Peer-to-Peer Botnet Detection Using NetFlow

Connor Dillon

System and Network Engineering
University of Amsterdam

Master thesis presentation, July 3rd 2014

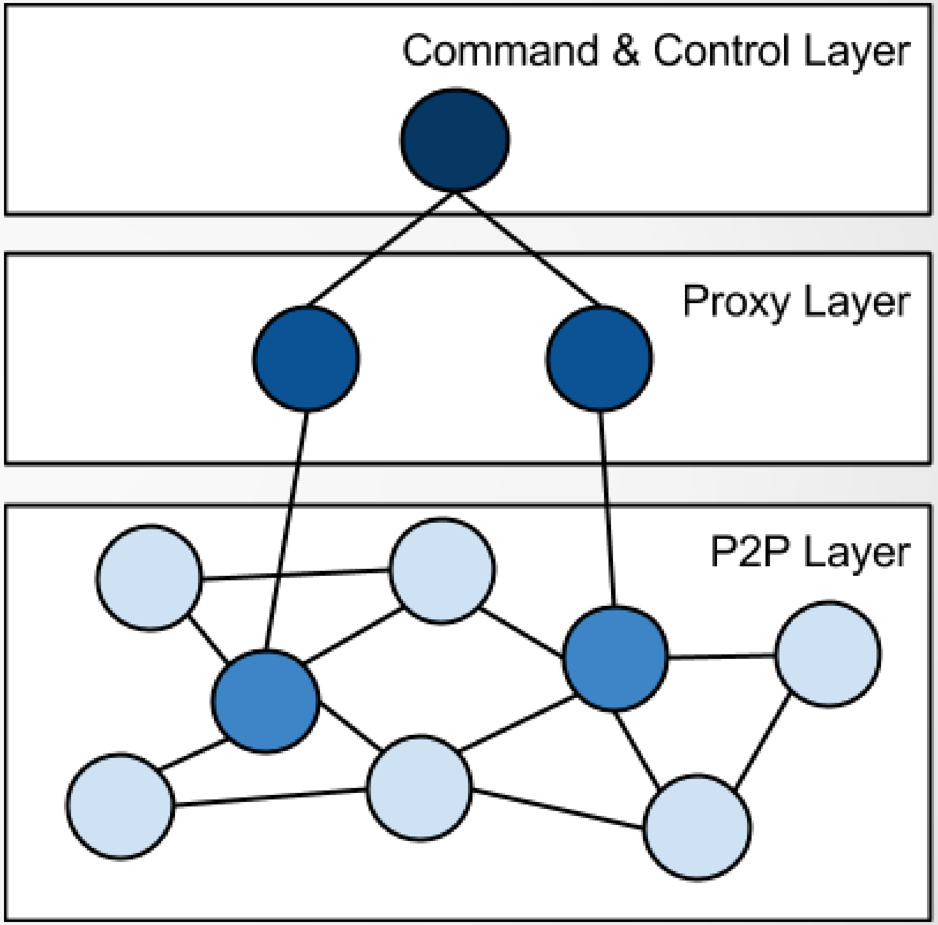
Supervisor: Pepijn Janssen
RedSocks

Botnets

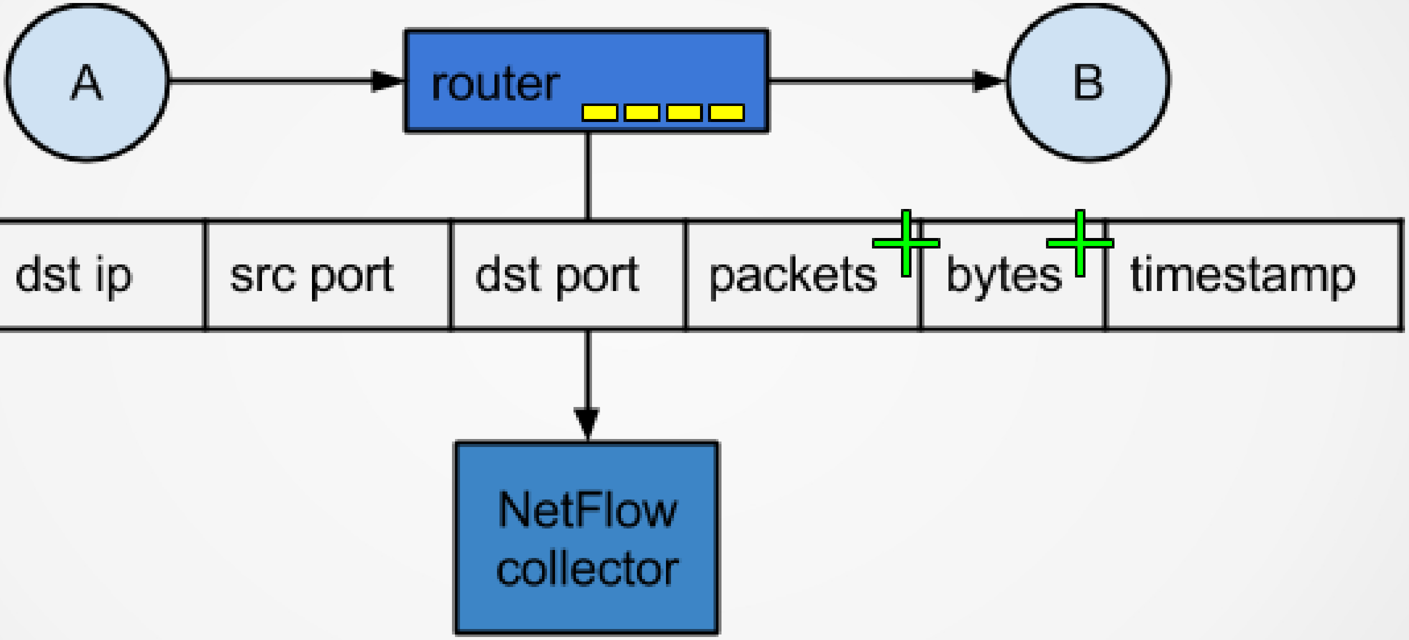
- Large group of infected computers, controlled by a criminal organization
  - Bots harvest information
  - Perform DDoS attacks
- Command & Control (C&C) botnets
  - Centralized architecture
  - C&C servers are weak point
- Peer-to-peer (p2p) botnets
  - P2p architecture
  - More robust
  - More stealthy

Zeus P2P Malware (aka Zeus Gameover)

- Trojan horse
- Financial fraud
- Botnet takedown on June 2nd 2014
    - P2P layer remains active



NetFlow



A

router

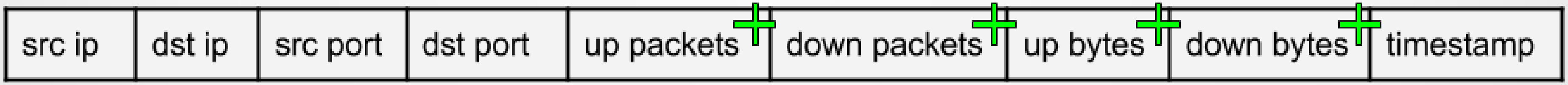
B

src ip	dst ip	src port	dst port	packets	bytes	timestamp

NetFlow collector

IP Flow Information Export

- IP Flow Information Export (IPFIX)
  - NetFlow v10
  - IETF RFCs 7011 through RFC 7015
  - Bidirectional flows RFC 5103



src ip	dst ip	src port	dst port	up packets +	down packets +	up bytes +	down bytes +	timestamp

Research Question

Can p2p bots be detected effectively by analyzing traffic flow data?

Related Research

- An Analysis of the Zeus Peer-to-Peer Protocol
  - Dennis Andriesse and Herbert Bos
  - Technical Report IR-CS-74, VU University Amsterdam, 2014
- Are Your Hosts Trading or Plotting? Telling P2P File-Sharing and Bots Apart
  - Ting-Fang Yen and Michael K. Reiter
  - Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference
- BotSuer: Suing Stealthy P2P Bots in Network Traffic through Netflow Analysis
  - Nizar Kheir and Chirine Wolley
  - Cryptology and Network Security vol. 8257, 2013

Approach

1. Acquire samples of active p2p malware
2. Install samples and capture NetFlow data of malicious traffic
3. Acquire NetFlow data of benign traffic
4. Analyze benign and malicious p2p traffic and find key differences
5. Design detection algorithm
6. Implement detection algorithm (Proof of Concept)
7. Test for false/true positives

Data Set: Benign Traffic

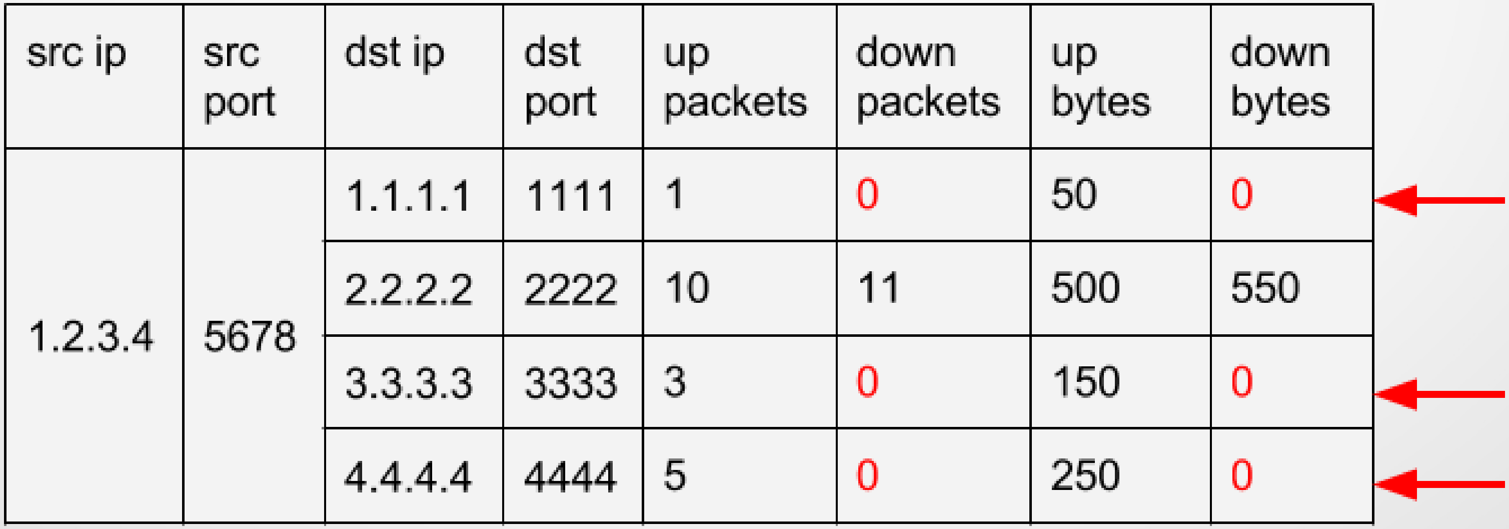
- Data generated specifically for this research
  - Web browsing traffic
  - Web streams
  - p2p traffic:
    - Multiple clients: uTorrent
    - FrostWire: BitTorrent
    - Bearshare: gnutella
    - iMesh: IM2Net
    - Ares Galaxy: own supernode/leaf protocol
    - Emule: eDonkey & Kademlia
    - Shareaza: multiple protocols

Data Set: Malicious Traffic

- Obtained active samples of Zeus P2P malware from public sandbox
- Installed samples in lab environment and captured traffic
- Data set contains:
  - Traffic from 3 different Zeus P2P binaries
  - Packet Captures (pcaps) of 100 mins, 2 hours and 12 hours

Isolating P2P Traffic

- UDP p2p protocols initiate connections from a single source port
- Peers try to connect to peers that are unreachable
- Result: lots of failed connections, to multiple destinations, from a single source IP/port

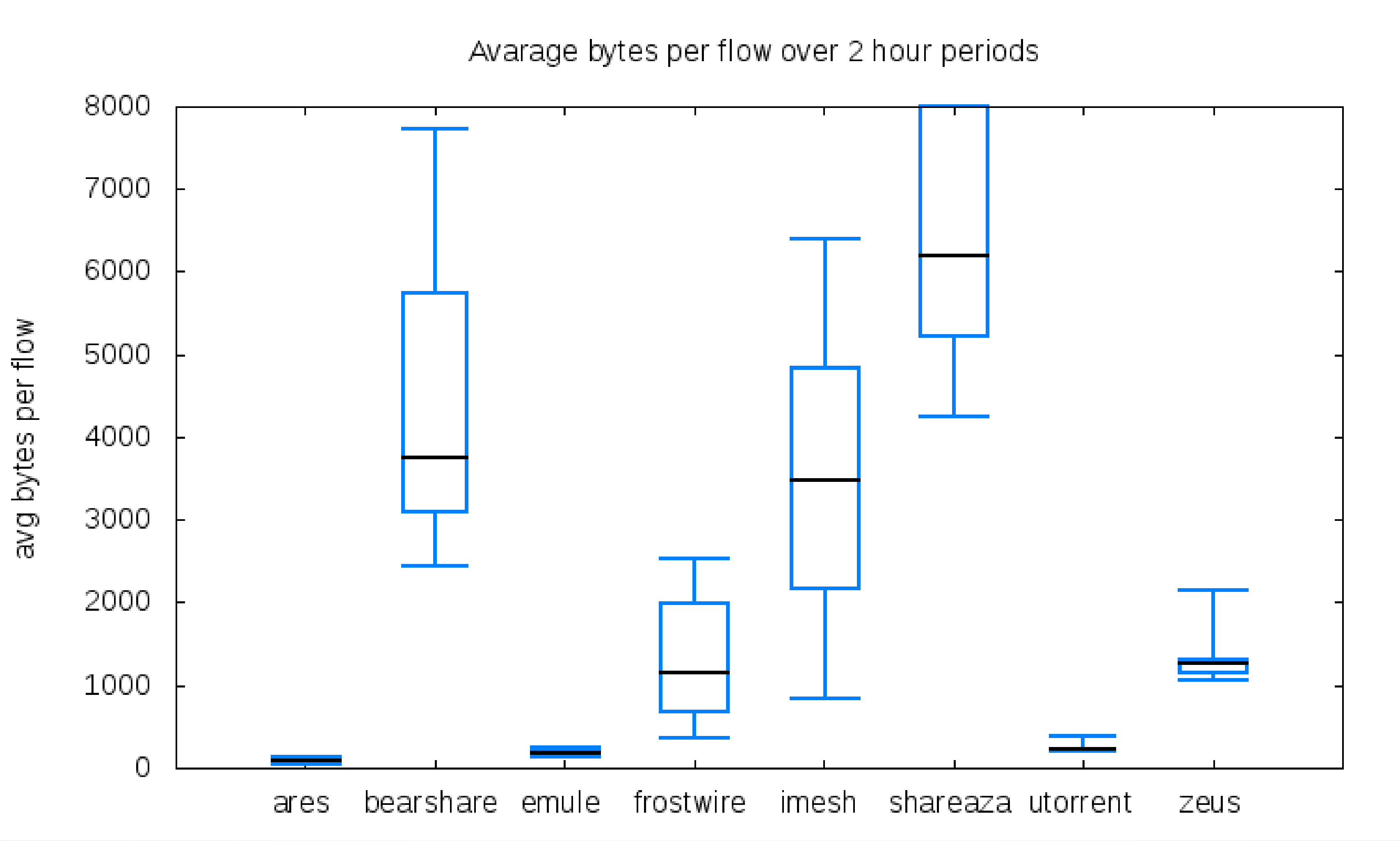


src ip	src port	dst ip	dst port	up packets	down packets	up bytes	down bytes
1.2.3.4	5678	1.1.1.1	1111	1	0	50	0
		2.2.2.2	2222	10	11	500	550
		3.3.3.3	3333	3	0	150	0
		4.4.4.4	4444	5	0	250	0

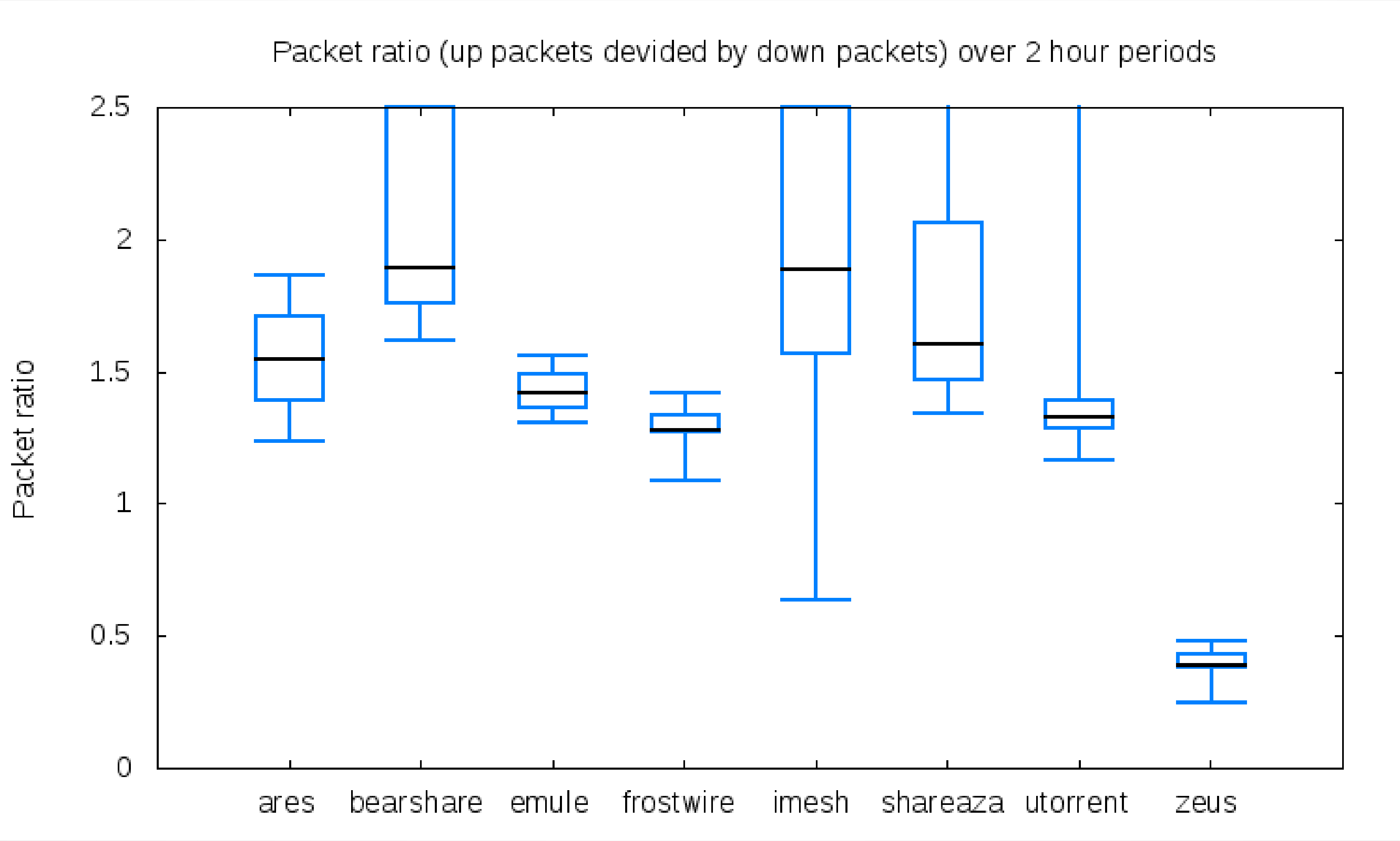
Benign vs Malicious: Finding Differences

- Per application, split up data in to 2 hour chunks
- Analyze
  - Amount of traffic generated
  - Average bytes/packets per flow
  - Protocol characteristics
  - Traffic patterns
  - Etc.

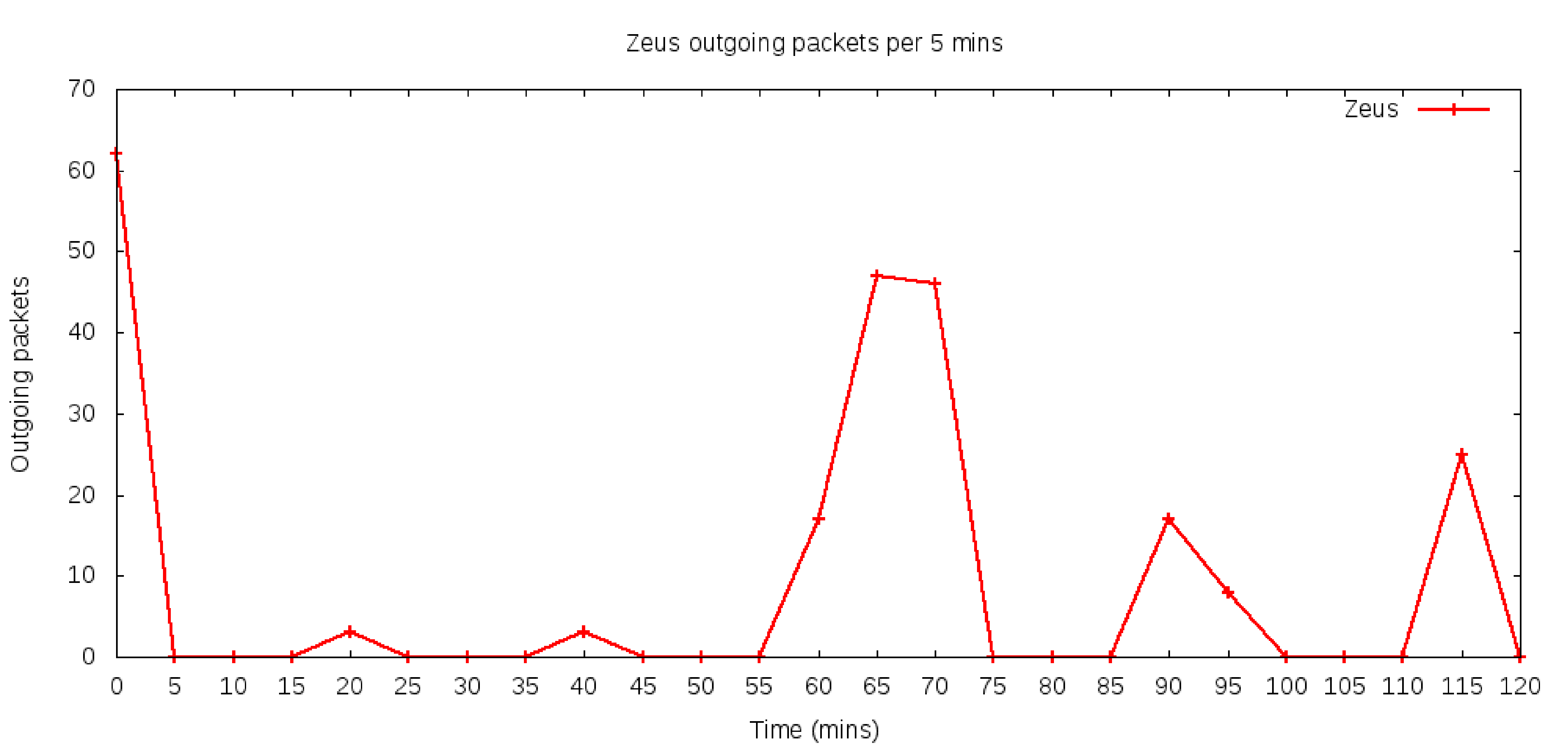
Benign vs Malicious: Traffic Volume



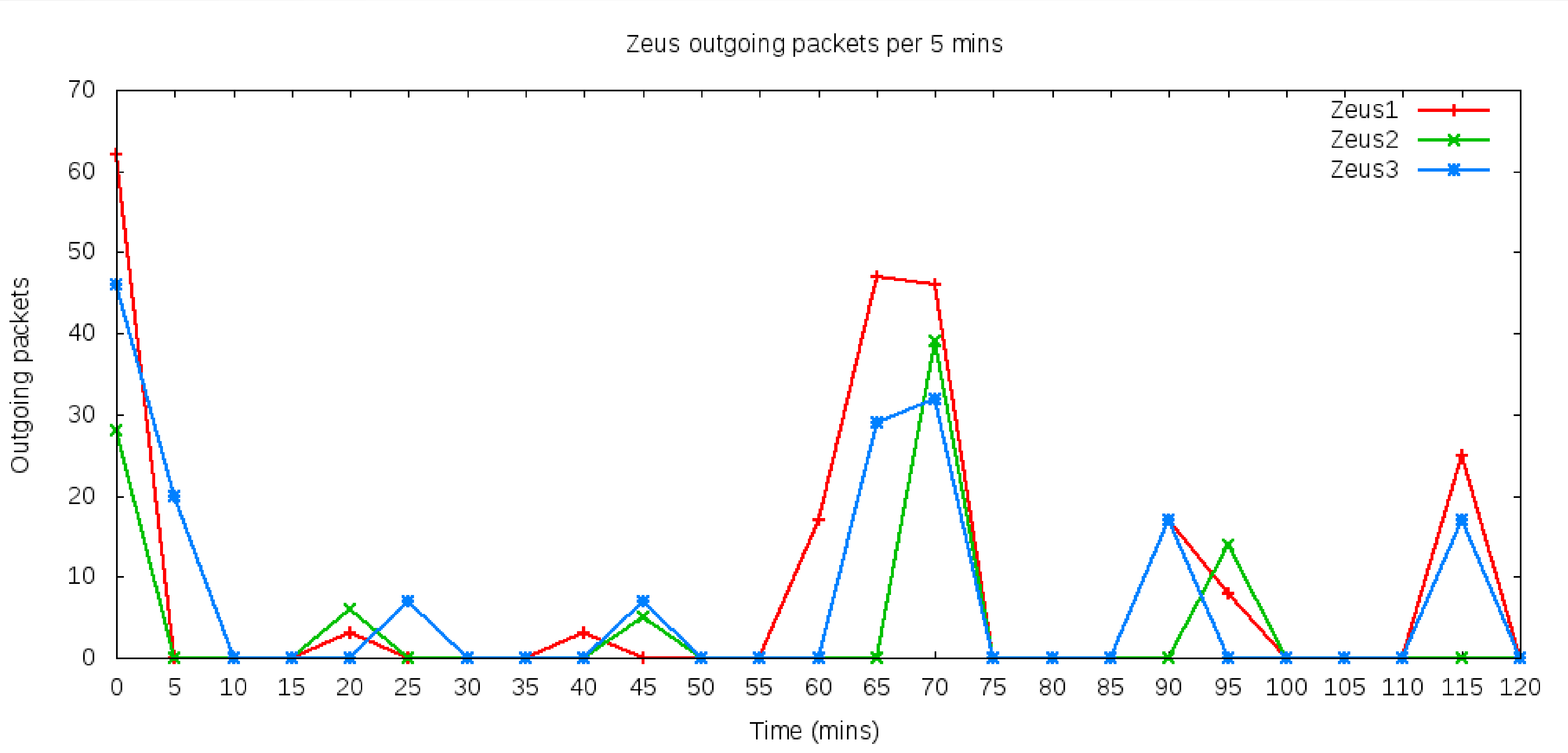
Benign vs Malicious: Packet Symmetry



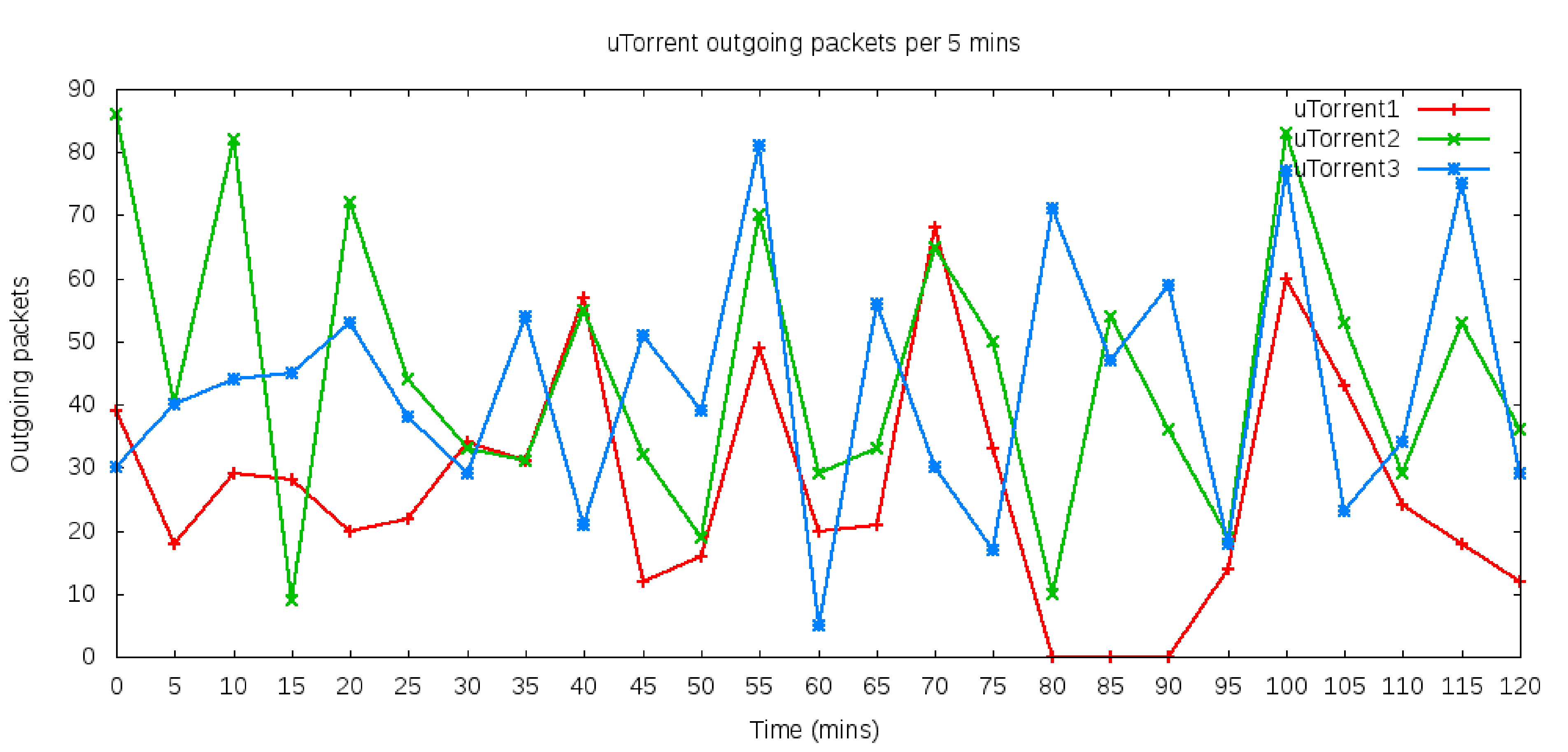
Benign vs Malicious: Traffic Pattern



Benign vs Malicious: Traffic Pattern



Benign vs Malicious: Traffic Pattern



Detection Algorithm

- Group all flows by source IP/port
- Sources with more than 3 failed flows to different hosts are marked as p2p
- Zeus p2p traffic is identified by either:
  - A packet ratio of less than 0.4
  - A traffic pattern of more than 3 approximately equal intervals of time of more than 5 mins

Detection Algorithm

```
def p2p_detect(flows):
    unreachables = set(
        flow.dst_ip
        for flow in flows
        if flow.up_pkts > 0 and flow.down_pkts == 0
    )

    if len(unreachables) > 3:
        return True


def zeus_ratio_detect(flows):
    up = sum(flow.up_packets for flow in flows)
    down = sum(flow.down_packets for flow in flows)

    if up / down > 0.4:
        return True
```

Detection Algorithm

```
def zeus_pattern_detect(flows):
    timestamps = list(flow.timestamp for flow in flows)
    intervals = list()

    previous_timestamp = timestamps[0]

    for timestamp in timestamps:
        if timestamp - previous_timestamp > 300:
            intervals.append(timestamp – previous_timestamp)

        previous_timestamp = timestamp

    if len(intervals) > 3:
        if stdev(intervals) < 150:
            return True
```

Proof of Concept

- NetFlow collector with detection algorithm implemented in Python
  - code will be available on GitHub
- Tested without false positives on available data
- Detects the Zeus P2P malware

Conclusion

- It's possible to detect p2p malware using flow data
  - Malware could change its behavior to avoid detection
- Detection algorithm:
  - Packet symmetry is probably specific to Zeus protocol
  - Traffic pattern might also be applicable to other malware
- Future research:
  - Other p2p malware
  - Testing more (real) benign p2p data for false positives

Thank you

Questions?