(Well-known) "Redirect the target domain's nameserver“ cache poisoning attacks

Kazunori Fujiwara, JPRS
fujiwara@jprs.co.jp

IEPG meeting, Toronto
2014/7/20

(Well-known) references

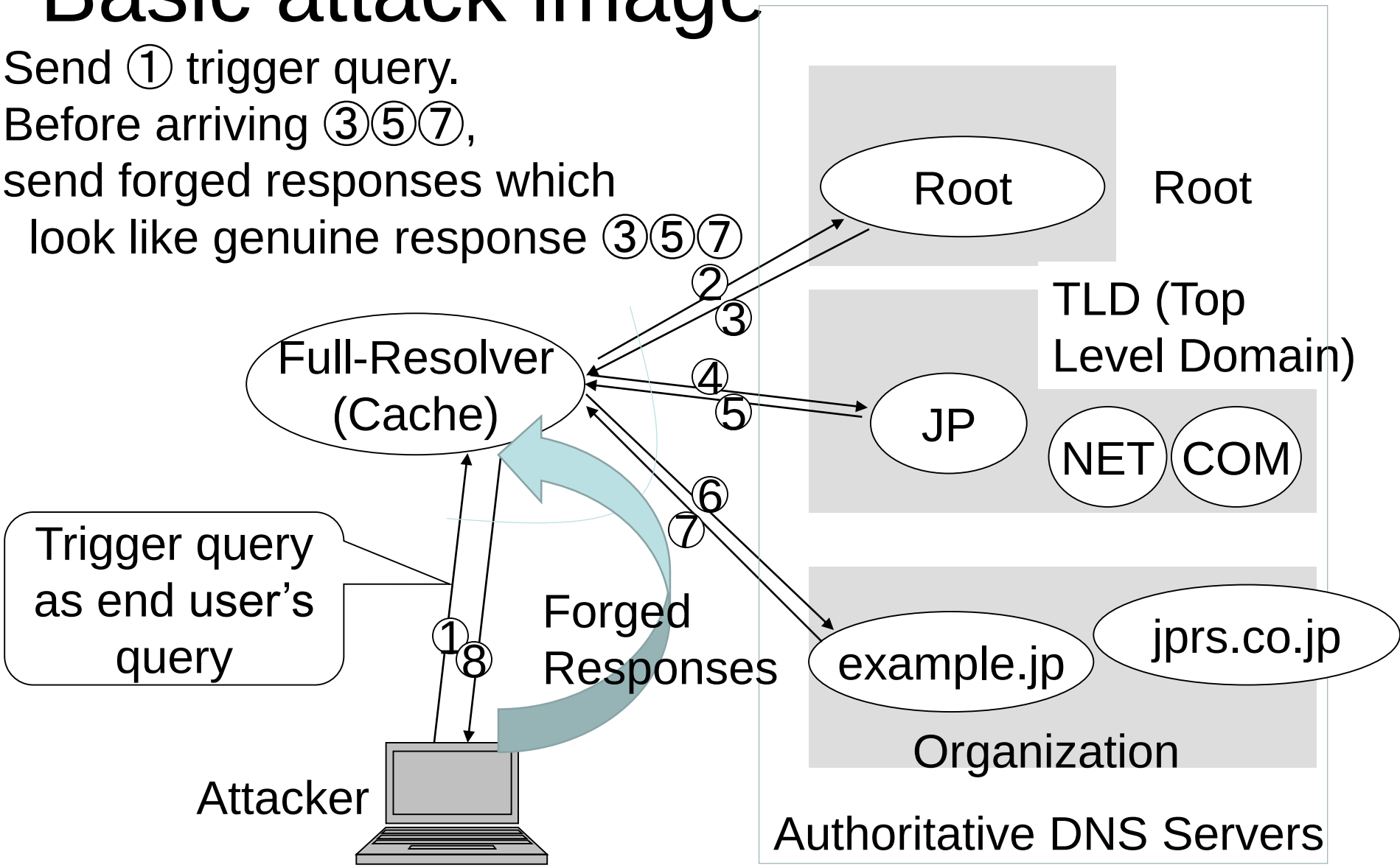
- Bernhard Müller, “Improved DNS spoofing using node re-delegation”, August 2008
- Wikipedia DNS spoofing
  - http://en.wikipedia.org/wiki/DNS_spoofing

> Redirect the target domain's nameserver
>
> The first variant of DNS cache poisoning involves redirecting the nameserver of the attacker’s domain to the nameserver of the target domain, then assigning that nameserver an IP address specified by the attacker.

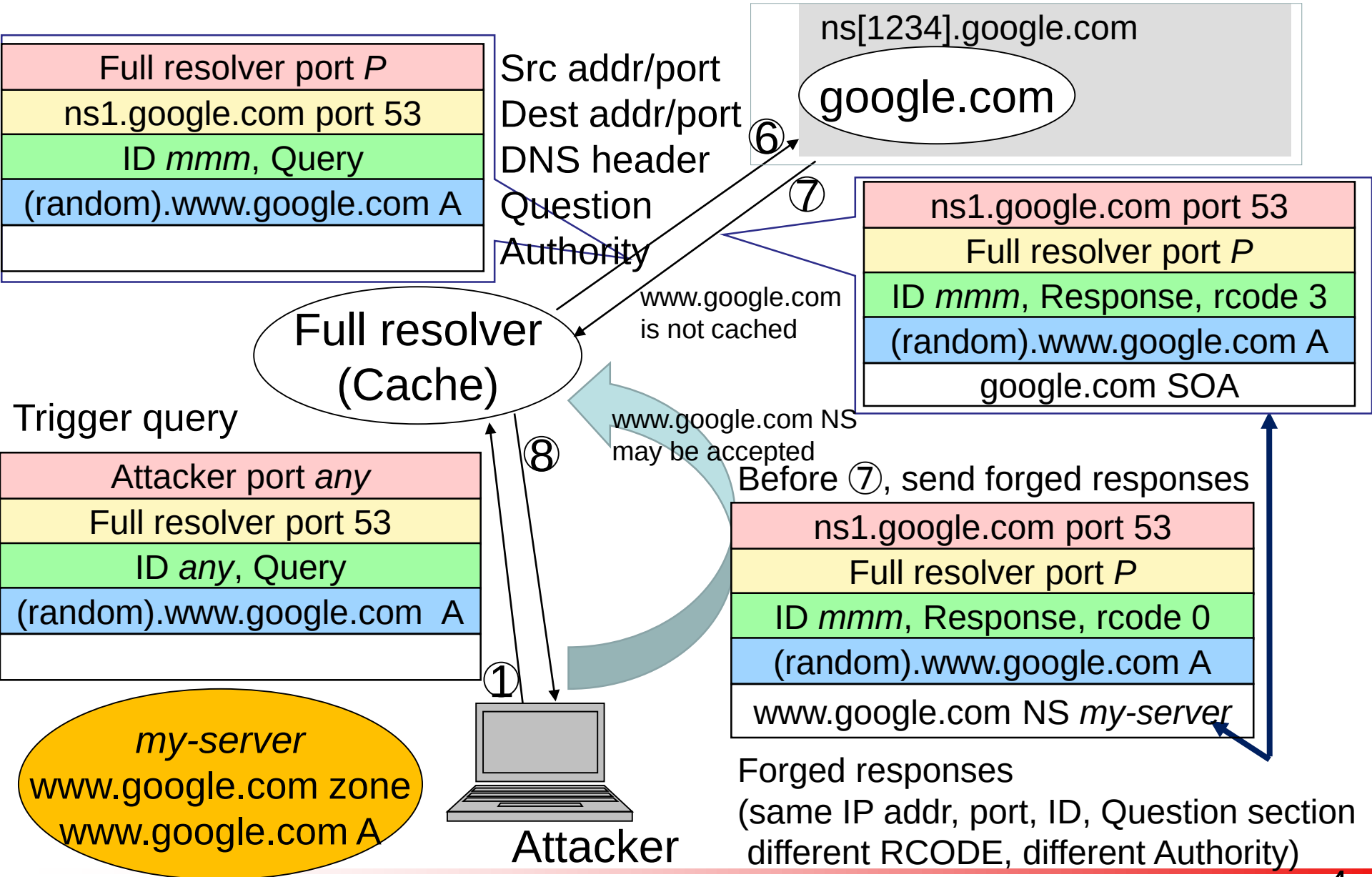
- Dan Kaminsky, “DNS 2008 and the new (old) nature of critical infrastructure”, July 2008
- RFC 3833 Threat Analysis of the Domain Name System (DNS)
  - 2.2. ID guessing and query prediction

Basic attack image

Send ① trigger query.
Before arriving ③⑤⑦,
send forged responses which look like genuine response ③⑤⑦



Detailed attack image (www.google.com)



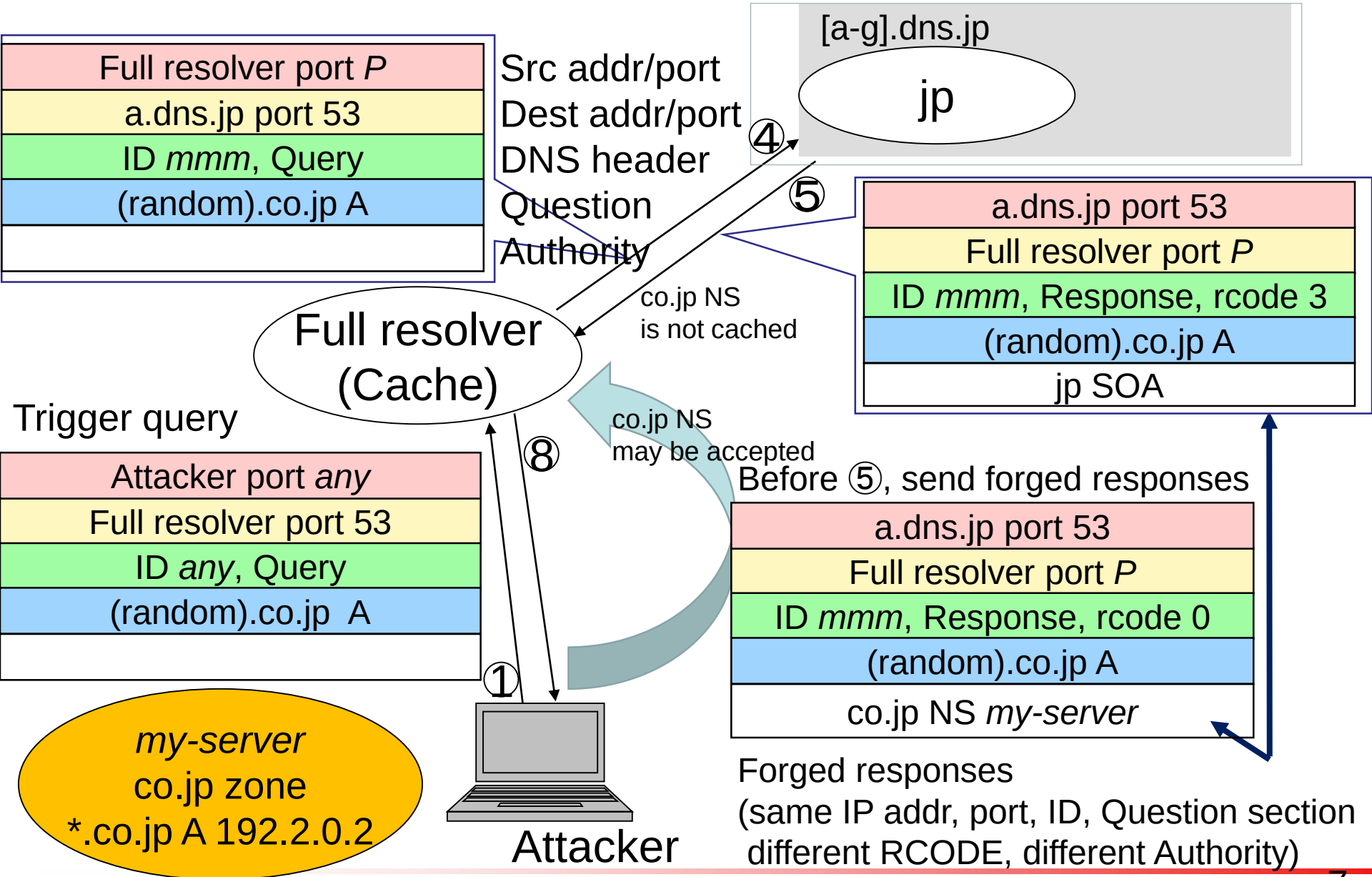
Attack details

- Choose a trigger domain name ($Trigger)
- Send a trigger query “(*random*).$Trigger A”
  - Authoritative server returns name error (rcode 3) for the query name
  - The trigger domain name is not cached by the full resolver
- Send forged responses
  - IP src = authoritative server addresses of Trigger domain
  - IP dst = full resolver’s address
  - src port = 53
  - dst port = full resolvers port (static, or random number)
  - DNS header: Rcode 0, response, ID = random number
  - Question section = “(*random*).$Trigger A”
  - Authority = what you want to inject : “Target NS my-server”
- Prepare Target zone in my-server
  - *.Target IN A 192.2.0.2

Target domain names

1. Domain names which does not have NS
   - Host names (www.google.com, www.iepg.org, ...)
   - Some non-terminals (co.jp, tokyo.jp, …)
2. Some zone cuts
   - When an authoritative server serves both a zone and its descendant zones
   - The shallowest zone apex is not a target
   - If a deepest name is a zone cut, it is not a target
   - Example 1: “net” and “root-servers.net”
     - Root servers serve both “.” and “root-servers.net”
   - Example 2: “co.uk”
     - uk servers serve both “uk” and “co.uk”

Attack example:co.jp



Attack example:net

Only when “net NS” is not cached by victim full resolvers

Full resolver port *P*
a.root-servers.net port 53
ID *mmm*, Query
(random).root-servers.net A

Src addr/port
Dest addr/port
DNS header
Question
Authority

[a-m].root-servers.net

② “.” and root-servers.net

③

a.root-servers.net port 53
Full resolver port *P*
ID *mmm*, Response, rcode 3
(random).root-servers.net A
root-servers.net SOA

net NS
is not cached

Full resolver
(Cache)

Trigger query

Attacker port *any*
Full resolver port 53
ID *any*, Query
(random).root-servers.net A

⑧

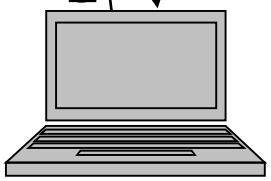
net NS
may be accepted

Before ③, send forged responses

a.root-servers.net port 53
Full resolver port *P*
ID *mmm*, Response, rcode 0
(random).root-servers.net A
net NS *my-server*

①

my-server
net zone
*.net A 192.2.0.2



Attacker

Forged responses
(same IP addr, port, ID, Question section
different RCODE, different Authority)

Attack example: root-servers.net

~~Only when "net NS" is not cached by victim full resolvers~~

Full resolver port *P*
a.root-servers.net port 53
ID *mmm*, Query
(random).root-servers.net A

Src addr/port
Dest addr/port
DNS header
Question
Authority

[a-m].root-servers.net

② "." and root-servers.net

③

a.root-servers.net port 53
Full resolver port *P*
ID *mmm*, Response, rcode 3
(random).root-servers.net A
root-servers.net SOA

root-servers.net NS is not cached

Full resolver (Cache)

Trigger query

Attacker port *any*
Full resolver port 53
ID *any*, Query
(random).root-servers.net A

⑧

root-servers.net NS may be accepted

Before ③, send forged responses

a.root-servers.net port 53
Full resolver port *P*
ID *mmm*, Response, rcode 0
(random).root-servers.net A
root-servers.net NS *my-server*

①

my-server
root-servers.net zone
***.root-servers.net A**

Attacker

Forged responses
(same IP addr, port, ID, Question section
different RCODE, different Authority)

Attack example:co.uk

Full resolver port *P*
ns1.nic.uk port 53
ID *mmm*, Query
(random).co.uk A

Src addr/port
Dest addr/port
DNS header
Question
Authority

[1-d].nic.uk

uk and co.uk

④

⑤

ns1.nic.uk port 53
Full resolver port *P*
ID *mmm*, Response, rcode 3
(random).co.uk A
co.uk SOA

co.uk NS
is not cached

Full resolver
(Cache)

Trigger query

Attacker port *any*
Full resolver port 53
ID *any*, Query
(random).co.uk A

⑧

co.uk NS
may be accepted

Before ⑤, send forged responses

ns1.nic.uk port 53
Full resolver port *P*
ID *mmm*, Response, rcode 0
(random).co.uk A
co.uk NS *my-server*

①

my-server
co.uk zone
*.co.uk A 192.2.0.2

Attacker

Forged responses
(same IP addr, port, ID, Question section
different RCODE, different Authority)

Attack success probability and expected value of attack time

	Variable	Unit	Values	Example values
Num of resolver's ports	$Nport$		$1\sim2^{16}$	1 or 64,000
Num of IDs	$Nqid$		2^{16}	65,536
Num of auth servers	Nns		$1\sim13$	4
RTT to auth servers	$Tauth$	second	$0.001\sim0.2$	0.039
Loop time	$Tloop$	second		0.020
Response send rate	$Rans$	packets/sec		100,000

- Success probability of first time trial

$$P = \frac{Tauth * Rans}{Nqid * Nport * Nns}$$

Example:
0.0148
Or
0.00000023

- Success probability under continuation attack is 1
- Expected attack time

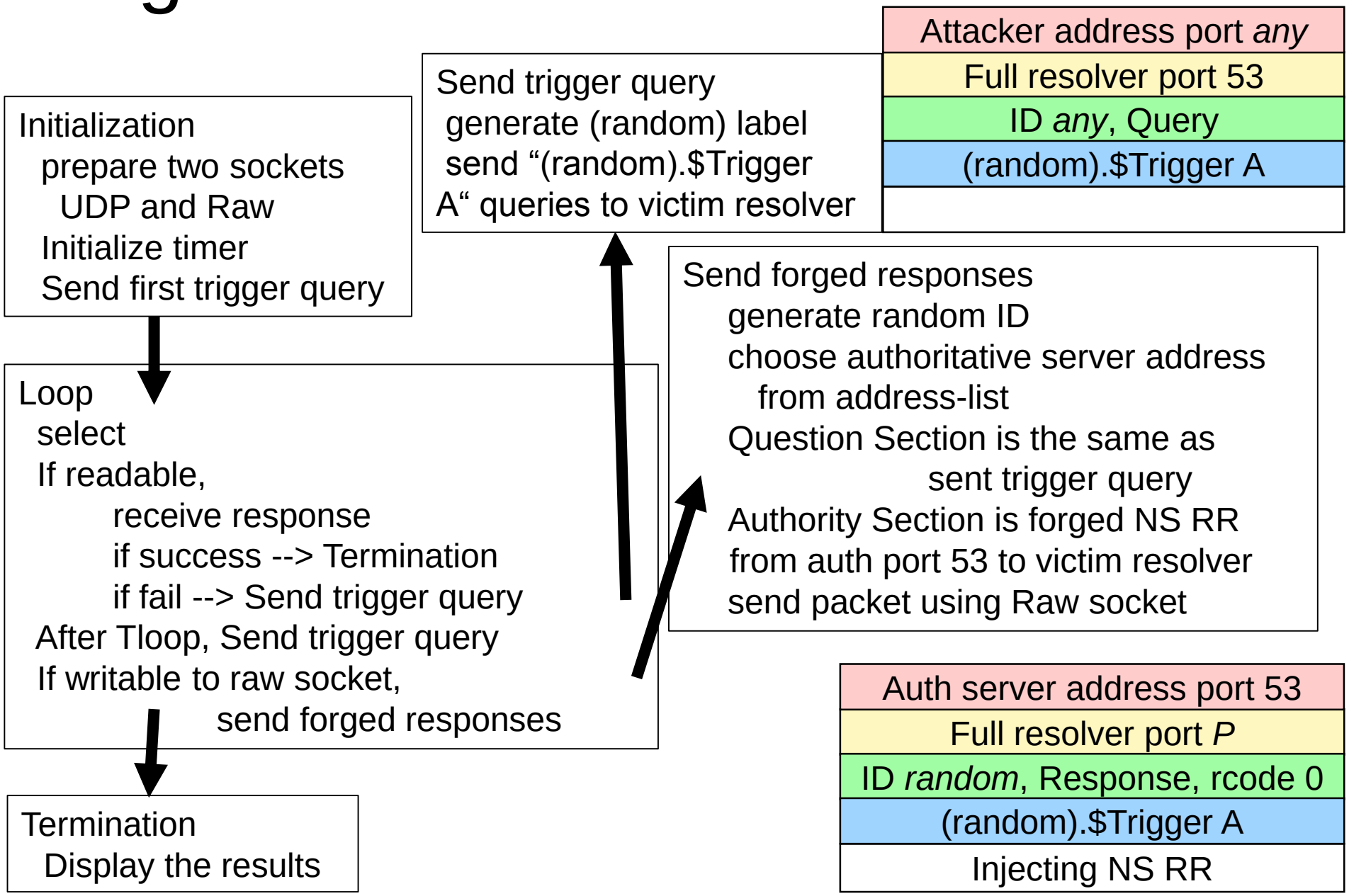
$$T = \frac{Nqid * Nport * Nns}{Rans}$$

Example:
2.62 seconds
Or
167,772 (2 days)

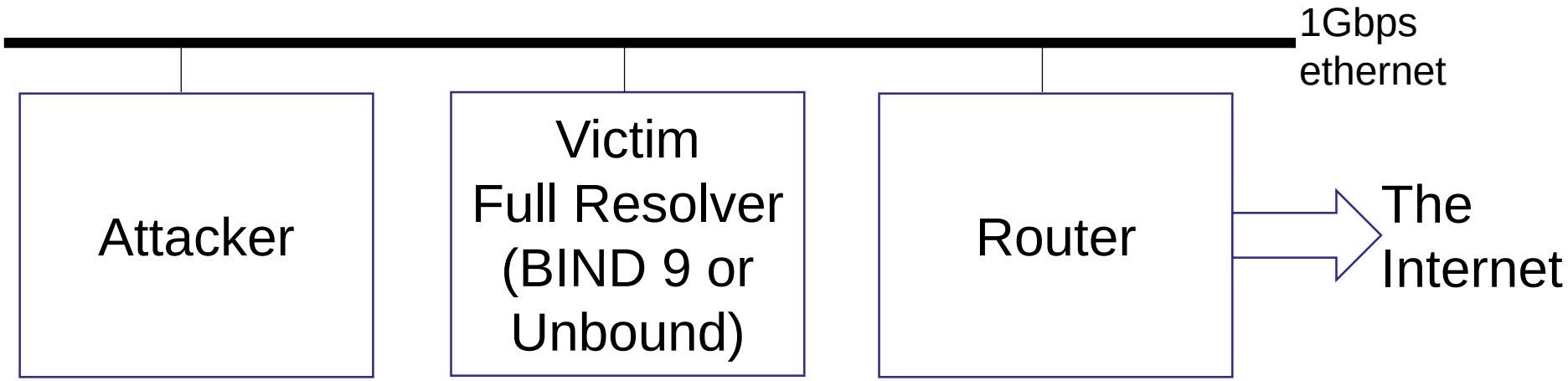
My attack tool

- Written in C, 500 lines, standard library only
  - Using select() to control timing and sockets
  - Send forged responses using raw socket
- Tool's parameters
  - Victim resolver's IP address and port number
  - Trigger domain name
  - NS RR information
  - Authoritative servers' IP address list
- Usage example
  - ./a.out 192.2.0.2 20001 www.google.com www.google.com ns.dnslab.jp 216.239.32.10/216.239.34.10/216.239.36.10/216.239.38.10

Diagram of the attack tool



Environment of attack experiments



- Two VMs (Attacker and Victim)
  - 2.5GHz Xeon, 2 cpu, 3GB memory
  - On same link (1Gbps)
  - Victim full resolver sends queries to the Internet
- Victim full resolver
  - BIND 9.9.5 (without source port randomization)
  - (Unbound 1.4.21 without source port randomization)

Two attack experiments

- www.google.com
  - % sudo ./a.out 203.178.129.2 20001 www.google.com www.google.com ns.dnslab.jp 216.239.32.10/216.239.34.10/216.239.36.10/216.239.38.10
  - The injection took 46 seconds,
  - 28 qps queries and 66,726 pps forged responses
- net
  - “rndc flush” on victim full resolver
  - sudo ./a.out 203.178.129.2 20001 root-servers.net net ns.dnslab.jp 198.41.0.4/192.228.79.201/…(root server addresses)
  - The injection took 8 seconds,
  - 30 qps queries and 69,158 pps forged responses

Expected attack time and results

	Var.	Unit	Range	www.google.com attack		net attack	
Port randomization				no	yes	no	yes
Num of resolver ports	Nport		1~ 2^16	1	64,000	1	64,000
Num of IDs	Nqid		2^16	65536	←	65536	←
Num of auth servers	Nns		1~13	4	←	13	←
RTT to auth servers	Tauth	sec	0.001 ~0.2	0.036~ 0.094	←	0.006~ 0.272	←
Loop interval	Tloop	sec		0.036	←	0.033	←
Response send rate	Rans	pps		66726	←	69158	←
First time trial Success possibility				0.00916	0.00000 014	0.00267	0.00000 004
expected attack time		sec		3.9	251,434	12.3	788,425
Attack time (result)		sec		46	×	8	×

If you detect poisoning

1. Flush poisoned RRSet
2. Resolve the poisoned RRSet
3. Compare the poisoned RRSet with other full resolvers
   - Your other full resolvers
   - Or, public DNS services (for example:Google)

   - Because cache flush may cause another poisoning

Measures

- Source port randomization
- Monitoring
  - Attacks may increase victim full resolver's load
  - Traffic
    - Number of packets of input and output are usually balanced on DNS
    - Victim full resolver receives many unmatched responses
- Many countermeasures are proposed/implemented
  - harden-referral-path on Unbound
  - nonce prefix on Google Public DNS
  - Use of TCP transport
  - DNS cookies
- See also
  - draft-fujiwara-dnsop-poisoning-measures-00

Detection by DNSSEC validation

- Cache poisoning is possible on BIND 9 and Unbound validators
- However, DNSSEC validation works well and they return validation error as “Server Failure”
- NSEC3 Opt-Out issue
  - Complex structured TLDs may have empty non-terminals
  - Empty non-terminals with NSEC3 are able to be validated because NSEC3 RRs say no NS
  - Some empty non-terminals do not have NSEC3 RRs
    - When there is no secure delegation for the name
    - One example: saitama.jp
    - They were not able to be validated
  - So, JPRS added TXT RRs to all empty non-terminals in JP zone
    - BIND 9’s dnssec-signzone generates NSEC3 RRs for non-empty non-terminals

Conclusion

- "Redirect the target domain's nameserver“ cache poisoning attack is easy
  - Most of domain names are conquerable
  - However, it is easy to trace attackers because it needs authoritative DNS servers
- Measures
  - Source port randomization works well
  - DNSSEC validation is effective to detect
  - Detection and correspondence of attacks are important

details

Success probability of first time trial

- Success if ID, port, address are matched
- Probability of first time trial P is probability within $Tauth$

$$P = \frac{Tauth * Rans}{Nqid * Nport * Nns}$$

- However, $Tauth$ is not controllable
- Use $Tloop$ instead of $Tauth$ under $Tloop < Tauth$

$$P = \frac{Tloop * Rans}{Nqid * Nport * Nns}$$

Success probability under continuation attack

- Success probability of not succeeding by $n-1st$ time, but succeeding n-th time

$$Pn = (1-P)^{n-1} * P \qquad (1)$$

- Success probability of succeeding by n-th time Qn is the sum of Pn

$$Qn = \sum_{i=1}^{n} (1-P)^{i-1} P \qquad (2)$$

$$Qn = 1 - (1-P)^{n} \qquad (3)$$

- n is brought close to infinity

$$Qn \to 1$$

Expected attack number of tries

- Success probability of not succeeding by $n-1st$ time, but succeeding n-th time

$$Pn = (1-P)^{n-1} * P \qquad (1)$$

- Expected attack number of tries by n-th time En is a sum of the multiply of value i and Pn

$$En = \sum_{i=1}^{n} i(1-P)^{i-1}P \qquad (2)$$

- Then

$$En = \frac{1}{P} - \frac{(1+nP)(1-P)^n}{P} \qquad (3)$$

- n is brought close to infinity

$$En \to \frac{1}{P}$$

Expected attack time

$$T = Tloop * E$$

$$= \frac{Tloop}{P}$$

$$= \frac{Nqid * Nport * Nns}{Rans}$$

Termination of attack

- Prepare “*.$Trigger IN A 192.2.0.1” on forged authoritative DNS server

- If the NS injection succeed, a response of a trigger query returns rcode 0, “(random).$D IN A” RR