Tor Hidden Services

Privacy Enhancing Technologies

Philipp Winter
4096R/2D081E16

June 8, 2012

Introduction to Tor

TorProject.org

What is it?

- Tor is a **low-latency** anonymity network (as opposed to high-latency networks, such as mix networks) consisting of thousands of relays
- The **most widely used** and deployed anonymity network
- Client bundles available for Linux, Windows, Mac and Android

How Does it Work?

- Tor implements 3rd (sometimes called 2nd) generation **onion routing**
- Clients build **circuits** consisting of **relays** and route TCP streams through them
- Relays are listed in **consensus** which is published by **directory authorities**
- Directory authorities and their keys are **hard-coded** into the Tor binaries

What Does an Attacker See?

https://www.eff.org/pages/tor-and-https

Facts

As of June 2012, approximately...

- **450.000** daily users
- **3000** relays contributed by volunteers
- **1000** bridges also contributed by volunteers
- Rough statistics available at: https://metrics.torproject.org

Try it!

- All that is needed: Tor Browser Bundle
- Zero-install, zero-configuration Tor bundle
- Contains Firefox without all the privacy assaults
- Vidalia, the GUI, allows the configuration of **hidden services** and a **bridge**

Hidden Services

In a Nutshell

- Tor's purpose is to provide **sender anonymity**
- Hidden services add **responder anonymity**
- That way, we can run a TCP service without revealing our IP address!
- Therefore: **Anonymous** clients can communicate with **anonymous** servers!
- In addition: **DoS** and **censorship** protection

How it is Used in Practice

- Whistleblowing websites need **censorship resistance** against mad governments
- Activist sites need to stay **anonymous** to resist against data center raids
- Resistance against **social graph analysis** (possible with data retention)

Hidden Services by Example: Bob

- Bob is a **journalist** who wants to publish **sensitive information**
- He wants to publish his articles **anonymously** and without getting **censored**
- So Bob decides to set up a **hidden service** (HS) in the Tor network
- There are 6 steps ranging from announcing the HS to using it

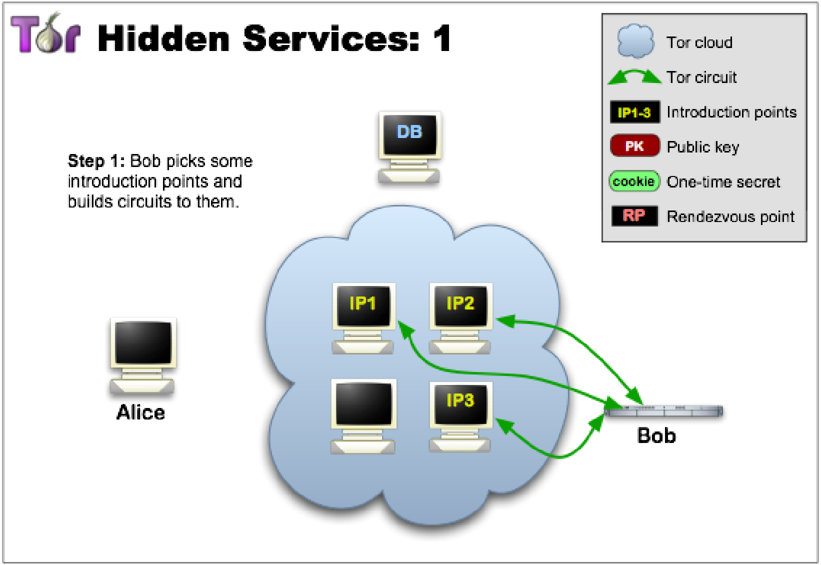
Step 0: Installation and Configuration

- Before Bob starts using Tor, he has to **install** the service
- So Bob sets up his own lighttpd **web server** which is **not** accessible over the Internet
- Also, Bob downloads the Tor binary and **configures** the hidden service

Step 1: Announcing Existance

- Bob's HS needs to **advertise** its existance in the Tor network
- The HS randomly picks **relays** , so called **introduction points** , in the network and establishes **circuits** to them
- Then, the HS asks these relays to act as introduction points by giving them its **public key**

Step 1: Announcing Existance



Step 1: Bob picks some introduction points and builds circuits to them.

Step 2: Upload of Hidden Service Descriptor

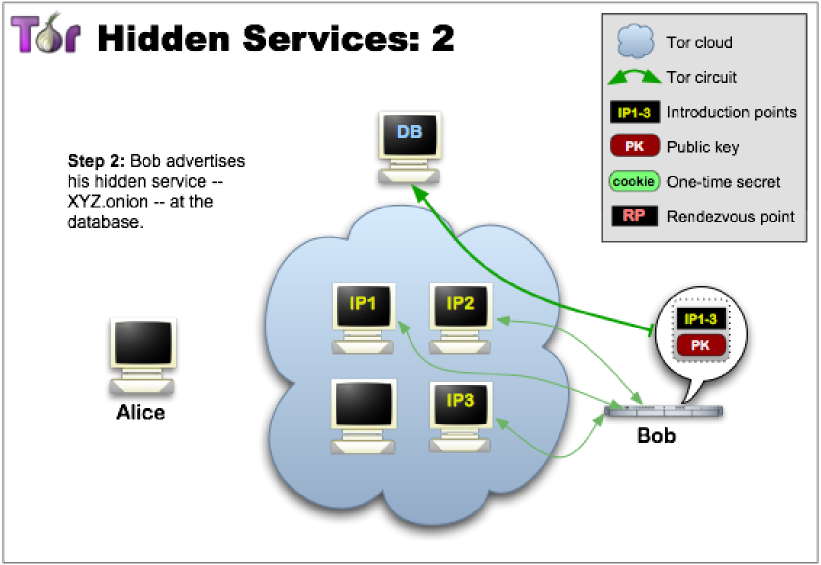
- Now, a **hidden service descriptor** must be built
- The descriptor maps the **name** of a HS to its **reachability** information
- It is uploaded to the **directory servers**
- Clients reach the HS by accessing KEY.onion where KEY (i.e. the name) is derived from the HSes public key
- Now, the HS is **set up** and ready to receive connections!

$$descriptor \mapsto (PK_{hs}, IP_1, IP_2, ..., IP_n)_{Sig_{PK_{hs}}}$$

Sample Onion Addresses

- http://idnxcnkne4qt76tg.onion/ — The Tor Project web site
- http://xqz3u5drneuzhaeo.onion/ — InspecTor
- http://eqt5g4fuenphqinx.onion/ — core.onion
- http://ci3hn2uzjw2wby3z.onion/ — Anonymous posting board

Step 2: Upload of Hidden Service Descriptor

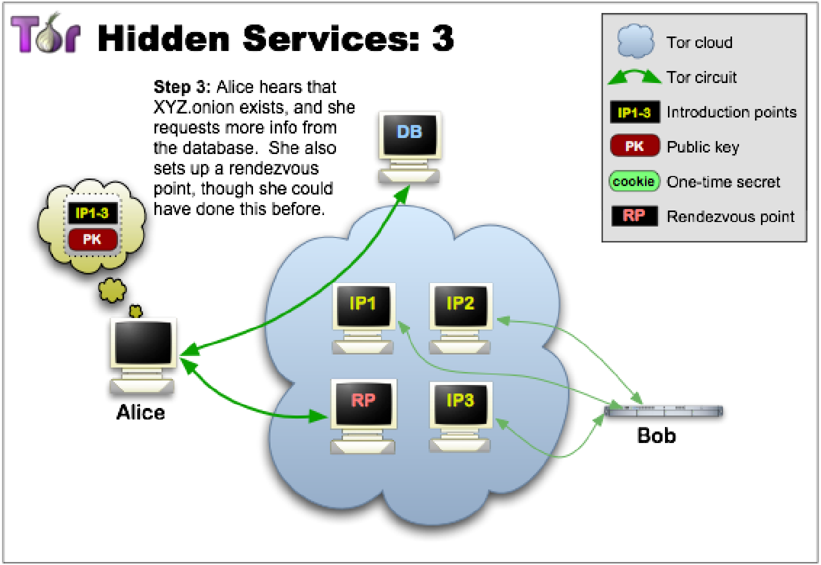


Step 2: Bob advertises his hidden service -- XYZ.onion -- at the database.

Step 3: Alice Prepares a Connection

- Alice now wants to **connect** to Bob's HS to read his **articles**
- Alice somehow learns about the onion address ynjeqmhe5j5tnzph.onion
- Alice's client **downloads the descriptor** from the directory authorities
- That way she obtained the **public key** and the **introductory points** !
- Finally, Alice randomly picks a **rendezvous point** and sends a one-time secret to it

Step 3: Alice Prepares a Connection



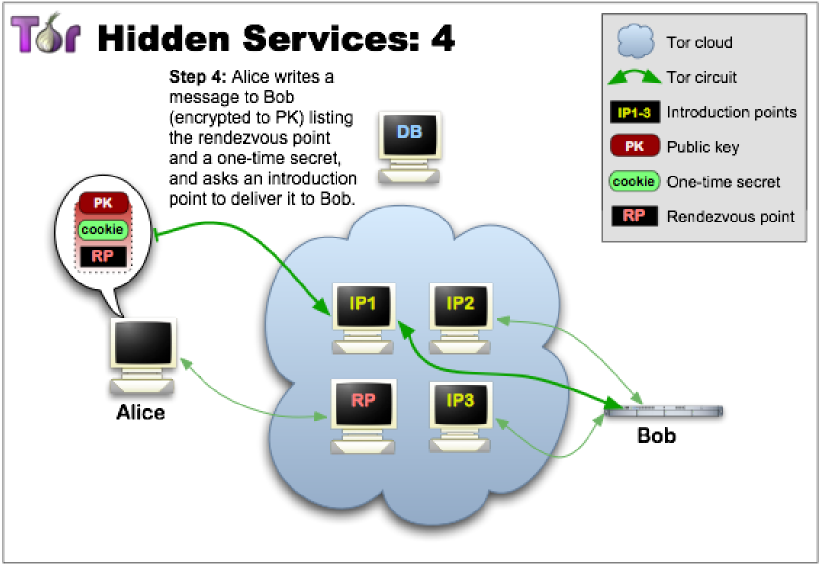
Hidden Services: 3

Step 3: Alice hears that XYZ.onion exists, and she requests more info from the database. She also sets up a rendezvous point, though she could have done this before.

Step 4: Alice Informs the Hidden Service

- Now Alice's client prepares an **introduce** message encrypted with the HSes public key
- The message contains the **address** of the **rendezvous point** and a **one-time secret**
- Alice sends this message to one of the HSes **introductory points** and they **forward** it to the HS
- Alice does all this over a Tor circuit so she remains **anonymous**

Step 4: Alice Informs the Hidden Service

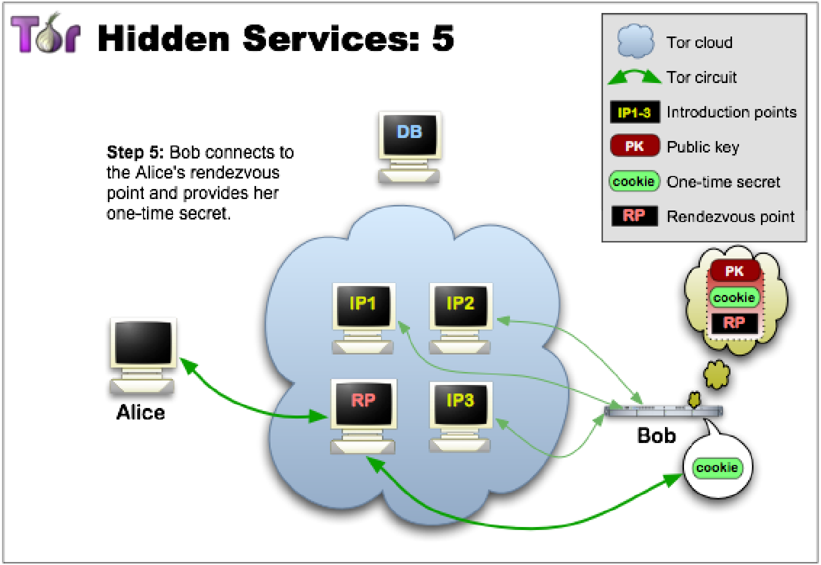


Step 4: Alice writes a message to Bob (encrypted to PK) listing the rendezvous point and a one-time secret, and asks an introduction point to deliver it to Bob.

Step 5: The Hidden Service Prepares a Connection

- The HS decrypts Alice's introduce message and obtains the **rendezvous point's address** as well as the **one-time secret**
- The HS creates a **circuit** to the rendezvous point and sends the **secret** to it

Step 5: The Hidden Service Prepares a Connection



Hidden Services: 5

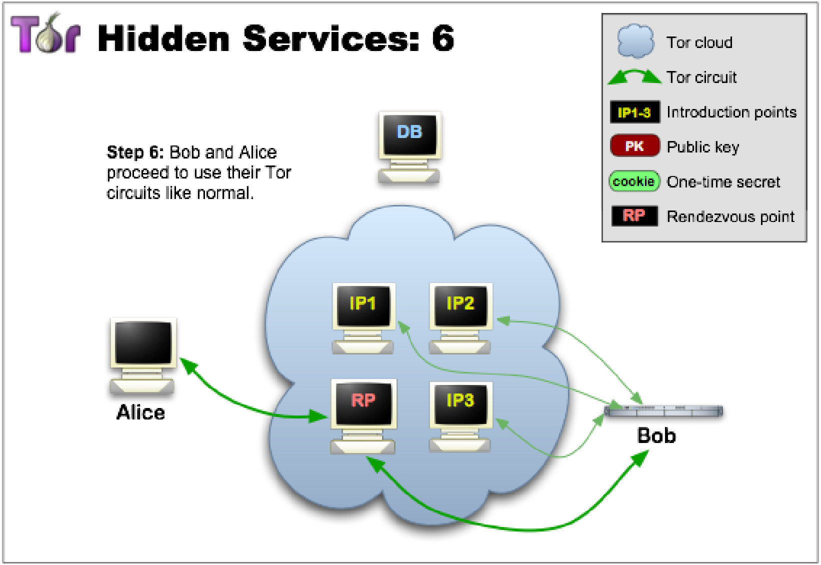
Step 5: Bob connects to the Alice's rendezvous point and provides her one-time secret.

Step 6: The Connection is Established

- Finally, the rendezvous point **notifies** Alice of the successful connection

- The rendezvous point now simply **forwards** data between Alice and the HS

Step 6: The Connection is Established



Step 6: Bob and Alice proceed to use their Tor circuits like normal.

Why Rendezvous Points?

- Rendezvous points only forward **connection information** and no actual traffic
- So they don't seem to be "responsible" for a hidden service
- Also, the traffic load could become **too high** if they would also forward traffic

What the Involved Parties Know

The Client...

- Does not know the location of the HS
- Knows the location of the rendezvous point

The rendezvous point...

- Does not know the location of both, the HS and the client
- Knows nothing about the nature of the HS or the data being transfered

The hidden service...

- Does not know the location of the client
- Knows the location of the rendezvous point

Accessing Hidden Services Without Tor

- The Tor2Web project provides access over the **plain web**
- To access Bob's articles, Alice can invoke ynjeqmhe5j5tnzph.tor2web.org
- Note that the **sender anonymity** is not the same as when accessed over Tor!
- Tor2Web trades off **security** for **convenience**

A More Practical Point of View

How Bob operates his HS...

- Bob runs lighttpd which is listening to localhost:80 and is hence **unreachable** to the wide Internet
- lighttpd is **not aware** of the fact that it is used as hidden service!
- The Tor process running on the same machine is accepting connections to the HS and **forwards** them to localhost:80
- The client application can **also** be **unaware** of Tor if it is used together with torsocks (e.g. `torsocks ssh u73zzkakuscok7zq.onion`)
- So client and server could be communicating completely **anonymous** over Tor without even **knowing**

Attacks on Hidden Services

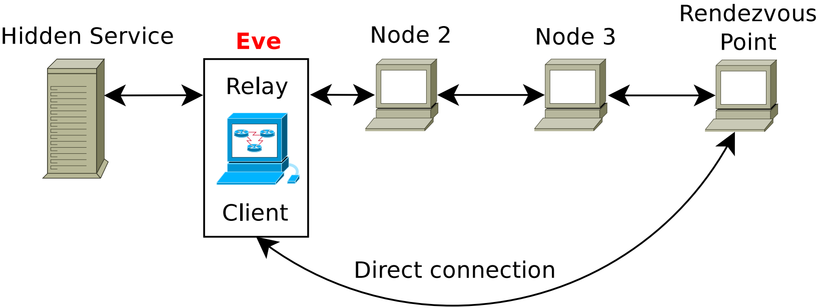
First Attack: Øverlier & Syverson

- In 2006, Øverlier and Syverson demonstrated how the **location** (i.e. IP address) of a HS can be **revealed**
- Attacker only needed a Tor **client** and a **relay** (trivial requirements) and the attack could work within minutes
- **Core vulnerability** : HS chose relays for its circuit at **random**
- **Goal of attacker** : Get chosen by HS as the **first hop** in the circuit

Øverlier & Syverson: How it Works in Practice

- Eve uses her Tor **client** to connect to the HS and she also runs a **relay**
- Eve continuously establishes connections to the HS and checks every time whether her relay was selected as first hop in the circuit $HS \rightarrow RP$
- As soon as her relay was chosen by the HS as first hop, she has the IP address!
- She can confirm whether her relay was selected by doing **traffic pattern analysis** using statistics
- **Solution** : Guard nodes for HSes

Øverlier & Syverson: Visualized



Second Attack: Murdoch

First we have to know...

- Computing devices have a so called **clock skew**, the ratio between the computer's actual and the nominal clock frequency
- So after x days, a computer's clock drifted off by y milliseconds
- Clock skew is a **very small** value but can even be **measured** over a network
- Computer's (even identical models) have **different** clock skews because the manufactoring process is not perfectly accurate → the clock skew can be seen as a **hardware fingerprint**

Second Attack: Murdoch

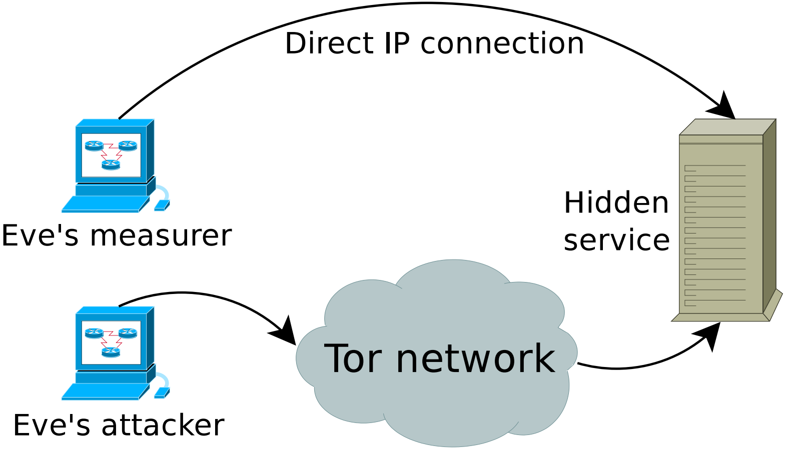
Clock skew and CPU load...

- Clock skew **changes** with temperature of the CPU (differences in 1–1.5°C are already measurable)
- The CPU's temperature can be influenced by controlling the **load**
- High load can be induced remotely by making the HS busy (e.g. fetching many websites)

Murdoch: How it Works in Practice

- Eve **suspects** several IP addresses to be the HS she wants to deanonymize
- She sends alternating traffic bursts through Tor to the HS and **measures** the clock skew of the suspected IPs (directly and not over Tor)
- Using **correlation techniques** , she can identify the HS if the IP addresses was in the set of suspects

Murdoch: Visualized



Conclusions

What You Should Keep in Mind

- HSes provide **responder anonymity** as well as **DoS** and **censorship protection**
- HSes can (and should) be accessed over **Tor** but they are also accessible over the **web**
- HSes are fairly **flexible** and do not require modifications of the underlying service (e.g. apache or sshd)

Literature

Dingledine, R., Mathewson, N., and Syverson, P.
Tor: The Second-Generation Onion Router.
In *USENIX Security Symposium* (San Diego, CA, 2004), USENIX Association, pp. 303–320.

Murdoch, S. J.
Hot or Not: Revealing Hidden Services by their Clock Skew.
In *Computer and Communications Security* (Alexandria, VA, 2006), ACM, pp. 27–36.

Øverlier, L., and Syverson, P.
Locating Hidden Servers.
In *IEEE Symposium on Security and Privacy* (Oakland, CA, 2006), IEEE, pp. 100–114.

The Tor Project.
Tor: Hidden Service Protocol.
https://www.torproject.org/docs/hidden-services.html.en.