Henry Clausen, Robert Flood, David Aspinall

Traffic generation using Containerization for Machine Learning

THE UNIVERSITY *of* EDINBURGH
informatics

EPSRC
Pioneering research and skills

The Alan Turing Institute

Contribution

ML applications offer significant progress to intelligent network security

Development currently held back by:

- lack of real-world data
- limitations of synthetic testbeds

We developed **Detgen**:

- Traffic generation suite based on containerized applications
- Four main improvements through program isolation and container independence

Agenda

(1) ML and network data

(2) Problems in current datasets

(3) Containerization

(4) Traffic generation suite

(5) Example use-case

(6) Limitation & conclusion

Agenda

(1) **ML and network data**

(2) Problems in current datasets

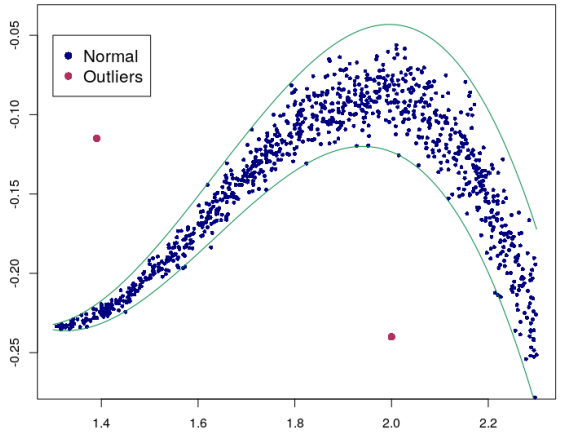
(3) Containerization

(4) Traffic generation suite

(5) Example use-case

(6) Limitation & conclusion

ML in Network Security

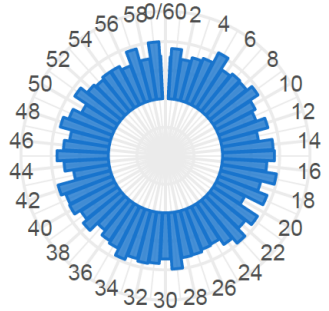


Intrusion detection:

- Misuse detection
- Anomaly detection
- Signature mining
- Temporal models
- …

Other applications:

- Traffic classification
- Vulnerability discovery
- Protocol verification
- …



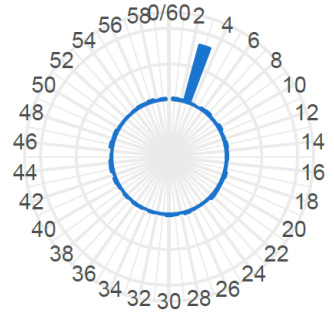
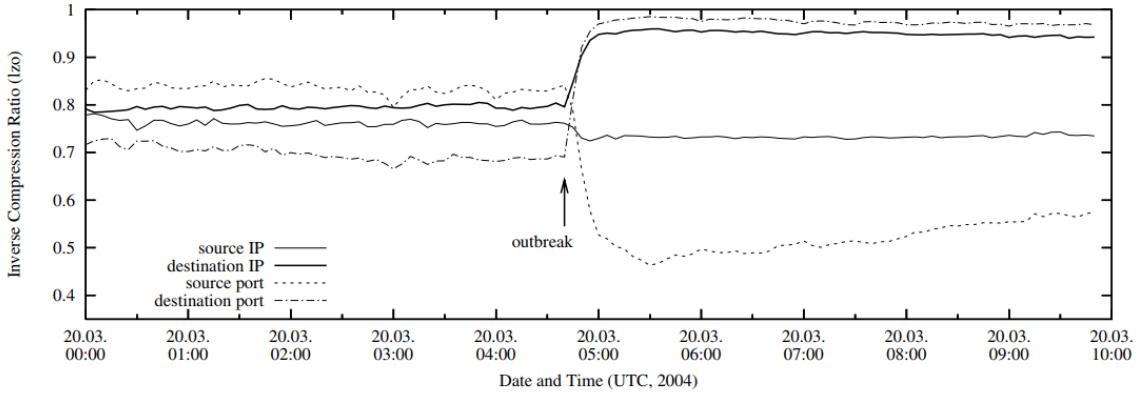
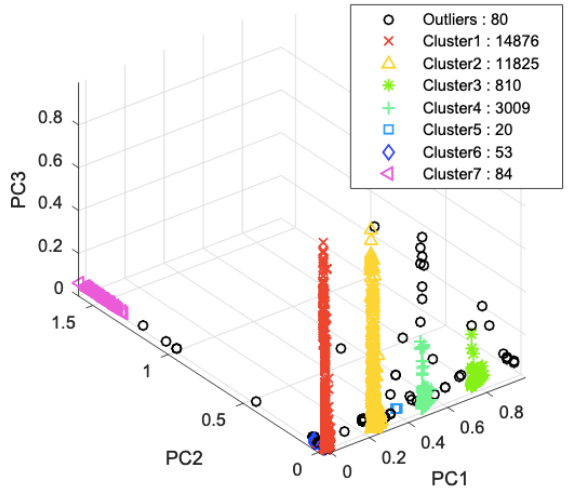


Figure 1. Blaster - TCP address parameter compressibility



Wagner et al., 2005



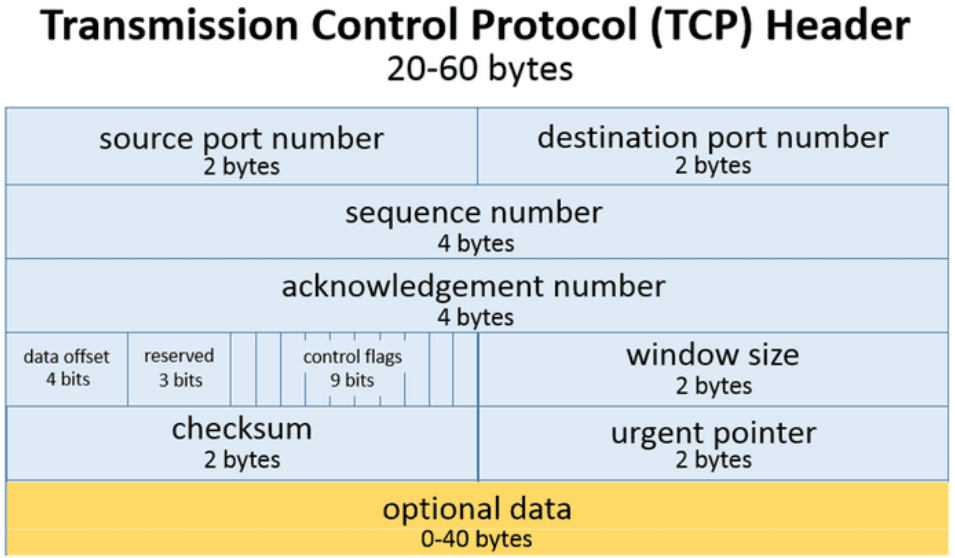
Yen et al., 2012

THE UNIVERSITY of EDINBURGH
informatics

Data

ML heavily dependent on datasets:

- Design
- Model training
- Evaluation



Format:

- raw packets
  - payload
  - meta-data
- network flow

```
Date flow start          Duration Proto  Src IP Addr:Port
2010-09-01 00:00:00.459    0.000 UDP    127.0.0.1:24920    ->
2010-09-01 00:00:00.363    0.000 UDP    192.168.0.1:22126 ->

                     Dst IP Addr:Port   Packets  Bytes Flows
                ->   192.168.0.1:22126      1     46     1
                ->   127.0.0.1:24920        1     80     1
```

THE UNIVERSITY of EDINBURGH
informatics

Data

Challenges for a dataset:

- lack of standardized conditions
- concept drift
- heterogeneity of traffic
- scarcity of attack traffic

Furthermore:

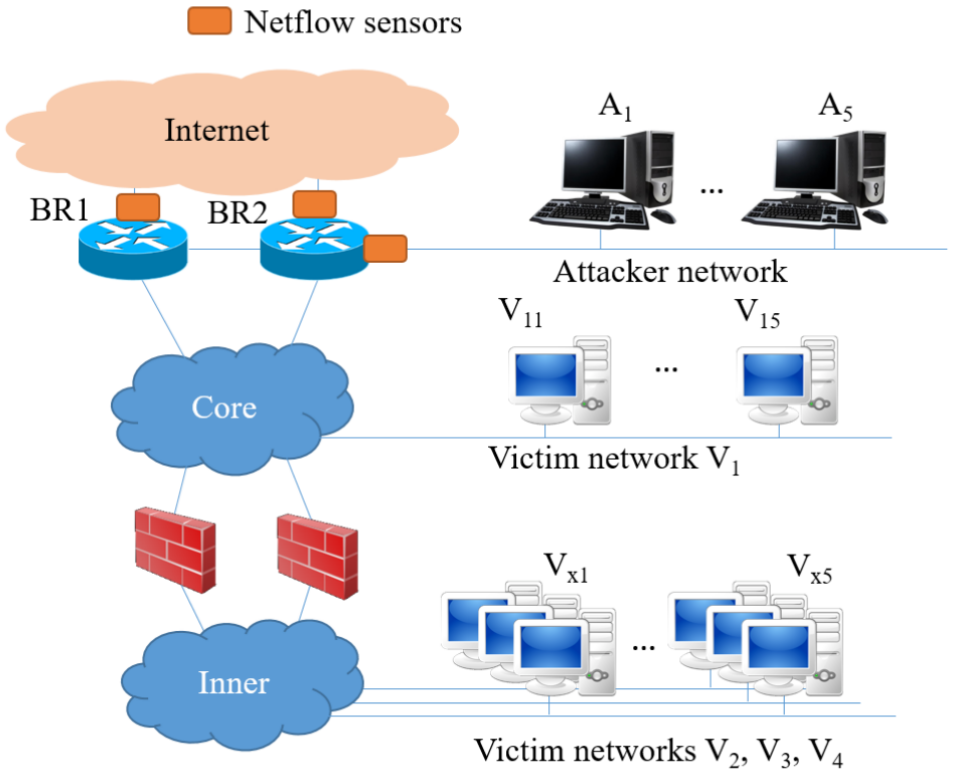
- privacy & security concerns
- attack isolation not guaranteed

→ lack of suitable real-world datasets

Synthetic dataset generation

Testbeds:

- Virtual machines arranged in isolated network
- Important services follow scripted tasks
- Data collected at router



UGR-16 testbed

Datasets: CIC-IDS 17, ISCX, UGR 16, UNSW-NB 15
Older: KDD 99, DARPA 98

THE UNIVERSITY of EDINBURGH informatics

Agenda

(1) ML and network data

(2) **Problems in current datasets**

(3) Containerization

(4) Traffic generation suite

(5) Example use-case

(6) Limitation & conclusion

Problems with current datasets

Successful ML application is held back by:

- Low traffic variation
- Lack of ground truth labels
- Limited size
- Static generation

Low traffic variation

Scripted activities:

- little exploration of protocol range

Example CIC-IDS 17:

- 99% of FTP-connections download same file (’wikipedia/encryption’)

Technical variations neglected:

- network congestion
- faults
  - out-of-order arrivals
  - connection restart
  - ….

Low traffic variation

Low variation leads to:

- homogenous data:
- models that do not generalise/overfit
- overoptimistic detection rates

Lack of ground truth

Association between traffic events and generating activity often impossible!

- Multiple programs on one machine
- No port binding

Static design

Set-up and capture performed once

Updates for specific protocols not possible

→ outdated traffic

→ no robustness to concept drift

Limited size

Traffic for individual protocols can be very small!

Protocol	UNSW-NB 15	CIC-IDS 17	ISCX	MAWI
HTTP	196195	276405	2372	156179
SSL	540	285760	141	591551
DNS	372748	1820105	200009	1581858
X509	459	2758590	331	Unknown
FTP	111685	5540	1989	278
SSH	31320	5600	434	5503
IRC	202	0	27	Unknown
SMTP	44455	0	125	4601

Agenda

(1) ML and network data

(2) Problems in current datasets

(3) **Containerization**

(4) Traffic generation suite

(5) Example use-case

(6) Limitation & conclusion

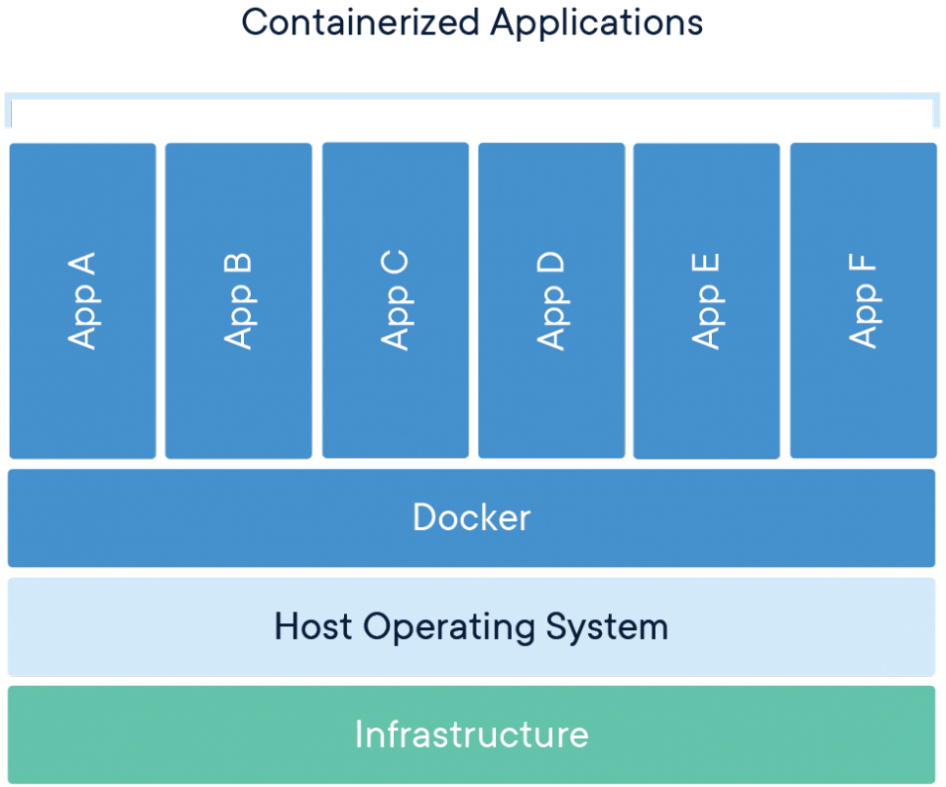
Containerization

Programs/process as standalone virtualised standard units

Advantages:

- lightweight
- runs uniformly
- safe through isolation

Containers can be arranged in virtual networks



Agenda

(1) ML and network data

(2) Problems in current datasets

(3) Containerization

(4) **Traffic generation suite**

(5) Example use-case

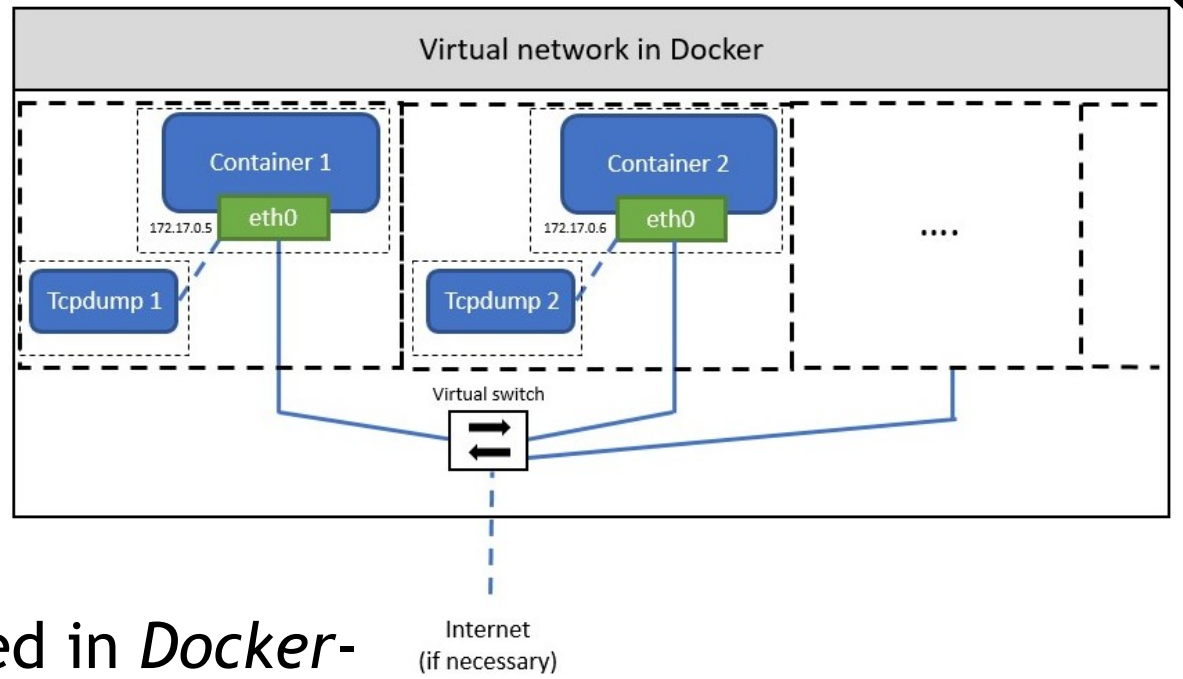
(6) Limitation & conclusion

Design principles

- High degree of traffic variation
- Ground truth labels through activity isolation
- Scalability
- Modularity

Capture scenario

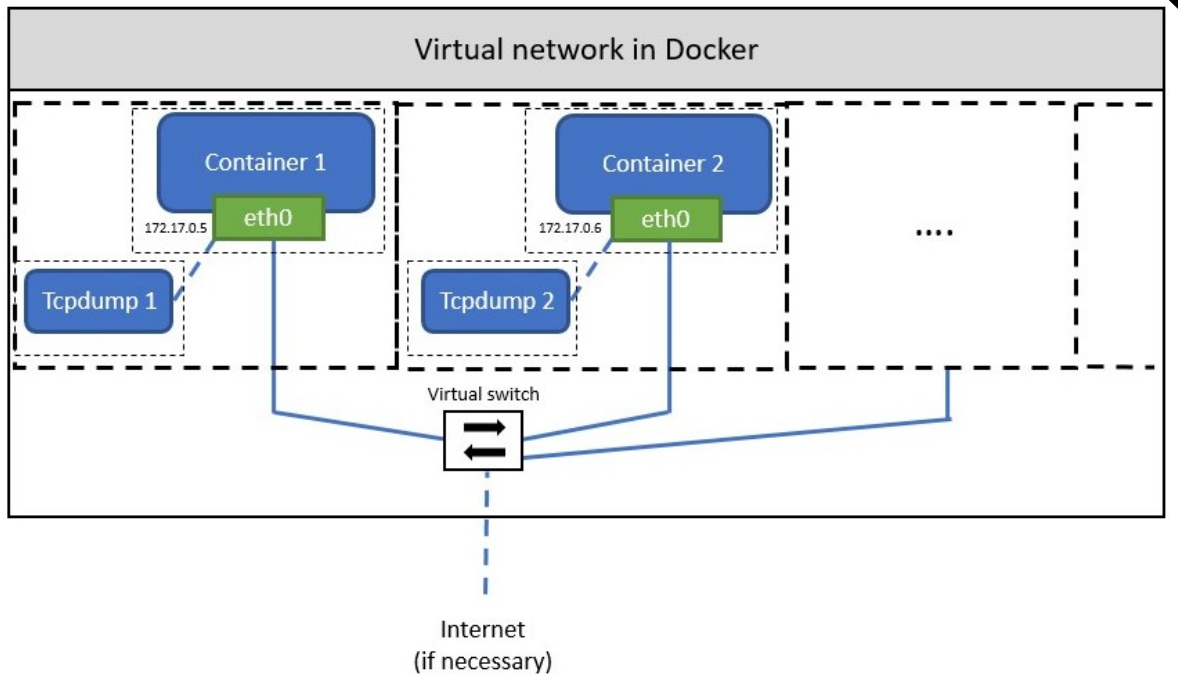
Arrangement of set of containers to provide a service and perform specific activity



- Each scenario saved in *Docker*-compose file
- Execution follows execution-script
- Traffic from several scenarios coalesced into whole dataset

Capture scenario

Arrangement of set of containers to provide a service and perform specific activity

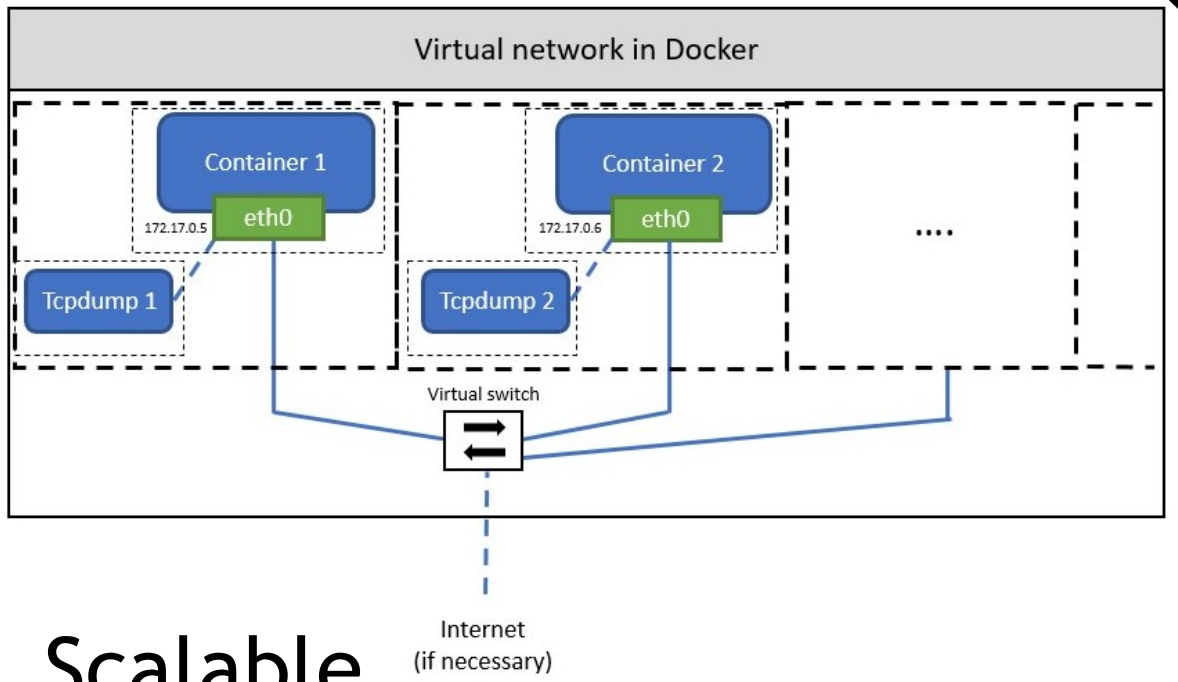


Ground truth

- traffic captured for each container
- scenario follows an execution script
- virtualisation shields from external influence

Capture scenario

Arrangement of set of containers to provide a service and perform specific activity



Modular

- scenarios independent of each other
- easy to add and update scenario

Scalable

- repeatable & consistent
- independent of host system
- lightweight

THE UNIVERSITY of EDINBURGH informatics

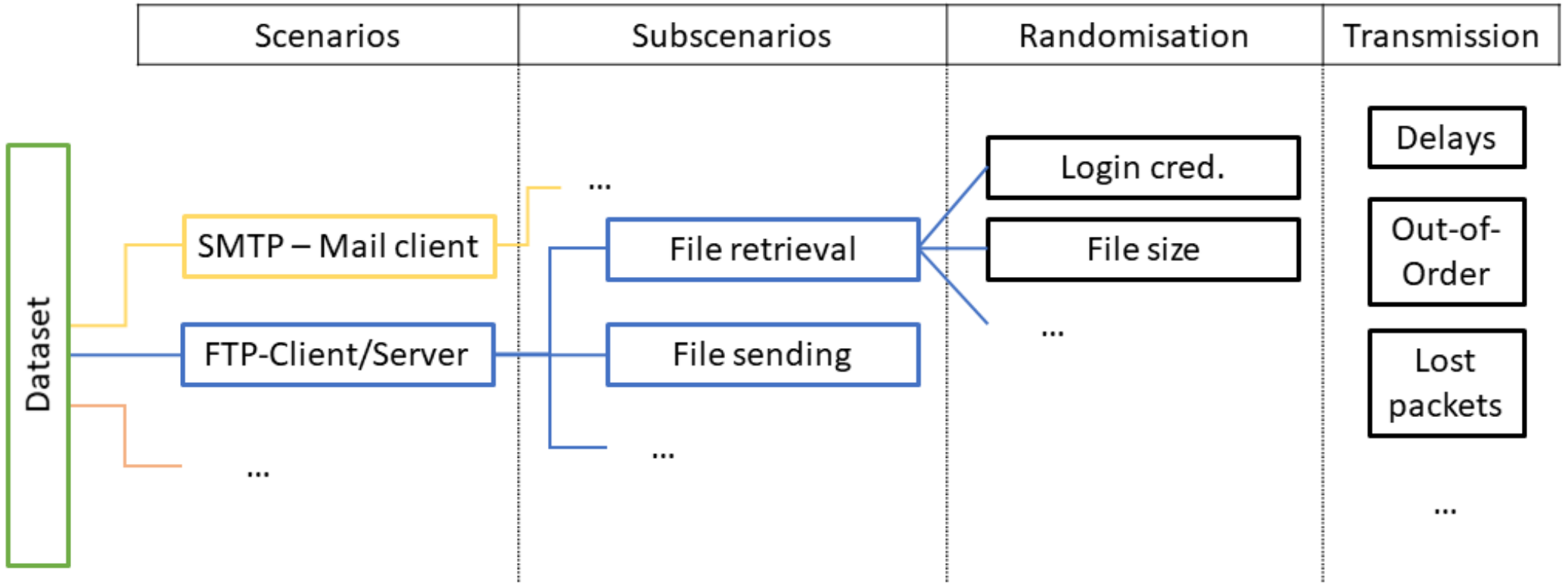
Variation is achieved through …

Exploration of service range
- Different tasks (file retrieval, sending, …)
- Login-failures, wrong file, …

TC/NetEm
- artificial packet delays, corruption, drops
- calibrated to emulate WAN characteristics

Input randomisation
- passwords
- transmitted files
- bash-commands
- …

Variation is achieved through …



Scenario implementation

(1) select primary and secondary containers

(2) identify different subscenarios of service

(3) identify variable input values and appropriate ranges

(4) create *Docker*-compose file

(5) write execution script

Current scenario suite

Further extension planned!

Scenario	#Sub	Scenario	#Sub	Scenario	#Sub
Ping	1	File-sync	6	Time sync	3
Web server	4	SMTP	5	Music stream	5
SSH	7	IRC	2	Video stream	1
FTP	12	BitTorrent	4	WAN wget	5
Web scraper	2	SQL database	4		
SSH bforce	3	Goldeneye DoS	1	Heartbleed	1
URL fuzz	1	Slow DoS	4	Backdoor	3
Auth bforce	2	Mirai	3	XXE	3
SQL-injection	2	Traffic relay	5	Crypto-miner	1

Agenda

(1) ML and network data

(2) Problems in current datasets

(3) Containerization

(4) Traffic generation suite

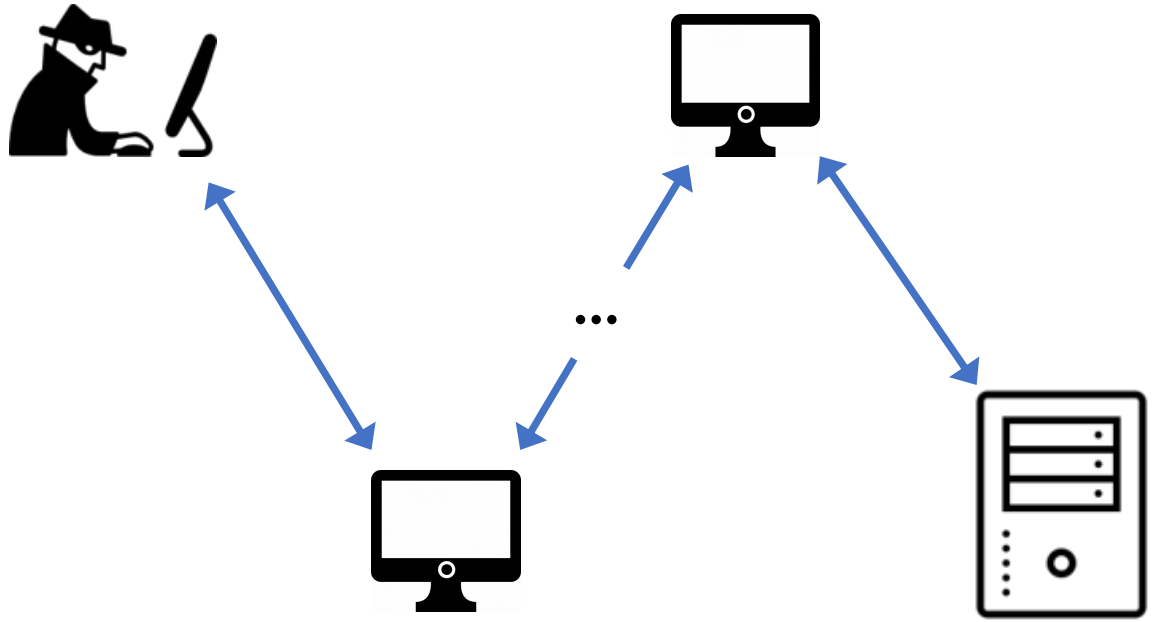
(5) **Example use-case**

(6) Limitation & conclusion

Example – stepping stone detection

Relayed attack to hide origin of attacker

Connection pair correlation for detection



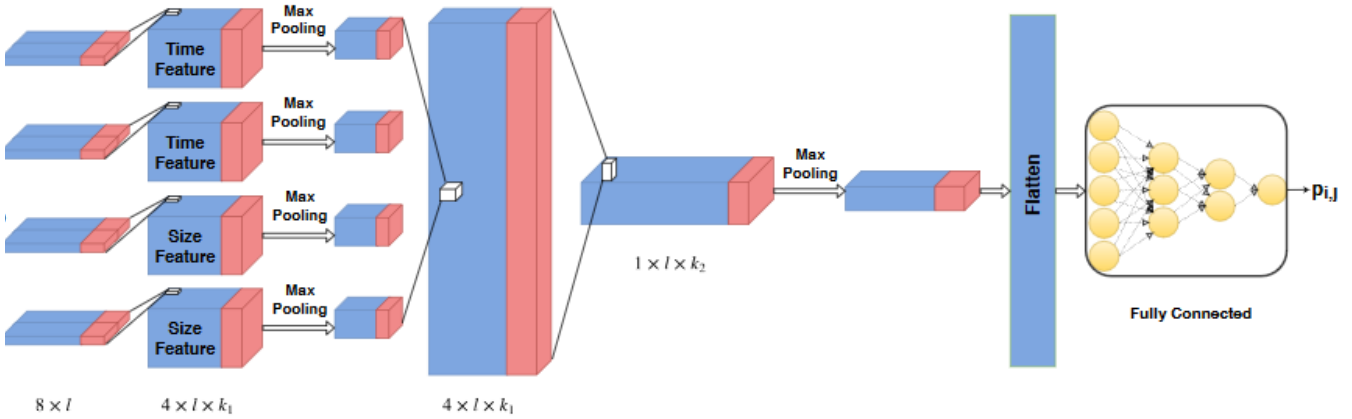
Example – stepping stone detection

Aim:

- train Conv. NN to detect correlation

Problems:

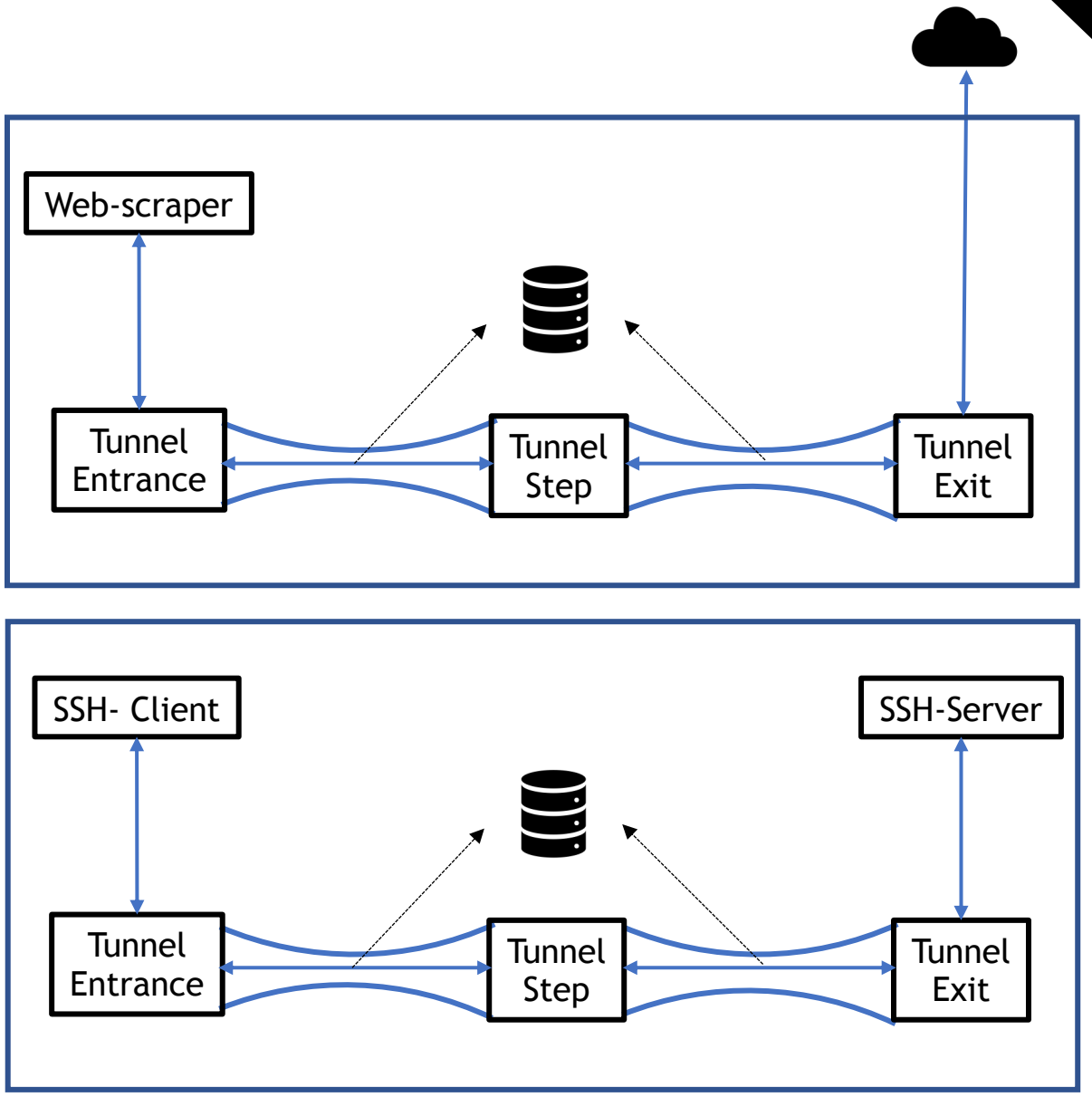
- needs a lot of data
- prone to overfitting
- different noise levels for evaluation



DeepCorr 2018, Nasr et al.

Docker data

- 50,000 connection pairs
- 3 different scenarios
- randomized input/congestion
- varied noise levels with labels



Agenda

(1) ML and network data

(2) Problems in current datasets

(3) Containerization

(4) Traffic generation suite

(5) Example use-case

(6) **Limitation & conclusion**

Limitations

Not replicated well:

- Network-wide distribution
- long-term temporal structures

Data volume huge

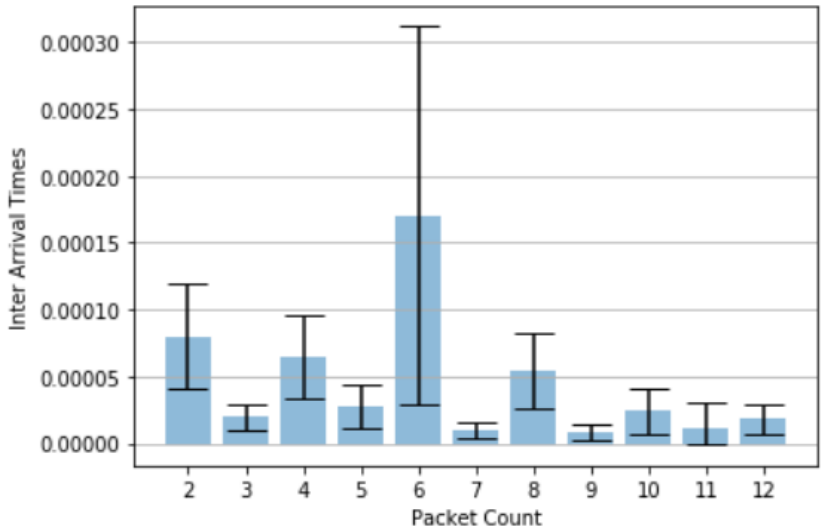
- preprocessing required

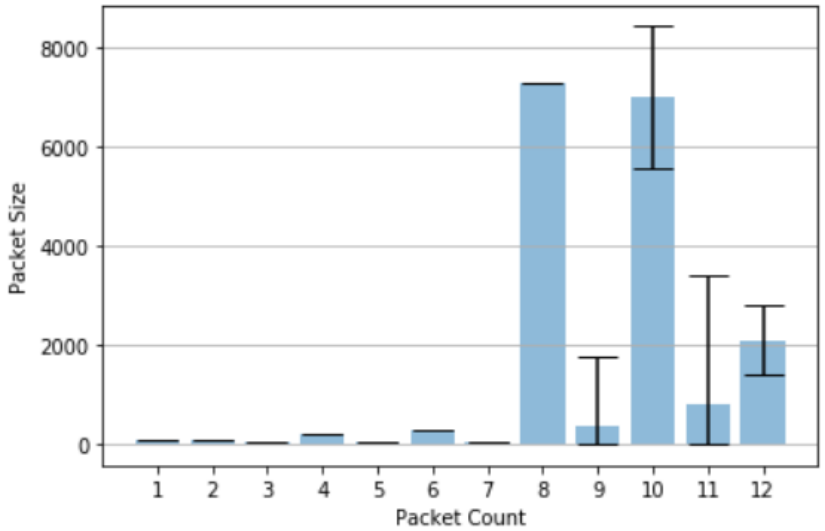
Manual implementation

Conclusion

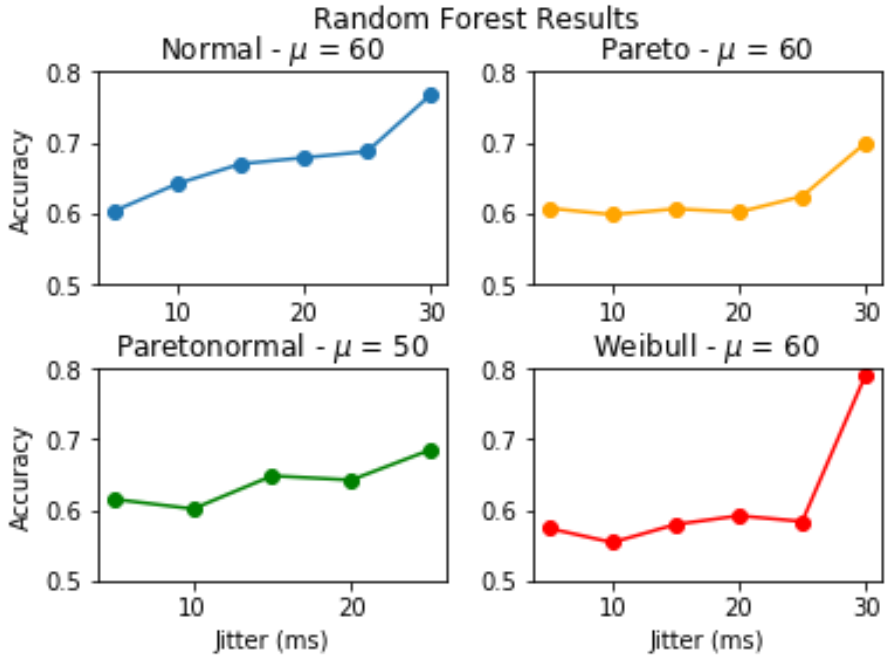
- Our traffic generation suite fuels ML through:
  - High degree of traffic variability
  - Ground truth labels through activity isolation
  - Scalability
  - Modularity
- github.com/detlearsom/detgen/
- Future work:
  - capture of syslogs
  - streamlined data coalescence

Reproducability





WAN-emulation



Distribution	Mean	Jitter	RF Accuracy
No Delays (Baseline)	0	0ms	0.8176
Constant Delay	40ms	0ms	0.6730
Normal	60ms	5ms	0.6028
Pareto	60ms	10ms	0.5979
Paretonormal	50ms	10ms	0.6015
Weibull	60ms	10ms	0.5540

THE UNIVERSITY of EDINBURGH
informatics

```
version: '2'
services:
  vsftpd:
    image: 'detlearsom/vsftpd'
    networks:
      capture:
        ipv4_address: 172.16.238.15
    volumes:
      - '$PWD/users:/home/vsftpd'
    environment:
      - FTP_USER=$User
      - FTP_PASS=$Password

  ftp-client:
    image: 'detlearsom/ftp-client'
    volumes:
      - $PWD/dataToShare:/dataToShare:ro
      - $PWD/receive:/usr/src/ftp
      - $PWD/scripts:/usr/src/scripts:ro
    networks:
      capture:
        ipv4_address: 172.16.238.20
    command: tail -F anything
```

```
  tcpdump_vsftpd:
    image: 'detlearsom/tcpdump'
    command: not(ip6 or arp or (udp and (src port 5353 or src port 57621))) -v -w "/data/dump-0
    volumes:
      - '${DATADIR}:/data'
    network_mode: "service:vsftpd"

  tcpdump_ftp-client:
    image: 'detlearsom/tcpdump'
    command: not(ip6 or arp or (udp and (src port 5353 or src port 57621))) -v -w "/data/dump-0
    volumes:
      - '${DATADIR}:/data'
    network_mode: "service:ftp-client"

networks:
  capture:
    driver: "bridge"
    ipam:
      driver: default
      config:
      - subnet: 172.16.238.0/24
        gateway: 172.16.238.1
```

Traffic example without chaff and delays

0.000000	FTP	Tunnel	74	34104 → 21 [SYN] Seq=0 Win=64240
0.010162	Tunnel	FTP	74	21 → 34104 [SYN, ACK] Seq=0 Ack=
0.010205	FTP	Tunnel	66	34104 → 21 [ACK] Seq=1 Ack=1 Wir
0.074863	Tunnel	FTP	86	Response: 220 (vsFTPd 3.0.2)
0.074887	FTP	Tunnel	66	34104 → 21 [ACK] Seq=1 Ack=21 Wi
0.074978	FTP	Tunnel	80	Request: USER mcdbmar
0.085066	Tunnel	FTP	66	21 → 34104 [ACK] Seq=21 Ack=15 W
0.115763	Tunnel	FTP	100	Response: 331 Please specify the
0.115822	FTP	Tunnel	83	Request: PASS f6BPRvk2rb
0.160513	Tunnel	FTP	89	Response: 230 Login successful.
0.160559	FTP	Tunnel	71	Request: PWD
0.201428	Tunnel	FTP	75	Response: 257 "/"
0.201527	FTP	Tunnel	72	Request: PASV
0.242470	Tunnel	FTP	117	Response: 227 Entering Passive M
0.262805	FTP	Tunnel	72	Request: LIST
0.303762	Tunnel	FTP	100	Response: 425 Security: Bad IP c
0.303810	FTP	Tunnel	72	Request: QUIT
0.344660	Tunnel	FTP	80	Response: 221 Goodbye.
0.344907	FTP	Tunnel	66	34104 → 21 [FIN, ACK] Seq=55 Ack
0.375502	Tunnel	FTP	66	21 → 34104 [FIN, ACK] Seq=186 Ac
0.375517	FTP	Tunnel	66	34104 → 21 [ACK] Seq=56 Ack=187

Traffic example without chaff and delays

6.120634	Entrance	Stepstone	158	Client: Encrypted pa
6.120954	Stepstone	Entrance	110	Server: Encrypted pa
6.131053	Entrance	Stepstone	66	52404 → 22 [ACK] Sec
6.164963	Stepstone	Entrance	126	Server: Encrypted pa
6.175017	Entrance	Stepstone	66	52404 → 22 [ACK] Sec
6.185255	Entrance	Stepstone	118	Client: Encrypted pa
6.205822	Stepstone	Entrance	134	Server: Encrypted pa
6.226119	Entrance	Stepstone	118	Client: Encrypted pa
6.250599	Stepstone	Entrance	126	Server: Encrypted pa
6.270854	Entrance	Stepstone	110	Client: Encrypted pa
6.291481	Stepstone	Entrance	110	Server: Encrypted pa
6.311829	Entrance	Stepstone	110	Client: Encrypted pa
6.332535	Stepstone	Entrance	158	Server: Encrypted pa
6.373084	Entrance	Stepstone	110	Client: Encrypted pa
6.393772	Stepstone	Entrance	134	Server: Encrypted pa
6.414121	Entrance	Stepstone	110	Client: Encrypted pa
6.434711	Stepstone	Entrance	118	Server: Encrypted pa
6.455201	Entrance	Stepstone	102	Client: Encrypted pa
6.465537	Stepstone	Entrance	138	Server: Encrypted pa
6.475715	Entrance	Stepstone	102	Client: Encrypted pa
6.522321	Stepstone	Entrance	66	22 → 52404 [ACK] Sec

6.121031	Stepstone	Exit	158	Client: Encrypted pacl
6.151449	Exit	Stepstone	110	Server: Encrypted pacl
6.151472	Stepstone	Exit	66	49254 → 22 [ACK] Seq=
6.164850	Exit	Stepstone	126	Server: Encrypted pacl
6.164859	Stepstone	Exit	66	49254 → 22 [ACK] Seq=
6.185375	Stepstone	Exit	118	Client: Encrypted pacl
6.205698	Exit	Stepstone	134	Server: Encrypted pacl
6.226252	Stepstone	Exit	118	Client: Encrypted pacl
6.250482	Exit	Stepstone	126	Server: Encrypted pacl
6.270961	Stepstone	Exit	110	Client: Encrypted pacl
6.291337	Exit	Stepstone	110	Server: Encrypted pacl
6.311961	Stepstone	Exit	110	Client: Encrypted pacl
6.332374	Exit	Stepstone	158	Server: Encrypted pacl
6.373198	Stepstone	Exit	110	Client: Encrypted pacl
6.393660	Exit	Stepstone	134	Server: Encrypted pacl
6.414233	Stepstone	Exit	110	Client: Encrypted pacl
6.434572	Exit	Stepstone	118	Server: Encrypted pacl
6.455318	Stepstone	Exit	102	Client: Encrypted pacl
6.465408	Exit	Stepstone	102	Server: Encrypted pacl
6.465511	Stepstone	Exit	102	Client: Encrypted pacl
6.475557	Exit	Stepstone	102	Server: Encrypted pacl
6.522327	Stepstone	Exit	66	49254 → 22 [ACK] Seq=