Botnet detection revisited: Theory and Practice of Finding Malicious P2P Networks via Internet Connection Graphs

Stefan Ruehrup, Pierfrancesco Urbano, Andreas Berger, Alessandro D’Alconzo
FTW, Vienna, Austria

5th IEEE Int. Traffic Monitoring and Analysis Workshop (TMA 2013)
1. Introduction and Motivation
   - P2P Botnets from an operators point of view
   - Graph-based botnet detection:
     Revealing P2P structures in Traffic Dispersion Graphs

2. Recent approaches
   - PageRank and Clustering
   - Finding Friend-of-a-friend structures

3. Experimental results
   - Limitations of existing approaches
   - Suggestions on improvements

4. Conclusion
What is a botnet?

- **Botmaster** runs the C&C server
- **Bots** malicious software runs on an infected hosts
- **C&C Server** commands and controls a group of bots
- address change to evade countermeasures

Client-server approach
P2P Botnets

→ Use of peer-to-peer (P2P) overlay networks already observed in practice

→ Example: Storm botnet
  - peak activity in 2007
  - millions of infected hosts
  - uses Overnet P2P network (based on Kademlia) in the setup phase

→ No central point of failure, but command dissemination slow

→ Theory: P2P networks form observable structures
What the network operator observes:

Problem: Which other hosts are infected?
Observable connections

Internet  monitored local network

invisible to the monitor

<src ip, src port, dst ip, dst port, protocol>

Goal: identify potential bots for further analysis
Traffic dispersion graph

Internet

monitored local network

monitored connections
(traffic dispersion graph, TDG)

monitor

invisible to the monitor

[src ip, src port, dst ip, dst port, protocol>

<monitor>

local IPs

external IPs

Edge: at least one IP packet transferred
Traffic dispersion graph - example

TDG with
~ 4000 nodes
thereof 500 internal
~ 20,000 edges

Known bot

How to find others?
Experimental setup

Traffic dispersion graph (TDG) from a network trace
a) DSL Network
b) UMass Campus Network

P2P botnet topology from a synthetic Kademlia P2P network

Merged network trace

Sample

superimposed nodes

(only one random bot is known)
Recent approaches to Botnet Detection

→ Finding fast mixing components by random walks in traffic dispersion graphs
  *BotGrep: Finding P2P bots with structured graph analysis*
  by Nagaraja, Mittal, Hong, Caesar, Borisov. USENIX Security 2010

→ Clustering hubs/authorities by pagerank on traffic dispersion graphs
  *BotTrack: Tracking botnets using netflow and pagerank.*
  by François, Wang, State, Engel. NETWORKING 2011

  complete knowledge of the network is assumed (holistic view)

→ Finding friends-of-a-friend in traffic dispersion graphs
  *Friends of an enemy: identifying local members of peer-to-peer botnets using mutual contacts*
  by Coskun, Dietrich, Memon. ACSAC ’10

  only partial knowledge is assumed (edge router’s view)
Hubs vs Authorities – results of BotTrack

PageRank on UMass trace with 50 bots, DBSCAN Clustering

Cluster 1
elevated nodes

Cluster 2

(no cluster)

Background nodes

Bots
Finding “Friends of an Enemy”

[Coskun et al. 2010]

1. monitored connections (full TDG)

   - external IPs
   - local IPs

Traffic Dispersion Graph (TDG)

graph extracted from 15 min traffic
TDG with “mutual contacts” only

2

only connections with mutual contacts

edge is included only if at least two nodes share an external destination
... and the resulting MCG

3 mutual contacts graph (MCG)

edge indicates mutual contact

external IPs  local IPs
“Dye pumping”

4. Dye pumping

- Start node (known infected host)

5. Result

- Set of nodes that likely belong to the same p2p network

Result depends on dye level threshold.
Dye Pumping details

- Algorithm as proposed by Coskun et al. 2010
- Compute transition matrix $T$
  \[ T(i,j) = \text{edge weight of } (i,j) \text{ over weighted degree of } i \]
- Multiply iteratively with Dye level vector $L$
- "Refresh“ dye level of the start node before each iteration

Algorithm 1 \textit{Dye--Pumping}(E, s, maxIter)

\begin{algorithmic}[1]
  \State $T \leftarrow \text{computeTransitionMatrix}(E)$
  \State $\bar{T} \leftarrow \text{normalize}(T)$
  \State $L \leftarrow [0, 0, ..., 0]^T$ \{initialize $L$ as a zero vector\}
  \For {iter = 1 to maxIter} \Do
    \State $L(s) \leftarrow L(s) + 1$ \{Pump dye from the seed node\}
    \State $L \leftarrow \sum_{i} L_i$ \{Normalize dye level vector\}
  \EndFor
  \State $L \leftarrow \bar{T} L$ \{Distribute dye in network for one iteration\}
  \State output $L$
\end{algorithmic}
Alternative to dye pumping

→ **MCG Search**: breadth-first search (limited by depth or dye)
  - each node in the visited set gives half of its dye
  - this dye is spread to unvisited nodes according to edge weight.

→ Similar detection rate to Dye Pumping
→ Saving matrix multiplications gives significant speed-up.
Results for campus network

ROC for UMass Campus Trace

False positives rate

True positives rate

variation of dye-level threshold

5% bots

10% bots

MCG Search 100 bots (~5%)
MCG Search 200 bots (~10%)
Dye Pumping 100 bots (~5%)
Dye Pumping 200 bots (~10%)
Louvain 100 bots (~5%)
Louvain 200 bots (~10%)

TPR 66%, FPR 1%
Results for DSL network

ROC for DSL Trace

- True positives rate vs. False positives rate
- MCG Search 20 bots (~5%)
- MCG Search 40 bots (~10%)
- Dye Pumping 20 bots (~5%)
- Dye Pumping 40 bots (~10%)
- Louvain 20 bots (~5%)
- Louvain 40 bots (~10%)

TPR 66%, FPR 1%
Runtime comparison

Average Number of Operations

Operations [#] vs Number of bots

Dye Pumping
Local MCG Search

Average Runtime

Runtime [s] vs Number of bots

Dye Pumping
Local MCG Search
Louvain
Some remarks...

on the impact of other P2P networks
Conclusion and open questions

→ Structured P2P networks form densely connected topologies

→ They can be found in Traffic Dispersion Graphs (TDG)
  - Only IP contacts needed (works also for encrypted c&c traffic)
  - Problem: nodes participate in botnet and legitimate communication

→ P2P networks with high interconnectivity can be separated
  - Performance depends on background traffic
  - Experiments: Local heuristics better than targeting high degree, page rank, ...

→ Network graph analysis on its own cannot detect botnets
  - but help to identify further bots, if some infected hosts are known
Thank you very much!

Stefan Ruehrup
ruehrup@ftw.at

FTW
Telecommunications Research Center Vienna
Vienna, Austria