BotGraph: Large Scale Spamming Botnet Detection

Web-account abuse attack

recent spamming technic

- New different approche for sending spam
- Basing on reputation of email providers
- Difficult to detect
  - signup detection
  - monitoring users' activity
- Very difficult to distinguish real user from bot

Solution?

tricky, with two challenges

1. **designing an algorithm**
2. **implementing working solution**
    - **milions of users**
    - **houndreds of gigabytes activity logs**

Solution!

bots != user

real user

- Rare and small corelations
- Variable and small sent emails per day rate
- Email size varies

bot user

- Tightly connected
- Spammers never fully control infected computers
- Higher and steady sent emails rate
- Emails templates

Problems

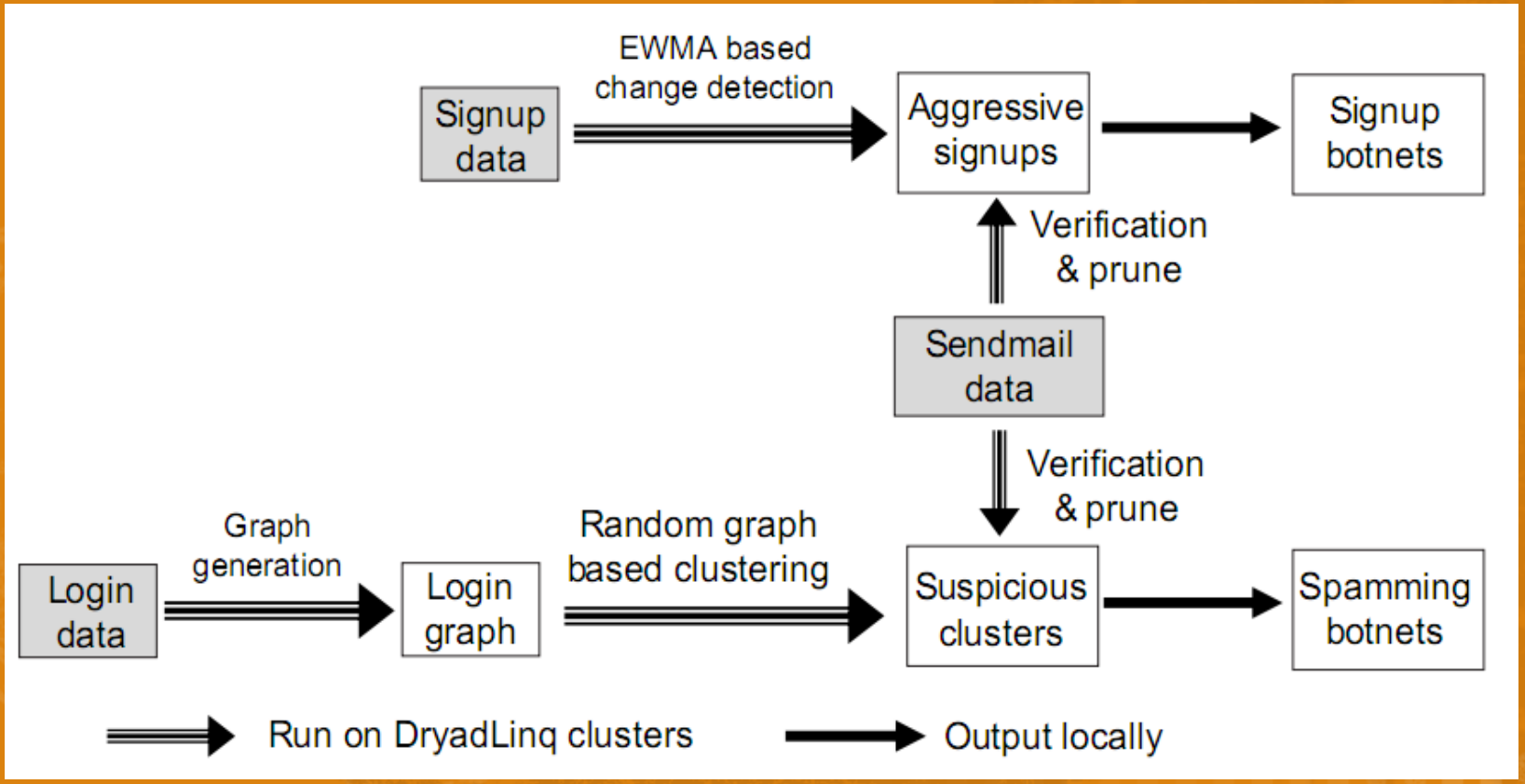
but...

real user

- mobile users, proxies and dynamic ips
- average is not every
- false positive bot classification unwanted

bot user

- stealthy
- possible counter technics



BotGraph architecture

User login graph

simple

- **bot-users login behaviour**
- **user login graph**
  - vertices - email accounts
  - edges - login from same ip address (ip-day)
- **sharing ip address**
  - single bot handles ~50 bot-users
  - single bot-user assigned to many bots over time
- **autonomous systems metric vs dynamic ips and proxies**

Giant connected component

- **random graph theorem**
  - average degree d = n*p
  - d < 1 => size = O(log n)
  - d > 1 => size = O(n)
- **bot-users forms giant connected component**
- **normal users' connected components are small (less then 100 nodes)**
- **components varies with sizes**
- **bot-users nets may intersect**
- **hierarchical extraction (increasing edges weight connection threshold)**

legitimate users pruning

- **based on the number of sent emails per day**
  - less then 10% users, sent more then 3 emails/day
  - BotGraph consider only nodes, where at least 80% of users sent more then 3 emails/day
- **validation based on emails size, account naming pattern**
  - **much more effective with users' groups analising**

Graph construction & analysis

- **Huge size**
  - over 500 milions of login data in one month (220GB)
    - userid, ip address, login timestamp
  - number of edges - hundreds of billions
- **240 machine cluster**
  - 1.5 hours
  - Dryad/DryadLINQ
- **Finding connected component**
  - simple divide and conquer
  - 7 minutes on cluster vs 4 hours on single computer

Two methods

i.e. "first didn't work"

method 1

- **partitioning by login ip address**
- **map phase: outputs an edge for every two users sharing an ip from AS**
- **reduce phase: weight aggregation of edges**

method 2

- **partition by user ID**
- **direct compare users in one partition**
- **generating *local summaries* of used IP-day keys in partition and broadcasting them**
- **upon reciving summary, sending related records**
- **merging recieved answers for broadcasted summaries**

comparison

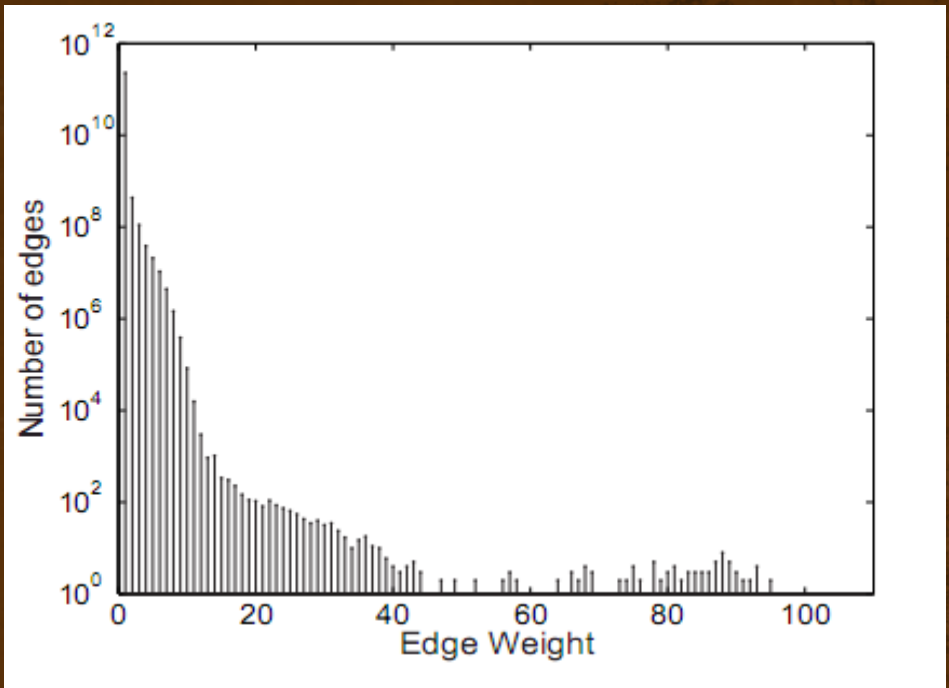
i.e. "why it didn't work"

method 1

- **sending edges of weight one. They can not be ignored**

method 2

- **directly computing edge of weight *w* or more**



performance

i.e. "how bad it didn't work"

method 1

- 12.0 TB communication
- interrupted 6+ hours
- 2.71 TB, 135 min (subset)
- 1.02 TB, 116 min (compression)

method 2

- 1.7 TB
- 95 min
- 460 GB, 28 min
- 181 GB, 22 min

Results

- **found 40 bot groups in January 2008**
  - botnet size from few houndrdes up to few milions
- **total of 20.58M of bot-users**
  - **16.41M** EWMA - 91.83% new findings
  - **8.68M** graph-based - 54.10% new findings
- **total of 1.84M of bot-IPs**
  - 240 784 EWMA
  - 1.60M graph-based
- **false positive rate estimated: 0.44%**

Questions?