Botnet Traffic Detection

Meng-Hsun Tsai (蔡孟勳)

Department of Computer Science & Information Engineering

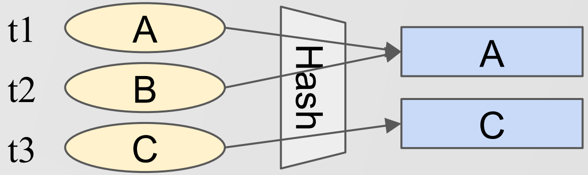
National Cheng Kung University

tsaimh@csie.ncku.edu.tw

Motivation

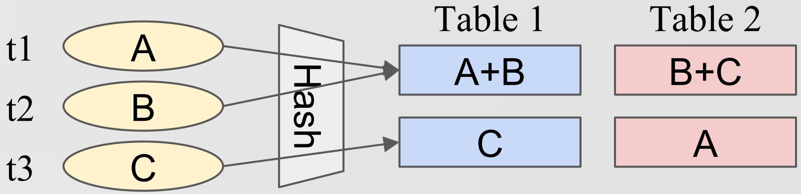
- Detecting and stopping the network scanning behavior is an efficient way to slow down the spread of IoT botnets.
- Given that controllers have more comprehensive network information and computation power, **switches are responsible for collecting network statistics, and controllers detect anomalies based on the data from switches** in our architecture.
- However, the **memory resources and operations (e.g. floating points) in P4 switches are limited**, but controllers require **data in high granularity** to detect and block the anomalies. Therefore, an efficient data collection method is required.

Related Work - Sampling Methods



- Estan et al. propose the ***Sample and Hold (SH) method*** to collect statistics of heavy hitters, which are major flows in a network [6].
- N. Hohn and D. Veitch propose the ***flow sampling method***, which samples flows instead of packets at random to collect statistics of heavy hitters [7].
- For an anomaly detector, **attacker network statistics are more important** than benign host network statistics. After all, an anomaly detector cannot detect attackers if no network statistics about the attackers are collected.
- Sampling methods collect only few attacker network statistics if attacker traffic is in the minority. In other words, **the storage utilization of sampling methods is low**.

Related Work - Sketch Methods



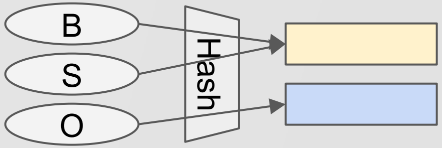
- Sketch methods create a data structure to estimate approximate statistics of flows.
- G. Cormode and S. Muthukrishnan propose the ***count-min sketch (CM sketch)*** [8].
  - The approximate statistic of an element is the minimum value among the hashed slots of the tables.
- Krishnamurthy et al. propose the ***k-ary sketch*** [9], which uses Equations 3.1 and 3.2 to estimate approximate statistics.
  - Assume there are **H** hash functions and **H** tables in a k-ary sketch. Each table ***T[i]*** has ***K*** slots. ***sum*(S)** is the sum of slot values in a table, and ***hi*** is theith hash function.

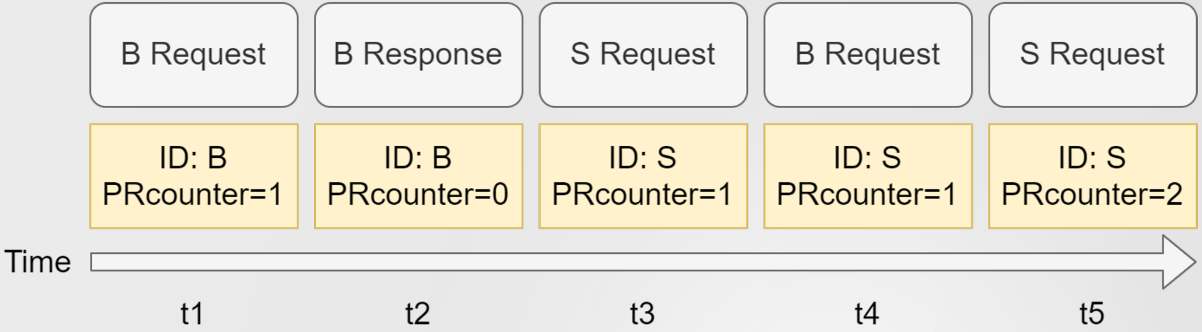
$$v_x^{est} = median_{i \in H}\{v_x^{h_i}\} \quad (3.1) \qquad v_x^{h_i} = \frac{T[i][h_i(x)] - sum(S)/K}{1 - 1/K} \quad (3.2)$$

Feature Selection - PRcounter

- Beigi et al. evaluate the efficiency of sixteen flow-based features with a decision tree classifier [23]. The results show that the **ratio between the number of incoming packets and the number of outgoing packets (IOPR) is the most efficient feature**.
- Inspired by IOPR, we propose Pending Request Counter (**PRcounter**) as a new feature. A PRcounter records **the difference between the number of TCP requests and the number of TCP responses for a device**.
- Each slot in a hash table contains **a PRcounter** and **an identifier (ID)**. The ID indicates the originator of a TCP connection.
- If **the ID of a packet is the same as that of a slot**, the PRcounter value of the slot is **increased** when the packet is a **request** and **decreased** when the packet is a **response**.

Proposed Method - 0-Replacement (1/2)





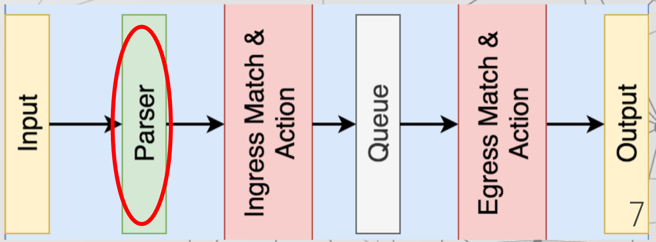
B: Benign Hosts
S: Scanner
upstep: 1, downstep: 1

- At t3, a scanner request arrives and **replaces the data** in the slot because the **PRcounter value in the slot equals zero**.
- We observe that **scanner data occupy slots after replacement**, so **benign host data are replaced with scanner data as time goes by**.
- We make the ***downstep* greater than the *upstep*** in 0-Replacement, so a **PRcounter value decreases to zero** with time even if a benign host has a connection failure.
- We define the ratio between the upstep and the downstep as the ***step ratio*** (γ). **The lower the step ratio is, the higher the tolerance of connection failure is.**

0-Replacement Parser

- ***hdr*** contains packet headers and ***m* (metadata)** contains information that will used in later control blocks.
- A parser in P4 starts with the *start* state and ends with the *accept* or *reject* states.
- The parser transits to *set_request* or *set_response* if the packet is a request or a response respectively. Otherwise, the parser transits to the *set_ignore* state.

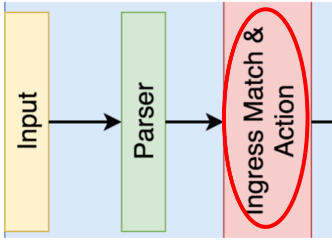
```
state start {
    transition: parse_ipv4;
}

state parse_ipv4 {
    packet.extract(hdr.ipv4);
    transition: parse_tcp;
}

state parse_tcp {
    m.Su = UPSTEP;
    m.Sd = DOWNSTEP;
    packet.extract(hdr.tcp);
    select(hdr.tcp.syn, hdr.tcp.ack) {
        1, 0: set_request;
        1, 1: set_response;
        default: set_ignore;
    }
}

state set_request {
    transition: m.tcp_status = 1;
    transition: accept;
}

state set_response {
    transition: m.tcp_status = 2;
    transition: accept;
}

state set_ignore {
    transition: m.tcp_status = 0;
    transition: accept;
}
```



0-Replacement Algorithm

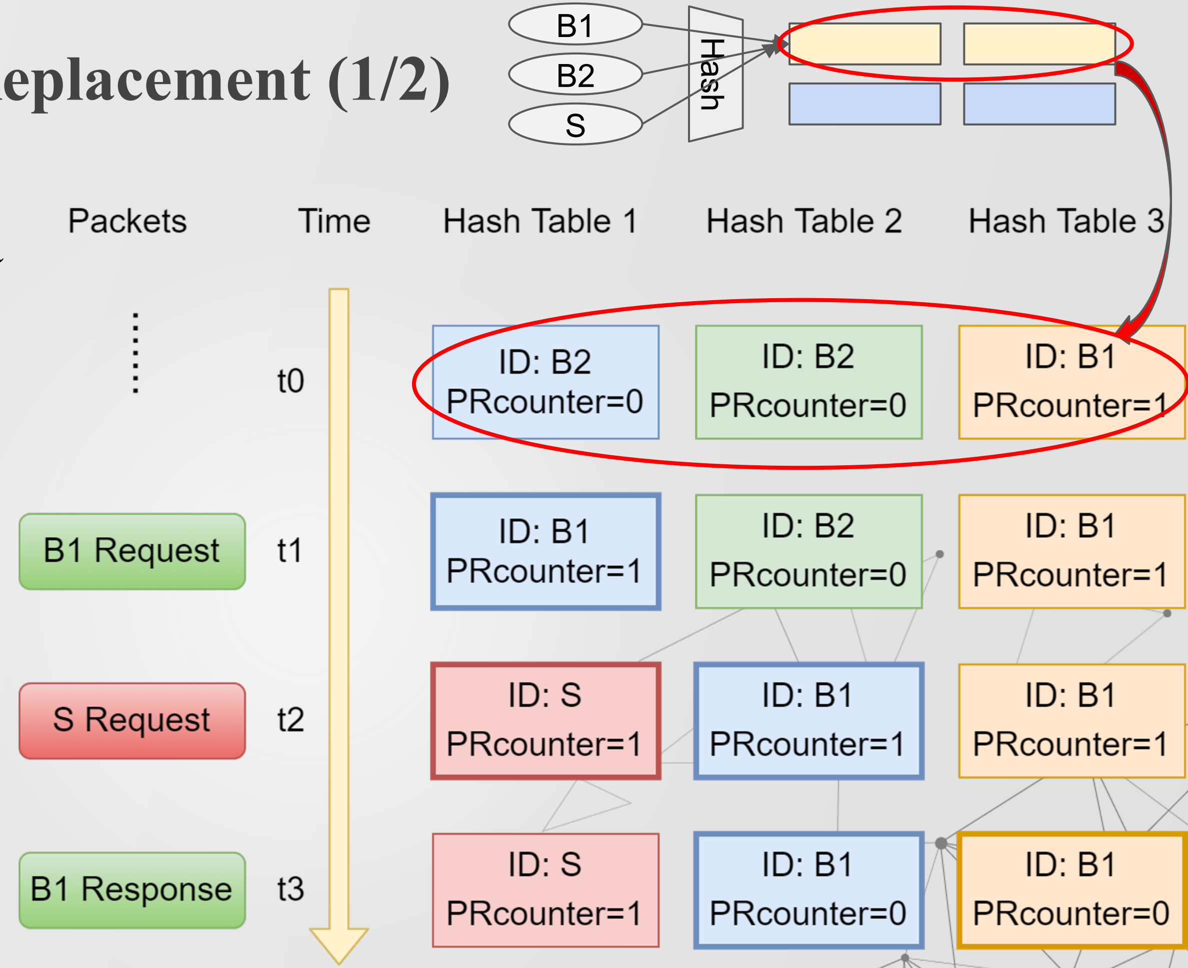
- The PRcounter value increases by *upstep (Su)* when the IDs of the slot and the packet are the same and the packet is a request. (**Lines 9-11**)
- The PRcounter value decreases by *downstep (Sd)* when the IDs are different and the packet is a response. If the PRcounter value is negative after the decrease, the value is set to zero. (**Lines 12-16**)
- Replacement happens when the IDs are different and the PRcounter value is zero. (**Lines 17-20**)

```
1:  procedure 0-REPLACEMENT(hdr, m)
2:      if m.tcp_status = 1 then
3:          m.id ← hdr.ipv4.src
4:      else if m.tcp_status = 2 then
5:          m.id ← hdr.ipv4.dst
6:      else
7:          return
8:      m.hid ← hash(m.id)
9:      if T(m.hid).id = m.id then
10:         if m.tcp_status = 1 then
11:             T(m.hid).cnt += m.S_u
12:         else if m.tcp_status = 2 then
13:             if T(m.hid).cnt > m.S_d then
14:                 T(m.hid).cnt −= m.S_d
15:             else
16:                 T(m.hid).cnt ← 0
17:     else
18:         if m.tcp_status = 1 and T(m.hid).cnt = 0 then
19:             T(m.hid).id ← m.id
20:             T(m.hid).cnt += m.S_u
```



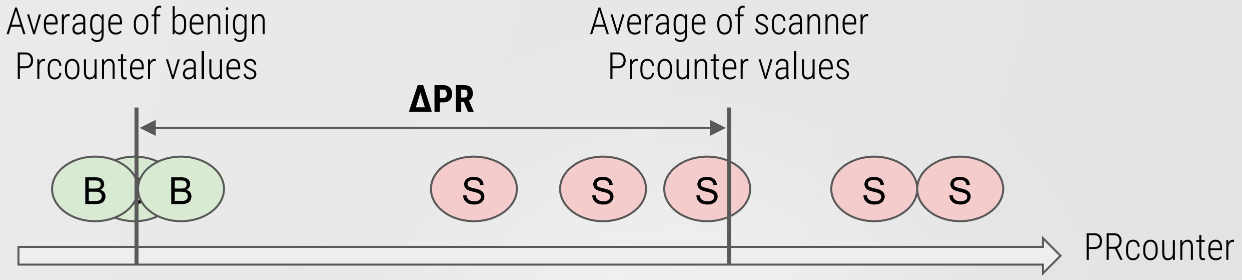
Proposed Method - E-Replacement (1/2)

- If **two scanner data are hashed to the same slot** in a table. Only the scanner data whose packet arrives earlier can occupy the slot. The **hash collision degrade performance**.
- Inspired by ***HashPipe* [10]**, we propose E-Replacement, which uses **multiple hash tables** to reduce the impact of hash collision.



(B1 and B2: Benign Hosts, S: Scanner, upstep: 1, **downstep: 2**)

K-Means Classifier



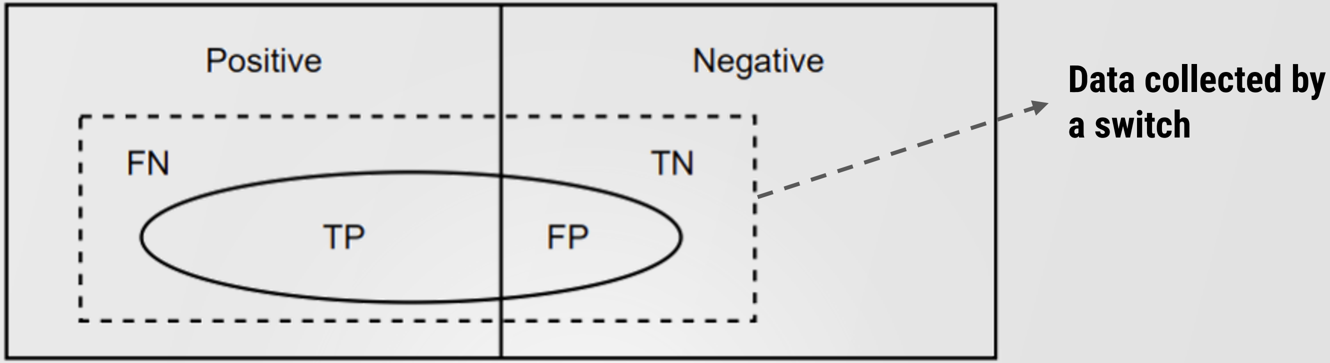
- After pulling network statistics from switches, controllers reset all PRcounter values in the switches to zero and detect scanners through the K-Means classifier.
- KMeans is a clustering algorithm that groups the data whose values are close.
- We set **the number of cluster *K* to two** because there are only two categories: scanners and benign hosts.
- We let **the group with the lowest PRcounter value be the benign host category**.

Simulation Settings

Notation	Default	Description
k	0.605	Shape Parameter of Weibull Distribution
λ	4.477	Scale Parameter of Weibull Distribution
μ	6.66	Mean of Weibull Distribution
RTT	0.28724	Round Trip Time (seconds)
τ	240	Polling Interval (seconds)
N_s	30	Number of Scanners
N_b	3000	Number of Benign Hosts
N_{slot}	1000	Number of Slots
N_{ht}	1, 5, 5	Number of Hash Table (optimized for CM/KA Sketch, 0R and ER)
θ_s	0.2	Average Successful Connection Rate of Scanners
θ_b	0.8	Average Successful Connection Rate of Benign Hosts
S_u	7, 8, 5, 6	Upstep (optimized for SH, CM/KA Sketch, 0R and ER)
S_d	10	Downstep

- We use random variables following the **Weibull distribution** to generate the inter-arrival of packets since the Weibull distribution can well model human network behaviors [11].
- We set the Weibull distribution parameters k and λ based on [11], RTT based on [12], the successful connection rates of scanners and benign hosts based on [13], and the polling interval based on [14].

Performance Metrics



- **Accuracy (Ra)**: (TP + TN) / (TP + FP + FN + TN)
- **Precision (Rp)**: TP / (TP + FP), **Recall (Rc)**: TP / (TP + FN)
- **Scanner Collecting Ratio (Rs)**: the ratio between the number of scanner slots and the total number of scanners in a network, (TP+FP)/Positive
- **Detection Rate (Rd)**: the ratio between the number of scanners correctly identified by the detector and the total number of scanners in a network, TP/Positive

Evaluation of Different Methods

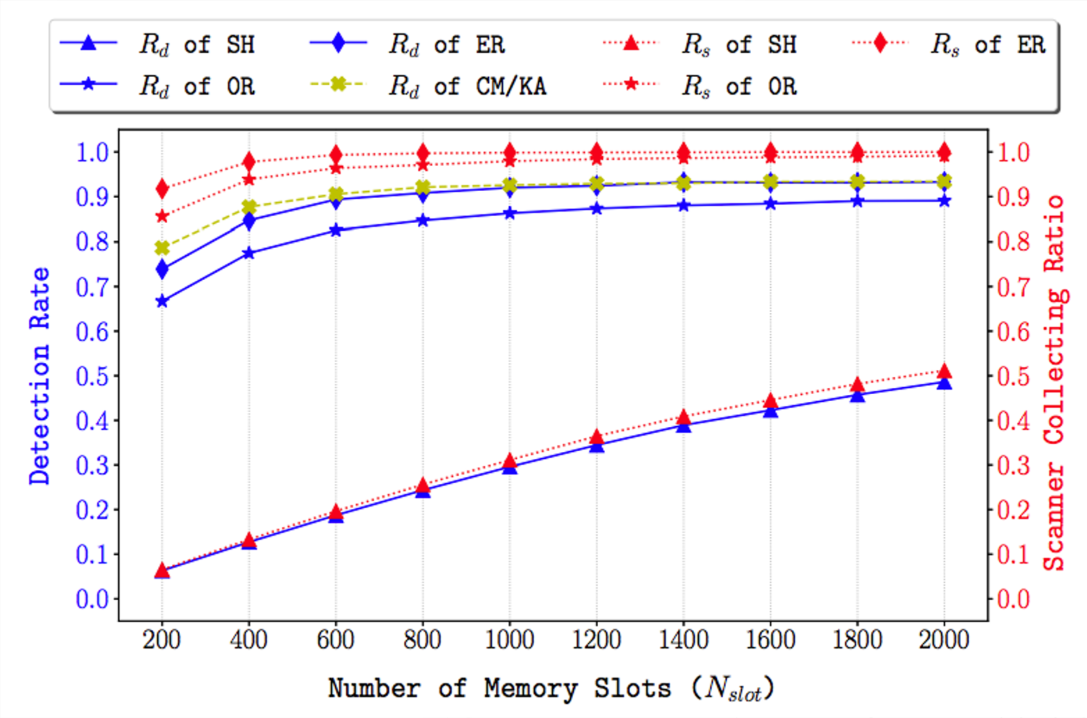
Method	R_s	R_a	R_c	R_p	R_d
Sample and Hold	0.312	0.999	0.949	0.999	0.296
0-Replacement	0.980	0.996	0.879	1.000	0.862
E-Replacement	0.998	0.997	0.921	0.999	0.920
CM/KA Sketch	-	0.942	0.923	0.159	0.923

Rs: scanner collecting ratio
Ra: accuracy
Rc: recall
Rp: precision
Rd: detection rate

- The **scanner collecting ratios of 0-Replacement and E-Replacement are higher** than the sample and hold method since benign slots are replaced with scanner data with time.
- The **precision of the two sketch methods is low** since **hash collision causes high estimation bias**; therefore, the detector mistakes the benign host data for scanner data.
- E-Replacement improves the **detection rate** by (0.92−0.862)/0.862 = **6.73%** and (0.92−0.296)/0.296 = **210.82%** compared to **0-Replacement** and the **sample and hold method** respectively.
- E-Replacement improves the **precision** by (0.999 − 0.159)/0.159 = **528.3%** compared to **the two sketch methods**.

Memory Usage

- Performances of all methods increase as the number of memory slot increases.
- After experiments, we observe that **scaling up Nslot, Ns, and Nb proportionally does not affect the results of E-Replacement**.
- The detection rate is **93.4%** in E-Replacement when **Nslot=1000, Ns=10 and Nb=1000**.



- Assume the ID and the PRcounter in a slot are **32-bit registers**. 66000 slots cost 66000∗64/220 = **4.02Mb SRAM**, which is 4.02/370 = **1.09%** SRAM, if we implement E-Replacement on the chip proposed by Bosshart et al [15].

Conclusions

- By leveraging the concept of replacement, E-Replacement **improves the detection rate by up to 210.82% compared to the sample and hold method**. Furthermore, E-Replacement **improves the precision by up to 528.2% compared to the count-min sketch method and the k-ary sketch method**.
- E-Replacement **mitigates the performance degradation caused by hash collision** by leveraging multiple hash tables. E-Replacement **improves the detection rate by up to 6.73% compared to 0-Replacement**.
- With only **4.02Mb SRAM**, E-Replacement can **detect around 93.4% scanners** in a class B network.

THANKS

Does anyone have any questions?

tsaimh@csie.ncku.edu.tw