Vuvuzela

Abhirath Mahipal

csjourney.com

@abhirathmahipal

Why?

- Existing approaches don’t hide metadata.
- Some do hide metadata but are not scalable.
- Vuvuzela tries to do both.

Goals

Subsidiary

- Prevent an adversary from distinguishing between real and possible adjacent actions in light of various analysis techniques.
- Don’t leak information in the form of probability.

End

- Hide content of the messages as well as metadata.
- Hence protect users over many rounds of communication.

What can a Powerful Adversary do?

- Modify / inject traffic. Read Verizon injecting HTTP traffic with tracking cookies.
- Block traffic from certain participants, delay traffic.
- Have unrestricted access to servers.
- Create fake traffic from certain users.
- Man power to monitor network usage over very long durations.
- DDoS servers.
- The above implies access to ISP, cloud service providers etc.

Overview

- Reduce the number of observable variables (example number of users online, number of requests from a given client etc). To outsiders, as well as amongst the servers themselves.
- Noise, noise & noise.
- Mixnets and onion encryption.
- 2 Protocols. Conversation and Dialing.
- Synchronous time based rounds, ephemeral storage in memory.
- Virtual dead drops (just named ID’s to identify which pocket to deposit and receive messages from).

Conversation Protocol

Client

- They compute the dead drop ID using the shared secret and round number. Onion wraps and sends it to the server.
- Construct fake message if not in a conversation.

Server

- Collect requests and decrypt. Add noise as per the parameters set.
- Shuffle the request and forward to the next server in the chain.
- If it’s the last server, match the dead drops and read and write accordingly.

How is info Hidden?

- **Users Active**
  - Every user exchanges information
  - Same length via padding etc
- **User and Dead Drop Correlation**
  - Different dead drop for every round
  - Mixnet design
- **Number of Messages Successfully Exchanged**
  - Cover traffic to hide the number of single accesses and double accesses

Dialing Protocol

- Assumed that the sender knows the receivers public key before hand.
- Fixed locations shared by multiple users to receive invitations.
- If Alice wants to talk to Bob, she drops an invite at Bob’s dead drop. Bob downloads many other invites (fake or belonging to Charlie) as well.
- Both the users generate a shared secret that can then be used in the conversation protocol.
- Public drop deads and known by everyone. So they proposed a P2P or CDN to download instead of routing via Vuvuzela servers.

How is info Hidden?

- **User Participation**
  - Invitations onion encrypted and shuffled
  - Even if not participating special no-op dead drop
- **Which Box did a User Send an Invite to?**
  - Random noise i.e fake invitations in every dead drop
  - Each dead drop shared by multiple users
- **How Many Invitations are in a Dead Drop?**
  - Fake invites and shared by multiple users
  - Tweakable to strike balance between privacy and performance

Client

- Client sends messages at a fixed interval no matter how fast a user types.
- Handles sending messages that could not be sent in the earlier round that the user missed.
- Onion encrypts and send the requests only to the first server, so the client should know the public keys of all the servers in the chain.
- User should leave the client online at all times (or should not leave clues as to when he comes online).
- Constructs fake requests in case not in an active conversation.
- Should not switch to a less secure protocol in case of DDoS (simple use case would be HTTP injection)

Scaling

- 1M users, 68,000 messages per second on beefy hardware.
- Constant number of fake messages helps scale.
- Costly in terms of bandwidth. Every client is constantly connected, fake cover traffic etc
- Increasing the number of servers reduces performance and increases latency because each request has to go through every server. Increases security though.
- Prone to DDoS attacks given the fixed chain of servers.

Other Material

- SOSP 2015 Slides
- Ember.js Web Client for Vuvuzela before Alpenhorn
- Slides Again (Smaller)
- Source Tree Without Alpenhorn

End