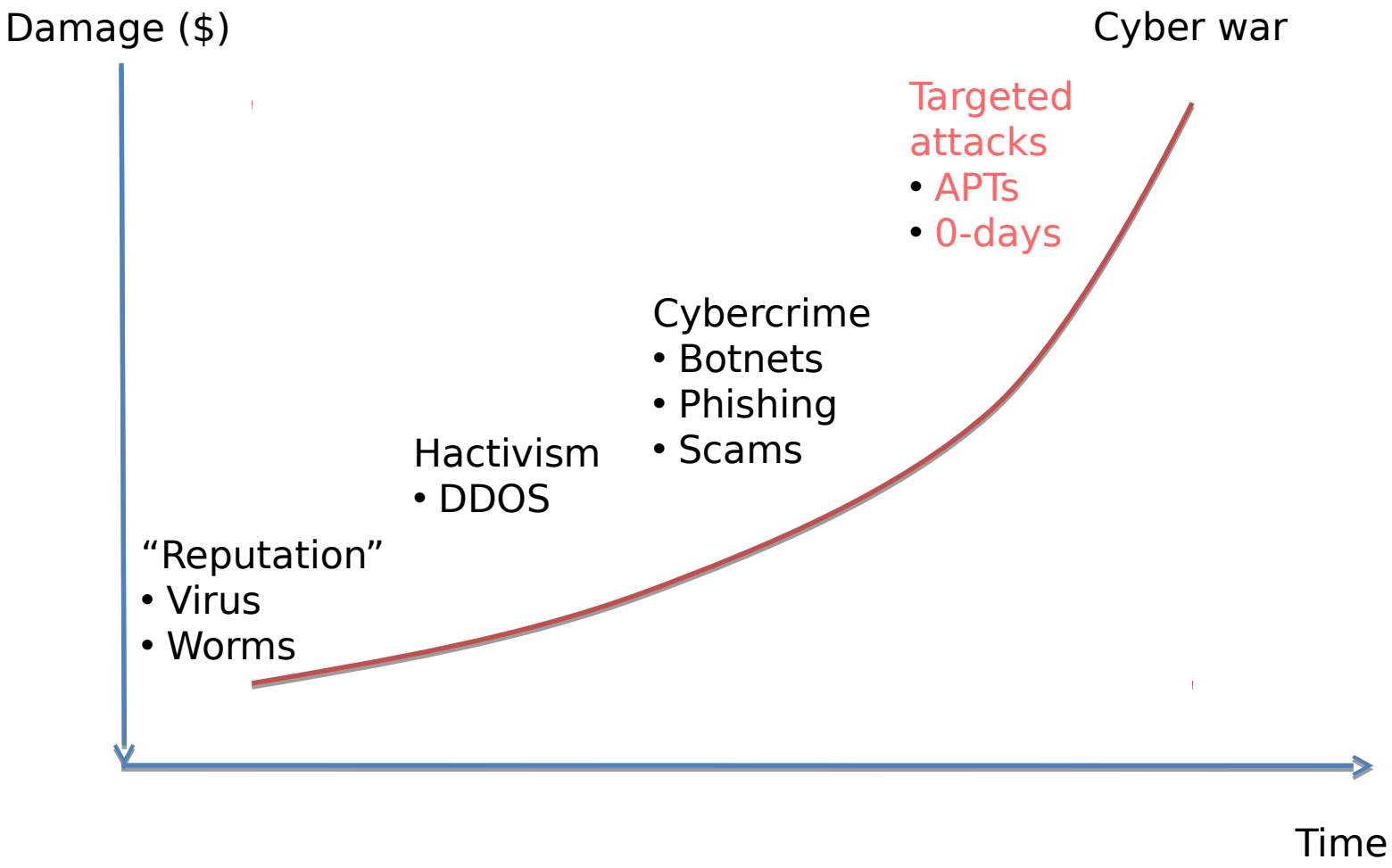
# BotYacc

## Unified structured graph analysis and behavioural analysis with C&C detection

ESORICS 2014

Shishir Nagaraja
University of Birmingham

# A significant network security problem



# Stats

- Rising importance of targeted attacks
  - 25% of total attacks are targeted [V]
  - 50-255/day [S]
- Ability to detect breaches
  - 67-- 69% of breaches are detected by *external parties* [V,M]
  - 66% of the breaches took *months* to be detected [S]
  - 243 days is the median time of attacker presence in compromised network [M]

Attacks are not detected by those affected.

Attacks remain unnoticed for long time.

# Stats roundup

- Targets [Symantec]

Manufacturing (24%)
Finance, insurance, real estate (19%)
Other services (17%)

- Targets [Mandiant]

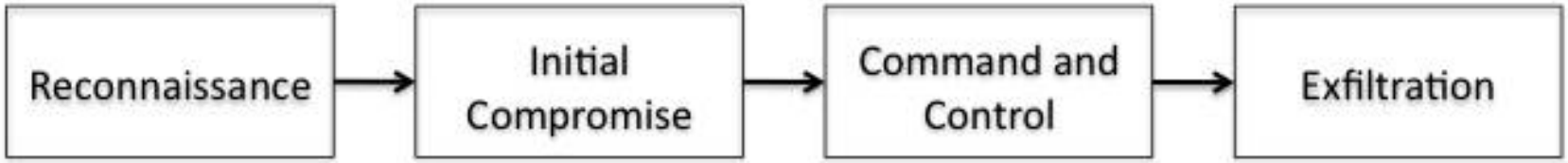
Aerospace & defence (17%)
Energy, oil, gas (14%)
Finance (11%)

Target size:
Large (2500+ employees): 50%
SMB (up to 250 employees): 31%

Targeted attacks span multiple sectors

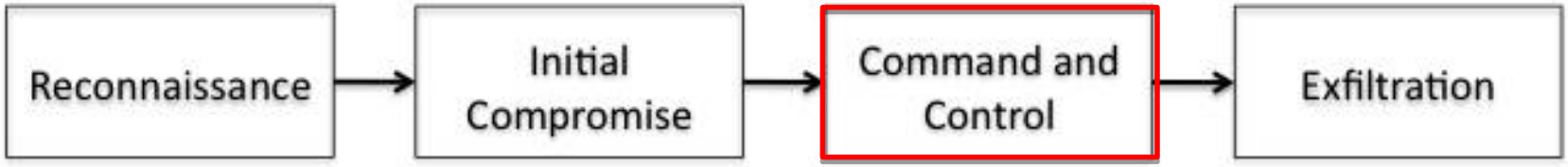
Org. size is not a predictor for attack likelihood

# Targeted attacks




Traditionally, lots of emphasis has been put on detecting the Initial Compromise step.

# Targeted attacks and C2 step



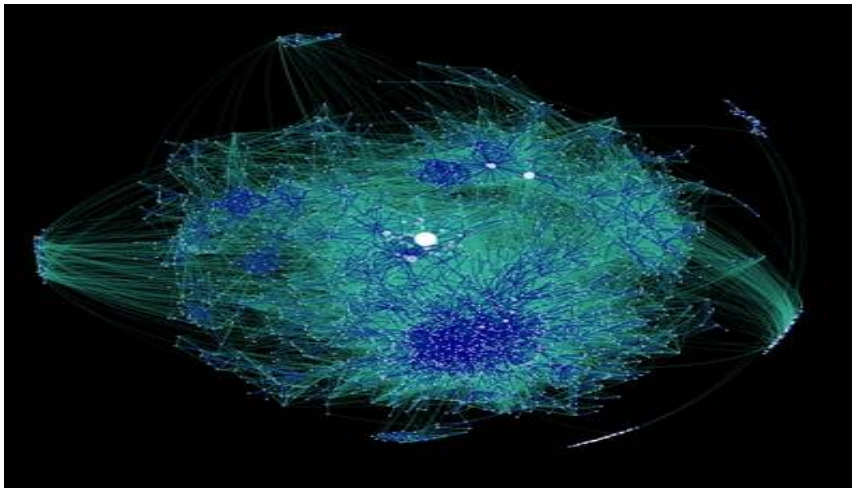

Command and control channel links compromised machines with attackers:

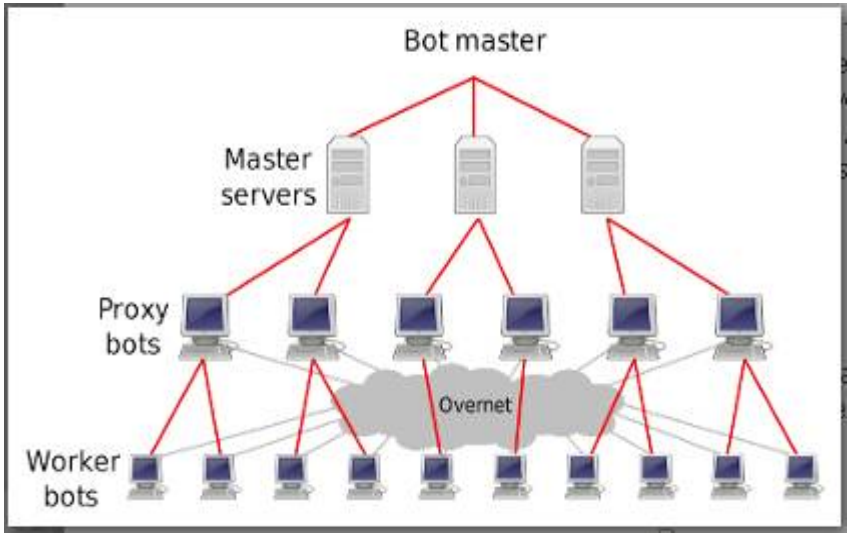
- Receive commands from attackers
- Exfiltrate data

What if we could detect and block C2 activity?

# C&C Landscape

- Targeted malware attacks
  - Coordinated attack platform
  - Avg size > 20,000 hosts
  - DDoS/Spam/Information theft
- C&C is going P2P
  - Robustness: no central nodes to find, attack
  - Storm, Conficker
- Structured P2P networks
  - Chord, Kademlia
  - Scalable
  - Resilient to churn
- Lots of beaconing traffic
- C&C also uses the Cloud (Google, Facebook,..)





# C&C communication

Evidence of C&C traffic

- `Who-talks-to-whom' – Patterns in communication structure (expanders such as chord and kademlia)
- `How they talk to each other – network **traffic characteristics** patterns
- 'When' they talk to each other – timing patterns

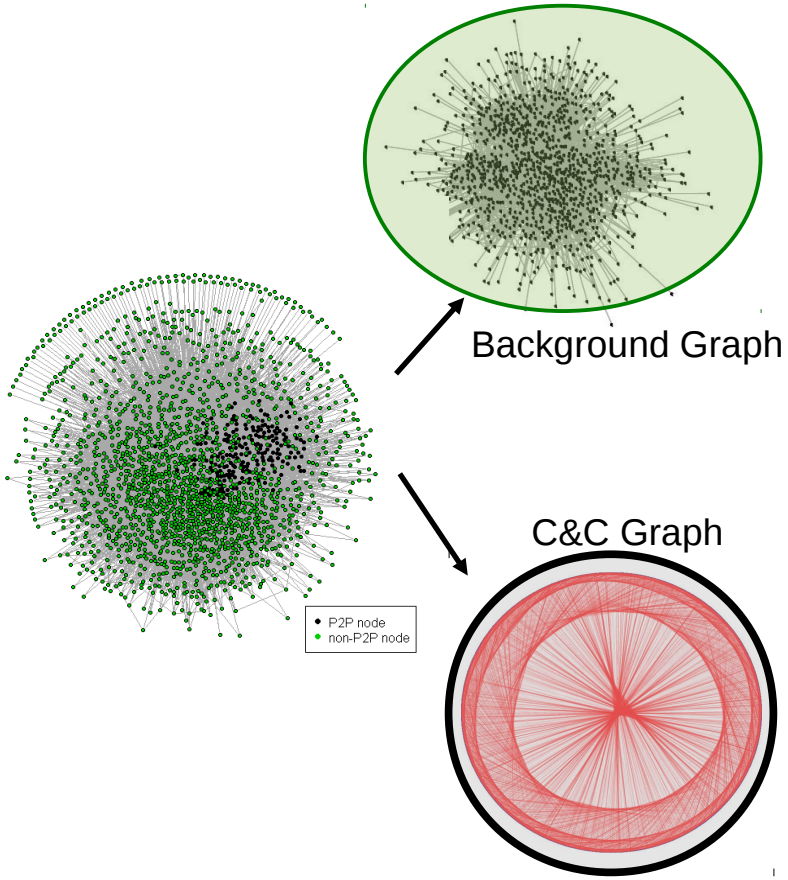
Isolation methodology: **Traffic analysis**

# Current Detection Approaches

- Misuse detection – Attack-traffic based classification
  - 0 day botnets & information theft hard to detect
  - partitioned misuse (Storm)
- Anomaly detection – Greater than threshold deviation
  - Each bot can fly under the radar (threshold)
- Clustering – Traffic classification based on traffic characteristics and content;
  - Doesn't work when content is encrypted or traffic is padded.
  - Traffic similarity is a noisy notion, this leads to false-positive rates ~ 0.5 to 1%

# Current Detection Approaches

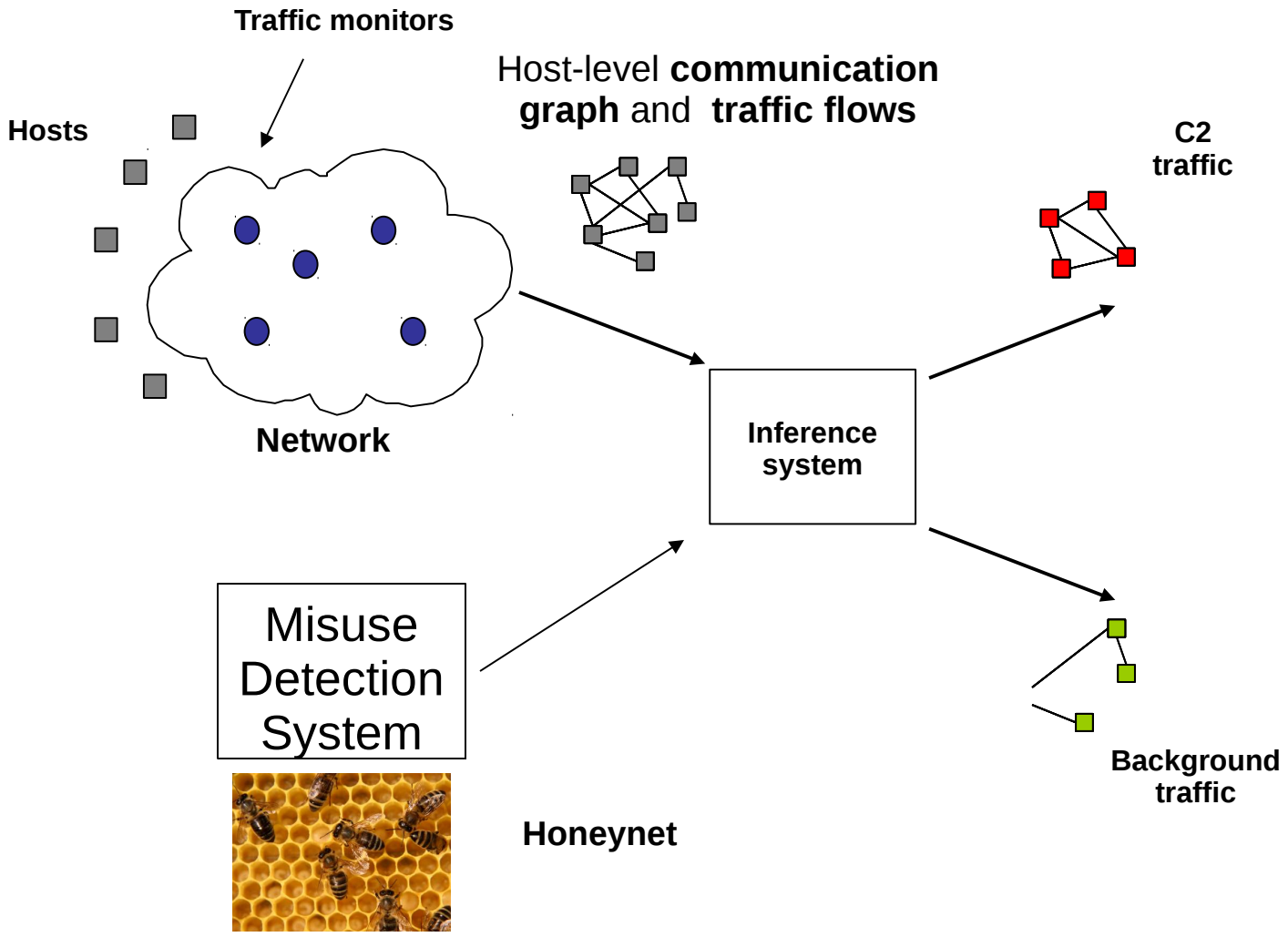
- Graph-based techniques
  - Can we isolate C&C traffic based on communication structure of which pairs of hosts communicate?
  - Inference is based on the P2P structure underlying C&C design.
- Past work has not considered graph noise (eg. beaconing traffic)
- Significantly high FPR if botnet overlaps a hub node (eg. Bots communicate with google.com)
- Dependence on ISP vantage points, not amenable to enterprise-level analysis.



# Clustering-based mechanisms

| System | Classifier-Based | Target | TP Rate | FP rate | ML Algorithms |
|---|---|---|---|---|---|
| Notos [3] | ✗ | Domains | 96.8% | 0.38% | $k$-means clustering |
| Kopis [4] | ✓ | Domains | 73.6-98.4% | 0.3-0.5% | Random Forest Classifier |
| Disclosure [6] | ✓ | Servers | 60-70% | 0.5-1% | Random Forest Classifier |
| BotHunter [22] | ✓ | Hosts | 95% | NGE(possibly low) | Pattern Matching |
| BotMiner [21] | ✓ | Hosts | 75-100%(specific botnets) | NGE(low, 0.0.03%) | $x$-means clustering |
| FluxBuster [51] | ✗ | FFSN | NGE | <1% | Hierarchical clustering, C4.5 Decision Trees |
| Antonakakis et al. DGA [5] | ✗ | DGA | 99.7% | 0.1% | $x$-means clustering, decision trees classifier |
| Botzilla [62] | ✓ | C2 Payloads | 94.5% | NGE (0.00001-0.00004%) | Pattern Matching |
| Zhang et al. Stealthy P2P [80] | ✗ | Stealthy P2P | 100% | 0.2% | Birch Streaming Clustering, Hierarchical Clustering |
| ProVex [63] | ✓ | Encrypted C2 | Per given malware, at least 78% (6 of 10 100%) | 1 per $2^{40}$ packets | Pattern Matching |
| Firma [61] | ✓ | Multi-protocol C2 | NGE | 0.00001% (live traffic) | Custom Clustering (matching specific features) and Signatures |
| BotGrep [47] | ✗ | P2P | 98% | 0.4% | $k$-Means + Algo Clustering |

# BotYacc: System Architecture: Cooperative C&C defense



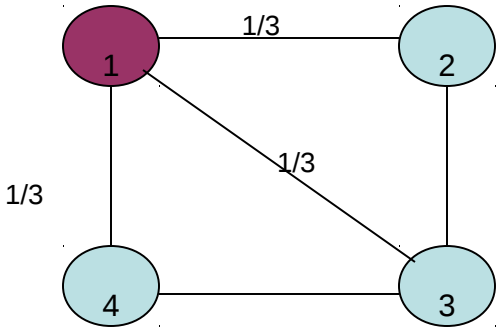
# Inference Algorithm: Structural Properties & Random Walks

- Model random walks on a graph as a Markov chain
  - $p_i$ : probability of being at node i
  - T: transition matrix

$$\Pi = \lim_{n \to \infty} T^n p$$

- Mixing time: convergence time to stationary distribution
  - Structured topologies are fast mixing

- Separate subgraphs based on relative Markovian mixing properties




$$T = \begin{matrix} 0 & .33 & .33 & .33 \\ .5 & 0 & .5 & 0 \\ .33 & .33 & 0 & .33 \\ .5 & 0 & .5 & 0 \end{matrix}$$

$$T_{ij} = \frac{1}{d_i}$$

(if i→ j is an edge)

$$\Pi = (0.3, 0.2, 0.3, 0.3)$$

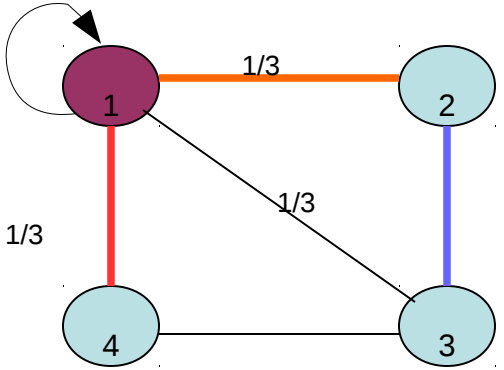
# Inference Algorithm: Structural Properties & Random Walks

- Separate subgraphs based on relative Markovian mixing properties

$$\Pi = \lim_{n-¿\infty} T^n p$$

- However walks are memoryless.

- Methodology: Walks (diffusion) over similar edges.




$$T = \begin{matrix} 0 & .33 & .33 & .33 \\ .5 & 0 & .5 & 0 \\ .33 & .33 & 0 & .33 \\ .5 & 0 & .5 & 0 \end{matrix}$$

$$T_{ij} = \frac{1}{d_i}$$

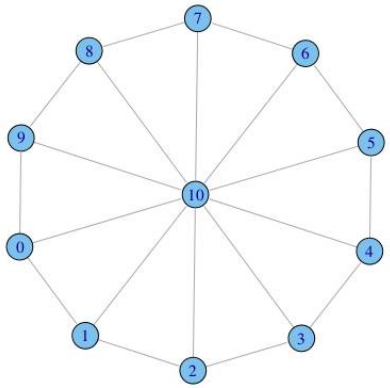
(if i→ j is an edge)

$$\Pi = (0.3, 0.2, 0.3, 0.3)$$

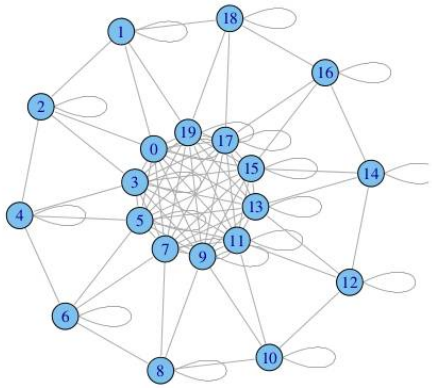
# 1) Dual creation step

- Edges of the communication graph become nodes in the dual.

- State probability mass ($q_i$) for P2P vertices closer to stationary distribution than non-P2P vertices




(a) graph $G$




(b) Dual $\mathcal{D}(\mathcal{G})$

# 2) Recursive Partitioning Step: a probabilistic model

- Generation of traces T
  - *n* random walks per node (traffic flow)
- Sample partitions based on the following probability distribution

Key intuition – **short walks** (log n), state probability mass is homogenous for p2p nodes

**Honeynet** input

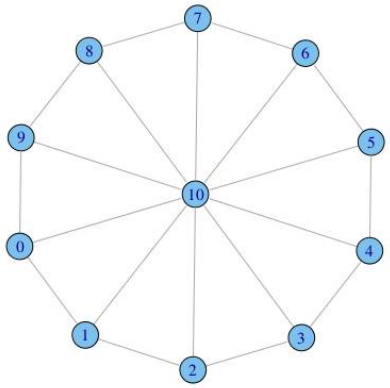
$$P(X = P2P|T) = \frac{P(T|X = P2P) \cdot P(X = P2P)}{Z}$$

Normalization constant

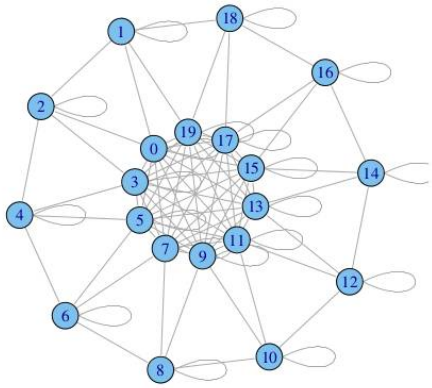
- Use Markov Chain Monte Carlo Techniques from SybilInfer[NDSS09]

# 3) Validation tests

- Heuristics to decide when we are done
  - High conductance of the partition
- Heuristics to decide if partition is P2P
  - Degree homogeneity
  - Fast mixing




(a) graph $G$




(b) Dual $\mathcal{D}(\mathcal{G})$

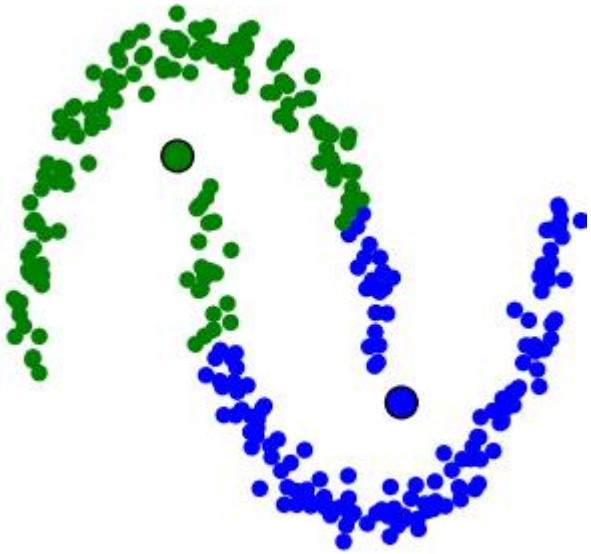
# 4) Combining graph analysis with traffic similarity

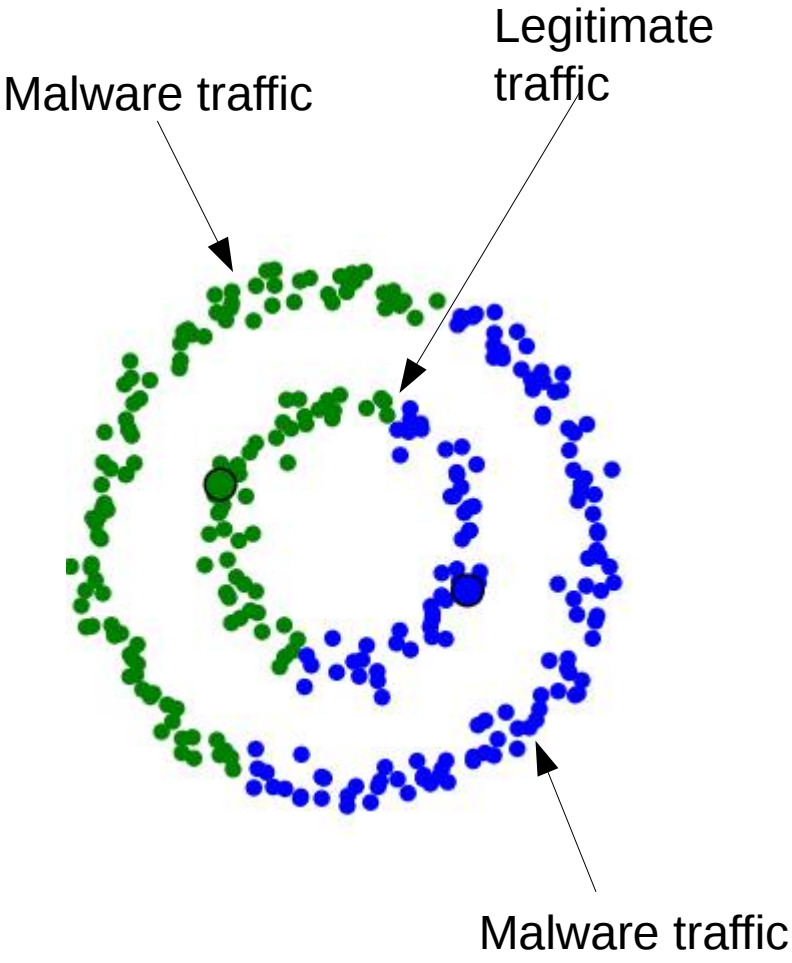
- Each node in the dual is a traffic flow.
- Edges in the dual represent comparisons between k-dimensional traffic flow vectors (use of a large number of flow features).
- We then carry out random walks as before, on the weighted graph. Weights are computed using the **laplace-beltrami** operator.

$$W_{ij} = \begin{cases} e^{-\frac{||e_i - e_j||}{t}} & \text{if} ||e_i - e_j||^2 \leq \varepsilon \\ 0 & \text{otherwise} \end{cases}$$

# Why the Laplace-beltrami operator?

We introduce a non-linear kernel to deal with non-linearity of the data.





# Experimental Methodology

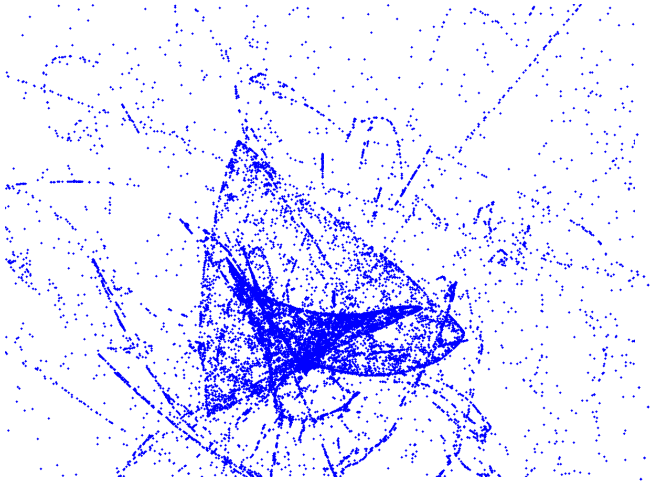
Real world traffic traces from an enterprise network containing malware C&C traffic

- We captured network traffic at a university gateway for a period between March and April 2012.
- This dataset has 113,576 unique source IP addresses and 11,643,993 traffic flows.
- This includes 432,257 embedded botnet flows from seeded malware.
- The malware testbed was seeded with Zeus-II, Spyeye, and Miner binaries.

# Evaluation

| | #Malicious flows | #gateway-flows | Detection% | % FP |
|---|---|---|---|---|
| Week1 | 3368 | 1211736 | 99.98 | 0.019 |
| Week2 | 8836 | 1392755 | 99.93 | 0.037 |
| Week3 | 3231 | 1109264 | 97.95 | 0.082 |
| Week4 | 8349 | 1312952 | 98.09 | 0.041 |
| Week5 | 8217 | 1130120 | 98.21 | 0.030 |

**Table 1.** Zeus in enterprise traffic – detection and error rates of inference



Pre-partitioning

Post-partitioning

# Evaluation

| | #Malicious flows | #gateway-flows | Detection% | % FP |
|---|---|---|---|---|
| Week1 | 8021 | 1346235 | 98.11 | 0.041 |
| Week2 | 6295 | 1327479 | 98.77 | 0.064 |
| Week3 | 4213 | 1180134 | 99.86 | 0.074 |
| Week4 | 3538 | 1396174 | 97.86 | 0.047 |
| Week5 | 5388 | 1186480 | 98.70 | 0.023 |

**Table 2.** Spyeye in enterprise traffic – detection and error rates of inference

| | #Malicious flows | #gateway-flows | Detection% | % FP |
|---|---|---|---|---|
| Week1 | 1050 | 1590306 | 97.50 | 0.018 |
| Week2 | 2735 | 1186212 | 96.64 | 0.064 |
| Week3 | 5341 | 1560028 | 94.89 | 0.048 |
| Week4 | 3099 | 1186929 | 95.52 | 0.062 |
| Week5 | 4566 | 1154067 | 97.76 | 0.072 |

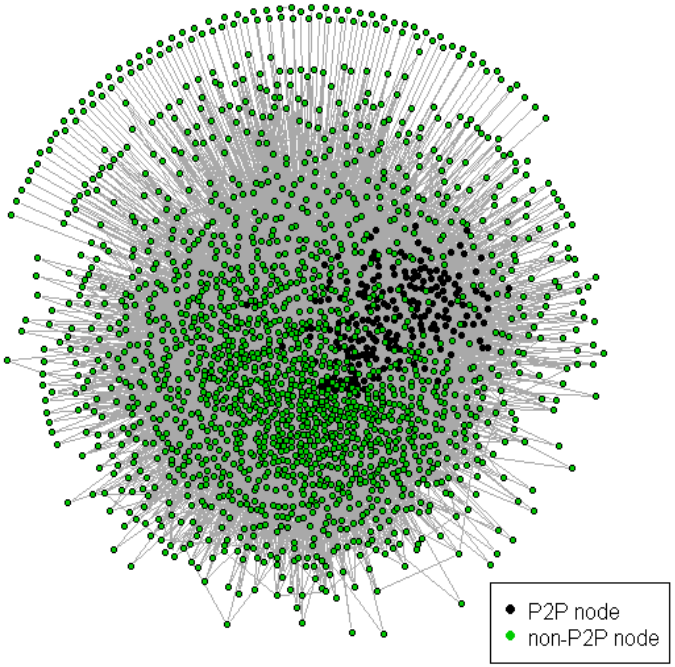
**Table 3.** Miner in enterprise traffic — detection and error rates of inference

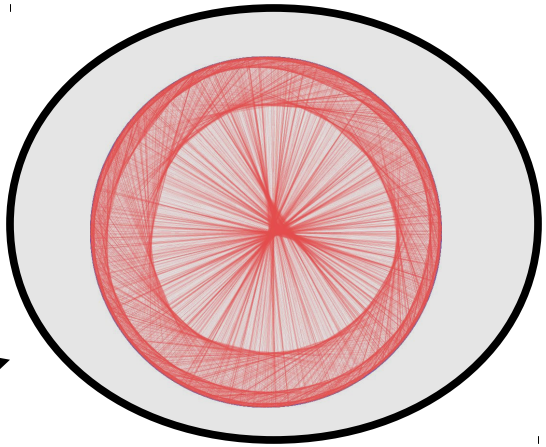
# Experimental Methodology - II

Abilene
104426 nodes,
547053 edges

+

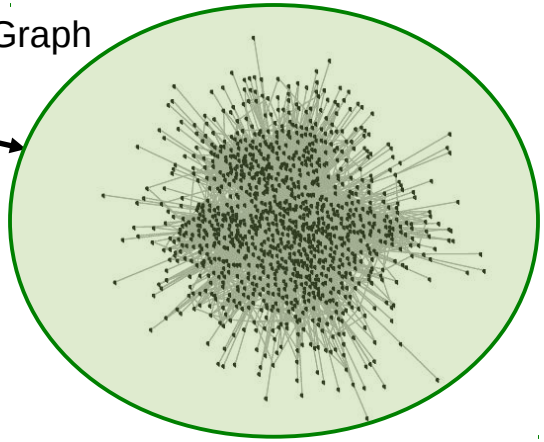
Synthetically
embedded P2P
C&C flows



P2P Component

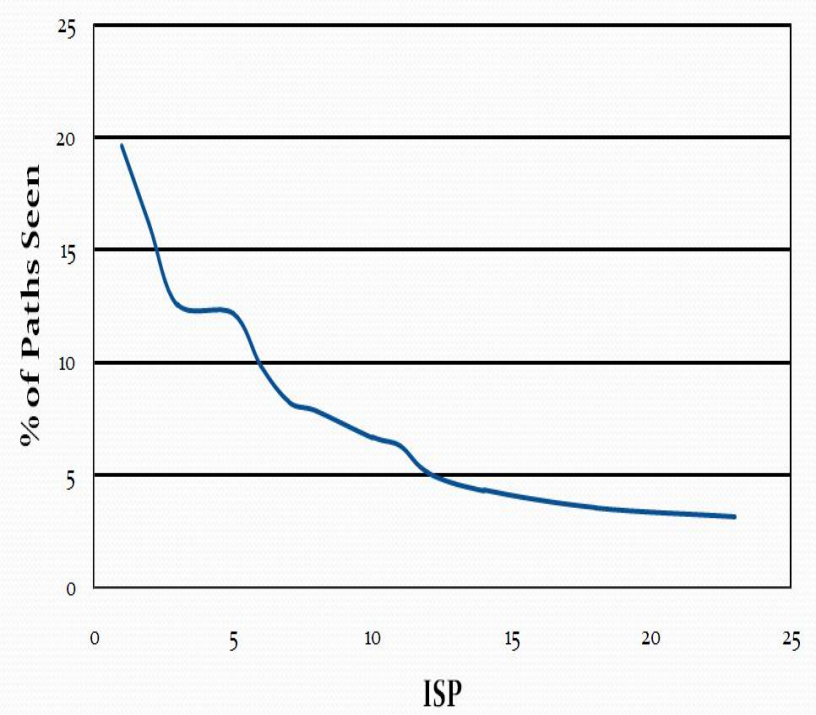
Background Graph



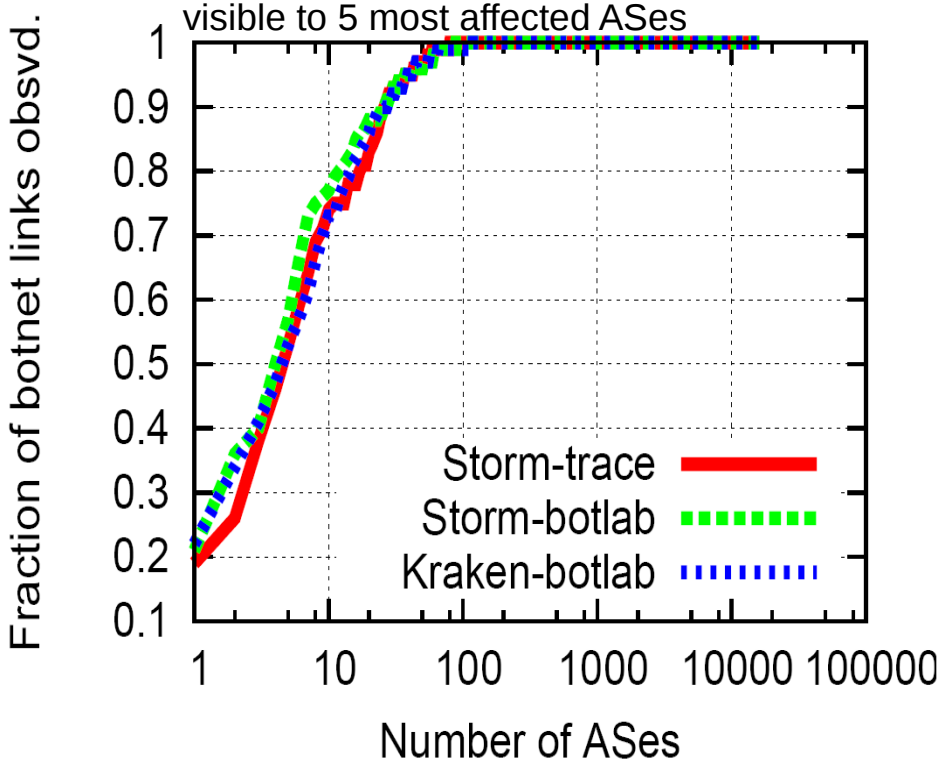
Metrics: False positives
Detection Percentage

# ISP Visibility

60% of paths visible to Tier 1 ISPs




57% Storm and 65% Kraken paths visible to 5 most affected ASes




Experimental Methodology: Remove 40 % C&C flows

# Evaluation – Abilene ISP

## |V| = 104,426 nodes, |E| =547053 edges, 60% visibility

| Topology | \|V\| | %FP | Detection % |
|---|---|---|---|
| deBruijn | 100 | 0.000 | 97 |
| | 1000 | 0.012 | 97 |
| | 10000 | 0.017 | 96 |
| Kademlia | 100 | 0.000 | 96 |
| | 1000 | 0.010 | 97 |
| | 10000 | 0.019 | 97 |
| Chord | 100 | 0.000 | 91 |
| | 1000 | 0.015 | 96 |
| | 10000 | 0.023 | 97 |

False positives are manageable

Detection > 90% for all topologies and botnet sizes

# Comparison with Community Detection Techniques

**Dataset:** dense 2000 node Abilene subgraph with embedded structured graphs

False Positives (%)/ Detection (%)

| Topology | BotGrep | Girvan-Newman Betweennes s | Fast Greedy Modularity | BotYacc | Modularity Eigenvector |
|---|---|---|---|---|---|
| deBruijn | 0.78/97.45 | 19.73/84.69 | 14.43/92.3 5 | 0.05/96.5 | 0.92/46.12 |
| Chord | 0.77/92.85 | 6.05/80.5 | 7.58/89.83 | 0.08/91.4 | 4.24/79.81 |
| Kademli a | 0.92/93.0 | 18.06/95.25 | 14.66/66.2 0 | 0.08/92.7 | 5.70/51.30 |

BotYacc significantly outperforms existing detection techniques

# Discussion

- Recognizing  Misbehavior
  - Start with a honeynet seed
    - Identify P2P network containing honeynet nodes
  - Anomaly/misbehavior detection
    - Statistical significance test

- BotYacc combines the idea of communication graph analysis with the notion of flow-similarity to yeild a unified traffic analysis technique.
  - It defaults to clustering in the absence of structural patterns.
  - It defaults to graph analysis in the absence of flow similarity patterns.

# Conclusion

- Unified graph analysis and machine learning techniques may be useful for finding malware C&C channels (one order of magnitude lower false positive rate).
- We have proposed one way of unifying these notions, which gives us the BotYacc technique. A nice property is that the same technique is usable for ISP and enterprise scales.
- Some evasion resistance is possible based on the difficulty of evading structured graph analysis whilst maintaining full flow anonymity to resist flow similarity detection.