Poison Over Troubled Forwarders:
A Cache Poisoning Attack Targeting DNS Forwarding Devices

Xiaofeng Zheng, Chaoyi Lu, Jian Peng, Qiushi Yang, Dongjie Zhou, Baojun Liu, Keyu Man, Shuang Hao, Haixin Duan and Zhiyun Qian

Presenter: Xiang Li Tsinghua University
Internet Access & Common Devices
How does DNS work on these routers and WI-FI networks?

They serve as DNS forwarders.
DNS Forwarder

- Devices standing in between stub and recursive resolvers
  - E.g., home routers, open Wi-Fi networks
  - **Gateways** of access control
  - **Load balancers** for upstream servers
DNS Forwarder: Prevalent Devices

- Prevalent devices
  - IMC ’14
    - 32M, 95% are forwarders
  - IMC ’15
    - 17.8M, 76.4% are residential devices
  - Enabled by various software and routers
    - BIND, Unbound, Knot Resolver, and PowerDNS
    - TP-Link, D-Link, and Linksys
DNS Forwarder: Prevalent Devices

● Prevalent devices

○ Part of the complex DNS infrastructure

Kyle Schomp, Tom Callahan, Michael Rabinovich, Mark Allman. On measuring the client-side DNS infrastructure. IMC ‘13
DNS Forwarder: Security

- **Security status**
  - Forwarder vs Recursive resolver
    - bailiwick check, DNSSEC validation
  - **Relies on the integrity of upstream resolvers**
  - Do not check too much by itself
  - E.g., fail to check the src port and TXID (PAM ’14)
    - simple **cache poisoning attacks**
    - DoS attacks
DNS Cache Poisoning Attacks

One of the most influential attacks targeting DNS resolvers
**DNS Cache Poisoning Attacks**

- **Forging a valid DNS response**
  - Matching the DNS query’s metadata
    - Address, Port, DNS transaction ID (TXID), Query name
  - Type 1: **Forging Attacks**
  - Type 2: **Defragmentation Attacks**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>4 bits, defines the IP version (IPv4 or IPv6)</td>
</tr>
<tr>
<td>IHL</td>
<td>4 bits, indicates the length of the header in 32-bit units</td>
</tr>
<tr>
<td>Type of Service</td>
<td>8 bits, defines the type of service (e.g., TCP, UDP)</td>
</tr>
<tr>
<td>Total Length</td>
<td>16 bits, total length of the packet</td>
</tr>
<tr>
<td>Identification</td>
<td>8 bits, identifies the fragment (if fragmented)</td>
</tr>
<tr>
<td>Time To Live</td>
<td>16 bits, time to live</td>
</tr>
<tr>
<td>Protocol</td>
<td>8 bits, the transport protocol (e.g., TCP, UDP)</td>
</tr>
<tr>
<td>Header Checksum</td>
<td>16 bits, the checksum of the header</td>
</tr>
<tr>
<td>Source Address</td>
<td>IP address of the sender</td>
</tr>
<tr>
<td>Destination Address</td>
<td>IP address of the receiver</td>
</tr>
<tr>
<td>Source Port</td>
<td>16 bits, the source port</td>
</tr>
<tr>
<td>Destination Port</td>
<td>16 bits, the destination port</td>
</tr>
<tr>
<td>Length</td>
<td>16 bits, the length of the data</td>
</tr>
<tr>
<td>Checksum</td>
<td>16 bits, the checksum of the data</td>
</tr>
<tr>
<td>Transaction ID</td>
<td>32 bits, unique identifier for each transaction</td>
</tr>
<tr>
<td>QDCOUNT</td>
<td>16 bits, number of questions received</td>
</tr>
<tr>
<td>NSCOUNT</td>
<td>16 bits, number of name servers returned</td>
</tr>
<tr>
<td>ANCOUNT</td>
<td>16 bits, number of answers received</td>
</tr>
<tr>
<td>Opcode</td>
<td>8 bits, type of query</td>
</tr>
<tr>
<td>Flags</td>
<td>8 bits, control flags</td>
</tr>
<tr>
<td>Z</td>
<td>8 bits, reserved</td>
</tr>
<tr>
<td>RCODE</td>
<td>16 bits, response code</td>
</tr>
</tbody>
</table>
DNS Cache Poisoning Attacks: Type 1

- **Type 1: Forging Attacks**
  - Guessing the metadata, e.g., TXID, src port
    - e.g., the BIND Birthday Attack, the Kaminsky Attack
    - others, e.g.,
    - attack with NAT, DNS proxy attack, sock overloading
  - Mitigation
    - randomize, randomize, randomize (RFC 5452)
    - src port, TXID, qname
Do randomization defenses end forging attacks?

Yes or No? Proud or Upset.

E.g., SAD DNS Attack with side-channels
DNS Cache Poisoning Attacks: Type 2

- **Type 2: Defragmentation Attacks**
  - *Circumventing the metadata*, e.g., TXID, src port

---

Diagram:
- **Attacker**
- **Recursive resolver**
- **Authoritative Server (victim.com)**

0. Spoofed 2nd fragment (no UDP and DNS headers)
1. DNS Query

Rogue response cached by recursive resolver

2nd Spoofed fragment cached

1st Fragmented response
2nd Forced fragmentation

Defragmented with Spoofed 2nd fragment
DNS Cache Poisoning Attacks: Type 2

- **Type 2: Defragmentation Attacks**
  - Forcing a fragmentation
  - Lower the MTU → difficult now
    - 0.7% Alexa Top 100k domains is willing to reduce the MTU to < 528 bytes
    - 0.3% of 2M open resolvers can reduce the MTU to < 512 bytes
  - Use the DNSSEC records → cannot target arbitrary domains
    - Non-validating recursive resolvers
    - DNSSEC deployment is still low
    - The attack only works for DNSSEC-signed domains
Our New Defragmentation Attack

Targeting DNS forwarders
Threat Model: Overview

- Defragmentation attacks targeting DNS forwarders
  - Reliably force DNS response fragmentation
  - Target arbitrary victim domain names
Threat Model: Overview

- Defragmentation attacks targeting DNS forwarders
  - Reliably force DNS response fragmentation
  - Target arbitrary victim domain names

1. Attacker & DNS forwarder locate in the same LAN (e.g., in open Wi-Fi networks)
2. Use attacker’s own domain name and authoritative server

Diagram:
- LAN
- Attacker
- DNS Forwarder
- Recursive resolver
- Authoritative Server (attacker.com)
Threat Model: Insight on Forwarder Roles

- Defragmentation attacks targeting DNS forwarders
  - **Reliably** force DNS response fragmentation
  - Target **arbitrary victim domain names**

1. Attacker & DNS forwarder locate in the same LAN (e.g., in open Wi-Fi networks)
   - Relies on recursive resolvers
   - Target of cache poisoning

2. Use attacker’s own domain name and authoritative server
   - Security checks (e.g., DNSSEC)
Motivation

Threat Model

Attack Workflow

Experiment

Discussion
Flow of Defragmentation Attack: Step 0&1

- Defragmentation attacks targeting DNS forwarders

1. Craft spoofed 2nd fragment
Crafting Spoofed 2nd Fragment

Challenge: guessing the IPID
Crafting Spoofed 2nd Fragment

- No UDP and DNS headers in the 2nd fragment
- IPID Prediction is needed
  - The IPIDs of the 2nd and 1st fragment should agree
Crafting Spoofed 2nd Fragment

- **IPIID assignment algorithms**
  - Global IPIID Counter
  - Random IPIID Counter
  - **Hash-based IPIID Counter**
    - key: <src IP, dst IP>
    - increased number: [1, the number of system ticks]

- **Predicting the hash-based IPIID**
  - same “NAT-ed” public src address
  - send the 2nd fragment quick
Crafting Spoofed 2nd Fragment

- Predictable IPID measurement results
  - Incremental IPID counter
    - Open DNS resolvers: 4.2M
  - Hashed-based IPID counter
    - OS: Windows 10 (version 1909), ubuntu (5.3.0.29-generic)
    - Public DNS services:
      - Cloudflare, Quad9, Comodo, OpenDNS, Norton

- Other header fields
  - Fragment offset
  - IP source address
  - UDP checksum
Flow of Defragmentation Attack: Step 2

- Defragmentation attacks targeting DNS forwarders

1. Craft spoofed 2nd fragment
2. Issue a DNS query
Flow of Defragmentation Attack: Step 3

- Defragmentation attacks targeting DNS forwarders

1. Craft spoofed 2nd fragment
2. Issue a DNS query
3. Authoritative returns oversized response (> Ethernet MTU)
Forcing a fragmentation of the DNS Response

Via oversized DNS responses
Attacker’s Oversized DNS Response

- CNAME chain
  - Use dummy **CNAME records** to enlarge attacker’s DNS response

<table>
<thead>
<tr>
<th>1st fragment</th>
</tr>
</thead>
<tbody>
<tr>
<td>a.attacker.com CNAME b.attacker.com</td>
</tr>
<tr>
<td>b.attacker.com CNAME c.attacker.com</td>
</tr>
<tr>
<td>c.attacker.com CNAME d.attacker.com</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>x.attacker.com CNAME y.attacker.com</td>
</tr>
<tr>
<td>y.attacker.com CNAME z.attacker.com</td>
</tr>
<tr>
<td>z.attacker.com A x.x.x.x</td>
</tr>
</tbody>
</table>

| 2nd fragment          |

> 1,500 Bytes (Ethernet MTU)
Always produce fragments
Attacker’s Oversized DNS Response

- **CNAME chain**
  - Use dummy **CNAME records** to enlarge attacker’s DNS response
  - Use CNAME to **point attacker’s domain to any victim**

**What the recursive resolver sees**

- **1st fragment**
  - a.attacker.com CNAME b.attacker.com
  - b.attacker.com CNAME c.attacker.com
  - c.attacker.com CNAME d.attacker.com
  - ... 
  - x.attacker.com CNAME y.attacker.com
  - y.attacker.com CNAME z.attacker.com
  - z.attacker.com A x.x.x.x

**What the DNS forwarder sees**

- **1st fragment**
  - a.attacker.com CNAME b.attacker.com
  - b.attacker.com CNAME c.attacker.com
  - c.attacker.com CNAME d.attacker.com
  - ... 

- **2nd fragment**
  - x.attacker.com CNAME y.attacker.com
  - y.attacker.com CNAME victim.com
  - victim.com A a.t.k.r

**Spoofed 2nd fragment**
Flow of Defragmentation Attack: Step 4

- Defragmentation attacks targeting DNS forwarders

1. Craft spoofed 2nd fragment
2. Issue a DNS query
3. Authoritative returns oversized response (> Ethernet MTU)
4. Defragment by forwarder
Flow of Defragmentation Attack: Bingo

- Defragmentation attacks targeting DNS forwarders

1. Craft spoofed 2nd fragment
2. Issue a DNS query
3. Authoritative returns oversized response (> Ethernet MTU)

---

**Flow Diagram**

- **0a. Any query (to recursive)**
  - **Current IPID**
  - **Predicted IPID**
    - 1. Spoofed 2nd fragment
      - Header: `victim.com A a.t.k.r`
    - 2a. Query `a.attacker.com`
  - 2b. Query `a.attacker.com`
  - 2c. Query `a.attacker.com`
  - 2d. Follow aliases (CNAME)
  - 3a. Responses

- **3b. Fragmented response**
  - 1st Fragmented response
  - 2nd Fragmented response

- **3c. Reassembled rogue response**
  - **Header (CNAME chain)**
  - **Header** `victim.com A a.t.k.r`

- **Lack Security Checks**

---

**Key Points**

- DNS forwarders
- oversized response (> Ethernet MTU)
- spoofed fragments
- reassembled rogue response
- lack of security checks
Conditions of Successful Attacks
Conditions of Successful Attacks: C1

● **EDNS(0) support**
  ○ Allows transfer of DNS messages > **512 Bytes over UDP**
  ○ To force a fragmentation
  ○ *Is being increasingly supported* by DNS software
    ■ BIND, Knot DNS, Unbound, and PowerDNS
  ○ Is supported by most recursive resolvers
Conditions of Successful Attacks: C2

- DNS caching by record
  - Caching the answers as a whole
    - a.attacker.com A a.t.k.r
  - Caching the answers by record
    - a.attacker.com CNAME b.attacker.com
    - ...
    - victim.com A a.t.k.r
Conditions of Successful Attacks: others

- No active truncation of DNS response
  - Ensures that the entire oversized response is transferred

- No response verification
  - DNS forwarders rely on upstream resolvers
  - No “re-query” for the aliases
Which DNS software is vulnerable?
Vulnerable DNS Software

Test results

- 2 kinds of popular DNS software are vulnerable
  - `dnsmasq` (used by OpenWRT), `Microsoft DNS`
  - others
    - DNRD caches DNS responses as a whole
    - BIND, Unbound, Knot, and PowerDNS re-query the CNAME chain

<table>
<thead>
<tr>
<th>Software</th>
<th>Version</th>
<th>EDNS(0) &amp; No truncation</th>
<th>Cache by Record</th>
<th>No Verification</th>
<th>Vulnerable</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>dnsmasq</code></td>
<td>2.7.9</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>MS DNS</td>
<td>2019</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Vulnerable Home Routers

- **Test results**
  - 16 models are tested (by real attacks in controlled environment)
  - **8 models** are vulnerable
  - others
    - either do not support EDNS(0) or truncate the large response
    - no one re-queries the aliases

<table>
<thead>
<tr>
<th>Brand</th>
<th>Model</th>
<th>EDNS(0)</th>
<th>No Truncation</th>
<th>Cache by Record</th>
<th>Vulnerable</th>
</tr>
</thead>
<tbody>
<tr>
<td>D-Link</td>
<td>DIR 878</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>RT-AC66U B1</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Linksys</td>
<td>WRT32X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Motorola</td>
<td>M2</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Xiaomi</td>
<td>3G</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>GEE</td>
<td>Gee 4 Turbo</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Wavlink</td>
<td>A42</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Volans</td>
<td>VE984GW+</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Real Attacks

- **Complex network experiment**
  - Home router: OpenWRT with dnsmasq
  - Client and attacker
    - in the same LAN
    - plus 13 other clients, e.g., mobile phones and tablets
    - 7.95Mbps/753.3Kbps of inbound/outbound traffic
  - Upstream recursive resolver: Norton public resolver
  - Authoritative resolver
  - **It takes 58s to complete a successful attack**
Real Attacks

- Complex network experiment
  - Home router: OpenWRT with dnsmasq
  - Client and attacker in the same LAN plus 13 other clients, e.g., mobile phones and tablets
  - 7.95 Mbps/753.3 Kbps of inbound/outbound traffic

- Upstream recursive resolver: Norton public resolver

- Authoritative resolver

- It takes 58s to complete a successful attack
How many real-world devices are affected potentially?
Measuring Clients Potentially Under Risk

- Collect vantage points
  - Implement measurement code in a network diagnosis tool
  - **20K clients**, mostly located in China

- Check the forwarder conditions
  - Ethical considerations: no real attack
  - 40% do not support EDNS(0) yet
  - **Estimated vulnerable clients: 6.6%**
Responsible Disclosure

- Responsible Disclosure
  - Submitting reports and connecting via emails
  - **ASUS** and **D-Link** release firmware patches
    - Caching the responses as a whole
  - **Linksys** accepts the issue via BugCrowd platform
  - **Microsoft** confirms the issue via Microsoft Bounty Program
Mitigation

● Mitigation for DNS forwarders
  ○ DNS caching by response (short-term solution)
    ■ Cache the responses as a whole
  ○ 0x20 encoding on DNS records
    ■ Encode names and aliases in all records
  ○ Perform response verification
    ■ DNSSEC
    ■ Re-query all names and aliases
      ● Should the forwarder do verification?
      ● Lack clear guidelines of DNS forwarders
So, what are DNS forwarders?

What role should they play?
What features should be supported?
DNS Forwarder Specifications

- RFC 1034
  - No discussion on DNS forwarding
- Now, multiple layers of server
  - stub resolver, *forwarder*, recursive resolver, authoritative resolver
- Different RFCs, different names
  - RFC 2136, 2308, 3597, 5625, 7626, 7871, 8499
- Two definitions of “forwarder”
  - **D1:** Serve as upstream servers of recursive resolvers
  - **D2:** Stand between stub resolvers and recursive resolvers
### DNS Forwarder Specifications: D1

- **Definition 1**
  - Serve as upstream servers of recursive resolvers

- **Uses**
  - Be leveraged to access authoritative servers
  - Have better Internet connection or bigger cache ability

<table>
<thead>
<tr>
<th>RFC</th>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
</table>
| 2136 | Dynamic Updates in the Domain Name System (DNS UPDATE) | When a zone slave forwards an UPDATE message…, enter the role of “forwarding server”.
| 2308 | Negative Caching of DNS Queries (DNS NCACHE) | … a bigger cache which may be shared amongst many resolvers.
| 7626 | DNS Privacy Considerations | ... these forwarders are like resolvers. |
DNS Forwarder Specifications: D2

- **Definition 2**
  - Stand between stub resolvers and recursive resolvers

- **Uses**
  - Take queries from clients, pass the requests on to another server

<table>
<thead>
<tr>
<th>RFC</th>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3597</td>
<td>Handling of Unknown DNS Resource Record (RR) Types</td>
<td>… forwarders used by the client.</td>
</tr>
<tr>
<td>5625</td>
<td>DNS Proxy Implementation Guidelines</td>
<td>(DNS) proxies are usually simple DNS forwarders …, relies on an upstream resolver ...</td>
</tr>
<tr>
<td>7871</td>
<td>Client Subnet in DNS Queries</td>
<td><strong>Forwarding Resolvers</strong>, … Recursive Resolver handles the query</td>
</tr>
<tr>
<td>8499</td>
<td>DNS Terminology</td>
<td>stand between stub resolvers and recursive servers.</td>
</tr>
</tbody>
</table>
DNS Forwarder Implementations

- Lack clear guidelines of DNS forwarders
  - The term of DNS forwarders is updated by RFC 8499
  - There are no implementation details -> diverse implementations

- What should a DNS forwarder do
  - How to handle DNS responses
  - Whether should they cache
  - Whether should they “re-query” some responses

- Only RFC 5625: DNS Proxy
  - DNS proxies should be as transparent as possible
  - Forward DNS packets (up to 4,096 octets)
Implementation guidelines of the DNS forwarder are needed.

To guarantee better security
● An attack targeting DNS forwarders
● Affects forwarder implementations extensively
● Call for more attention on DNS forwarder security

Any Questions?

zxf19@mails.tsinghua.edu.cn
Poison Over Troubled Forwarders: A Cache Poisoning Attack Targeting DNS Forwarding Devices

Xiaofeng Zheng, Chaoyi Lu, Jian Peng, Qiushi Yang, Dongjie Zhou, Baojun Liu, Keyu Man, Shuang Hao, Haixin Duan and Zhiyun Qian

Xiang Li