Alice’s goal: communicate to server without letting others know

- Privacy requirements:
  1. Routers on the path from her to server learn nothing
  2. Bob who can eavesdrop on packets learns nothing
  3. Server does not know her IP
Anonymous Surfing

- Alice’s goal: communicate to server without letting others know

Routers may be honest-but-curious (semi-honest) where they follow protocols, but may collect data and attempt to find more information.
Why Your IP Matters?

Your IP address is Your ID.

The Most Advanced & Secure Internet Privacy Service

PROBLEM
- Consumer Profiling
- Cybercrime
- Personal Data Exposed
- Identity Theft

SOLUTION
- Anonymous Surfing
- Protected Wi-Fi
- Encrypted Virtual Private Networks (VPN)
- Untraceable IP Rotation

Are you a Home User? Are you a Business? Customized Solutions for

SIGN UP NOW! $79.99 1 Year Subscription

EASY TO INSTALL • EASY TO USE

Promotional Code: [Code]

Privacy Policy
Terms of Service
FAQ
Customer Support
A simple solution - VPN to change your IP
How to relax our trust assumption on VPN server?

- VPN prevents service providers from knowing the client’s IP
  - Your real IP is hided away

- User has to trust the VPN server – not ideal

- Our goal:
  - to prevent the adversary from learning whom we talk to

- Attack model:
  - Adversary may monitor all traffic, control some intermediary proxies (such as VPN server)
Basic idea for anonymous two-way communication: hide your message in a crowd

- Basic idea: use semi-honest proxies to permute your and others’ messages, decrypt, and resend
- Your computer also mixes packets for others
- More messages, better privacy
- More participants, better privacy
Mix network [David Chaum ‘80]

- Alice wants to send a message to Bob -- a third person should not find out who the sender or recipient is

Alice wants to send a message to Bob:

1. Encrypts her message to Bob with Bob’s key
2. Encrypts it 3 more times with key a, b, and c, respectively

Proxies (or resenders) assumed to follow the protocol (honest-but-curious)

A single eavesdropper or proxy does not learn who the sender or receiver is

Figure from F.W.J. van Geelkerken -http://www.iusmentis.com/
Mix network pros and cons

- **Pros**
  - Effective if proxies process and permute enough messages at a given time or during a period
  - Good for email (not time-sensitive application)

- **Cons**
  - The use public key crypto makes it slow for real-time communication, e.g., web surfing
The Onion Router (TOR)

- Thousands of relays simultaneously
- > 200,000 active users per week

Typically 3 nodes used for each route, called TOR onion routers
Privacy and anonymity continued

![Number of relays](https://metrics.torproject.org/)

The Tor Project - [https://metrics.torproject.org/](https://metrics.torproject.org/)
Alice wants to communicate with Bob

How Tor Works: 1

Step 1: Alice's Tor client obtains a list of Tor nodes from a directory server.
Alice wants to communicate with Bob

Step 2: Alice’s Tor client picks a random path to destination server. **Green** links are encrypted, **red** links are in the clear.
Alice wants to communicate with Bob

Step 3: If at a later time, the user visits another site, Alice's Tor client selects a second random path. Again, green links are encrypted, red links are in the clear.
How exactly does it work?

- **Alice knows**
  - Self, node1, node2, node3, Bob
- **Node1 knows**
  - Alice, node2
- **Node2 knows**
  - node1, node3
- **Node3 knows**
  - node2, Bob
- **Bob knows**
  - node3
Details: Alice -- constructing a circuit

Alice constructs circuits incrementally, one hop at at time

1) Alice ↔ node1: Diffie-Hellman to exchange a session key (symmetric)

2) node1 extends the circuit to node2
   - Node1 ↔ node2: Diffie-Hellman to exchange a session key (symmetric)
   - After that, Alice has a 2-hop circuit with a shared session key with node1 and node2

3) node2 extends the circuit to node3
   - After that, Alice has a 3-hop circuit who all share the same session key

4) Alice use the circuit to talk to Bob
   - The last hop is unencrypted (node3 → Bob)

Using a circuitID to recognize the circuit
Details: Bob - reply the message

- Bob: reply to node3 (unencrypted)
- Node3 → node2 (encrypted once using the shared session key)
- Node2 → node1 (encrypted once using the shared session key)
- Node1 → Alice (encrypted once using the shared session key)
- Alice remove the encrypted layers using the shared session key
  - Get the reply messaged from Bob
Use 2 nodes as an example, can be easily extended to 3 nodes
TOR changes the route periodically (e.g., every 10 minutes) to avoid traffic analysis
Some security analysis

- Each TOR node routes messages for many hosts
- It is difficult to keep track of how messages are routed within TOR network
  - Assuming majority of TOR nodes are not corrupted or collude
- However, there are some issues in a stronger adversary model:
  - All TOR nodes are semi-honest, i.e., follow the protocols but curious about the data passing through.
Attacker may know who initiates or receives the traffic

- Attacker knows Alice starts some communication.
- Entry node knows Alice starts some communication.
- Exit node knows Bob is the receiver.
Solution: Alice and Bob become TOR nodes as well

How Tor works: 4

Nine Tor nodes and 4 users / Tor nodes

A: Alice connects to Bob - B: Bob connects to Dave
J: Jane connects to Alice - D: Dave connects to Jane
Still need to use end-to-end encryption in onion routing

- TOR Used to Snatch Embassy Passwords in 2007
  - A Swedish security researcher used TOR and sniffed 100 embassy, government and Fortune 500 e-mail passwords
- Cause: unencrypted email
- Not a TOR vulnerability

Need both confidentiality and privacy – not the same

http://www.eweek.com/c/a/Security/ToR-Used-to-Snatch-Embassy-Passwords/
TOR location hidden service

Alice can connect to Bob's server without knowing where it is

Server needs to
- Be accessible from anywhere
- Resist censorship
- Require minimal redundancy for resilience in DoS attack
- Can survive to provide selected service even during full blown DDoS
- Resistant to physical attack (you can't find them)

How is this possible?
Basic ideas of hidden services

- Use an intermediary to marry client and server
- Similar ideas used in Skype as well
  - For users behind NAT (network address translator)
Location Hidden Service

1. Server Bob creates onion routes to **Introduction Point (IP)**
2. Bob gets **Service Descriptor** to include introduction point
3. Alice obtains Service Descriptor at a Lookup Server
Location Hidden Service

4. Client Alice creates onion route **Rendezvous Point (RP)**
5. Alice sends **RP** address and any authorization through Intro Point to Bob
6. If Bob chooses to talk to Alice, connect to Rendezvous Point
7. Rendezvous point mate the circuits from Alice to Bob

RP simply relays (end-to-end encrypted) messages from client to service and vice versa.

Does the RP know where Bob is (i.e., Bob’s IP address)?
Silk Road Shutdown

- Ross Ulbricht, alleged operator of the Silk Road Marketplace, arrested by the FBI on Oct 1, 2013
How Was Silk Road Located?

- FBI agent Tarbell’s testimony:
  - Agents examined the headers of IP packets as they interacted with the Silk Road’s login screen, noticed an IP address not associated with any Tor nodes
  - As they typed this address into the browser, Silk Road’s CAPTCHA prompt appeared
  - Address led to rented server in a data center in Iceland

- Common problem: misconfigured software does not send all traffic via Tor, leaks IP address
  - Is this really what happened with the Silk Road server?
Attacks on Tor
Main (?) Tor Problem

Traffic correlation and confirmation
Traffic Confirmation Techniques

- Congestion and denial-of-service attacks
  - Attack a Tor relay, see if circuit slows down
- Throughput attacks
- Latency leaks
- Website fingerprinting
A realistic model of Tor adversaries needs to incorporate:

- Autonomous systems and Internet exchange points
- Evolution of Internet topology over time
- Traffic generated by typical applications over time

[Johnson et al. “Users Get Routed”. CCS 2013]
Node Adversaries
TorPS: The Tor Path Simulator

- Realistic client software model based on the current Tor
- Reimplemented path selection in Python
- Major path selection features:
  - Bandwidth weighting
  - Exit policies
  - Guards and guard rotation
  - Hibernation
  - /16 and family conflicts
**TorPS: The Tor Path Simulator**

- **Network Model**
- **User Model**
- **Streams**
- **Client Software Model**
- **Relay statuses**
- **Stream ➔ Circuit mappings**
**TorPS: User Model**

- Gmail/GChat
- Gcal/GDocs
- Facebook
- Web search
- IRC
- BitTorrent

**Typical Session Schedule**
- One session at 9:00, 12:00, 15:00, and 18:00 (Su-Sa)

**Worst Port**
- (6523)

**Best Port**
- (443)

**20-minute traces**

**Repeated Sessions**
- 8:00-17:00, M-F
- 0:00-6:00, Sa-Su
**TorPS: The Tor Path Simulator**

Network Model

Streams

User Model → Tor → Client Software Model

Relay statuses

Stream → Circuit mappings
TorPS: The Tor Path Simulator

Network Model

metrics.torproject.org

Hourly consensuses

Monthly server descriptors archive
TorPS: The Tor Path Simulator

Network Model

User Model

Streams

Client Software Model

Relay statuses

Stream ➔ Circuit mappings
Node Adversary Results

Time to first compromised circuit

![Graph showing cumulative probability over days from first stream at different bandwidth rates.](chart.png)
Not a Theoretical Threat!

- Sybil attack + traffic confirmation
- In 2014, two CMU CERT “researchers” added 115 fast relays to the Tor network
  - Accounted for about 6.4% of available guards
  - Because of Tor’s guard selection algorithm, these relays became entry guards for a significant chunk of users over their five months of operation
- The attackers then used these relays to stage a traffic confirmation attack
Fighting Internet Censorship

- Key use of anonymity networks – circumventing Internet censorship
Using Tor for Circumvention

Easily recognizable at the network level

Deep packet inspection (DPI)

“Classic” Tor may not be effective anymore!
Let’s Play Hide-and-Seek

For example, make this look like a Skype connection
Goal: Unobservability

Censors should not be able to identify circumvention traffic, clients, or servers through passive, active, or proactive techniques.
Unobservability by Imitation

- “Parrot systems”
- Imitate a popular protocol like Skype or HTTP
  - SkypeMorph (CCS 2012)
  - StegoTorus (CCS 2012)
  - CensorSpoofer (CCS 2012)
SkypeMorph

Censorship region

The Internet

SkypeMorph client

Traffic shaping

SkypeMorph bridge

A Tor node
Missing Control Channels

Censorship region

The Internet

TCP control

SkypeMorph client

SkypeMorph bridge

A Tor node
Let’s imitate the missing parts!

- Problem: hard to mimic *dynamic behavior*
  - Active and proactive tests
Dropping UDP Packets

![Graph showing the number of UDP packets dropped over time with and without available bandwidth changes. The graph includes bars for packets dropped per 10 seconds, with red dashed lines indicating changes in available bandwidth. The x-axis represents time in seconds, ranging from 250 to 650, and the y-axis represents the number of packets, ranging from 0 to 10.]
### Other Tests

<table>
<thead>
<tr>
<th>Test</th>
<th>Skype</th>
<th>SkypeMorph+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flush Supernode cache</td>
<td>Serves as a SN</td>
<td>Rejects all Skype messages</td>
</tr>
<tr>
<td>Drop UDP packets</td>
<td>Burst of packets in TCP control</td>
<td>No reaction</td>
</tr>
<tr>
<td>Close TCP channel</td>
<td>Ends the UDP stream</td>
<td>No reaction</td>
</tr>
<tr>
<td>Delay TCP packets</td>
<td>Reacts depending on the type of message</td>
<td>No reaction</td>
</tr>
<tr>
<td>Close TCP connection to a SN</td>
<td>Initiates UDP probes</td>
<td>No reaction</td>
</tr>
<tr>
<td>Block the default TCP port</td>
<td>Connects to TCP ports 80 and 443</td>
<td>No reaction</td>
</tr>
</tbody>
</table>
StegoTorus

Censorship region

The Internet

StegoTorus client

HTTP

HTTP

HTTP

Skype

StegoTorus bridge

Ventrilo

HTTP

A Tor node
StegoTorus Chopper

- Dependencies between links

![Graph showing packet rate over time with indication of starting to drop the blue flow.](image)
**StegoTorus-HTTP**

- Does not look like any HTTP server!
- Most HTTP methods not supported!

<table>
<thead>
<tr>
<th>HTTP request</th>
<th>Real HTTP server</th>
<th>StegoTorus’s HTTP module</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET existing</td>
<td>Returns “200 OK” and sets Connection to keep-alive</td>
<td>Arbitrarily sets Connection to either keep-alive or Close</td>
</tr>
<tr>
<td>GET long request</td>
<td>Returns “404 Not Found” since URI does not exist</td>
<td>No response</td>
</tr>
<tr>
<td>GET non-existing</td>
<td>Returns “404 Not Found”</td>
<td>Returns “200 OK”</td>
</tr>
<tr>
<td>GET wrong protocol</td>
<td>Most servers produce an error message, e.g., “400 Bad Request”</td>
<td>No response</td>
</tr>
<tr>
<td>HEAD existing</td>
<td>Returns the common HTTP headers</td>
<td>No response</td>
</tr>
<tr>
<td>OPTIONS common</td>
<td>Returns the supported methods in the Allow line</td>
<td>No response</td>
</tr>
<tr>
<td>DELETE existing</td>
<td>Most servers have this method not activated and produce an error message</td>
<td>No response</td>
</tr>
<tr>
<td>TEST method</td>
<td>Returns an error message, e.g., “405 Method Not Allowed” and sets Connection=Close</td>
<td>No response</td>
</tr>
<tr>
<td>Attack request</td>
<td>Returns an error message, e.g., “404 Not Found”</td>
<td>No response</td>
</tr>
</tbody>
</table>
Unobservability by imitation is fundamentally flawed!
Imitating a Real System Is Hard

- A complex protocol in it entirety
- Inter-dependent sub-protocols with complex, dynamic behavior
- Bugs in specific versions of the software
- User behavior

Not enough to mimic a "protocol," need to mimic a specific implementation with all its quirks
Lesson #2

Partial imitation is worse than no imitation

Bad imitation of Skype is easier to recognize than Tor
This is an ex-parrot!
This parrot is no more
This is a late parrot
It’s stone dead