
DNS Cache-Poisoning: New Vulnerabilities and Implications, or: **DNSSEC, the time has come!**

Amir Herzberg and Haya Shulman
Dept. of Computer Science
Bar Ilan University

8/1/2013

About me: Amir Herzberg

- Associate professor, Computer Science @ Bar Ilan
- Previously: many years in industry (e.g. 10 @ IBM)
- Main areas:
 - Network security, esp. Internet protocols
 - Denial of Service: attacks and defenses
 - Applied cryptography
 - Secure e-commerce and payments
 - Secure usability, esp. phishing
 - Anonymity



About us

Bar Ilan University
NetSec group



Haya Shulman:

Fresh Graduate
PhD Thesis:
DNS Security
(and more...)

Amir Herzberg:

NetSec/Crypto
Researcher
Attacks: DNS,
TCP/IP, DoS, ...

2013... DNSSEC, IPSEC:15yrs old

Yet: < 6% of traffic encrypted,...

➔ Insecure against MitM attacker

WHY???

False hope: attackers are `off-path`

Can send spoofed packets but not intercept

Reality: MitM attackers are common

Open WiFi, **route hijacking**, mal-devices, **DNS poisoning**

False belief: DNS, TCP immune to off-path attacks

Reality: **TCP hijacking**, **DNS poisoning**

Outline

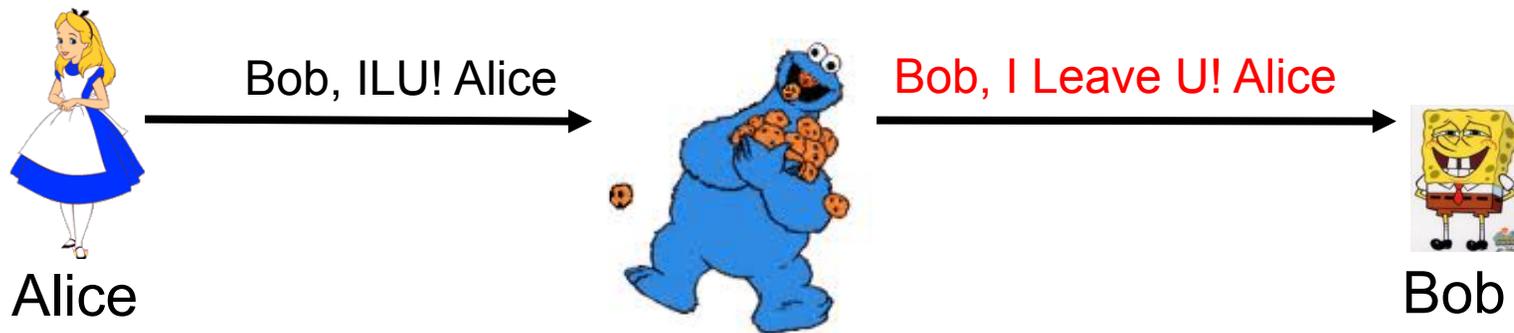
- **Attack model: MitM vs. Off-path**
- DNS poisoning: Background
- **Source-port de-randomization** attacks
 - Resolver-behind-NAT, proxy-using-upstream
- **1st-fragment piggybacking** attacks
- Implications and defenses
 - Patches: to resolvers, name-servers, registrars
 - Deploy DNSSEC – correctly... [and fix it, too??]

Everyone is worried about Security...

- So, why isn't crypto used more?
 - SSL/TLS/IPsec <6% of traffic, DNSSEC <1%, BGPsec ~ 0%, ...
- Why? Illusion of security due to two false myths:
 - Most attackers are only off-path, not MitM
 - Simple, client-only challenge-response defenses suffice against off-path attackers

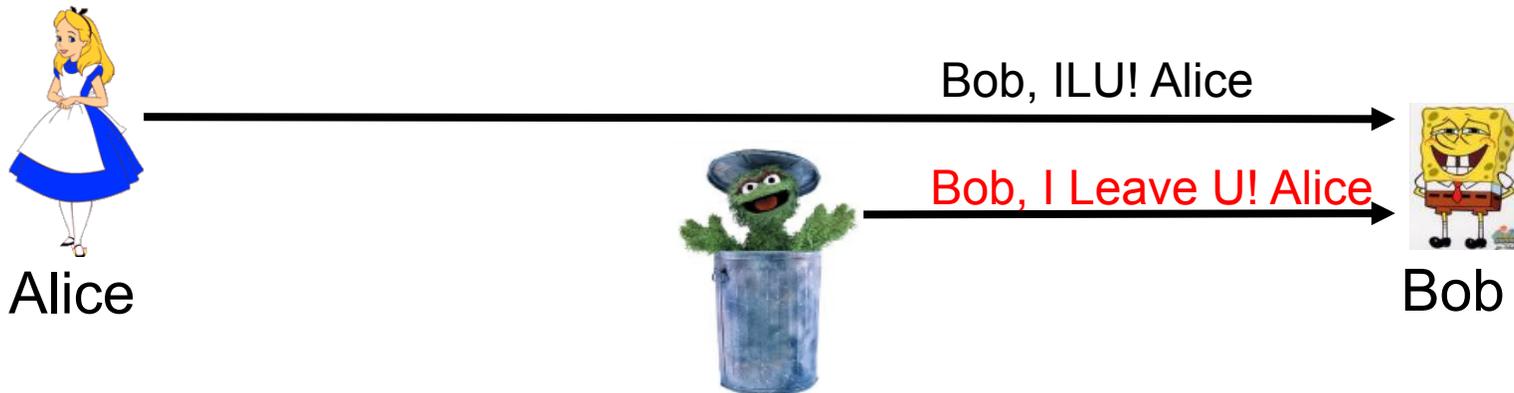
Attacker Model: MitM or Off-Path?

- **Man-in-the-Middle** attacker
 - On path
 - Harder but possible: wifi, route hijack, vulnerable router, ...
 - Or: give wrong address – **DNS poisoning**
 - Prevent with **crypto**: overhead, complexity, PKI ...
 - Why bother?



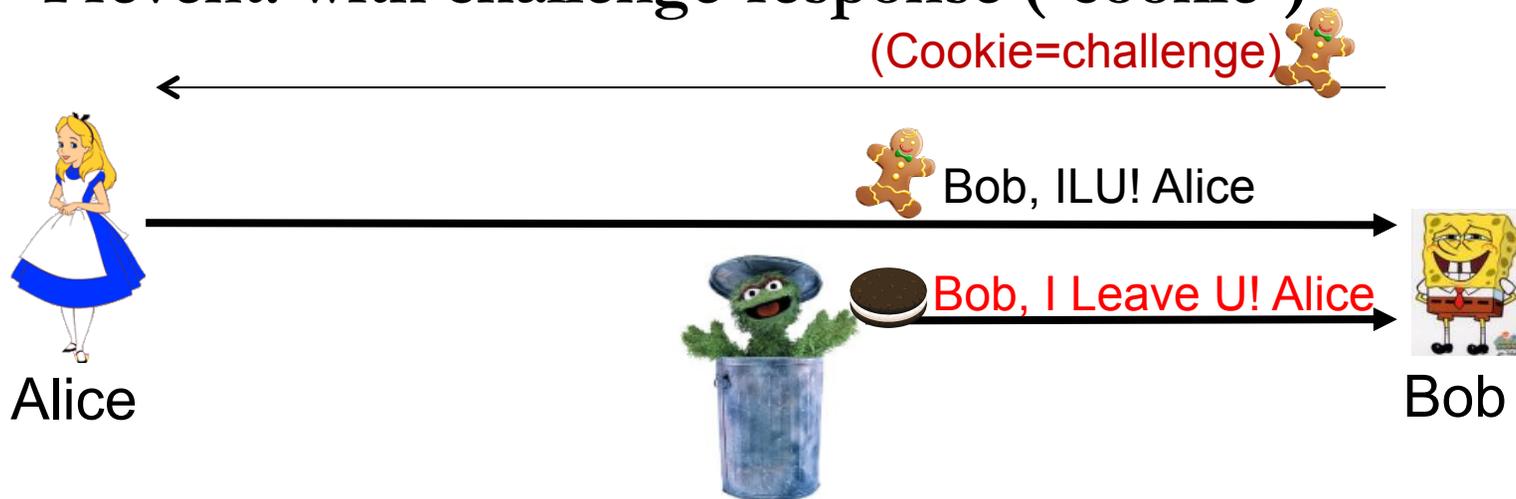
Attacker Model: MitM or Off-Path?

- Folklore: most attackers are weak, off-path
- `Security` is often against **Off-Path Oscar**
 - Do not control devices en-route
 - Cannot intercept/modify/block traffic
 - **Prevent: with challenge-response (`cookie`)**



Attacker Model: MitM or Off-Path?

- Folklore: most attackers are weak, off-path
- `Security` is often against **Off-Path Oscar**
 - Do not control devices en-route
 - Cannot intercept/modify/block traffic
 - **Prevent: with challenge-response (`cookie`)**



Challenge-Response: What Can Go Wrong?

- Attacker **has** MitM capabilities
- **Insufficient entropy**: too short or non-uniform
 - TCP [Zalewski01, Watson04]
 - DNS [Klein03, Kaminsky08]
- Side-channel: reused field (source port)
 - DNS [HS12, HS13], TCP [GH12, GH13, QM(X)12]
- Cut-&-paste: use real cookie in spoofed packet
 - DNS [HS13]

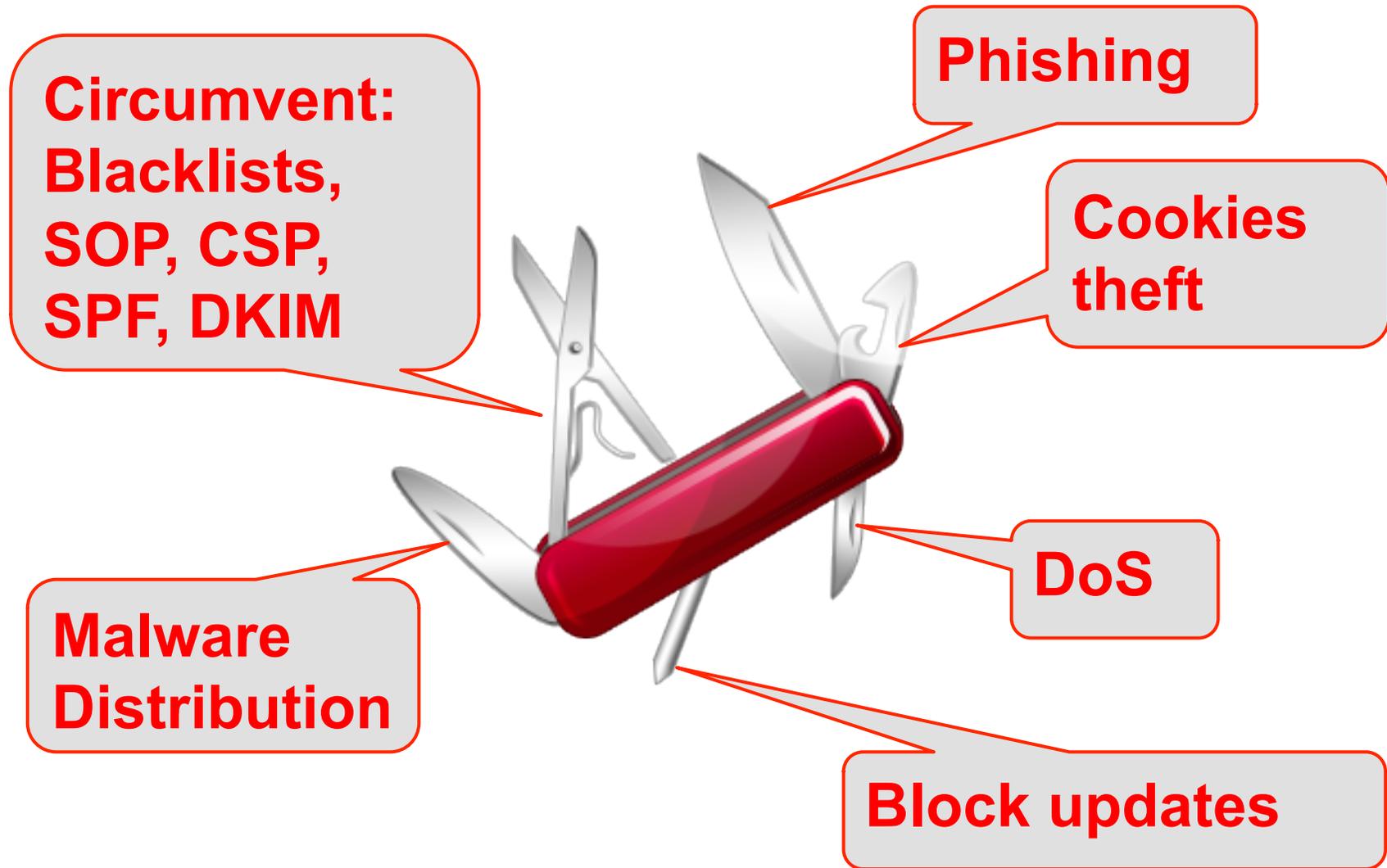
Everyone is worried about Security...

- So, why isn't crypto used more?
 - DNSSEC/IPsec <6% of traffic, DNSSEC <1%, BGPsec ~ 0%, ...
- Why? Illusion of security due to two false myths:
 - Most attackers are only off-path, not MitM
 - Simple, client-only challenge-response defenses suffice against off-path attackers
- Reality:
 - MitM capabilities: via WiFi, BGP hijacking, ...
 - **Off-path attacks against TCP & DNS**
[Today: simplified]

Outline

- Attack model: MitM vs. Off-path
- **DNS poisoning: Background**
- **Source-port de-randomization** attacks
 - Resolver-behind-NAT, proxy-using-upstream
- **1st-fragment piggybacking** attacks
- Implications and defenses
 - Patches: to resolvers, name-servers, registrars
 - Deploy DNSSEC – correctly... [and fix it, too??]

DNS Poisoning: the Hacker's Knife



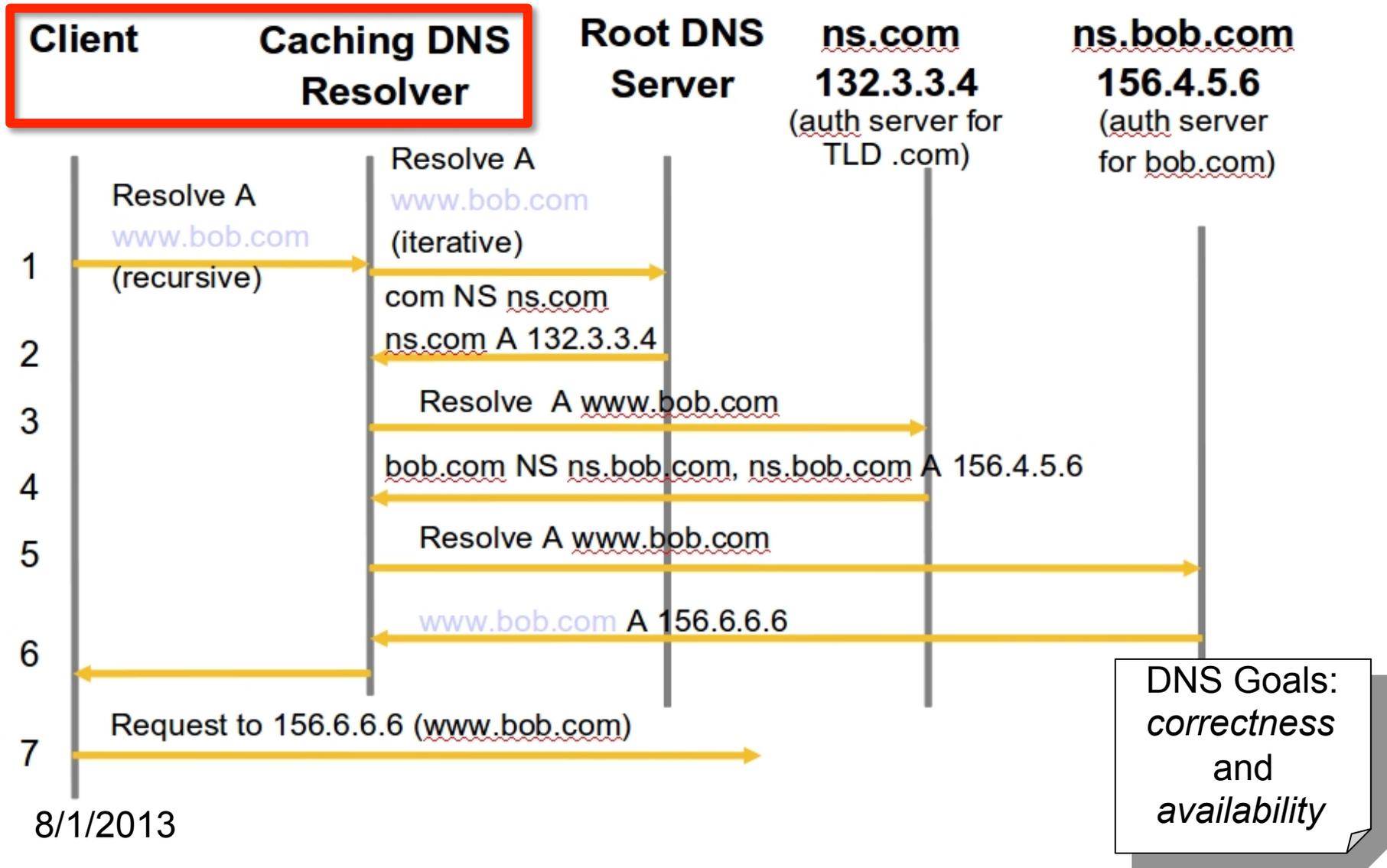
Exploiting Poisoning (and Injecting)

- Circumvent Name/Address server identification
 - Browser's Same Origin Policy (SOP) defenses
 - XSS (Cross-Site Scripting)
 - Steal `HTTP cookies/credentials`
 - Phishing, defacement, malware distribution
 - Fake policies: CSP, SPF, DKIM, black-lists
- Long-lived, multi-user attacks: exploit caching of...
 - DNS mappings (resolver/client cache)
 - HTTP objects (in browser/proxy; scripts, HTML, ...)

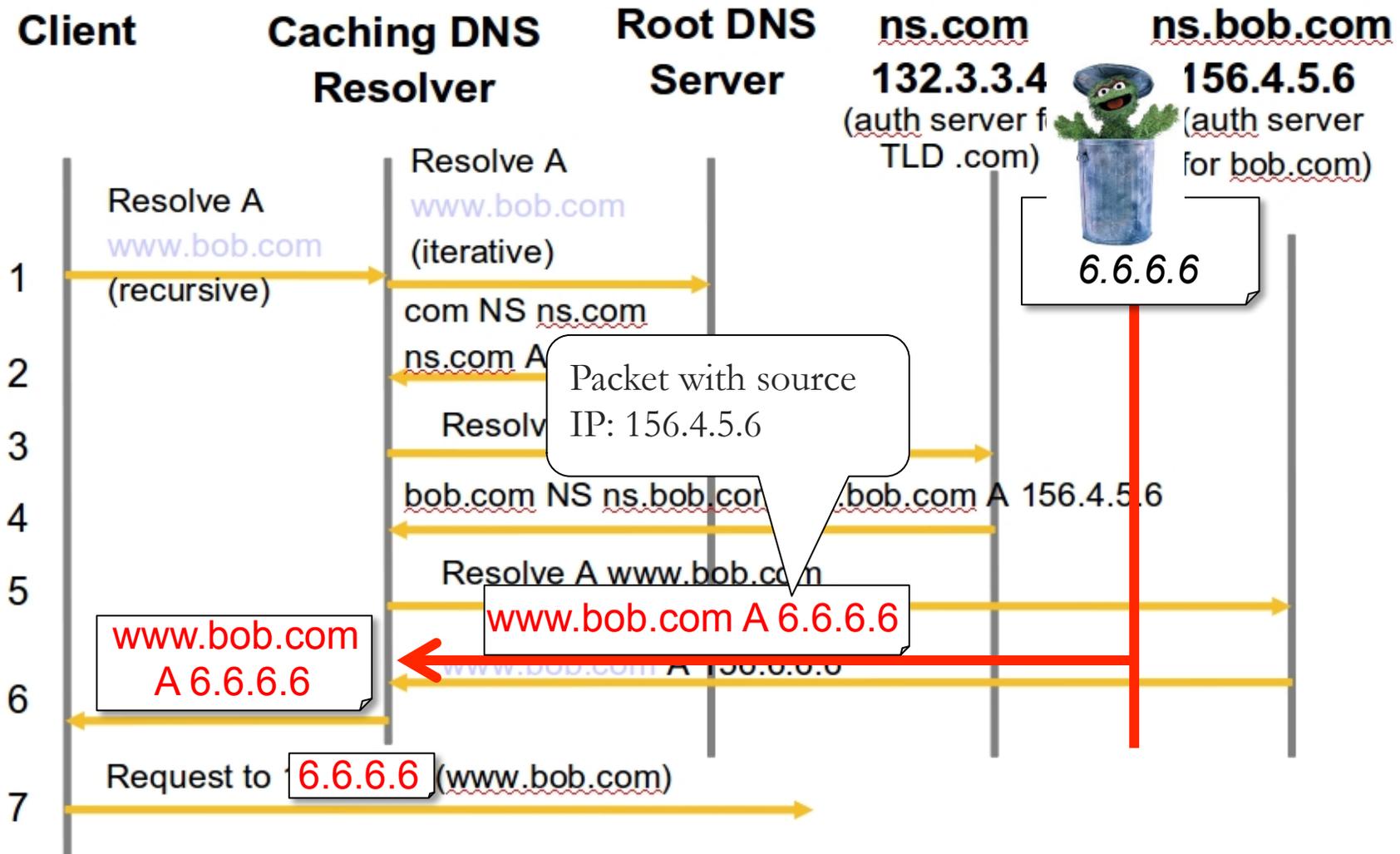
DNS Poisoning

- DNS: Internet directory (domain names → IP,...)
- Maps: Domain-Names to IP addresses, policies, ...
- Caching critical for efficiency
 - At clients and at DNS Resolvers (aka proxies, local DNS)
- Poisoning : cache with **fake** mapping:
`www.google.com A 6.6.6.6`
- Simple request-response (over UDP), efficient, caching
- Myth: `can't poison' – TTL, 16-bit TXID, source port

Domain Name System (DNS)



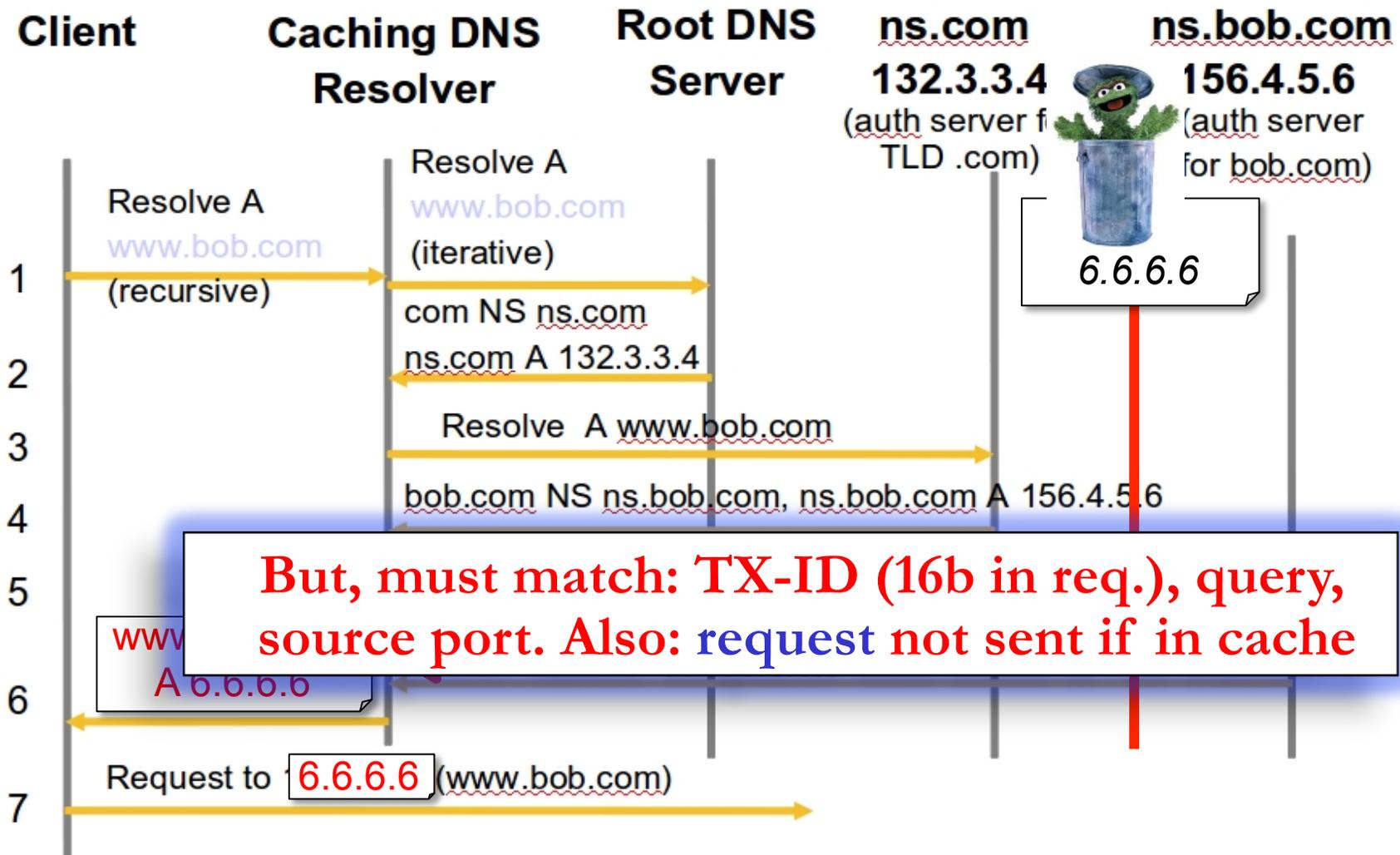
DNS Cache Poisoning



8/1/2013

Herzberg and Shulman: DNSSEC, the time has come!

DNS Cache Poisoning



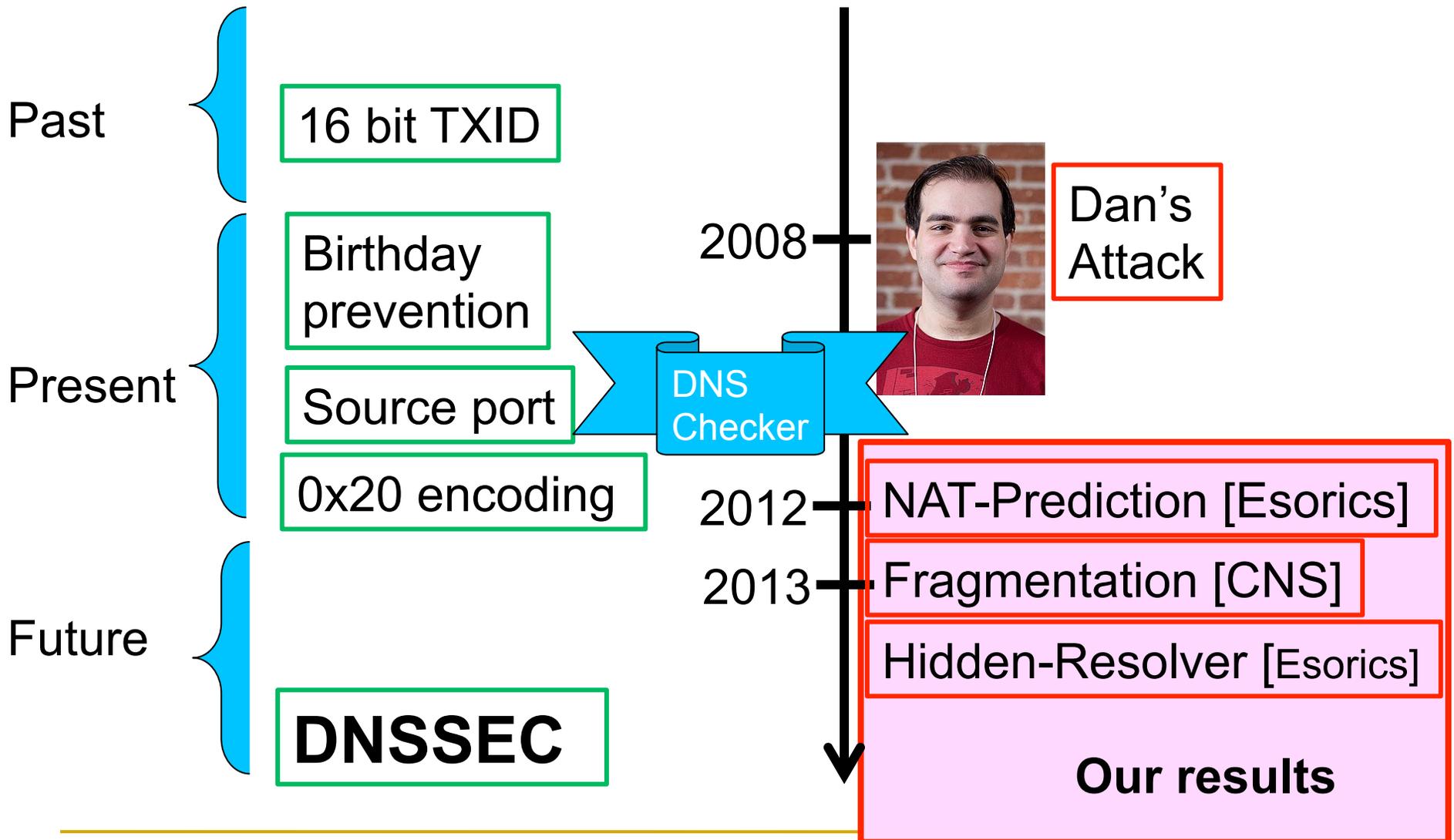
8/1/2013

Herzberg and Shulman: DNSSEC, the time has come!

Defenses against DNS Poisoning

- **Currently**, mostly Challenge-response defenses:
 - Unilateral (in resolver): `challenges' using existing request fields echoed in responses
 - TX-ID (16b), Source port (16b), Query [0x20]
- Cryptographic defenses (**DNSSEC**): limited use
 - Root and many TLDs signed
 - Many resolvers request signatures, but few **validate**
 - **Why?** Myths (rare MitM, weak Oscar)

DNS Poisoning Timeline



8/1/2013

Herzberg and Shulman: DNSSEC, the time has come!

Outline

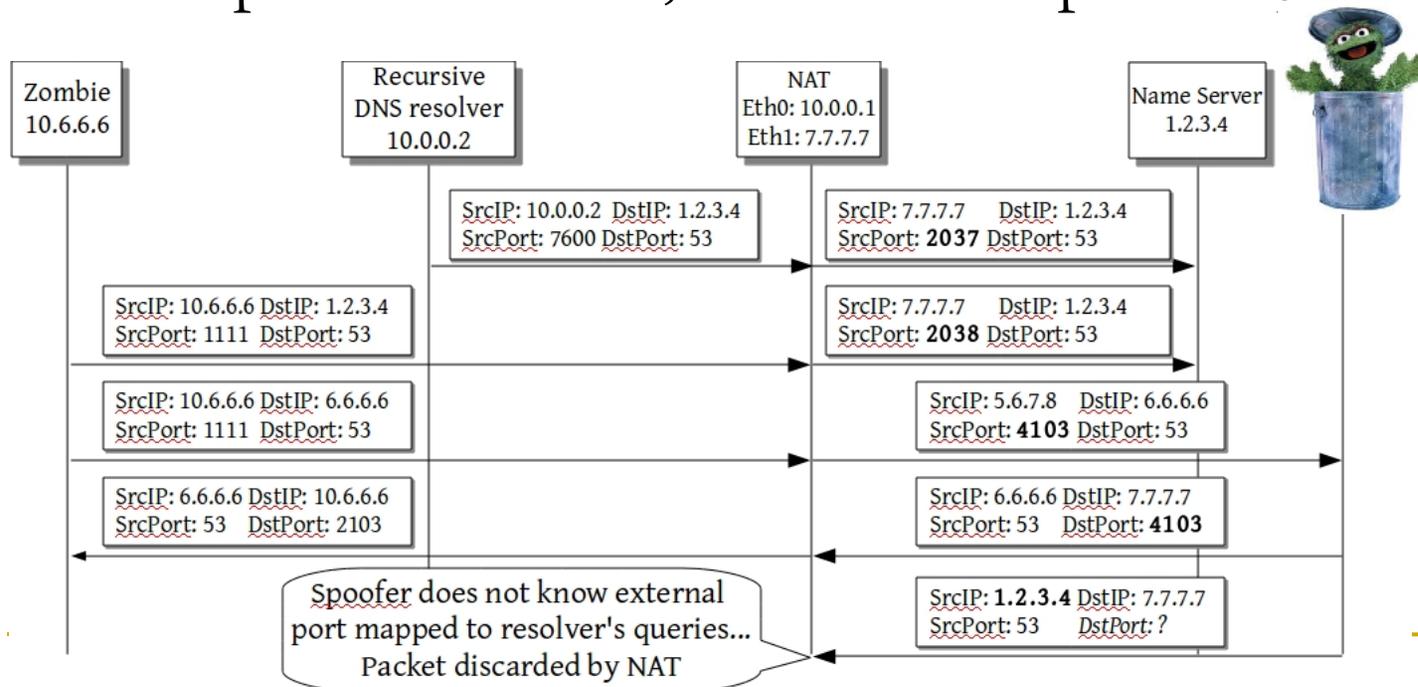
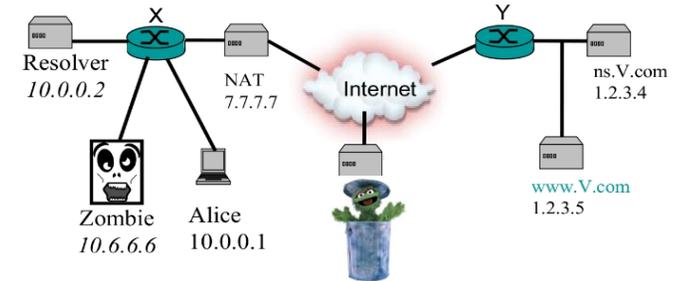
- Attack model: MitM vs. Off-path
- DNS poisoning: Background
- **Source-port de-randomization** attacks
 - Resolver-behind-NAT, proxy-using-upstream
- **1st-fragment piggybacking** attacks
- Implications and defenses
 - Patches: to resolvers, name-servers, registrars
 - Deploy DNSSEC – correctly... [and fix it, too??]

Source Port De-Randomisation Attacks

- **Learn source-port via side channel**
- Attacks on two common configurations:
 - Resolver-behind-NAT [Esorics'12]
 - Attacks for most types of NATs (only one was secure)
 - Upstream resolver (e.g., OpenDNS) [Esorics'13]
 - Learn resolver's IP address, too [often enough for DoS !]

Resolver-behind-NAT

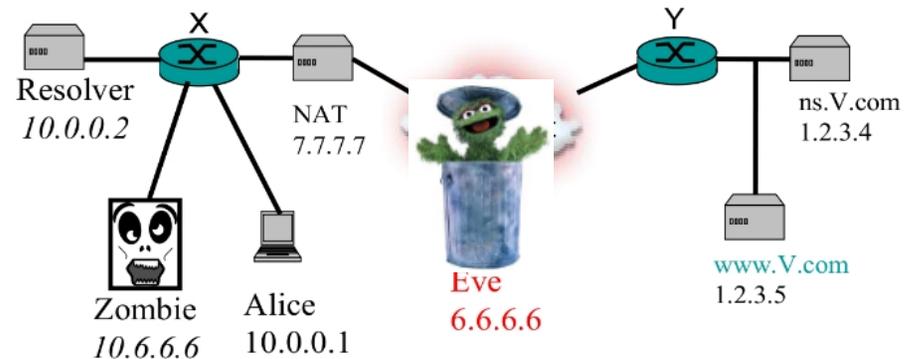
- Port re-allocated by NAT
- Few methods; most vulnerable
- E.g., **per-dest incrementing** (Linux)
- Initial port is random; can attacker predict port?



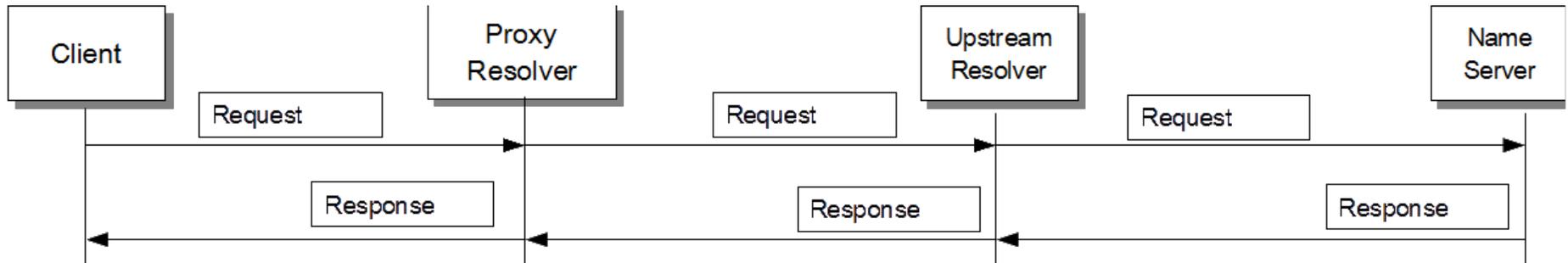
Herzberg and Shulman: DNSSEC, the time has come!

Resolver-behind-NAT: Attack

- Example: attack on **per-dest incrementing** (e.g., Linux)
- Initial port is random; can attacker predict/trap port?
- Attack phases:
 - Hole-punch the NAT
 - Exploit assigned mapping to guess port
- Variations apply to different NAT devices



Upstream DNS Resolver



- Upstream DNS resolvers:
- Popular: Google's public-DNS, OpenDNS, many others
- Recommended by experts, vendors
 - E.g., Akamai: 'Customer's primary DNS are not directly exposed to end users, so the risk of cache poisoning and DoS attacks is mitigated'...
- Proxy resolvers often has lower bandwidth, weaker security
 - We found (CAIDA): 54% incrementing ports, 30% fixed port
 - And... both types are vulnerable!

Herzberg and Shulman: DNSSEC, the time has come!

Upstream DNS Resolver - Attack

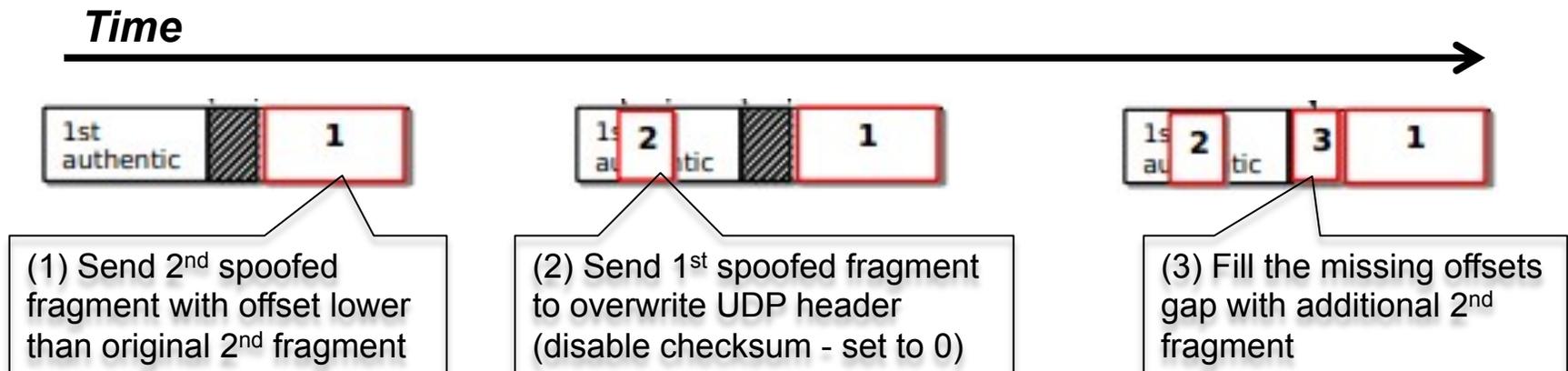


- Poisoning attack in three phases
- Phase 1: find proxy's IP address
 - Many requests with fragmented response... `kill` with spoofed frag
 - Suffices for DoS attack on proxy!
- Phase 2: find fixed/current port #
 - By a more complex frag attack, or by `port overloading`
- Phase 3: `regular` (`Kaminsky`) poisoning

Herzberg and Shulman: DNSSEC, the time has come!

Defragmentation-Cache Poisoning

- Response is sent in two fragments:
- Sample each port via 3 fragments:



- Query retransmission when incorrect port
- Referral request: port found

DNS	TXID	✓
	0x20	✓
UDP	Port X'	?
	chksum:0	✓
IP	IP-ID: <i>i</i>	✓
	Addresses	✓

Herzberg and Shulman: DNSSEC, the time has come!

Outline

- Attack model: MitM vs. Off-path
- DNS poisoning: Background
- **Source-port de-randomization** attacks
 - Resolver-behind-NAT, proxy-using-upstream
- **1st-fragment piggybacking** attacks
- Implications and defenses
 - Patches: to resolvers, name-servers, registrars
 - Deploy DNSSEC – correctly... [and fix it, too??]

1st-fragment piggybacking attacks

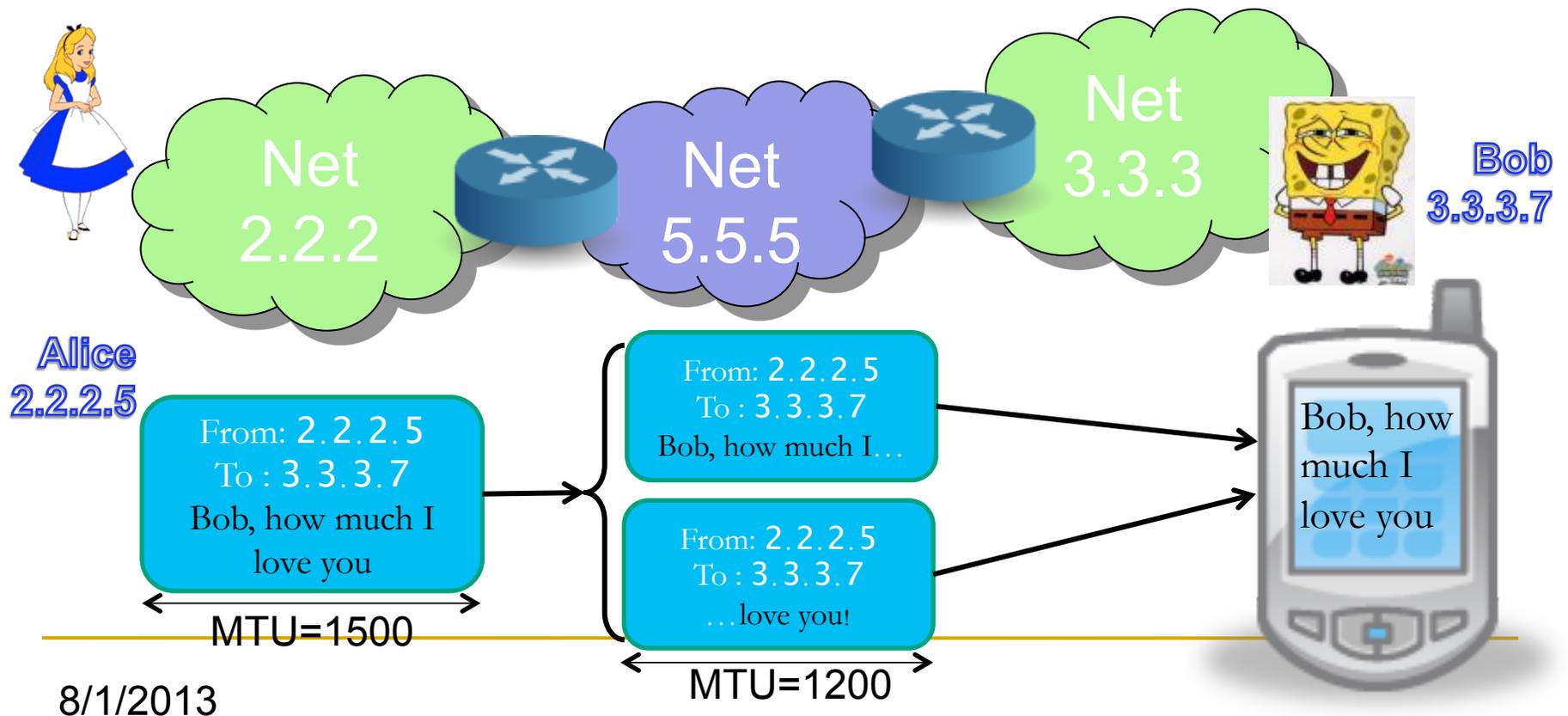
- Cut'n'Paste attack:
- Poison a long, **fragmented** DNS response
 - Source fragmentation will do [works even for IPv6]
- **All `challenges` are in the first fragment!**
 - **TXID, “src” port, even query [e.g., 0x20 defense]**
- Replace 2nd fragment with a fake one!
- Few details and quick recap on IP fragmentation

IP Fragmentation

Nets have a limit on maximal packet size

If the packet is larger than the limit: fragmentation

Reassemble at the receiver



Fragment Reassembly

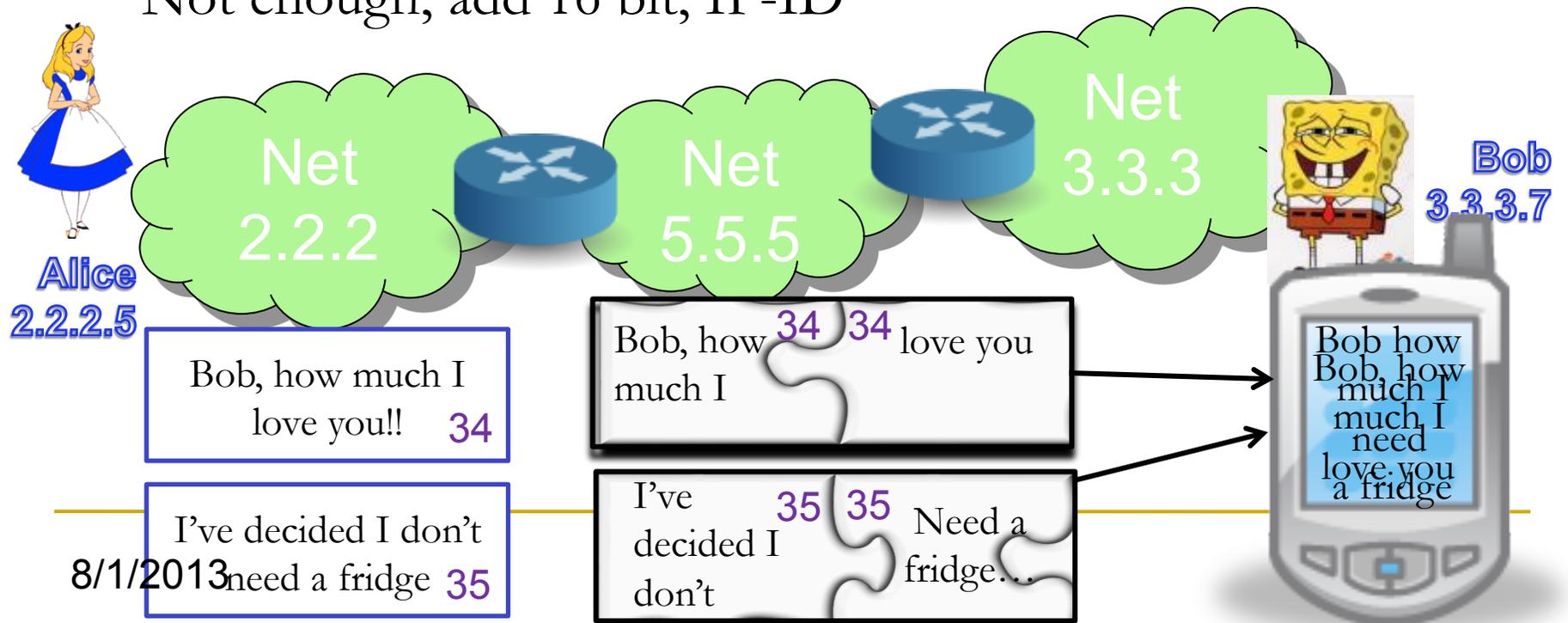
Bob receives fragments of a packet

How to reassemble without introducing mistakes

Identify fragments of the same packet

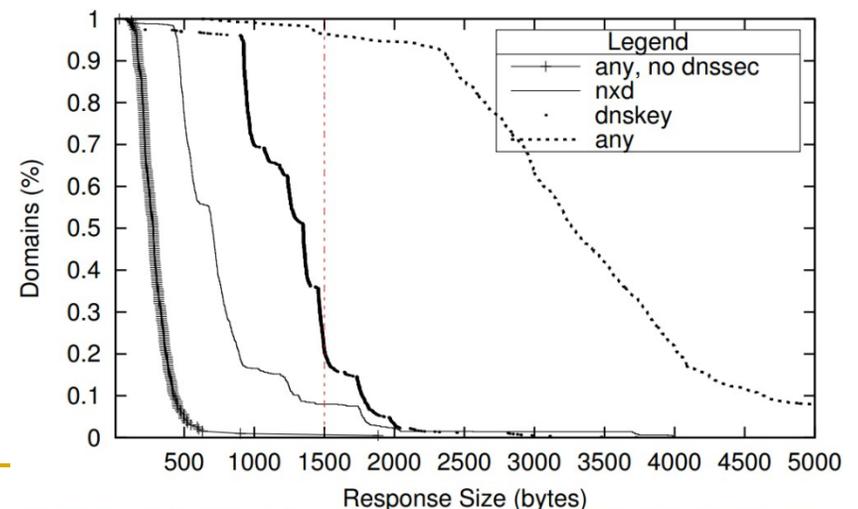
By sender/receiver addresses and protocol (TCP/UDP)

Not enough, add 16 bit, IP-ID

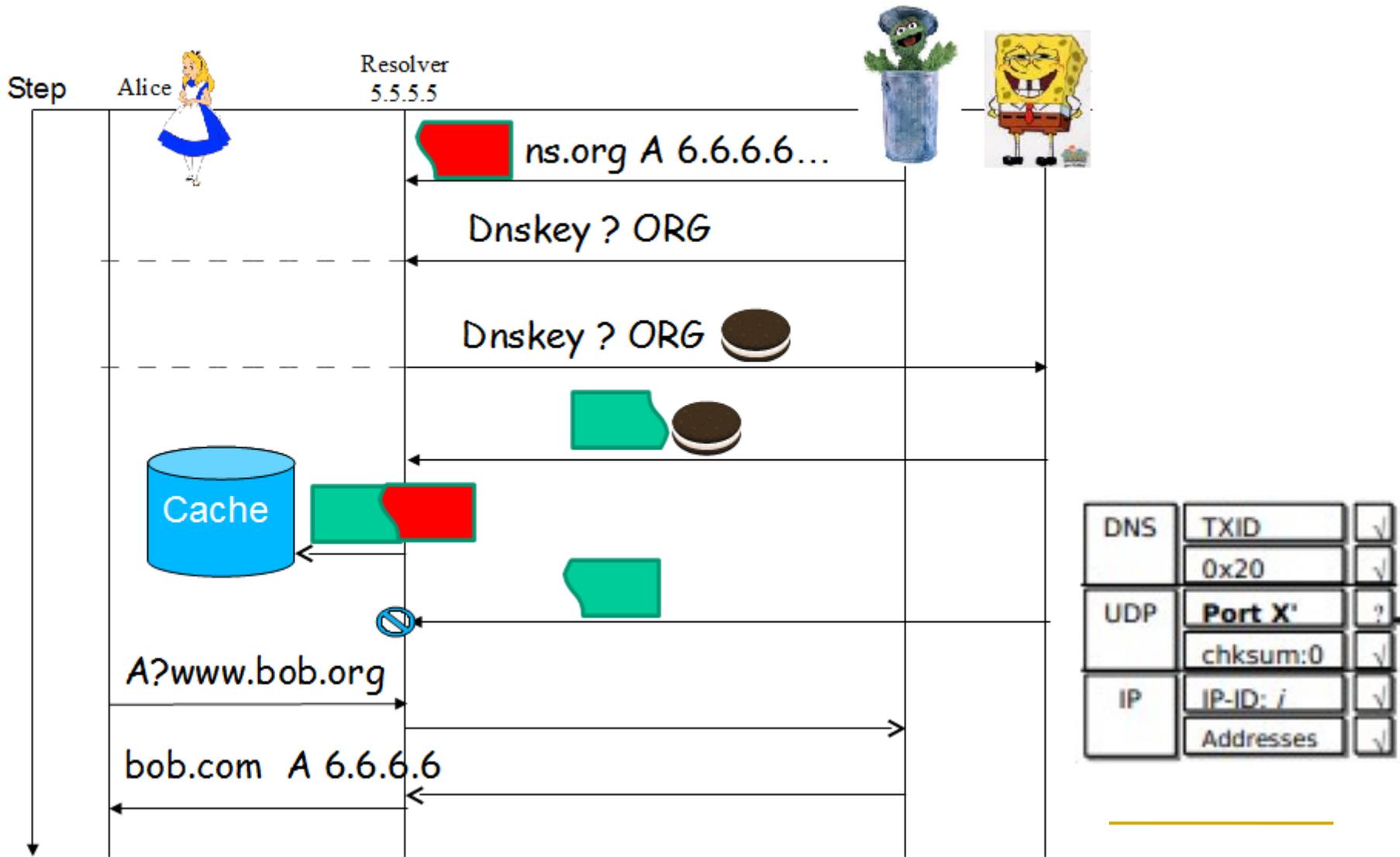


Off-Path Discarding and Modifying

- We show off-path can **discard** and **modify** fragments!!
 - Exploit fragmentation for poisoning!
 - In reality fragmentation is rare (<1%)
 - But, off-path attacker can **cause** fragmentation!!
 - Two methods:
 1. Trigger requests whose responses fragment
 - E.g., DNSSEC protected
 2. Attacker registered domain

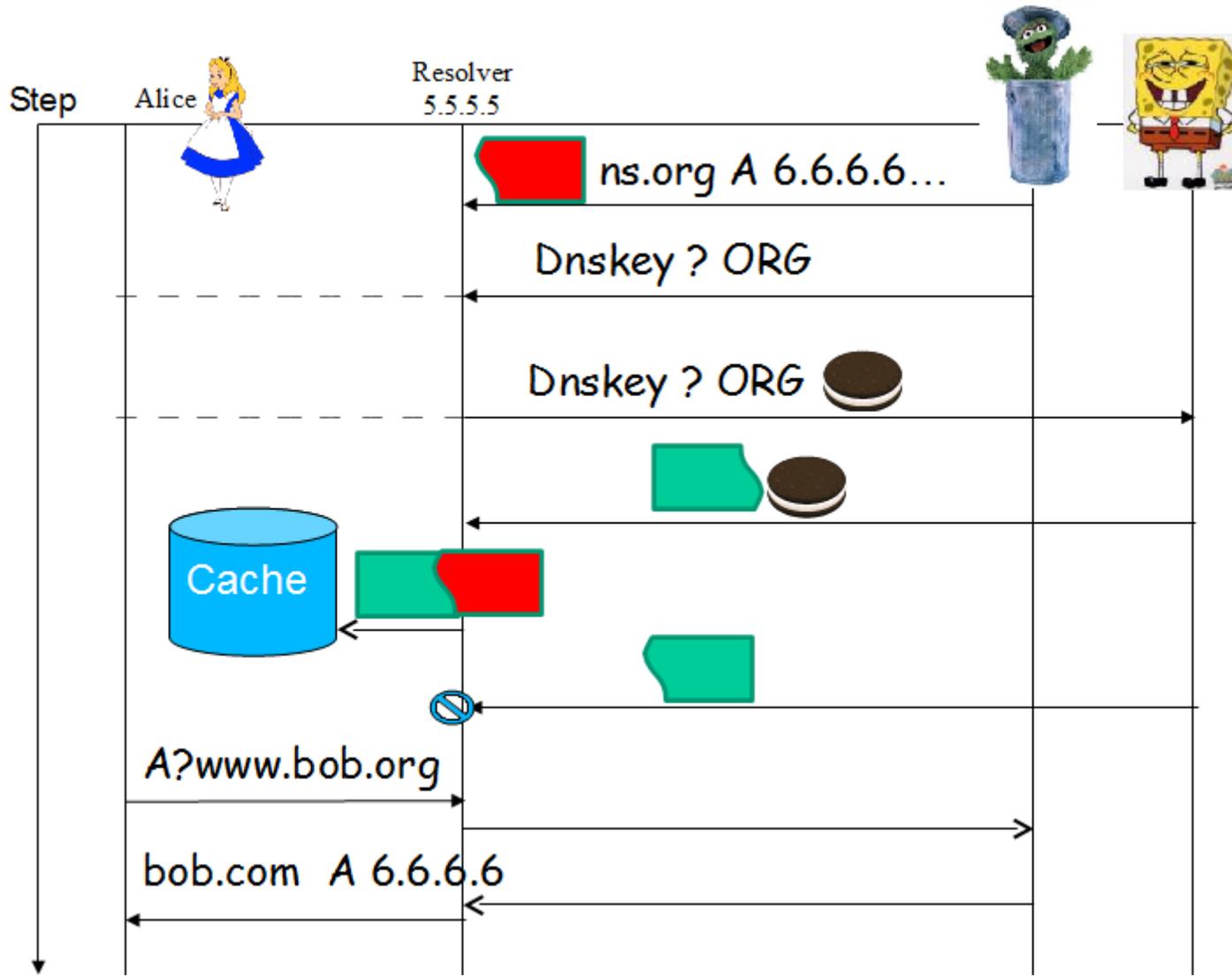


Modify Long DNSSEC Responses



8/1/2013

Modify Long DNSSEC Responses



8/1/2013

Poisoning DNSKEY Response

poisoning-org-not-cached - Wireshark

File Edit View Go Capture Analyze Statistics Telephony Tools Help

Filter: `(udp.port == 53 or (ip.frag_offset==0) or (ip.frag_offset==1480))` Clear Apply

No.	Time	Source	Destination	Protocol	Info
122	15:27:55.806647	199.249.112.1	132.70.6.202	IP	Fragmented IP protocol (proto=UDP 0x11, off=1480, ID=7c6e) [Reassembled in #134]
133	15:27:56.194478	132.70.6.202	199.249.112.1	DNS	Standard query response DNSKEY ORG
134	15:27:56.255698	199.249.112.1	132.70.6.202	DNS	Standard query response DNSKEY DNSKEY DNSKEY DNSKEY RRSIG RRSIG
135	15:27:56.255728	199.249.112.1	132.70.6.202	IP	Fragmented IP protocol (proto=UDP 0x11, off=1480, ID=7c6e)

Annotations:

- DNS request DNSKEY?ORG.** (Callout to packet 133)
- Spoofed second fragment** (Callout to packet 134)
- First fragment is reassembled with the (spoofed) second fragment** (Callout to packet 135)
- Authentic second fragment (cannot be reassembled and is discarded after 30 seconds)** (Callout to packet 134)
- Forged A RRs of DNS servers of ORG. Authentic RRs were:**
b0.org.afiliast-nst.org: type A, class IN, addr 199.19.54.1
d0.org.afiliast-nst.org: type A, class IN, addr 199.19.57.1

Domain Name System (response)

[Request: To: 133]

[Time: 0.061220000 seconds]

Transaction ID: 6x55f6

Flags: 0x8400 (Standard query response, No error)

Questions: 1

Answer RRs: 6

Authority RRs: 7

Additional RRs: 5

- Queries
 - ORG: type DNSKEY, class IN
- Answers
 - ORG: type DNSKEY, class IN
 - ORG: type RRSIG, class IN
 - ORG: type RRSIG, class IN
- Authoritative nameservers
 - ORG: type NS, class IN, ns a0.org.afiliast-nst.info
 - ORG: type NS, class IN, ns a2.org.afiliast-nst.info
 - ORG: type NS, class IN, ns b0.org.afiliast-nst.ORG
 - ORG: type NS, class IN, ns b2.org.afiliast-nst.ORG
 - ORG: type NS, class IN, ns c0.org.afiliast-nst.info
 - ORG: type NS, class IN, ns d0.org.afiliast-nst.ORG
 - ORG: type RRSIG, class IN
- Additional records
 - b0.org.afiliast-nst.ORG: type A, class IN, addr 132.70.6.201**
 - d0.org.afiliast-nst.ORG: type A, class IN, addr 199.19.112.155**
 - b0.org.afiliast-nst.ORG: type AAAA, class IN, addr 2001:500:c::1
 - d0.org.afiliast-nst.ORG: type AAAA, class IN, addr 2001:500:f::1
 - <Root>: type OPT

Frame (1514 bytes) Reassembled IPv4 (1731 bytes)

Text item (), 16 bytes Packets: 202 Displayed: 4 Marked: 0 Profile: Default

Causing Long, Fragmented Responses

- Often, attacker doesn't need to find a long response
- Attacker **causes** a long, fragmented response
 - From a victim NS of a TLD (.ORG, .CO.UK, ...)
 - By **registering** an 'appropriate' subdomain
- To cause fragmentation:
 - Register many name servers
 - With long names
- Example? One-Domain-to-Rule-them-All .ORG
 - Or see paper [CNS2013]... or next foil 😊

88423	199.249.120.1	IPv4	480	Fragmented IP protocol (proto=UDP 0x11, off=1480, ID=b063) [Reassembled in #207715]
207714	132.70.6.119	DNS	102	Standard query NS one-domain-to-rule-them-all.org
207715	199.249.120.1	DNS	1514	Standard query response
207716	199.249.120.1	IPv4	480	Fragmented IP protocol (proto=UDP 0x11, ff=1480, ID=b063) [Reassembled in #207715]
<ul style="list-style-type: none"> ▶ one-domain-to-rule-them-all.org: type NS, class IN, ns i23456789101112131415161718192021222324252627282930313233343536.123456789.one-domain-to-rule-them-all.org ▶ one-domain-to-rule-them-all.org: type NS, class IN, ns j23456789101112131415161718192021222324252627282930313233343536.j23456789.one-domain-to-rule-them-all.org ▶ one-domain-to-rule-them-all.org: type NS, class IN, ns sns-pb.isc.org ▶ one-domain-to-rule-them-all.org: type NS, class IN, ns pdns3.ultradns.org ▶ h9p7u7tr2u91d0v0ljs9llgidnp90u3h.org: type NSEC3, class IN ▶ h9p7u7tr2u91d0v0ljs9llgidnp90u3h.org: type RRSIG, class IN ▶ <u>o64vmqp2rn5ef3aou4g3hruir3ijhis4.org: type NSEC3, class IN</u> ▶ o64vmqp2rn5ef3aou4g3hruir3ijhis4.org: type RRSIG, class IN 				
<p>▼ Additional records</p> <ul style="list-style-type: none"> ▶ a34353.123456789101112131415161718192021222324252627282930313233343536.123456789.one-domain-to-rule-them-all.org ▶ b34353.123456789101112131415161718192021222324252627282930313233343536.123456789.one-domain-to-rule-them-all.org ▶ b34353.123456789101112131415161718192021222324252627282930313233343536.123456789.one-domain-to-rule-them-all.org ▶ b34353.123456789101112131415161718192021222324252627282930313233343536.123456789.one-domain-to-rule-them-all.org ▶ b34353.123456789101112131415161718192021222324252627282930313233343536.123456789.one-domain-to-rule-them-all.org ▶ a23456789101112131415161718192021222324252627282930313233343536.a234567891011121.one-domain-to-rule-them-all.org ▶ c23456789101112131415161718192021222324252627282930313233343536.c234567891011121.one-domain-to-rule-them-all.org ▶ d23456789101112131415161718192021222324252627282930313233343536.d234567891011121.one-domain-to-rule-them-all.org ▶ e23456789101112131415161718192021222324252627282930313233343536.e234567891011121.one-domain-to-rule-them-all.org ▶ f23456789101112131415161718192021222324252627282930313233343536.f234567891011121.one-domain-to-rule-them-all.org ▶ g23456789101112131415161718192021222324252627282930313233343536.g234567891011121.one-domain-to-rule-them-all.org ▶ h23456789101112131415161718192021222324252627282930313233343536.h234567891011121.one-domain-to-rule-them-all.org ▶ i23456789101112131415161718192021222324252627282930313233343536.i234567891011121.one-domain-to-rule-them-all.org ▶ j23456789101112131415161718192021222324252627282930313233343536.j234567891011121.one-domain-to-rule-them-all.org ▶ sns-pb.isc.org: type A, class IN, addr 132.70.6.244 ▶ pdns3.ultradns.org: type A, class IN, addr 132.70.6.202 				

DNS query sent by resolver

Sp spoofed second fragment

DNS response: First authentic fragment reassembled with spoofed second fragment

Authentic second fragment (discarded after timeout)

ad c4 72 60 e0 ed fd
8f 85 9f 7f cb 7a b8
a5 28 7e 29 a9 08 9f
0020 d1 92 86 22 4e 13 ca
0630 80 00 04 84 46 06 c8
0640 80 00 04 84 46 06 c8
0650 80 00 04 84 46 06 c9
0660 80 00 04 84 46 06 ca
0670 80 00 04 84 46 06 f4
0680 80 00 04 84 46 06 ca
0690 80 00 04 84 46 06 ca
06a0 80 00 04 84 46 06 f4
06b0 80 00 04 84 46 06 f4
06c0 80 00 04 84 46 06 ca
06d0 80 00 04 84 46 06 f4
06e0 80 00 04 84 46 06 77
06f0 80 00 04 84 46 06 f4
0700 80 00 04 84 46 06 f4
0710 80 00 04 84 46 06 f4
0720 80 00 04 84 46 06 ca
0730 80 00 10 20 01 0d b8
0740 70 73 34 c2 eb 00 1c
0750 01 0d b8 85 a3 00 42
0760 21 00 01 00 01 00 01
0770 00 70 10 00 00 00 00

Outline

- Attack model: MitM vs. Off-path
- DNS poisoning: Background
- **Source-port de-randomization** attacks
 - Resolver-behind-NAT, proxy-using-upstream
- **1st-fragment piggybacking** attacks
- Implications and defenses
 - Patches: to resolvers, name-servers, registrars
 - Deploy DNSSEC – correctly... [and fix it, too??]

Still patching after all these years...

- All attacks: real, practical, validated (by others too)
- Resolvers
 - (Smart) pseudo-random port allocation (see paper)
 - Prepend random-length prefix to referral queries
- Name servers:
 - Append random RR
 - Or send random value of EDNS buffer size from NS
 - But...advanced frag attacks may change checksum field – see Esorics'13 paper
- Either: small (non-frag) limit on EDNS (use TCP)
- Registrars: Limit length of subdomain responses

Or... can we just use SSL/TLS ?

- Tempting: forget DNS, just use secure connection!
- Using secure connection **is** a good idea, sure
- But not complete solution:
 - Is web's PKI secure? Hmm...
 - Overhead
 - Unrealistic to expect all web to be fixed
 - Phishing
 - Denial-of-service
 - Non-web applications: **SMTP**, P2P, ...
Even **security**: e.g.: blacklists, SPF, DKIM...

DNSSEC, the time has

- These patches are **come!** too much, too complex, and:
 - Maybe there's another vulnerability/attack?
 - And what about MitM attacker? Like, is BGP secure?
- And... who said they'll suffice??
- We say: **time to properly use DNSSEC**
- But... some improvements may be needed, too
 - Abolish (insecure) NSEC3 OPT-OUT
 - Add **crypto-agility**, esp. critical to adopt ECDSA !
 - More... See our paper on this (and/or talk to us 😊)

Questions ?

Thank you!

Herzberg and Shulman: DNSSEC, the time has come!