Introducing SOR: SSH-based Onion Routing

André Egners  Dominik Gatzen  Andriy Panchenko
Ulrike Meyer

28th of March 2012
“On the Internet...
...nobody knows you’re a dog” [1993]
encryption protects only contents of communication
relationship between communicating parties remains visible:
- local network administrator
- service provider (e.g. webserver)
- search engines
- ISPs
- governments

"Anonymity is the state of being not identifiable within a set of subjects, the anonymity set" [Pfitzmann]
Who needs anonymity?

- privacy-aware individuals
- political dissidents/journalists
- organizations and companies
- law enforcement
- governments and military
- ... 
- YOU?
Introduction
Attacker Classification

EXTERNAL PARTY
Introduction
Attacker Classification

EXTERNAL PARTY

SERVICE PROVIDER
*BOFH - Bastard Operator From Hell
Introduction
Attacker Classification

*BOFH - Bastard Operator From Hell*
*BOFH - Bastard Operator From Hell
*BOFH - Bastard Operator From Hell
messages are sent not directly, but through several “middle nodes”

**ideal**: no knowledge whether source is a real originator or just forwarding messages on behalf of the others

**state-of-the-art**: onion routing

**well known examples**: *Tor*, *JAP*, *I2P*

*OR – Onion Router*
messages are sent not directly, but through several “middle nodes”

ideal: no knowledge whether source is a real originator or just forwarding messages on behalf of the others

state-of-the-art: onion routing

well known examples: *Tor, JAP, I2P*

*OR – Onion Router*
messages are sent not directly, but through several “middle nodes”

**ideal**: no knowledge whether source is a real originator or just forwarding messages on behalf of the others

**state-of-the-art**: onion routing

**well known examples**: Tor, JAP, I2P

*OR – Onion Router*
messages are sent not directly, but through several "middle nodes"

**ideal**: no knowledge whether source is a real originator or just forwarding messages on behalf of the others

**state-of-the-art**: onion routing

**well known examples**: Tor, JAP, I2P

*OR – Onion Router*
messages are sent not directly, but through several "middle nodes"

ideal: no knowledge whether source is a real originator or just forwarding messages on behalf of the others

state-of-the-art: onion routing

well known examples: *Tor, JAP, I2P*

*OR – Onion Router*
Background

Tor (1)

- **low-latency** anonymization network
- protection from **non-global** adversary
- based on **the onion routing**
- most popular and widespread one
- originally developed by US Navy
- currently about 2,000 nodes, hundreds of thousands clients
Background

Tor (2) – Circuit Through the Tor Network

Alice

Entry Node (OR1)

Middle Node (OR2)

Exit Node (OR3)

Bob

Directory Mirror

Onion Router (OR)

Encrypted Link

Unencrypted Link
Motivation
Tor & Other Low-Latency Anonymization Networks

- high complexity and poor performance of the protocols
- requires very high development effort
  - at most a few, or just one implementation
  - risk of monoculture
  - failure in the single implementation can paralyze/destroy the whole network
  - modification of the protocol every now and then
- complex properties’ analysis
- vague knowledge (by the masses) how it really works

Complexity kills security, thus also anonymity

Idea/challenge: use standardized protocols with well tested implementations
Motivation
Tor & Other Low-Latency Anonymization Networks

- high complexity and poor performance of the protocols
- requires very high development effort
  - at most a few, or just one implementation
  - risk of monoculture
  - failure in the single implementation can paralyze/destroy the whole network
  - modification of the protocol every now and then
- complex properties’ analysis
- vague knowledge (by the masses) how it really works

complexity kills security, thus also anonymity

idea/challenge: use standardized protocols with well tested implementations
Foundations

SSH Tunnels

**SSH Forwarding Tunnel**

- Client
  - local_port
  - SSH tunnel
  - SSH server
  - TCP connection
  - remote_port
  - Remote host

**SSH SOCKS Server Request**

- Client
  - local_port
  - SSH tunnel
  - SSH server
  - SOCKS server
  - service_1
  - service_2
Introducing SOR
SSH-based Onion Routing

- Uses telescoped SSH tunnels
  - Supports multiple client applications based on SOCKS
  - Dynamic reconstructions of onion chain
  - Application transparency
- Data integrity, authentication, key establishment provided by SSH
- TCP connection multiplexing is included in SSH
SOR Onion Chain: Sender Anonymity

1. `ssh -L lp1:Middle:22 Entry`
2. `ssh -L lp2:Exit:22 localhost -p lp1`
3. `ssh -D lp3 localhost -p lp2`
**SOR Onion Chain: Sender Anonymity**

1. `ssh -L lp1:Middle:22 Entry`
2. `ssh -L lp2:Exit:22 localhost -p lp1`
3. `ssh -D lp3 localhost -p lp2`
**SOR Onion Chain: Sender Anonymity**

1. `ssh -L lp1:Middle:22 Entry`
2. `ssh -L lp2:Exit:22 localhost -p lp1`
3. `ssh -D lp3 localhost -p lp2`
1. `ssh -L lp1:Middle:22 Entry`
2. `ssh -L lp2:Exit:22 localhost -p lp1`
3. `ssh -D lp3 localhost -p lp2`
SOR
Receiver Anonymity

- desc = hidden service descriptor
- introPoints = List of introduction points
- pubKey = hidden service public key
- \( \text{Sig(desc)} \) = signature of descriptor

- Onion chain with reverse port forwarding
- Onion chain with local port forwarding, TLS secured

SSH onion routing network

1a

1b

1c

database server

introduction point 1

introduction point 2

introduction point 3

hidden service

2 desc(introPoints, pubKey), \( \text{Sig(desc)} \)
SOR
Receiver Anonymity

- Enc = encrypted with public key hidden service
- RP = rendezvous point
- -> direction of message
- onion chain with local port forwarding
- onion chain with reverse port forwarding
- TLS secured
- local port forward to hidden service over RP

1: hidden service id ->
2: descriptor <-
3: 
4: Enc(RP) ->
5: forward ->
6: 
7: ACK, port <-
8: forward <-
9: connect to hidden service, port -> rendezvous point

SSH onion router network
Global user account for SOR routers

Single public/private key pair for user authentication

Disable command execution and pty of SSH

Add *no-pty, command="/bin/false"* to authorized_keys
Limit outgoing connections to other SOR routers

\[ \text{permitopen=host:port} \]

Limit outgoing connections to specific ports

Currently not possible using vanilla SSH (but with iptables)
SOR
Database Server

- Manages List of available onion routers and hidden services
- OR sends ADD_SERVER_REQ (Desc(IP, port, public key, exit policy), Sig(Desc)) to database server
- DBS verifies signature; on success adds OR
- Clients fetch the list by sending GET_SERVERS_REQ over TLS connection to DBS
Dynamic Known Host File

- For SSH server authentication client needs host key
- SSH checks if offered host key already in list and IP matches
- But OR reached over [localhost]:local_port
- Dynamically create file with chosen OR and ports
Evaluation
Sender Anonymity Performance

SOR vs. Tor in Lab Network

- Time [ms]
- Throughput [Mbps]
- SOR
- Tor

Bar chart showing the comparison between SOR and Tor in terms of time and throughput.
Evaluation
Receiver Anonymity Performance

SOR vs. Tor Hidden Service in Lab Network

- SOR
- Tor

RTT Buildup Bandwidth

Time [ms] Throughput [Mbps]
Conclusion

- SOR is completely based on open standards
- Well tested and performance-tuned SSH protocol
- Out of the box nested SSH connections
- Support of sender and receiver anonymity
- The current system outperforms the state of the art Tor
Q?