DNS Response Rate Limiting

LISA14

13 November 2014
About the Presenter

Eddy Winstead

- Senior Systems Engineer, Internet Systems Consortium
- Sales Engineer, Configuration Inspector, Consultant
- BIND & ISC DHCP Trainer
- 20+ years of DNS, DHCP and sysadmin experience
ISC at a Glance

Open Source
- BIND DNS server
- ISC DHCP client, relay, server
- Kea new DHCP server

Network Services
- F-Root, one of 13 root server systems world-wide
- Hosted@, public-benefit hosting

Commercial Services
- Subscription Support Services
- BIND and DHCP
- Training
State of the Net - Cyber Attacks

• Cyber attacks against US businesses increased 42% compared to the previous year.

• Over 50% of the significant online operations experience five or more 2-6 hour DDoS attacks per month.

• DDoS attacks increased 20% in Q2, 2013, and have risen across the board in size, strength, and duration.
Distributed Denial of Service Attack

• DDoS attacks are used by malicious parties to force a computer resource—a website, network, or application—to stop responding to legitimate users.

• Motives
  – Ideology/Vendetta
  – Politics
  – Competition
  – Cloaking Criminal Activity
  – Extortion
  – Because we can…

• Examples
  – Smurf Attack
  – (S)SYN flood
  – Reflected DoS
Reflected DoS Attacks

- rDoS involves sending forged requests of some type to a very large number of computers that will reply to the requests.

Two steps are taken to conduct such an attack:

1. Attacker modifies IP packet data through Internet Protocol address spoofing.

2. Attacker searches for responses that are several times bigger than the request.
DDoS and DNS

• DNS is easily used for DDoS:
  – DNS lacks any source validation features
  – Most ISPs don’t check the source address of packets they send
  – Small DNS queries can generate large responses
    • DNS Amplification Attacks
Normal Traffic

- Source IP Address: 5.6.7.8
- Responds to 5.6.7.8

Internet

© 2014 www.isc.org
rDoS Attack

Source IP Address: 5.6.7.8
1.2.3.4

Internet

Responds to 1.2.3.4
Accidental(?) DNS Attacks

Poor Network Hygiene

- Non-caching name servers
- Too frequent flushing
- Open recursive servers (some ~25-30 Million, in fact!)
Cost of DDoS Attacks

- Revenue loss and lost sales
- Operational expenses related to downtime
- Decreased employee productivity
- Impact on customer experience
- Brand and reputation damage
- Breach of contract and violation of service level agreements
A SOLUTION ON THE AUTHORITATIVE SIDE OF THINGS...
How did RRL come about?

- ISC signed our zones in 2006
- Observed queries that were occurring too frequently from the same IP
- Defensive strategy sessions at ISC with Paul Vixie led to RRL

EDNS0 query for isc.org of type ANY is 36 bytes long
Response is 3,576 bytes long
Response Rate Limiting

• An Enhancement to the DNS
  – A mechanism for limiting the amount of unique responses returned by a DNS server
  – A mitigation tool for the problem of DNS Amplification Attacks
  – The only practical defense available for filtering in the name server
    • BIND 9.9.4 includes RRL as a key feature
      – Available for download at https://www.isc.org/downloads/
Benefits of RRL

• Improved efficiency and ability to deflect attacks
  – Huge reductions in network traffic
  – Huge reductions in server load

• Brand protection
  – Servers are no longer seen as participating in abusive network behavior.

• Smoother network traffic
  – Impact on legitimate traffic has been minimal
  – Significant drop in attack traffic
  – No dropped DNS queries
Boundaries of RRL

- At present, RRL implementation is recommended for *authoritative servers only*.

- RRL cannot identify which source addresses are forged and which are not.

- We can use the information from pattern analysis to throttle responses
  - Incoming queries are **NOT** throttled by RRL
Use Case

• **Symptom:**
  – ISP identifies a significant increase in the number of queries
  – Attackers use ISP’s response query to amplify attack
  – ISP’s DNS infrastructure contributes to the attack

• **Solution:**
  – Network operator at ISP enables RRL
  – Defines parameters to mitigate queries and response time

• **Result:**
  – ISP experiences huge reduction in traffic
  – Upholds positive corporate image; doesn’t contribute to the attack
ISC RRL DEPLOYMENT EXPERIENCE
RRL on ISC’s network

• Deployed on isc.org and SNS in Spring of 2012
• Deployed on F-root in Summer of 2013
ISC F-Root

f-ams1 traffic (~1 day, bits)

<table>
<thead>
<tr>
<th></th>
<th>Max</th>
<th>Avg</th>
<th>Max</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IN</strong></td>
<td>11.48Mb</td>
<td>4.43Mb</td>
<td>11.48Mb</td>
<td>0b</td>
</tr>
<tr>
<td><strong>OUT</strong></td>
<td>56.53Mb</td>
<td>9.21Mb</td>
<td>56.53Mb</td>
<td>0b</td>
</tr>
</tbody>
</table>

(as of Fri Jul 19 06:09:42 2013 GMT)
ISC F-Root

f-ams1 traffic (~1 day, bits)

- max input
- avg input
- max output
- avg output

(as of Fri Jul 19 06:09:42 2013 GMT)

IN Max(Max) = 11.48Mb Avg(Max) = 4.43Mb Max(Avg) = 11.48Mb Cur(Avg) = 0b
OUT Max(Max) = 56.53Mb Avg(Max) = 9.21Mb Max(Avg) = 56.53Mb Cur(Avg) = 0b
ISC F-Root

f-ams1 traffic (~1 day, bits)

Implemented RRL

Attackers gave up in frustration

max input  avg input  max output  avg output
(as of Fri Jul 19 06:09:42 2013 GMT)

IN  Max(Max)= 11.48Mb  Avg(Max)= 4.43Mb  Max(Avg)= 11.48Mb  Cur(Avg)= 0b
OUT  Max(Max)= 56.53Mb  Avg(Max)= 9.21Mb  Max(Avg)= 56.53Mb  Cur(Avg)= 0b
ENABLING & CONFIGURING RRL IN BIND
Enabling RRL

• RRL is available in ISC’s BIND 9.9.4 Software
  – Download: https://www.isc.org/downloads/
  – RRL support must be enabled with –enable-rrl prior to compiling
  – Documentation: https://kb.isc.org/article/AA-01000

```plaintext
options {
    directory "/var/named";
    rate-limit {
        responses-per-second 5;
        log-only yes;
    }
};
```
**K.I.S.S. (ISC’s RRL deployment philosophy)**

- **SLIP**
  - How many UDP requests can be answered with a truncated response.
  - Setting to “2” means every other query gets a short answer
    (much more on this topic later)

- **Window**
  - 1 to 3600 second timeframe for defining identical response threshold
  - Highly variable based on conditions

- **Responses-per-second**
  - How many responses per second for identical query from a single subnet
  - Highly variable based on conditions
rate-limit {
  slip 2; // Every other response truncated
  window 15; // Seconds to bucket
  responses-per-second 5; // # of good responses per prefix-length/sec
rate-limit {
    slip 2;       // Every other response truncated
    window 15;    // Seconds to bucket
    responses-per-second 5; // # of good responses per prefix-length/sec
    referrals-per-second 5; // referral responses
    nodata-per-second 5; // nodata responses
    nxdomains-per-second 5; // nxdomain responses
    errors-per-second 5; // error responses
    all-per-second 20; // When we drop all
rate-limit {
    slip 2; // Every other response truncated
    window 15; // Seconds to bucket
    responses-per-second 5; // # of good responses per prefix-length/sec
    referrals-per-second 5; // referral responses
    nodata-per-second 5; // nodata responses
    nxdomains-per-second 5; // nxdomain responses
    errors-per-second 5; // error responses
    all-per-second 20; // When we drop all
}

log-only no; // Debugging mode
rate-limit {
    slip 2;  // Every other response truncated
    window 15;  // Seconds to bucket
    responses-per-second 5;  // # of good responses per prefix-length/sec
    referrals-per-second 5;  // referral responses
    nodata-per-second 5;  // nodata responses
    nxdomains-per-second 5;  // nxdomain responses
    errors-per-second 5;  // error responses
    all-per-second 20;  // When we drop all

    log-only no;  // Debugging mode
    qps-scale 250;  // x / query rate * per-second  
        = new drop limit
    exempt-clients  {127.0.0.1; 192.153.154.0/24;};
rate-limit {
  slip 2;       // Every other response truncated
  window 15;    // Seconds to bucket
  responses-per-second 5;  // # of good responses per prefix-length/sec
  referrals-per-second 5; // referral responses
  nodata-per-second 5;  // nodata responses
  nxdomains-per-second 5; // nxdomain responses
  errors-per-second 5;   // error responses
  all-per-second 20;     // When we drop all

  log-only no;  // Debugging mode
  qps-scale 250; // x / 1000 * per-second
  // = new drop limit

  exempt-clients { 127.0.0.1; 192.153.154.0/24; 192.160.238.0/24 }

  ipv4-prefix-length 24; // Define the IPv4 block size
  ipv6-prefix-length 56; // Define the IPv6 block size
rate-limit {
    slip 2; // Every other response truncated
    window 15; // Seconds to bucket
    responses-per-second 5; // # of good responses per prefix-length/sec
    referrals-per-second 5; // referral responses
    nodata-per-second 5; // nodata responses
    nxdomains-per-second 5; // nxdomain responses
    errors-per-second 5; // error responses
    all-per-second 20; // When we drop all

    log-only no; // Debugging mode
    qps-scale 250; // x / 1000 * per-second
                   // = new drop limit
    exempt-clients { 127.0.0.1; 192.153.154.0/24; 192.160.238.0/24 }
    ipv4-prefix-length 24; // Define the IPv4 block size
    ipv6-prefix-length 56; // Define the IPv6 block size

    max-table-size 20000; // 40 bytes * this number = max memory
    min-table-size 500; // pre-allocate to speed startup
};
The SLIP=1 vs SLIP=2 debate

- ANSSI (CVE-2013-5661) recommends SLIP=1. Knot sets this as default.
- BIND & NSD defaults remain at SLIP=2

Let’s talk about why…
The SLIP=1 vs SLIP=2 debate

• The ANSSI (CVE-2013-5661) findings indicate SLIP=2 lowers the time needed for successful cache poisoning.

• While an authoritative server is suppressing responses, an attacker has an increased window to send malicious “responses” to a resolver.

• The findings aren’t surprising or disputed, but the recommendation (SLIP=1) is…
Additional data for the SLIP debate

• The ANSSI tests weren’t just Kaminsky-style attacks – but assumed only one authoritative nameserver in play due to SRTT trickery and/or Shulman fragmentation attack.

• 1 authoritative server, SLIP=2 lowered the time to successful poisoning from “days” to “hours”. ~16 hours at 100Mbit/sec.
Additional data for the SLIP debate

• Well… we already have a solution for cache poisoning!

DNSSEC

• Of course, deployment remains a challenge.
Final thoughts on SLIP

• ISC decided to keep the default at SLIP=2 in BIND as we think this best provides protection against the problem RRL was designed to address.

• Your SLIP decision will be based on finding the right balance of competing security concerns in your environment.
Use of Logfiles

- Initially use logging
- Use a separate logging channel to segregate data from regular logs

Log only “dry run” feature to view behavior before going live with RRL
logging {

  channel query-error_log {
    file "log/query-error.log" versions 7 size 100M;
    print-category yes;
    print-severity yes;
    print-time yes;
    severity info;
  };
  category query-errors { query-error_log; };
}
Additional Considerations

- Window length – interrupt self-monitoring
  - Whitelist option ‘exempt clients’
- Not responding to legitimate queries
RRL Classifier

• Expansion of RRL Basic
  – RRL Basic filters on Destination Address of Response (source of attack traffic is assumed to be forged, but provides address of attack target)

• 2014
  – Name Requested (QNAME)– allows for whitelisting and supports possible expansion to recursive use case
  – Size of the Response– limits amplification potential
Additional RRL General Information

• A Quick Intro to RRL: [https://kb.isc.org/article/AA-01000/189/](https://kb.isc.org/article/AA-01000/189/)

• What is a DNS Amplification Attack: [https://kb.isc.org/article/AA-00897](https://kb.isc.org/article/AA-00897)
Additional RRL Advanced Information

• Response to SLIP issue

• Vixie Article on DNS Security
WHAT ARE WE SEEING & DOING ON THE RECURSIVE SIDE?
What are we seeing on the recursive side these days?

• ‘Collateral Damage’ Client DDoS traffic
  <randomstring>.www.abc123.com
  <anotherstring>.www.abc123.com

The queries are unique and originate from a large range of different client addresses. Typically, the servers for abc123.com do not respond at all, or only sporadically to the recursive server handling the client query.

A flurry of queries will run for a day or two, then stop. The domains are genuine, and the majority appear to be for online commercial sites, often hosted in China.
Problem statement

• Authoritative servers under attack are non-responsive and tie up resolver resources wanting for replies
• So far, the impact on recursive server resources appears to be accidental - primarily due to open resolvers.
• This is a wake-up call that we need to better manage recursive resources
Resolver impact

1. Request for string.abc123.com
2. Attempt to resolve request
3. Server is unresponsive
4. Reply (NXDOMAIN or SERVFAIL)

Insecure Home gateway

Home user is probably oblivious

Initiator of DDoS traffic

Waiting for response from D

ISP resolver

Target of the DDoS Authoritative provider or their host

abc123.com
Mitigation Approaches

• Traffic patterns impacting all recursive servers (not just BIND)
• Mitigations suggested/introduced:
  – Network infrastructure/environment
  – Some generic to all DNS servers
  – Some specific to BIND (currently experimental) but could be adopted by other DNS server software manufacturers.
Mitigation Approaches - 1

• Eliminate open resolvers
  – Is your recursive server an open resolver?
  – Open client CPE devices
  – Small business users forwarding local open caches to your servers

• Compromised/infected clients
  – ‘hearsay’ evidence that these exist now
  – But it’s only a matter of time…
Mitigation Approaches – 2

• Locally-created authoritative answers
  – Detect ‘bad’ domain names
  – Make recursive server temporarily authoritative for the domain being used
  – Prevents valid queries (which wouldn’t succeed anyway)
  – Problem of false-positives – might need white-lists if using scripted detection
  – Need to undo the mitigation afterwards
Mitigation Approaches – 3

• Response Policy Zones (DNS-RPZ)
  – Detect ‘bad’ domain names
  – Update RPZ zone to blacklist domains
  – Prevents valid queries (which wouldn’t succeed anyway)
  – Problem of false-positives – might need white-lists if using scripted detection
  – Need to undo the mitigation afterwards
Experimental Approaches – 1

• **Hold-down Timer** *(since writing, deprecated and replaced with fetches-per-server)*
  – One timer each per server per zone
  – Count how many consecutive times a server fails to respond *(holddown-threshold)*
  – When threshold reached, don’t send queries to that server for *holddown-timer* seconds (doesn’t abort any currently waiting queries)
  – Quick check – if next ‘response’ from server is a timeout, then hold-down immediately
  – Helpful, but less effective with intermittent outages.
Experimental Approaches – 2

• Rate limiting *fetches-per-server*.
  – Configurable upper limit (default 0 = unlimited)
  – Per-server quota dynamically re-sizes itself based on the ratio of timeouts to successful responses
  – Completely non-responsive server eventually scales down to fetches quota of 2% of configured limit.
Experimental Approaches – 3

- Rate-limiting *fetches-per-zone*
  - Similar to clients-per-query
  - Works with unique clients
  - Tune larger/smaller depending on normal QPS to avoid impact on popular domains
  - Could be less effective against non-responding server for many zones
QUESTIONS?
Thank You