Kaan Özdinçer

Danışman: Hacı Ali Mantar

Seminer I - BİL591

Yazılım Tanımlı Ağlar ile Güvenlik

SDN & Security

- Security for SDN
  - Security issues in SDN itself
  - Heads
- Security with SDN
  - Security applications based on SDN
  - Tails

NetFuse: Short-circuiting traffic surges in the cloud

Authors: Ye Wang, Yueping Zhang, Vishal Singh, Cristian Lumezanu, Guofei Jiang

Published in: Communications (ICC), 2013 IEEE International Conference

Date of Conference: 9-13 June 2013

Date Added to IEEE Xplore: 07 November 2013

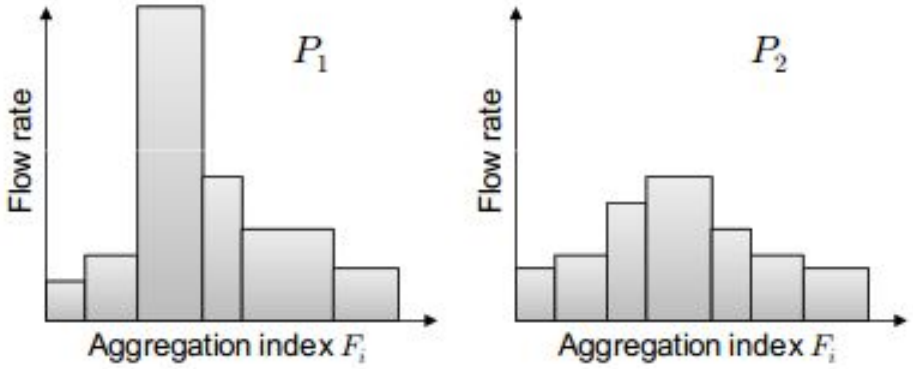
NetFuse: Introduction

- Netfuse is a mechanism to protect against traffic overload in OpenFlow-based data center networks.
- First Goal; Identifying flows with suspicious behavior.
- Proxy between switches and controller.
- Similar to how fuse boxes protect electrical circuits from surges.

- Use Openflow control messages to achieve overload detection and reaction.
- To detect, use multi-dimensional flow aggregation.
- Finally, limits rating with adaptive control using toxin-antitoxin mechanism.

Netfuse: Flow Aggregation

- Current Practice: Single-Dimensional Aggregation
- Input is a set of *n* flows: {*f1,f2,f3*} = Fx
- Each valid flow aggregation (set partition) P= {*F1,F2...*}
- Overloading score S(P) = max{F}-m^{F}/m^{F}
  - max: maximumu rate accross all aggregated flows
  - m^: low median



	Aggregation condition	Example overloading reason
P1	ingress and egress	end-to-end flooding
P2	source subnet	compromised VMs
P3	destination subnet	flash crowd to specific VMs
P4	destination port	attack against specific services
P5	routing	routing misconfiguration
P6	start time range	correlated data transfers
P7	frequency threshold	new traffic load
P8	duration threshold	short/failed connection attempts
P9	burstiness threshold	buggy customized TCP

- Reasonable Aggregations
- Spatial: P1-P6
- Temporal: P7-P9

Netfuse: Flow Aggregation ++

- In practice, the overloading can be caused by specific applications at specific network regions, corresponding to a combination of multiple aggregation conditions on different flow properties.
- Ex. Database attack.
  - “Start time < 30s” and “dPort=1433”
  - If we aggregate the flows by either start time **or** dPort, we cannot identify the right set of flows.
- Develop a breadth-first search algorithm with branch pruning to enumerate the multi-dimensional flow aggregation.
  - Input = {*P1,P2,P3*}
  - P1={{*f1,f2*}{*f3*}} | *f1,f2* old | *f3* new
  - P2={{*f1*},{*f2,f3*}} | *f1* port 22 | *f2,f3* port 80
  - Algorithm generates new *P3* by combinin *P1* and *P2* | P3={{f1},{f2},{f3}}
    - F1 old, port 22 | F2 old, port 80 | F3 new, port 80
  - If P3 has higer score than P1 and P2, algorithm considers P3 better and keeps it.
  - Algorithm continues to combine partitions to find the best flow aggregation.

Netfuse: Flow Aggregation +++

Algorithm 1: maxAgg (π)

```
1  π* = {P | P ∈ π, S(P) > th};
2  while |π*| > 1 do
3      select P from π*, foreach P' ∈ π* do
4          if not (P ≤ P' or P' ≤ P) then
5              P'' = PP';
6              if P'' ∉ π* and S(P'') > S(P) and
               S(P'') > S(P') then
7                  π* ← π* ∪ P'';
8      if max{S(P) | P ∈ π*} > S(P) then
9          π* ← π* − {P};
10 if π* == ∅ then
11     P* ← flows;
12 else
13     P* ← the only aggregation in π*
```

- Complexity: O(NK^m)
  - N: number of active flows
  - K: number of pre-determined aggregation rules
  - m: dimension of optimal aggregation.

Netfuse: Adaptive Control

- Basic Control
  - For each identified overloading flow, NetFuse instructs the associated switches to reroute the flows to the NetFuse box.
  - Rerouting, different from mirroring, frees the network resource originally occupied by the overloading flows.
  - Then, NetFuse can delay or selectively drop packets of the redirected flows to reduce their rates.

Netfuse: Adaptive Control ++

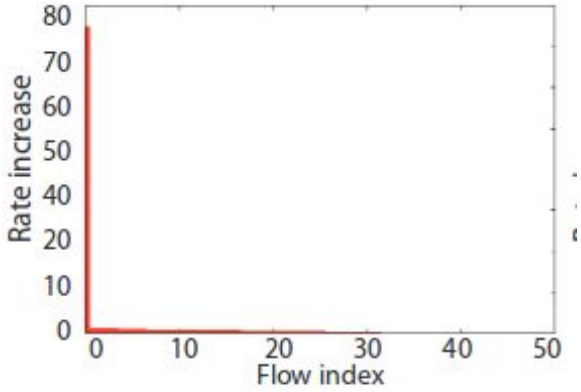
- Toxin-Antitoxin Mechanism
  - NetFuse should regulate the identified flows differently and adapt the reaction according to the system feedback.
  - Adaptive control mechanism, inspired by toxin-antitoxin biological systems.
  - When it delays an overloading flow f, Netfuse tests the aggressiveness of f' response.
  - If f reduces its rate Netfuse also reduse aggressiveness. Of delay it no longer delays.
  - Otherwise starts delaying f more aggresively unti it eventually fills the buffer, dropping all packets off.
    - Target rate of f is r (avg rate of other normal flows.)
    - Current rate is rf
    - Extra delay puts on f: (rf-r)*rf*RTD/r (RTD: round trip delay time)

Netfuse: Design

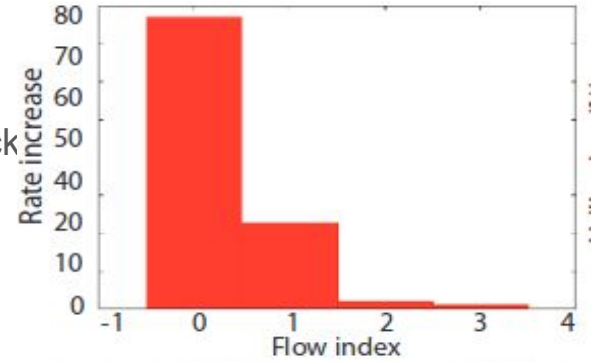
- İmplement as a proxy device between the OF switches and controller.
- Monitoring: **passive listening** and **active query**
  - NetFuse intercepts all the control messages between the switches and controller, thereby obtaining a global view of the network.
- Netfuse Box work as transparent between controller and switches.
  - Each switch sends a PacketIn message to controller and cont. Replies with FlowMod (source IP port) (Vlan tag, ingress interface) (flow start time etc.) passively listen as a proxy
  - Also use ReadState messages to periodically query the switches.

Netfuse: Preliminary Experiments

- 1) Single Compromised Flow
  - Pick up one normal flow and suddenly increase its rate.
  - Suspicious flows uses more resources and easily identified without aggragation.
- 2) DDoS Attack
  - İnject flows from different switches to an edge switch
  - “Freq < 4” and “start < 30s” aggragation.
  - Also mis-identified normal flows. False Alarm %9
  - Adaptive Netfuse reaction will eventually block real bad attack and let the normal flows pass.



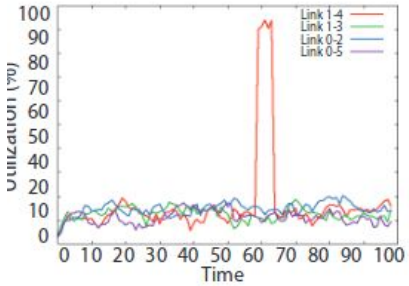
(a) Expt 1: NetFuse easily identifies single misbehaving flow, without aggregation.



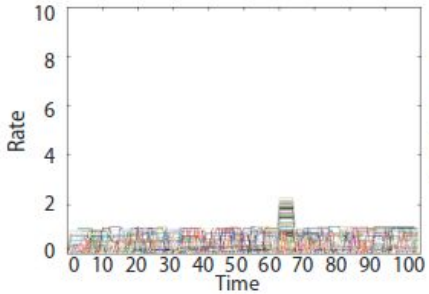
(b) Expt 2: Multi-dimensional flow aggregation reveals misbehaving aggregate flows (index 0) but with few false positives (index 1).

Netfuse: Preliminary Experiments ++

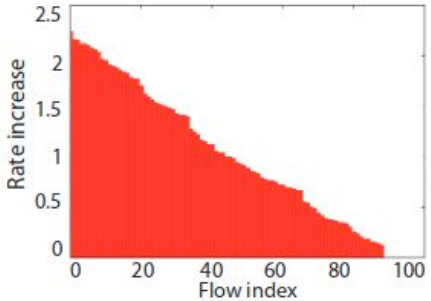
- Routing Misconfiguration
  - Let around 50 new flows. (c)
  - Rate of each flow. (d)
  - To identify new flows that are causing overload. Without aggregation (e)
  - With aggragation by combining freq and routing(A to B) (f)



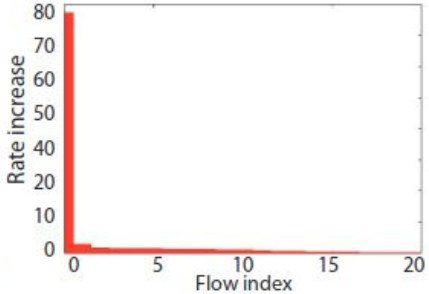
c) Expt 3: Utilization of the four core-to-aggregation links.



(d) Expt 3: Rates for all flows.



(e) Expt 3: NetFuse cannot easily identify the misbehaving flows when there is no flow aggregation.



(f) Expt 3: With flow aggregation, NetFuse identifies the aggregation responsible for overload.

Towards load balancing in SDN-networks during DDoS-attacks

Authors: M. Belyaev, S. Gaivoronski.

Published in: Science and Technology Conference (Modern Networking Technologies) (MoNeTeC), 2014 First International

Date of Conference: 28-29 Oct. 2014

Date Added to IEEE Xplore: 15 January 2015

Callophrys: Introduction

- SDN for a ”survival” mitigation during DDoS attacks, the load balancing problem.
- two-level balancing solution in SDN networks
  - traditional balancing between servers **L7**
  - load balancing between network devices **L4**
- Focus on traffic ballancing and network resources utilization.
- Experiments show that our solution significantly increase survival time of the system under DDoS-attack. Mitigation.

Callophrys: Propesed Solution

- The algorithm itself is based on the fact that we can use the SDN switch-level flows to redirect traffic based on destination and source IP-address information.
- This allows for dividing the traffic between different routes in the network regardless of the packets' actual contents.
- The algorithm goes as follows;
  - Acquire the load and topology information for the network.
  - Ovverride routing for the network acquired by Bellman-Ford pathfinding algorithm.
  - Iteratively keep splitting traffic paths for routes that are;
    - Overloaded
    - Have alternate routes available

Callophrys: Propesed Solution ++

- Let network contain switches ***1-N***
- A is channel between ***i*** and ***j*** will be addressed as ***(i,j)***
- Bandwith is $\mathbf{b_{ij}}$
- Current channel load $\mathbf{w_{ij}}$
- e, constant small load value param. Input. (not studied, future work)
- If $\mathbf{w_{ij} + e >= b_{ij}}$, channel is overloaded.

- Endpoint servers $\boldsymbol{B_1 ... Bk}$
- Bandwith matrix: $\mathbf{M_{maxload}}$ ($N_x N$) contains all $\mathbf{b_{ij}}$
- Load matrix: $\mathbf{M_{load}}$ ($N_x N$) contains all load values $\mathbf{w_{ij}}$.
- Available resource matrix: $\mathbf{M_{free}}$ defined as $\mathbf{M_{maxload} - M_{load}}$

Callophrys: Propesed Solution +++

- Phase 1: Update network load mask $\mathbf{M}_{\mathbf{load}}$. Executed before the need. $\boldsymbol{w_{ij}}$ numbers of bytes of coming from ***i*** to ***j*** during single update period.
- Phase 2: Applied once to ovverride packet routing and use statically routes that we can later modify.
- Phase 3: Current path table $\mathbf{T}_{\mathbf{path}}$ based on Phase 2. Set of triples $\{ips_{src}, ip_{b}, path\}$ address mask, address to server $\boldsymbol{B_i}$. First iteration, 0.0.0.0/0 wildcard accepting, subsequent iterations.

Callophrys: Propesed Solution ++++

1. Update $\mathbf{M_{load}}$ and $\mathbf{M_{free}}$ with current load information.
2. Find the first overloaded link in $\mathbf{M_{load}}$ ($\mathbf{w_{ij} + e >= b_{ij}}$)
3. Find the first path r_q in $\mathbf{T_{path}}$ such that it contains link (i,j).
4. Find a new shortest path from entry to server B_i. assume that (i,j) is closed in current topology. If there is no such path go back 3 and find a new path for same link. If there are no more path, should go back 2 and select a new link.
5. Calculate the maximum available additional load for $path_q$. Lookup every link in $\mathbf{M_{free}}$.
6. Calculate new sets of masks ips_{old} and ips_{new}. Remove corresponding entry from $\mathbf{T_{path}}$, insert all new entries.
7. Commit the changes in $\mathbf{T_{path}}$ to all switches.
8. Wait for the timeframe and go back to 1.

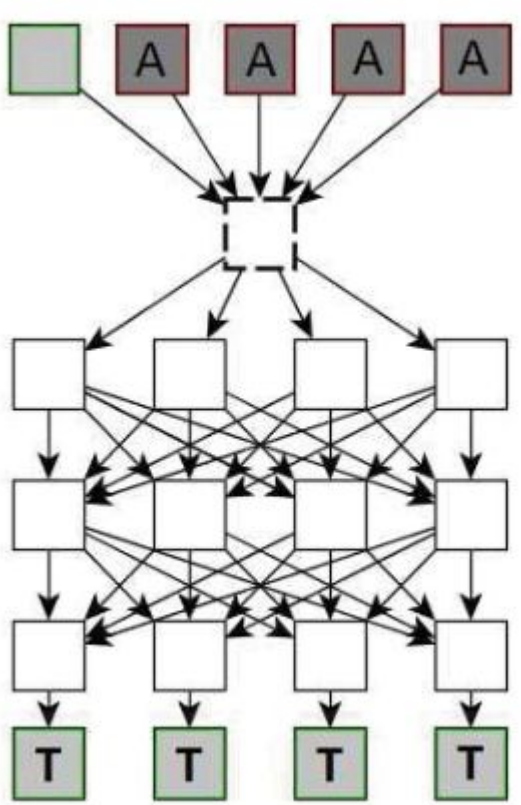
Callophrys: Implementation

- The system aims at both identifying, detecting and mitigating DDoS attacks at early phases.
- It uses the Floodlight Openflow controller.
- Distributed software system employing a number of async agents.
- Handles the following kinds of messages.
  - Timeframe: signaling that a timeframe is reached. (sent by program scheduler)
    - Send a query message for the current network topology.
    - Send a query message for the current load information.
  - Topology: signaling that topology information has changed. (sent by Floodlight)
    - Update topology information.
  - Load: Periodic load information is received (sent by Floodlight)
    - Validate and apply current load info.
    - İf balancer is active, Algorithm phase 3.
  - Alert: signaling that attack has started (sent by Callophrys)
    - Turn on active mode.
    - Force send a timeframe message to self.

Callophrys: Evaluation & Conclusion

- Evaluated using Mininet and Floodlight SDN controller
- Test topology ---------->
- Attackers (A) generate high traffic (iperf)
- Varierty of bandwidths

- Experiments show that proposed solution helps to increase survival time of defended system during DDoS attack.
- Want to test at real network (future work)
- It is efective as DDoS mitigation solution in SDN, but can also be used as a general load balancing system.



Other Researches

Revisiting traffic anomaly detection using software defined networking
Syed Akbar Mehdi, Junaid Khalid, Syed Ali Khayam.
ACM Conference on SIGCOMM. Aug. 2014.

Software-Defined-Networking-Enabled Traffic Anomaly Detection and Mitigation
Daojing He, Sammy Chan, Xiejun Ni, Mohsen Guizani.
IEEE IoT Journal. Apr. 2017.

Enabling fast, dynamic network processing with clickOS
Joao Martins, Mohamed Ahmed, Costin Raiciu, Felipe Huici.
HotSDN, Aug. 2013.

Kandoo: a framework for efficient and scalable offloading of control applications
S. H. Yeganeh, Yashar Ganjali.
HotSDN, Aug. 2012.

Fleet: defending SDNs from malicious administrators
S. Matsumoto, S. Hitz, A. Perrig.
HotSDN, Agu. 2014.

Other Researches ++

SDN Based Architecture for IoT and Improvement of the Security
O. Flauzac, C. Gonzalez, A. Hachani, F. Nolot.
IEEE WAINA, Mar. 2015.

A fuzzy logic-based information security management for software-defined networks
S. Dotcenko, A. Vladyko, I. Letenko.
IEEE ICACT, Feb. 2014.

Deployment of Intrusion Prevention System based on Software Defined Networking
L. Zhang, G. Shou, Y. Hu, Z. Guo.
IEEE ICCT. Nov. 2013.

DynPaC: A Path Computation Framework for SDN
A. Mendiola, J. Astorga, E. Jacob, M. Higuero, A. Urtasun, V. Fuentes.
IEEE EWSDN, Oct. 2015.

OpenFlow: enabling innovation in campus networks.
N. McKeown and T. Anderson.
ACM SIGCOMM, 2008.

Kaan Özdinçer

Danışman: Hacı Ali Mantar

Seminer I - BİL591

Yazılım Tanımlı Ağlar ile Güvenlik