

Visualizing Clustered Botnet Traffic Using t-SNE on Aggregated NetFlows



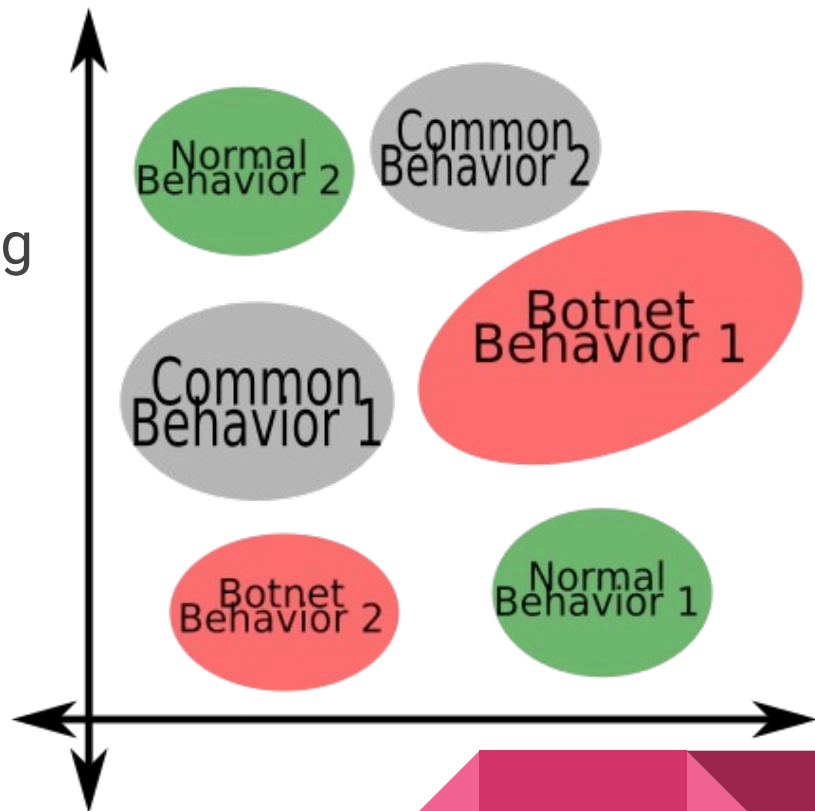
Studying high
volume flood of
network traffic,
droplet by droplet



Objective

Visualize network traffic by clustering similar behaviors together so that researchers can easily analyse and study interesting clusters.

An ideal clustering would look like this figure: →



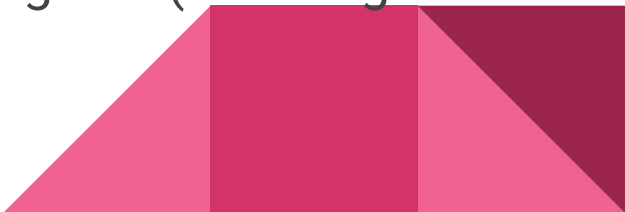


Objective is to visualize
not to detect

Analysis of the visualization might help in
detection

Introduction

Rule-based Detection

- Rule examples:
 - Blacklist of IPs
 - Blacklist of content checksums
 - Require previous knowledge of malware
 - Known previous compromise
 - Can't detect new, different or adaptive malware
 - Botnets are dynamic, their IPs can change (as nodes join and leave)
 - Malware payloads might include random data regions (resulting in changing checksums)
- 

Anomaly Detection

- Hard to define baseline in dynamic changing environment
- Require previous knowledge of environment or baseline

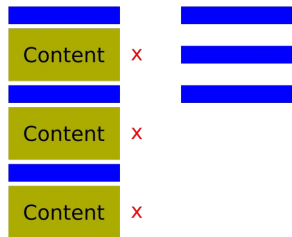


High-level Methodology

Collect

Capture traffic

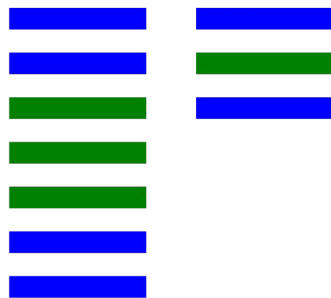
Collect network traffic and store it. In this research we do not rely on contents but rather on the flow.



Aggregate

Group many rows

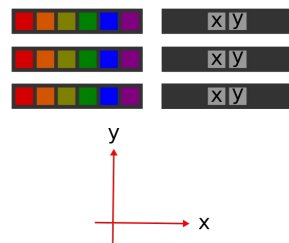
Reducing number of rows into collective figures (total/mean/...)



Embed & Visualize

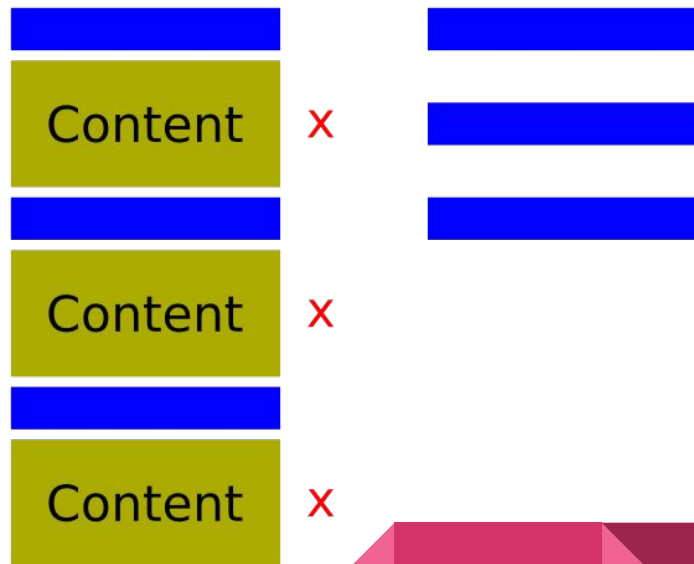
Dimensionality reduction

Reducing number of columns from higher dimensions into 2D or 3D coordinates in order to visualize the data.



Netflows

- Like phone bill
 - Who contacted who
 - When
 - For how long
 - But NOT: what was said



How does Netflows file look like?

We have used the same file format as the data published in:

"An empirical comparison of botnet detection methods" Sebastian Garcia, Martin Grill, Honza Stiborek and Alejandro Zunino. Computers and Security Journal, Elsevier. 2014. Vol 45, pp 100-123. <http://dx.doi.org/10.1016/j.cose.2014.05.011>

It's 15 column CSV file that looks like this:

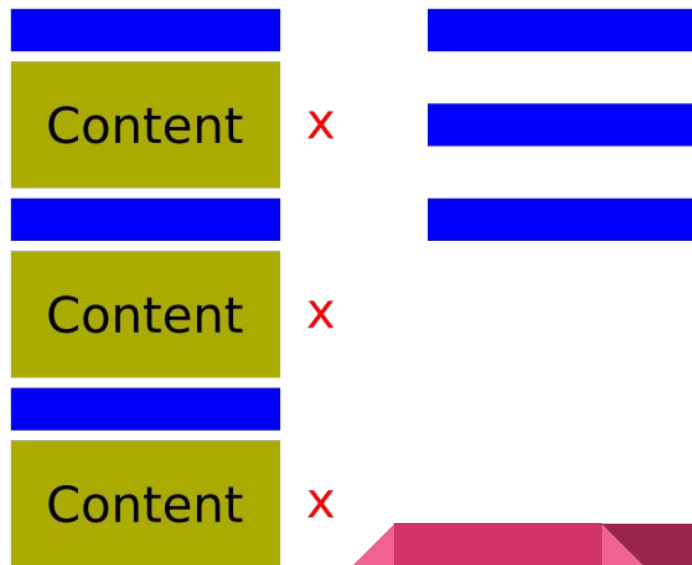
```
StartTime, Dur, Proto, SrcAddr, Sport, Dir, DstAddr, Dport, State,  
sTos, dTos, TotPkts, TotBytes, SrcBytes, Label
```

```
2011/08/10 09:46:53.047277, 3550.182373, udp, 212.50.71.179, 39678,  
<->, 147.32.84.229, 13363, CON, 0, 0, 12, 875, 413,
```

```
flow=Background-UDP-Established
```

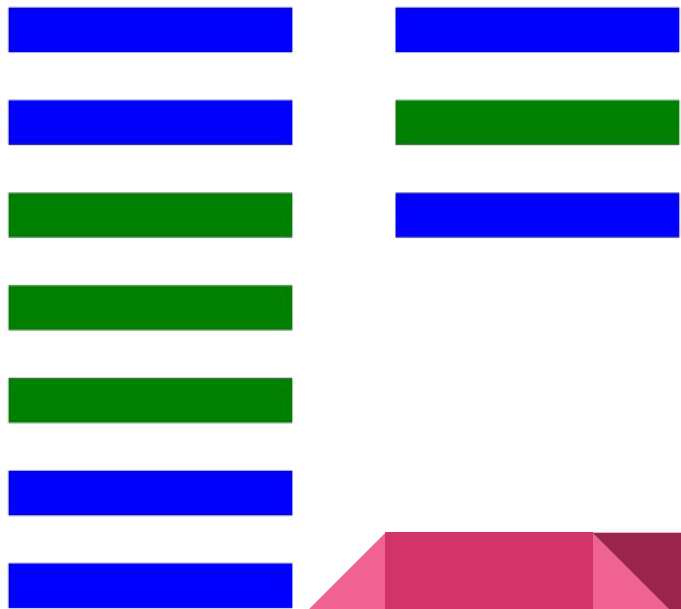
Are Netflows Indicators Effective?

- Without looking at content:
 - periodic, small, fixed-sized, short-duration, single destination
 - Single source, multiple destinations, fixed-size
 - Non-periodic, variable sized, variable duration, ..etc.



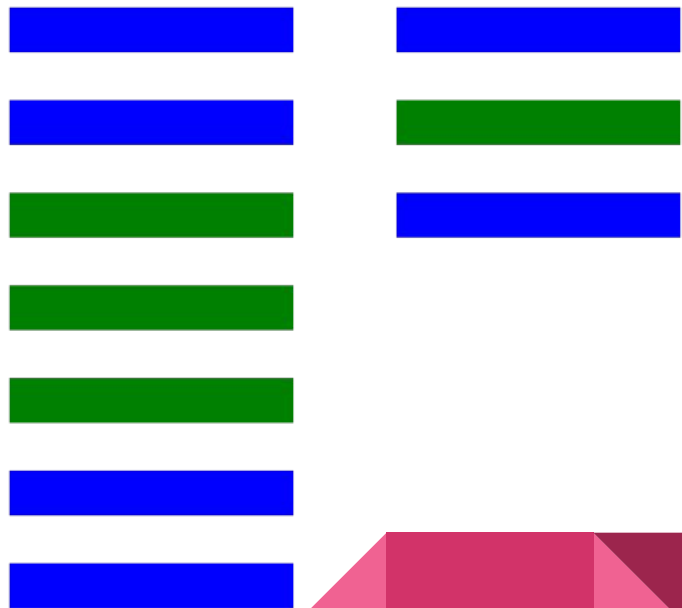
Aggregation of rows

- Taking the input netflows, and picking some aggregation key like source IP address then aggregate some numbers like
 - Count of netflows
 - Average bytes
 - Summation of packets
 - ...etc



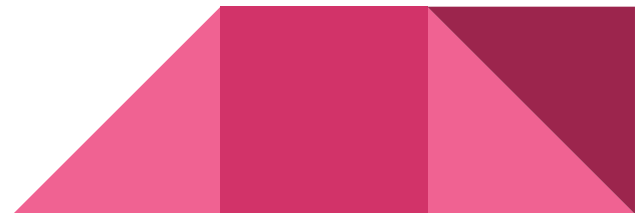
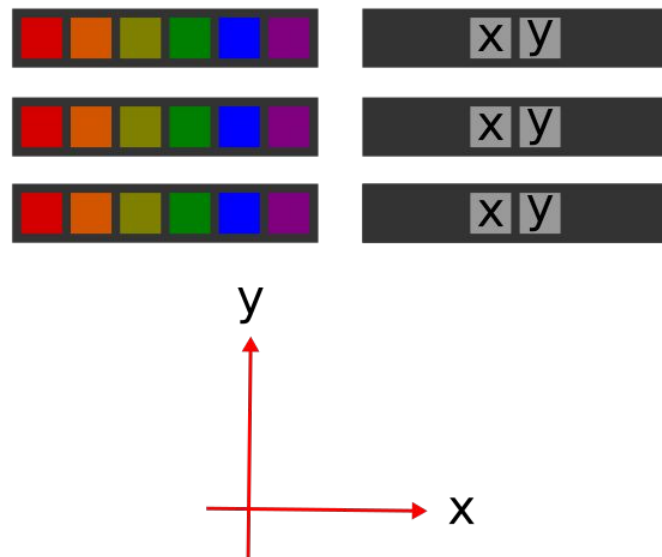
Choice of aggregations

- Choice of aggregation key
 - Should we include source port?
- Being robust and independent of varying factors
 - Capture duration
 - Incremental or Real-time



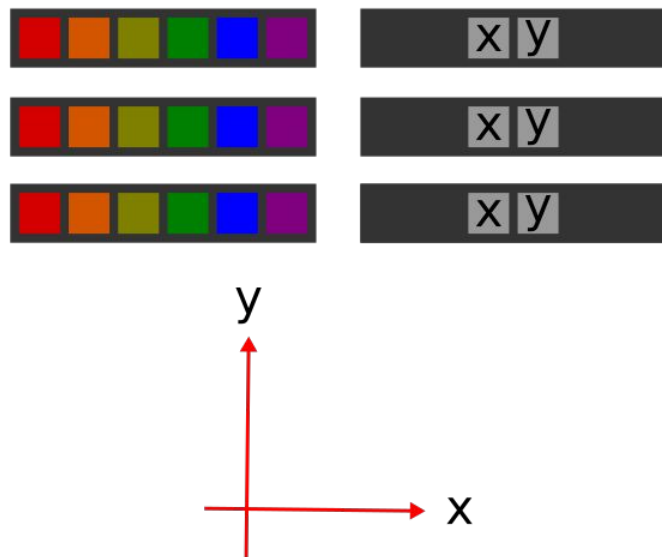
Dimensionality reduction

- Given many dimensions for a single data point
- We can study them by looking at only two of those dimensions at a time (ignoring others) which is called **projection**
- Or use **PCA** or **t-SNE** to transform data from high dimension to low dimension (2D) preserving as much



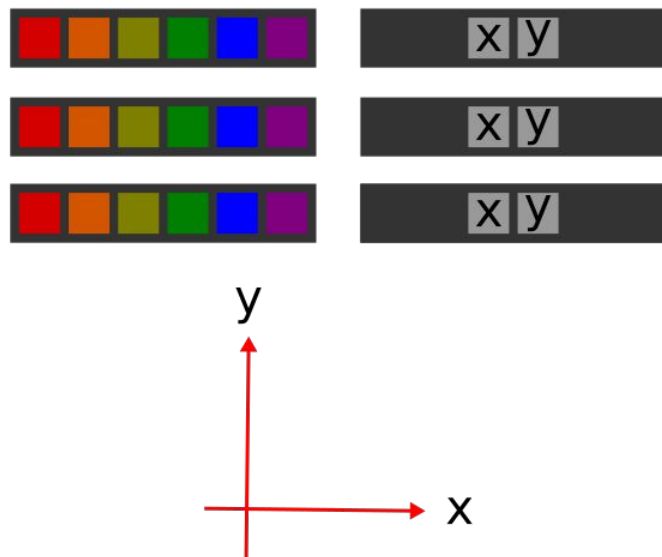
Dimensionality reduction

- Principal component analysis (**PCA**)
 - Invented in 1901
 - Statistical method
 - orthogonal linear transformation
 - can be thought of as fitting an n-dimensional ellipsoid to the data, where each axis of the ellipsoid represents a principal component.



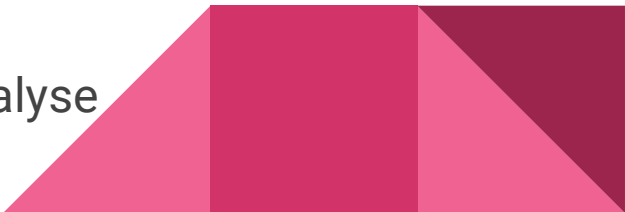
Dimensionality reduction

- t-distributed Stochastic Neighbor Embedding (t-SNE)
 - Modern 2008
 - Stochastic (depends on initial random state)
 - Not only linear relation
 - Tries to preserve distances

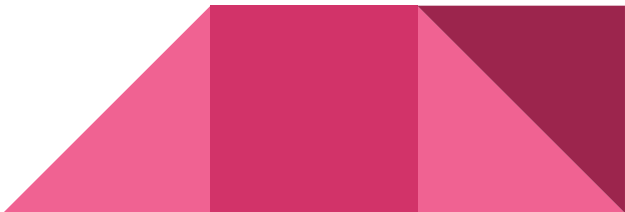


Introduced methodology

Choice of aggregation: aggregation key

- Our aggregation key
 - the type of transport layer protocol (TCP/UDP/ICMP)
 - source address
 - destination address
 - destination port
 - Source port is not included in our aggregation key but it's not ignored (later)
 - Other papers used different keys like only IP Address or a fixed moving window of 10 minutes traffic ...etc. (see "Literature review" section in our paper)
 - Our choice to include destination port allow us to analyse process behaviors not just hosts
- 

Choice of aggregated fields: before normalization


- Number of Netflows
 - Number of Distinct source ports
 - Summation of total packets
 - Summation of total bytes
 - Summation of source bytes (bytes in forward direction)
 - Summation of destination bytes (bytes in backward direction)
 - Average bytes balance ratio (where bytes balance ratio is source bytes / total bytes)
 - Average wait time between flows
 - Average flow duration
- 

But they are not robust

- The scale along different dimensions is not the same
 - having 1 byte off in total bytes is not the same as having 1 second off in delay between flows
 - IMPORTANT: t-SNE tries to preserve distances
- The longer the capture period, the more bytes we capture and aggregate
 - Can't be real-time
 - Can't be incremental



Normalization: flows relative metrics

- Make numbers relative to number of flows, for example
 - Average number of packets per flow
 - Average total bytes per flow
 - The ratio of number of distinct source ports over number of flows
 - A client that uses different ephemeral source port each time would have the ratio to be 1.0
 - A client the uses the same port over and over would have a value that approaches 0.0
 - No matter how long we capture or how many flows we have aggregated
 - The scale is not yet normalized
- 

Normalization: scale and center

- Range normalized
 - $x' = (x - x_{\min}) / (x_{\max} - x_{\min})$
 - Pros: scale [0 - 1.0]
 - Cons: not average centered
- Make metrics average centered
 - $x' = (x - x_{\text{avg}})$
 - Pros: center is 0
 - Cons: Scale is $(-\infty, \infty)$
- Make metrics average centered and inverse max scale
 - $x' = (x - x_{\text{avg}}) / x_{\max}$
 - Pros: center is 0
 - Cons: Scale
- Use log
 - $x' = \log(x+1)$
 - Make 1,10,100,1000 into 0,1,2,3
- Standard Score
 - $x' = (x - x_{\text{avg}}) / \sigma$
 - Pros: center is 0, step is σ
 - Cons: open-ended range $(-\infty, +\infty)$
- Sigmoid-family squashing
 - $x' = \tanh(x - x_{\text{avg}})$
 - Pros: center is 0, range is $(-1.0, 1.0)$

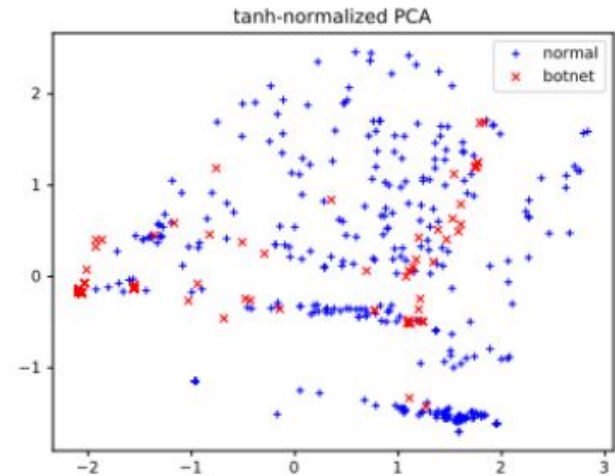
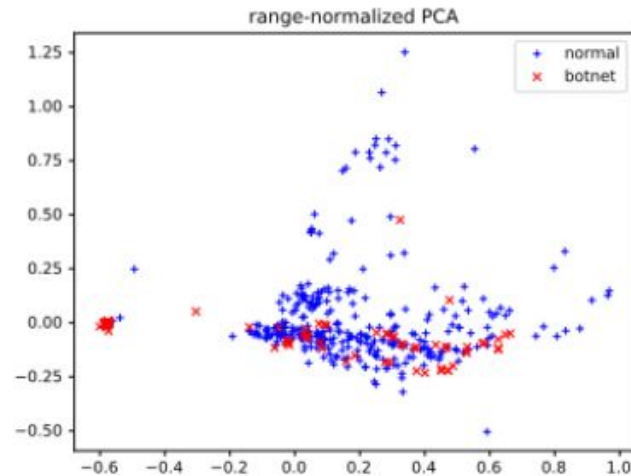
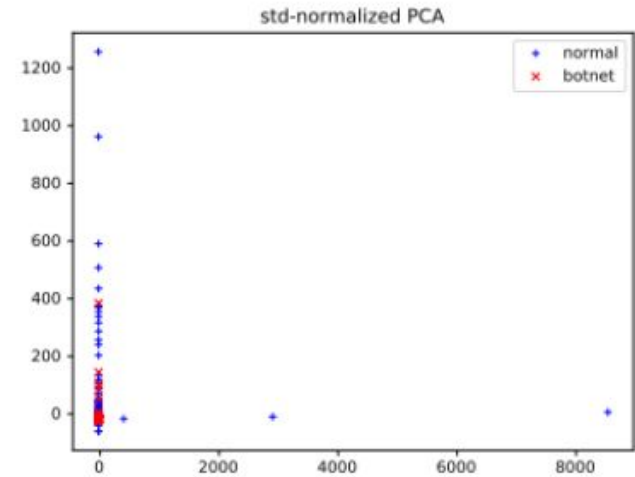
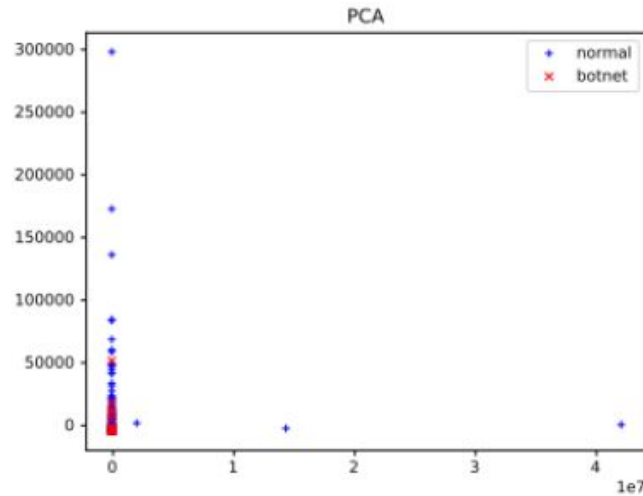




Evaluation of normalization methods

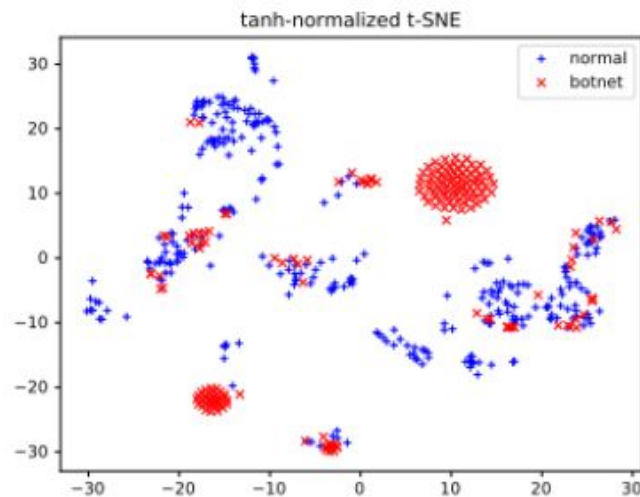
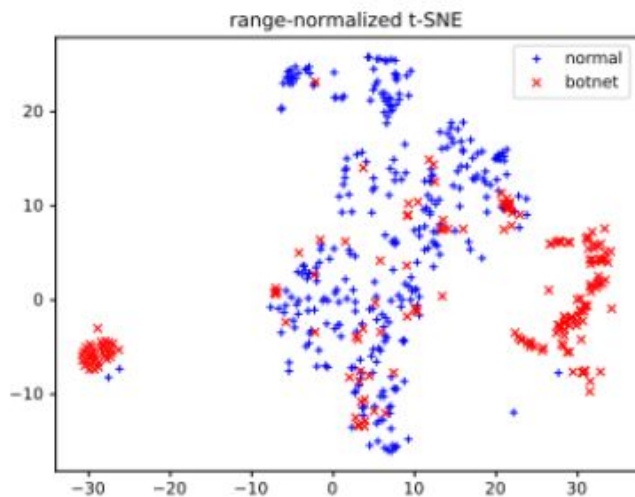
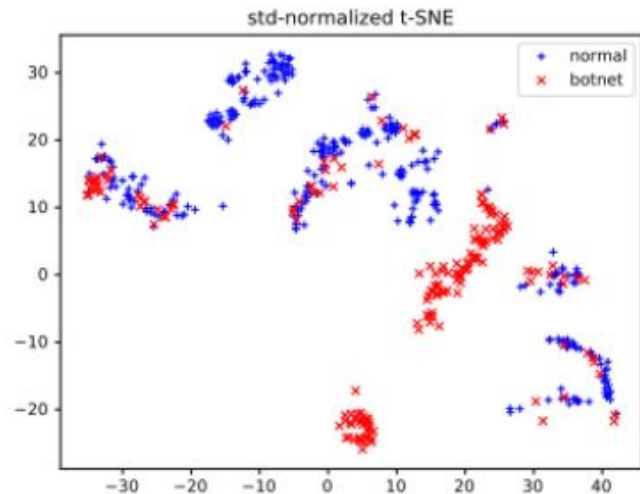
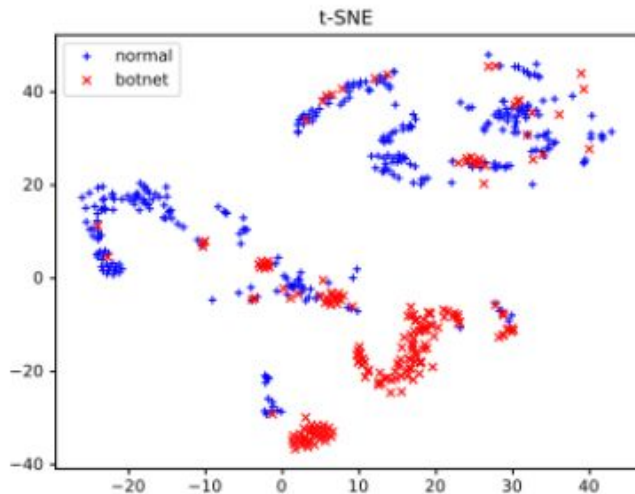
PCA applied to scenario 5 of CTU-13 dataset with different normalization methods

TOP: PCA with no normalization (left) standard score (right)
BOTTOM: range normalized (left), tanh-normalized (right)



t-SNE applied to scenario 5
of CTU-13 dataset

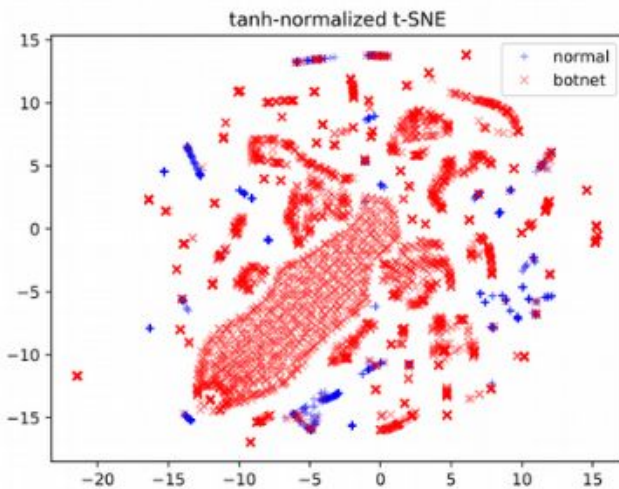
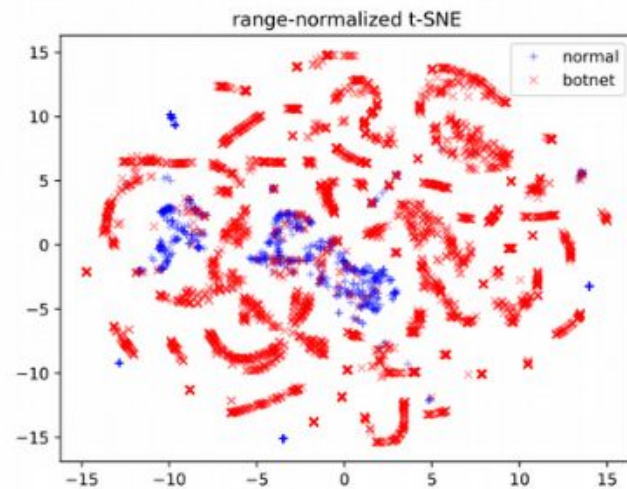
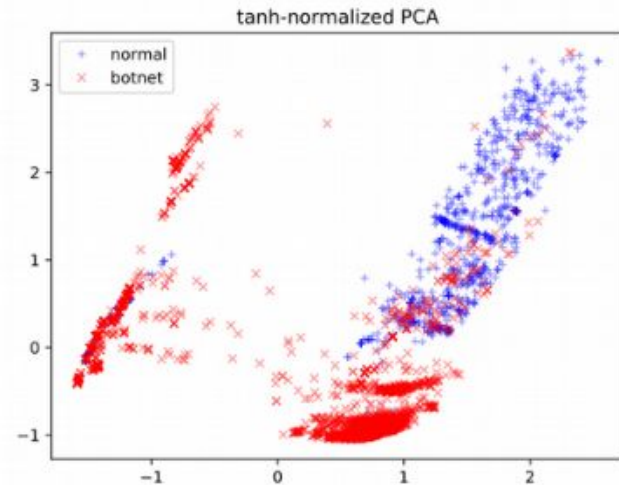
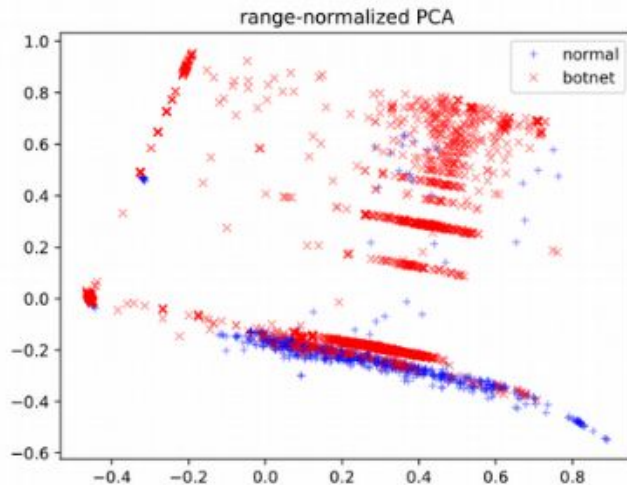
TOP: t-SNE with
no normalization
(left) standard
score (right)
BOTTOM: range
normalized (left),
tanh-normalized
(right)



Evaluation of embedding methods

Scenario 1 in the CTU-13 dataset

PCA (top) vs. t-SNE
(bottom) applied on both
range normalized (left)
and tanh normalized
(right)



Include variance as a feature

Botnets tend to be periodic, being able to identify periodic behaviors is desired.

Some papers try to calculate frequencies, but our paper use “variance” as a metric or feature which would be 0 for uniform values and increase when data is not uniform.



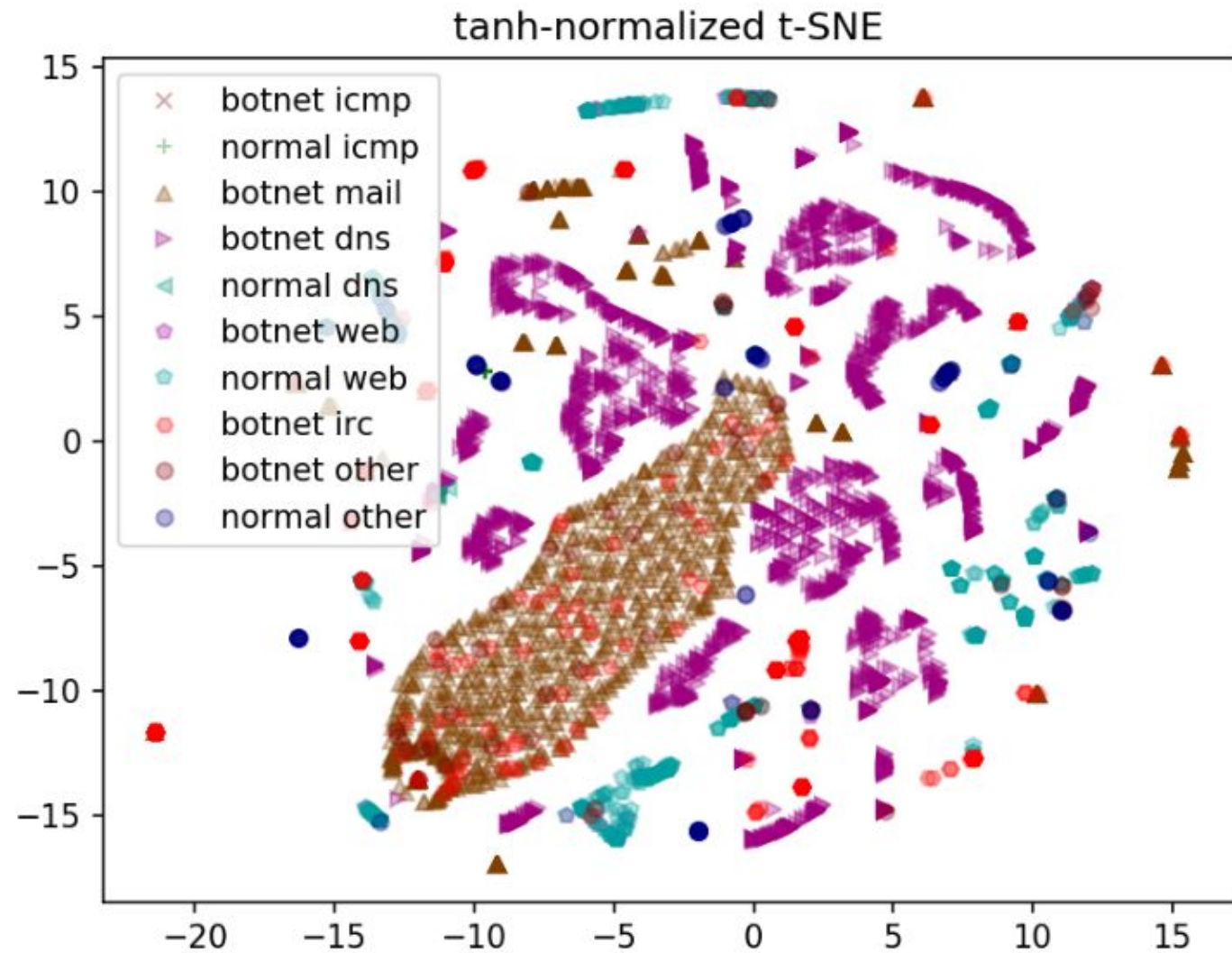
Final Normalized Aggregated Features

1. $\tanh(\text{flows}_i - \text{flows}_{\text{avg}})$
2. $\tanh(\text{srcPorts}_i - \text{srcPorts}_{\text{avg}})$
3. $\tanh(\text{avgPacketsPerFlow}_i - \text{avgPacketsPerFlow}_{\text{avg}})$
4. $\tanh(\text{varPacketsPerFlow}_i - \text{varPacketsPerFlow}_{\text{avg}})$
5. $\tanh(\text{avgBytesLogPerFlow}_i - \text{avgBytesLogPerFlow}_{\text{avg}})$
6. $\tanh(\text{varBytesLogPerFlow}_i - \text{varBytesLogPerFlow}_{\text{avg}})$
7. $\tanh(\text{avgSrcBytesLogPerFlow}_i - \text{avgSrcBytesLogPerFlow}_{\text{avg}})$
8. $\tanh(\text{varSrcBytesLogPerFlow}_i - \text{varSrcBytesLogPerFlow}_{\text{avg}})$
9. $\tanh(\text{avgDstBytesLogPerFlow}_i - \text{avgDstBytesLogPerFlow}_{\text{avg}})$
10. $\tanh(\text{varDstBytesLogPerFlow}_i - \text{varDstBytesLogPerFlow}_{\text{avg}})$
11. $\tanh(\text{avgBalanceRatio}_i - \text{avgBalanceRatio}_{\text{avg}})$
12. $\tanh(\text{varBalanceRatio}_i - \text{varBalanceRatio}_{\text{avg}})$
13. $\tanh(\text{avgWaitTime}_i - \text{avgWaitTime}_{\text{avg}})$
14. $\tanh(\text{varWaitTime}_i - \text{varWaitTime}_{\text{avg}})$
15. $\tanh(\text{avgDuration}_i - \text{avgDuration}_{\text{avg}})$
16. $\tanh(\text{varDuration}_i - \text{varDuration}_{\text{avg}})$

Results

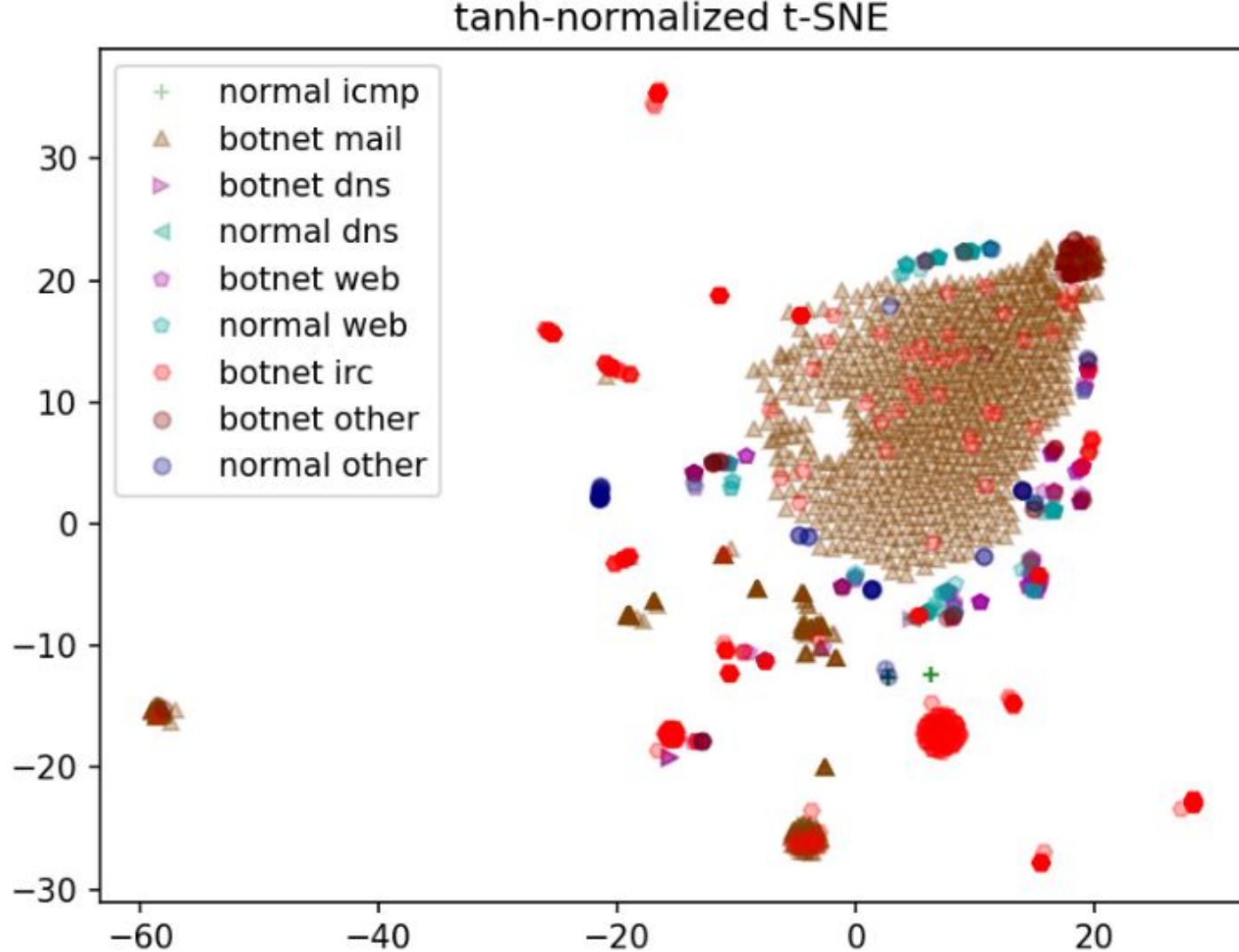
t-SNE applied to
scenario 1 of CTU-13
with colored ports
and protocols

With a glance at the
figure, this botnet
easily identified as a
spam botnet that
send mass number
of email possibly
based on IRC C2C
trigger. Beside IRC
this botnet also rely
heavily on DNS and it
rarely use ICMP.



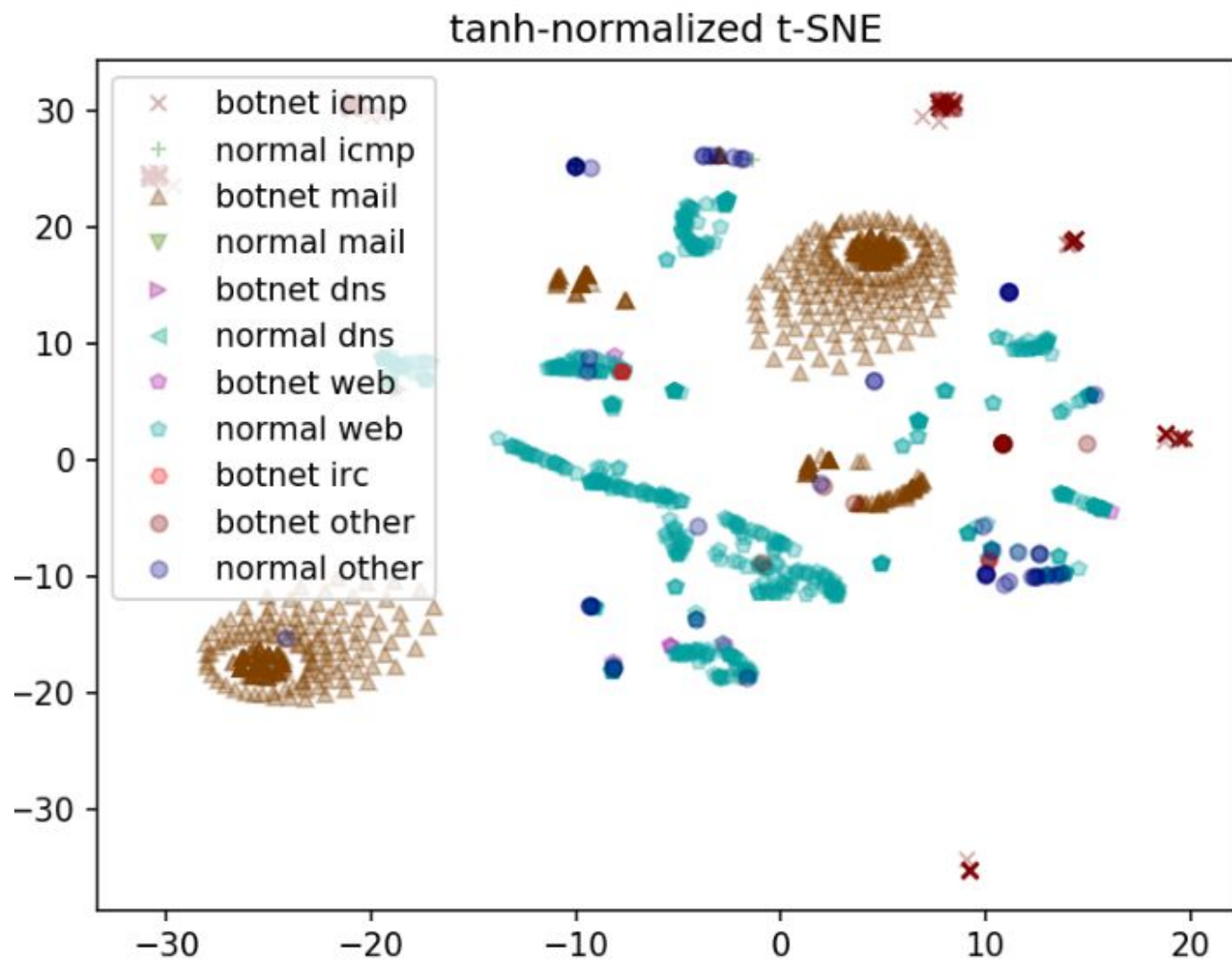
t-SNE applied to
scenario 2 of CTU-13
with colored ports
and protocols

Like previous botnet
except that it does
not rely on DNS but
mainly on IRC



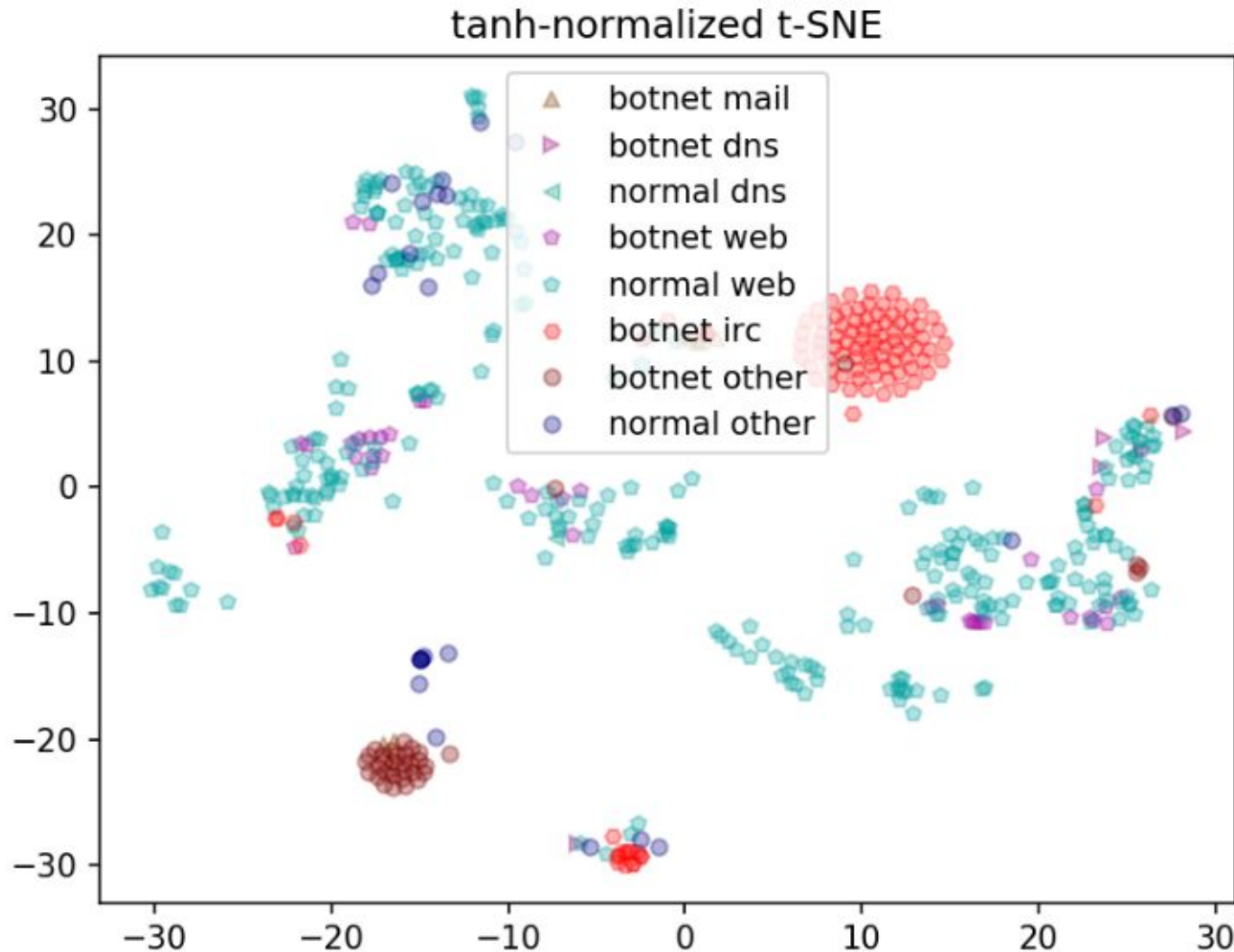
t-SNE applied to
scenario 4 of CTU-13
with colored ports
and protocols

Clearly this is a spam
botnet which uses
ICMP for its
operation



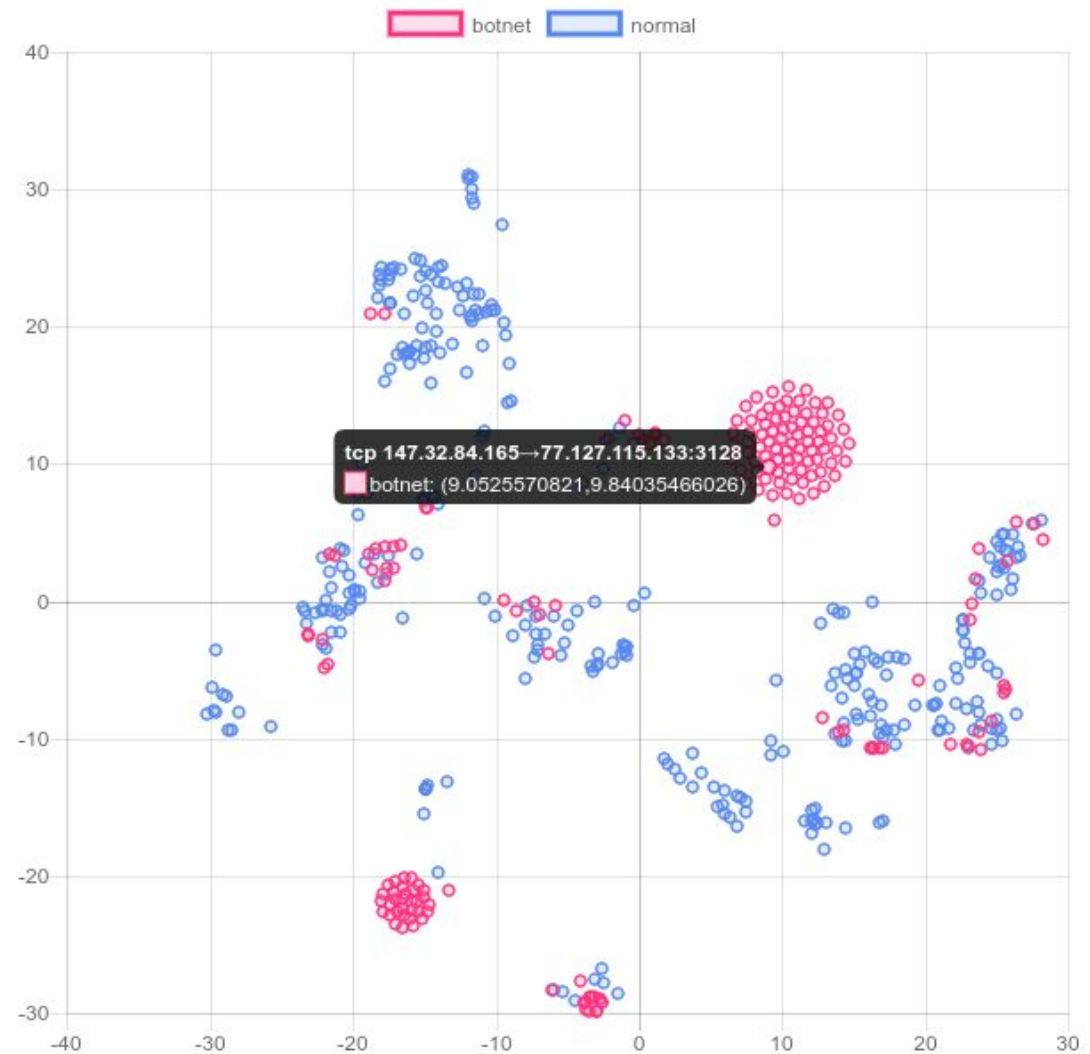
t-SNE applied to
scenario 5 of CTU-13
with colored ports
and protocols

Unlike previous
botnets, it's not a
spam botnet, it
operates mainly over
IRC and it uses port
3128 (squid proxy
port) found bottom
left



Interactive tool published on
github

<https://github.com/muayyad-alsadi/flowvis>





Q & A