DNS: Victim or Attacker

Paul Ebersman
Paul_Ebersman@cable.comcast.com, @paul_ipv6
ICANN49, 24 Mar 2014, Singapore
Attacking your cache
Recursion

DNS queries are either recursive or nonrecursive

1) Recursive query for www.google.com/A
2) Nonrecursive query for www.google.com/A
3) Referral to com name servers
4) Nonrecursive query for www.google.com/A
5) Referral to google.com name servers
6) Nonrecursive query for www.google.com/A
7) A records for www.google.com
8) A records for www.google.com
Cache Poisoning

- What is it?
  - Inducing a name server to cache bogus records

- Made possible by
  - Flaws in name server implementations
  - Short DNS message IDs (only 16 bits, or 0-65535)

- Made easier on
  - Open recursive name servers
Cache Poisoning Consequences

• A hacker can fool your name server into caching bogus records

• Your users might connect to the wrong website and reveal sensitive information

• Your users’ emails might go to the wrong destination

• Man in the middle attacks, phishing, credentials theft
The Kashpureff Attack

Eugene Kashpureff’s cache poisoning attack used a flaw in BIND’s additional data processing.
DNS Message IDs

• Message ID in a reply must match the message ID in the query

• The message ID is a “random,” 16-bit quantity
Amit Klein of Trusteer found that flaws in most versions of BIND’s message ID generator (PRNG) don’t use sufficiently random message IDs.

- If the current message ID is even, the next one is one of only 10 possible values.
- Also possible, with 13-15 queries, to reproduce the state of the PRNG entirely, and guess all successive message IDs.
Birthday Attacks

• Brute-force guessing Msg ID is a birthday attack:

• 365 (or 366) possible birthdays, 65536 possible message IDs

• Chances of two people chosen at random having different birthdays:

• Chances of n people (n > 1) chosen at random all having different birthdays:

\[ \frac{364}{365} \approx 99.7\% \]

\[ p(n) = \left(1 - \bar{p}(n)\right) \]

\[ \bar{p}(n) = \frac{364}{365} \times \frac{363}{365} \times \ldots \times \frac{366 - n}{365} \]
Birthday Attacks (continued)

<table>
<thead>
<tr>
<th>People</th>
<th>Chances of two or more people having the same birthday</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>12%</td>
</tr>
<tr>
<td>20</td>
<td>41%</td>
</tr>
<tr>
<td>23</td>
<td>50.7%</td>
</tr>
<tr>
<td>30</td>
<td>70%</td>
</tr>
<tr>
<td>50</td>
<td>97%</td>
</tr>
<tr>
<td>100</td>
<td>99.99996%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number of reply messages</th>
<th>Chances of guessing the right message ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>~20%</td>
</tr>
<tr>
<td>300</td>
<td>~40%</td>
</tr>
<tr>
<td>500</td>
<td>~80%</td>
</tr>
<tr>
<td>600</td>
<td>~90%</td>
</tr>
</tbody>
</table>
The Kaminsky Vulnerability

How do you get that many guesses at the right message ID?
The Kaminsky Vulnerability (continued)

How does a response about q00001.paypal.com poison www.paypal.com’s A record?

Response:

;; -->>HEADER<<-- opcode: QUERY, status: NOERROR, id: 61718
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; QUESTION SECTION:
; q00001.paypal.com. IN A

;; AUTHORITY SECTION

;; ADDITIONAL SECTION
www.paypal.com. 86400 IN A 10.0.0.1
Initial Kaminsky fixes

• To make it more difficult for a hacker to spoof a response, we use a random query port in addition to a random message ID
• If we use 8K or 16K source ports, we increase entropy by 13 or 14 bits
• This increases the average time it would take to spoof a response substantially
• However, this is not a complete solution
• Spoofing is harder, but still possible
• Evgeniy Polyakov demonstrated that he could successfully spoof a patched BIND name server over high-speed LAN in about 10 hours
Defending your cache
Defenses

- More randomness in DNS msg IDs, source ports, etc.
- Better checks on glue
- DNSSEC
Overwhelming your authoritative servers
Sheer volume and persistence

- 10s of thousands of bots
- 10s of millions of open resolvers
  - (see http://openresolverproject.org/)
- Gbps of traffic generated
- 45% of ISPs experience 1-10 DDoS/month, 47% experience 10-500 DDoS/month
High yield results

- Small queries, large responses (DNSSEC records)

- Using NSEC3 against you
Make sure they’re your servers…

- Vet your registry/registrar
- Think about NS TTLs
How to defend your servers
Harden your server

- Perimeter ACLs
- Higher capacity servers
- Clusters or load balanced servers
- Response Rate limiting (RRL)
Spread yourself out

• Fatter internet pipes (but makes you more dangerous to others)

• More authoritative servers (up to a point)

• Anycast

• High availability
Being a good internet citizen
It’s not just you being attacked

• If you allow spoofed packets out from your network, you are part of the problem…

• Use BCP38/RFC3704 Ingress filtering

• Implement RFC5358

• http://openresolverproject.org/
Revise DNS Standards?
Changing RFCs?

- Glaciers start to look speedy
- Source Address Validation
- TCP vs UDP
- DNS Cookies
DNS use by the bad guys
DNS use by bad guys

- Command and control
- DNS Amplification
- Fastflux
  - single flux
  - double flux
- Storm, Conficker, etc.
Protecting your users
Dealing with malware

- Prevent infections (antivirus)
- Block at the perimeter (NGFW, IDS)
- Block at the client (DNS)
Antivirus

• Useful but has issues:
  - Depends on client update cycles
  - Too many mutations
  - Not hard to disable
  - Poor catch rates for new viruses
Perimeter defenses

- Necessary but not complete:
  - Limited usefulness after client is already infected
  - Detection of infected files only after download starts
  - Usually IP-based reputation lists
  - Limited sources of data
RPZ DNS

• Uses a reputation feed(s) (ala spam)

• Can be IP or DNS based ID

• Fast updates via AXFR/IXFR

• Protects infected clients, helps ID them

• Can isolate infected clients to walled garden
There is *not* only one

Use all methods you can!
Q & A
Thank you!