DEMO: SDN-based System to Filter Out DRDoS Amplification Traffic in ISP Networks

By
Priyanka.G.Dodia & Yury Zhauniarovich

Background

- Main Paper:
  - Sorting the Garbage: Filtering Out DRDoS Amplification Traffic in ISP Networks
  - Published in the Proceedings of IEEE NetSoft 2019
  - Conference held in Paris, France in June 2019

- Demo Paper:
  - Accepted at ACM's Computer & Communications Security (CCS) Conference
  - To be held in London, UK from Nov 11th-15th

Problem

- DRDoS attacks are major threat to Internet with large scale impact
- Goal: Bring down victim network by bombarding it with garbage traffic
- Popular among attackers:
  - Low resource requirements: Small spoofed request packets can reflect large response to victim
  - Attacker stays anonymous: SRC IP spoofed with victim ip
- Attackers use vulnerable servers with open UDP ports (dns, ntp, ssdp..) to reflect and amplify traffic to victim
  - Negative impact on benign users; owners of vulnerable machines
- ISPs host numerous vulnerable servers, if abused, can generate garbage (reflected amplified traffic) in Terabytes
  - Hard to detect
  - Loss of bandwidth and QOS to customers

Proposed Solution

- Our proposed solution:
  - Shield ***Amplifier Network* vs *Victim Network***
    - Most existing DRDoS solutions try to protect *victim networks*
    - Protect vulnerable amplifiers from spoofed amplification requests
  - Stop attack Midway
    - Detect **spoofed** traffic and filter out at edge of an ISP before it reaches amplifiers
    - Thereby reducing the storm of attack traffic directed towards the victim
  - Benefit ISPs and their customers
    - Reduce wasteful bandwidth consumption of ISP
    - Prevent loss of money for ISPs (asymmetric traffic agreements)
    - Prevent QoS degradation due to amplification during ongoing DRDoS attacks

Our Solution Prototype

- **Software Defined Networking** based system to filter out garbage traffic from an ISP network
- Simulate the test network in GNS3
- **Network Components**
  - Python based POX - SDN Controller
  - OpenFlow Edge Switch
  - Cisco Routers
  - Ubuntu based Host devices in ISP network
  - Amplification Honeypot installed on one of the ISP hosts
    - Listens to any incoming spoofed attack requests abusing udp services

GNS3

- **GNS3:**
  - Network Simulator software that seemlessly glues together different open source software
  - Allows to emulate a network that includes cisco routers, switches, cisco devices and any other devices that can run on QEMU or virtual box emulator
  - It also allows to connect the virtual network to the physical network, it is possible to access Internet in the emulated environment

- Each device in GNS3 is a **docker** image
  - DOCKER containers are similar to virtual machines but ***light weight***
  - They run on same kernel as the host
    - Quick Startup: Doesn’t simulate entire OS
    - More efficient in host resource usage

Demonstration

1. Main network components

2. Details of initial configuration steps

3. Launch DRDoS from attacker machine
    a. We show **spoofed** attack traffic sent:
        i. **ATTACKER -> REFLECTOR HOST at ISP**
        ii. **REFLECTOR HOST -> VICTIM**

Demonstration

4. We show how honeypot detects amplification requests and issues **block rule**

    a. We show reflected traffic from ISP stops reaching victim machine
        i. Spoofed requests from **ATTACKER -> AMPLIFIER** are dropped at ISP
        ii. No reflection after block rule is added

    b. Attack packets observable at honeypot
        i. **Proactive honeypot rule implemented with high priority**
        ii. Traffic to honeypot is not blocked so it can monitor attack end **to remove block rule**

5. We show once attack ends, the **block rule** is dropped from switch table and packets to victim SRC IP resume flow normally.

Demo Steps : Initial Set up

1. **Initial Setup:** Install Python on Controller and Honeypot machines
   - ❏ Controller runs python based DRDoS server script
   - ❏ Honeypot runs client script
   - ❏ Install honeypot configuration *(python setup.py)*

2. Terminal to **network components:**
   - ❏ Attacker, Host, Victim
   - ❏ Controller, Switch, Honeypot

3. **Startup:** Start DRDoS App and server script at **Controller**
   - ❏ Ping controller -> switch and all connected machines
   - ❏ ./pox.py log.level --DEBUG forwarding.controller openflow.discovery
   - ❏ Connectivity between machines:
     - ❏ Ping controller -> switch
     - ❏ Ping honeypot (10.0.0.4) -> host (10.0.0.5)
     - ❏ Ping Attacker (20.0.0.2) -> honeypot/host
     - ❏ Ping Victim (30.0.0.2) -> honeypot/host

Demo Steps

4. **Attack Start:** Abusing reflector at ISP
   - ❏ **Host :** tcpdump host 10.0.0.5 -nnS
   - ❏ **Victim :** tcpdump host 30.0.0.2 -nnS
   - ❏ **Attacker -> Host:** nping --icmp -S 30.0.0.2 10.0.0.5 -c 20(spoofed SRC IP)

5. Start attack monitor and client script at **Honeypot**
   - ❏ **Honeypot client script:** src/DDoSHoneypot.py
   - ❏ **Attacker -> Honeypot:** nping --udp -p123 -S 30.0.0.2 10.0.0.4 -c 4 (spoofed SRC IP)

6. **Attack Detection** at Honeypot
   - ❏ Show **block rule** at **Openflow switch :** Blocks reflection of packets to victim SRCIP
     - ❏ **Attacker -> Host:** nping --icmp -S 30.0.0.2 10.0.0.5 -c 20 (spoofed SRC IP)
   - ❏ Honeypot proactive rule with high priority **(cookie #6)**
     - ❏ Show packets received at honeypot during blockage
     - ❏ **Attacker -> Honeypot (Honeypot continues to receive packets):**
     - ❏ nping --udp -p123 -S 30.0.0.2 10.0.0.4 -c 4 (spoofed SRC IP)

7. Attack ends and block rule is dropped as instructed by honeypot

Extra Commands

- Port open:
  - apt install xinetd
  - nano /etc/xinetd.d/chargen
  - /etc/init.d/xinetd restart
  - nping --udp -S 30.0.0.2 10.0.0.5 -c 10 -p 19
  - tcpdump host -nnS 10.0.0.5 <host>
  - tcpdump host -nnS 30.0.0.2 <victim>