Herbivore: An Anonymous Information Sharing System

Emin Gün Sirer
August 25, 2006
Need Anonymity Online

- Current networking protocols expose the identity of communication endpoints.
- Anyone with access to backbone Internet traffic can determine communication patterns.
- Encryption helps conceal content, but not identity.
- Constitutes a military vulnerability.
- Easy to determine C&C centers.
- Opportunities for industrial espionage.
Goals

- Anonymity
- Broadcast Networks
- DC-Nets

Scale
Source Rewriting
Performance
Source Rewriting

- Packets sent through an intermediary to mask origin
  - E.g. MIXes, Crowds, Onion Routing, Tarzan, AP3B
- Long paths and time delays make it difficult to trace back
- Practical, implemented
- High latency
- A powerful adversary, through observations, ultimately traces back

“Attack at dawn”
Broadcast Networks

- Every node sends to every other node all the time
  - E.g. P5
- Strong anonymity: cannot tell who or when
- Must constantly send at peak bandwidth
- Low throughput
- High network load
- Never implemented
Herbivore Overview

- Herbivore builds on dining cryptographer networks (DC-Nets)
  - Elegant scheme for anonymous communication
    [Chaum 1981]
- Strong anonymity guarantee
  - Even an adversary that has tapped the entire network and observed every packet cannot determine packet origin
- Herbivore makes DC-Nets practical
  - Efficient and scalable, with the same strong anonymity guarantee
DC-Net Operation

- Every pair of participants tosses a coin in secret.
- Every participant reports the XOR of all their coins and messages.
- XORing all reported values reveals the message.
  - XOR of all messages if more than one transmitter.
DC-Net Example

\[ \text{Pa} \oplus \text{Pb} \oplus \text{Pc} = \]

\[ (\text{AB} \oplus \text{AC}) \oplus (\text{AB} \oplus \text{BC} \oplus \text{m}) \oplus (\text{BC} \oplus \text{AC}) = \]

\[ \text{AB} \oplus \text{AB} \oplus \text{AC} \oplus \text{AC} \oplus \text{BC} \oplus \text{BC} \oplus \text{m} = \text{m} \]

\[ 0 \oplus 0 \oplus 1 = 1 \]

\[ 0 \oplus 1 \oplus 1 = 0 \]
DC-Net Properties

Why does it work?
- All nodes participate in the computation of the packet
- All nodes equally culpable
- Information theoretic guarantee

Shared anonymous broadcast channel
- Like Ethernet, but virtual

As described so far, it is not a practical system
- Lacks protocol, scale and performance
Herbivore DC-Net Protocol

- Use PRNG instead of coins
  - Derive stream of coin tosses efficiently
- Fully-connected key graph
  - Every pair has a unique key, no weak points
- Communication occurs in rounds, of three phases
  - Reservation
  - Transmission
  - Voting
Herbivore Reservation Phase

Goal: anonymously acquire exclusive access to the channel

Divide time into transmission slots
- A node with a message to send
  - selects a transmission slot, i, at random
  - broadcasts a bit vector, with 0's everywhere and a 1 for the ith bit
  - everyone receives XOR of all reservations
  - transmits in reserved slot, if succeeded

Collisions trigger Ethernet-like backoff
Herbivore Transmission Phase

- A node transmits its message in the slot it has reserved
  - Unreserved slots are skipped
- Collisions may occur during the transmission phase
  - If an odd number of nodes select the same slot, or if there is a malicious node
  - Every packet carries data and hash
  - Provides collision detection & ensures packet integrity
- Multiple rounds in parallel
Herbivore Voting Phase

- **Goal:** signal to other nodes that a node is in the middle of a long transaction
- **Delay departure until transaction is completed, if possible**
- **Herbivore voting is bandwidth efficient (2 bytes)**
  - **Special case for anonymous 1-bit voting**
Herbivore Overlay Topology

Chaumian DC-Nets use a Fully-Connected Graph

- $O(1)$ latency, $O(N^2)$ load
Herbivore Overlay Topology

◆ Chaumian DC-Nets use a Fully-Connected Graph
  ▪ $O(1)$ latency, $O(N^2)$ load

◆ Or Ring
  ▪ $O(N)$ latency, $O(N)$ load
Herbivore Overlay Topology

- Chaumian DC-Nets use a Fully-Connected Graph
  - $O(1)$ latency, $O(N^2)$ load
- Or Ring
  - $O(N)$ latency, $O(N)$ load
- Herbivore uses a Star topology
  - All nodes send their packets to a “center” node in each round
  - Center duties rotate deterministically at each round
  - $O(1)$ latency, $O(N)$ load
Herbivore Protocol Efficiency

- **Topology**
  - Low latency, low load overlay organization

- **Reservation**
  - We derive and use optimal vector size

- **Transmission**
  - We run multiple transmission rounds concurrently

- **Voting**
  - We extend system lifetime with efficient 1-bit voting
Herbivore Scale

Traditional DC-Nets do not scale
- Protocol is too heavy-weight for use at planetary scale

Divide and conquer!
- Self-organize the network into cliques of k-nodes
- Use the relatively heavy-weight protocol in small cliques

Decouple protocol cost from system size
Herbivore Clique Management

- Use a P2P overlay to organize N participants into cliques of minimum size k
  - Clique size ranges from k to 3k, for k = 20 or so

- Every node solves a crypto-puzzle to obtain a node-id and join the system
  - Puzzle solution randomizes entry into cliques
  - Nodes demonstrate solution of the puzzle to each preexisting clique member
  - No central authority is involved

- Use Pastry to map nodes to clique
Herbivore Cliques

- A clique of more than $3k$ nodes is split into 2 cliques.
- When nodes depart and clique size drops below $k$, the nodes depart and join closest existing cliques.
Interclique Operation

- Within a clique, all communication is anonymous
  - Uses the Herbivore DC-Net protocol

- Between cliques, packets are forwarded via randomly selected proxies

- Interfacing with the outside world also occurs through randomly selected proxies
DC-Net Filesharing

Naïve solution is simple

- Every node has a list of files it offers for downloads
- Queries are broadcast from clique to clique
- Files are transferred back if query hit

Naïve solution is open to intersection attacks

- RIAA queries for "Metallica", examines clique membership of all cliques that respond, takes the intersection over time
- Whoever remains is guilty of placing Metallica songs online
Herbivore Filesharing

- Batch download system with a simple user interface
  - List of files to publish
  - List of files to acquire
- Every node has two file stores
  - A-list: files available to others but not yet disseminated to anyone
  - B-list: LRU cache of files recently sent in response to queries
Herbivore Filesharing

- When a query arrives for a file held in the A or B-list, the node responds with the file
  - If on A-list, the file is transferred to the B-list

- When a file is overheard on the broadcast channel, it is placed on the B-list
  - Hence, all nodes in the clique have state identical to the originator
  - Can be done probabilistically, with $p < 0.5$

- No way to determine the originator, despite use of small anonymization groups!
  - Can search or sue everyone in the clique (not under US law)
  - Published files may get dropped for lack of interest
Herbivore Status

- Implemented the system
  - Anonymous filesharing, instant messaging and web browsing
  - YIM-like interface for FS and IM + web proxy
  - ~27,000 lines of code

- Deployed on Planetlab

- The system is practical
  - First known deployment of DC-Nets
  - Scales well, efficient protocol
Herbivore Bandwidth

Herbivore Bandwidth

Bandwidth (Kb/s)

Clique Size
Herbivore Latency

Herbivore Latency (s)

- Series 1
- Series 2
- Series 3
- Series 4

Clique Size

Latency (s)

1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2

10
15
20
25
30
35
40
Summary

Herbivore provides strong anonymity, scalability and performance
- DC-Nets are practical!

Enables participants to share information anonymously, even in the presence of omnipotent adversaries
Further Information

E. Gün Sirer
egs@cs.cornell.edu
http://www.cs.cornell.edu/People/egs/herbivore/
Attacks and Defenses

Sybil: use cryptopuzzles
Jamming: use commitment and trap
Intersection: use A and B-lists
Statistical: DC-Nets
Sloth: accrues strikes
Center: accrues a fractional strike
Eclipse: check adjacent clique members on clique creation
Abuse: selective revocation with secret sharing
Anonymity and Abuse

What if someone uses the system to perform nefarious activities?
- E.g. plot a terrorist attack

Serious problem
- But not new, police have mechanisms for tracking down criminals with similar anonymous channels in the real world

Technical solution
- Share secret keys using \((n, k)\)-secret sharing
- Revoke anonymity when \(k\) out of \(n\) participants agree