DDoS Defense in Depth for DNS (DDIDD)

John Heidemann (PI)
joint work with Wes Hardaker and Jeleana Mirkovic (co-Pi)
and ASM Rizvi and Robert Story

USC/ISI

2019-12-09

This research is sponsored by the project “CICI: RSARC: DDoS Defense in Depth for DNS”, NSF OAC-1739034.

Copyright © 2019 by Hardaker, Heidemann, Mirkovic; Release terms: CC BY-NC 4.0

DDoS is Bad… and Getting Worse

big
2012 innovation: automated botnets for extortion

bigger
2013 innovation: DNS amplification

biggerer (as of Feb. 2018)
2016: 620 Gb/s KrebsOnSecurity.com
800 Gb/s (or more?) OVH
innovation: 145k-node botnet from hacking IoT devices

cheap: booters offer DDoS-as-a-service starting at $1/attack
[Santanna et al, 2015]

biggest (so far)
2018: 1.3Tb/s memcached
Years of Research… the Problem Remains

fixing the problem at the root:  
⇒ but misaligned cost and benefits

mitigating the problem with services  
⇒ loses autonomy and can be expensive

• source address filtering (BCP38)  
  – hard to deploy for big ISPs  
  – only ~50% after 10 years of work

• attack traceback  
  – requires cooperation across ISPs

• better security in end-devices  
  – fundamentally hard to be perfect  
  – counter to the economics of commodity devices and IoT

• traffic scrubbing  
  – NTT, etc.  
  – re-route traffic, “clean it” (proprietary), forward it to you

• huge infrastructure with automated traffic shifting  
  – Akamai, Cloudflare, etc.

DDoS Fundamental Problem

• any open service must accept queries from everywhere

• end-devices will never be fully secure

• millions of devices exist (more every day)

• each attack is easy (DDoS-as-a-Service exists)

⇒ huge advantages for attacker
and no silver bullet
Our Approach: Defense In Depth

- no one silver bullet
- **Deep Layers**: a collection of countermeasures to mitigate attacks — chip away at *each part* of problem
- components
  - 1. hop-count filtering: anti-spoofing
  - 2. existing-name query whitelisting
  - 3. known client whitelisting
  - 4. aggressive client detection
  - 5. scale-out to cloud
- we will open source components

---

Project Status

- supported by NSF CICI
- started in Fall 2017
- builds on
  - B-Root revitalization (supported by USC and others)
  - prior NSF projects: FRADE ([https://steel.isi.edu/Project/frade/](https://steel.isi.edu/Project/frade/), USC)
  - prior studies of Root DNS DDoS (USC, U. Twente, and SIDN)
- complements other anti-DDoS projects
  - PAADDOS ([https://ant.isi.edu/paaddos/](https://ant.isi.edu/paaddos/); USC and U. Twente): anycast
  - LEADER (USC): anti-low-rate DDoS
  - DIINER ([https://ant.isi.edu/diiner/](https://ant.isi.edu/diiner/); USC): shared DNS testbed around B-Root
Target Application: DNS

- our work should apply to many apps, but DNS is our focus

- why? DNS is important and particularly challenging
  - most queries are UDP => spoofing is easy
  - service-level expectations often require answers
    - particularly for Root DNS
    - (* although exceptions for under attack)
  - amplification can make outbound traffic a bottleneck

Testing and Transition using B-Root DNS

- Root DNS is a key Internet service
  - has been DDoS’ed multiple times

- Steps in transition plan:
  - Test on B-Root infrastructure first (committed to support research)
  - Work with other DNS operators
    - Letters of interest from two other root operators
    - Joint collaboration with .nl
  - Publish results and release software as open source
Current Results

• specific filters
  – (1) source address filtering
  – (2) hop-count filtering
  – (3) client modeling
  – (4) response-code blacklisting
• (5) automatic filter selection
• curated datasets to support research
• future plans
(1) Source Address Filtering

- **idea**: (not new)
  - build a *whitelist* of typical service users
  - when attacked, keep *only traffic from the whitelist*

- **pros**:
  - simple, safe
  - will reduce outgoing traffic volume

- **cons**:
  - some false rejection (if whitelist is not perfect)
  - volumetric attacks can overwhelm *incoming* traffic

Source Address Filtering Status

- testing of ipsets at scale *source address filtering*
  - plug-in to Linux kernel
  - extends iptables (firewall) to support millions of filters

- **results**:
  - *yes* we can handle the typical B-Root customer set

- **deployment for B-Root completed**
  - automated whitelist construction; module deployed; attack playbook updated
(2) Hop-Count with (3) Client Modeling

• idea:
  – learn typical hop-count and rate from each source IP
  – filter by hop-counts
  – filter remaining traffic by rate
• pro:
  – hop-counts are stable, so good filter with low false positive
  – traffic with spoof known clients gets wrong hop-count
  – client-modeling catches anything that slips through
• con:
  – need new iptables module to hop-count filter at scale
  – client modeling may not be easy

Hop-count Filtering Status

• preliminary analysis looks promising
  – high precision (0.1% false drops after 1h training)
  – very high recall: drops 99.4% of attack traffic with random spoofing
  – concern: must track ~10M values/~3M if working with /24 prefixes
• new ipset extension to handle hop-count filtering
  – prototyped and evaluated in testbed
  – not yet deployed
Hop-count Filtering – Performance

learning from 90 minutes is enough

filtering accuracy is very high
(even for a naïve approach and a smart attack)

Hop-count Filtering vs. Smart Attackers

<table>
<thead>
<tr>
<th>TTL</th>
<th>Source</th>
<th>Entry size</th>
<th>Percent dropped</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>In table</td>
<td>/32</td>
<td>98.4%</td>
</tr>
<tr>
<td></td>
<td>In table</td>
<td>/24</td>
<td>98%</td>
</tr>
<tr>
<td></td>
<td>Not in table</td>
<td>/32 or /24</td>
<td>100%</td>
</tr>
<tr>
<td>Most popular TTL</td>
<td>In table</td>
<td>/32</td>
<td>40%</td>
</tr>
<tr>
<td></td>
<td>In table</td>
<td>/24</td>
<td>70%</td>
</tr>
<tr>
<td></td>
<td>Not in table</td>
<td>/32 or /24</td>
<td>100%</td>
</tr>
<tr>
<td>Exact TTL</td>
<td>In table</td>
<td>/32 or /24</td>
<td>0%</td>
</tr>
<tr>
<td></td>
<td>Not in table</td>
<td>/32 or /24</td>
<td>100%</td>
</tr>
</tbody>
</table>

very accurate vs. naïve attacker
somewhat accurate vs. adversary
ineffective vs. omniscient oracle (but impractical adversary)
(3) Client Modeling Details

- model request and error rate from each client
- *filter when client’s query rate increases suddenly*
  - intuition: tolerate typical aggressive users
  - but filter new ones
- also filter if client’s *error rate increases* (NXDOMAIN)
  - intuition: attackers often use fake names to avoid caching
- status: tested on several 2017 B-Root events
  - Good attacker identification, acceptable collateral damage

Client Modeling – Performance

![Graphs showing minimal collateral damage and large reduction in attack traffic](image-url)
Client Modeling – Performance

<table>
<thead>
<tr>
<th>Date</th>
<th>Precision</th>
<th>Recall</th>
<th>F1 score</th>
</tr>
</thead>
<tbody>
<tr>
<td>2017-11-30</td>
<td>0.99</td>
<td>0.99</td>
<td>0.99</td>
</tr>
<tr>
<td>2017-02-21</td>
<td>0.97</td>
<td>0.89</td>
<td>0.93</td>
</tr>
<tr>
<td>2017-03-06</td>
<td>0.93</td>
<td>0.93</td>
<td>0.93</td>
</tr>
<tr>
<td>2017-04-25</td>
<td>0.96</td>
<td>0.89</td>
<td>0.92</td>
</tr>
</tbody>
</table>

very effective (high accuracy) against all 2017 attacks vs. B-Root

(4) Response Code Blacklisting

- idea: some attacks send random strings (all fail)
- when attacked, *ignore replies that are failure* (NXDOMAIN)
  - challenge: normal replies (like typos) are also NXDOMAIN

- pro:
  - greatly cuts outgoing bitrate
- con:
  - lots of legitimate queries are NXDOMAIN (typos!), so defense has a high false positive rate
- result: defense of last resort
Automating Defenses: the Need for Choice

- in general, need *combination* of approaches
- possible filters on prior slides
- need to *automate* selection
  - to react quickly
  - and to keep re-evaluating
- how?
  - measure resource consumption directly
  - deploy most promising countermeasure
  - measure response and try alternative if unsuccessful

### (5) Automating Defenses

**require different defenses (no single method works all the time)**

<table>
<thead>
<tr>
<th>Event</th>
<th>Source Whitelisting</th>
<th>Response Blacklisting</th>
<th>Query Blacklisting</th>
<th>Converge to the best?</th>
<th>Latency to detect attack (s) (from start)</th>
<th>Latency to select the best filter (s) (from start)</th>
<th>No. of selected filters before the best choice</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015-11-30</td>
<td>Good</td>
<td>No</td>
<td>Good</td>
<td>Yes</td>
<td>13.17</td>
<td>13.33</td>
<td>1</td>
</tr>
<tr>
<td>2015-12-01</td>
<td>Good</td>
<td>No</td>
<td>Good</td>
<td>Yes</td>
<td>5.05</td>
<td>5.22</td>
<td>1</td>
</tr>
<tr>
<td>2016-06-25</td>
<td>Fair</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>10.24</td>
<td>10.24</td>
<td>0</td>
</tr>
<tr>
<td>2017-02-21</td>
<td>No</td>
<td>Fair</td>
<td>Good</td>
<td>Yes</td>
<td>6.67</td>
<td>38.81</td>
<td>3</td>
</tr>
<tr>
<td>2017-03-06</td>
<td>No</td>
<td>Fair</td>
<td>Good</td>
<td>Yes</td>
<td>14.33</td>
<td>15.37</td>
<td>1</td>
</tr>
<tr>
<td>2017-04-25</td>
<td>No</td>
<td>Fair</td>
<td>Good</td>
<td>Yes</td>
<td>11.73</td>
<td>12.03</td>
<td>1</td>
</tr>
</tbody>
</table>

**we always find the best defense, although sometimes it takes several tries**
Automated Defense: Dynamic Adaptation

<table>
<thead>
<tr>
<th>Case</th>
<th>Ingress network b/s (Gb/s)</th>
<th>CPU usage (%)</th>
<th>Egress network b/s (Gb/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without system</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Non-adaptive</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>With system</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This DDoS stresses the target’s egress link.

Automatic defenses work (see drop in egress traffic and CPU) but it fails when the attack changes.

We re-assess during attack to change filter => handles polymorphic attacks.

(6) Towards the Cloud

- IoT-based attackers can hit (nearly) any bitrate
- defense must be able to scale capacity
- => “fail to the cloud”
  - when under attack, add capacity in the cloud

- very positive discussions with 3 different cloud providers
- challenges:
  - requires anycast that spans us and cloud
  - want to use our own DDoS defenses in the cloud
  - while not harming other tenants
Cloud Status

• working on cloud-native implementation of B-Root for AWS
  – one VM provides all services
  – scales vertically (bigger instance) and horizontally (many instances)
• work in progress
  – prototype in place
  – but needs integration with our instrumentation and measurement
  – and need to be very careful with BYO-IP mixed with anycast

Curating Datasets from B-Root

• 5 events so far (attacks or large traffic bursts)
• 10 DITL events (each 2 days long)
  – 48-hour period, synchronized with other root letters
• new full week of data

• DITL and other DDoS datasets distributed through IMPACT
  – https://impactcybertrust.org
  – https://ant.isi.edu/datasets/
Relationship to Other DDoS Projects

- **LEADER** (NSF, started 2018)
  - PIs: Mirkovic and Hauser (ISI)
  - Looking into low-rate DDoS attacks and OS mechanisms to prevent them
  - May be useful to harden OS on root servers

- **PAADDoS** (started 2018)
  - PIs: Heidemann and Pras (U. Twente in .nl)
  - will examine anycast routing
  - ideas:
    - active use of anycast to adapt to attack load
    - anycast planning with Verfploeter

- **DIINER** (started 2019-10)
  - PIs: Heidemann and Hardaker (USC)
  - ideas:
    - leverage B-Root into an open testbed
    - data availability
    - experiments on live traffic

Conclusion

- DDoS is important but hard problem
- earliest tools deployed in B-Root
- additional tools and cloud are underway
- tools and data for you to use
  - open-source tool release in 2020q1
  - datasets available today

- [https://ant.isi.edu/ddidd/](https://ant.isi.edu/ddidd/)