CSCE 463/612
Networks and Distributed Processing
Spring 2018

Application Layer IV

Dmitri Loguinov
Texas A&M University

February 13, 2018

Chapter 2: Roadmap

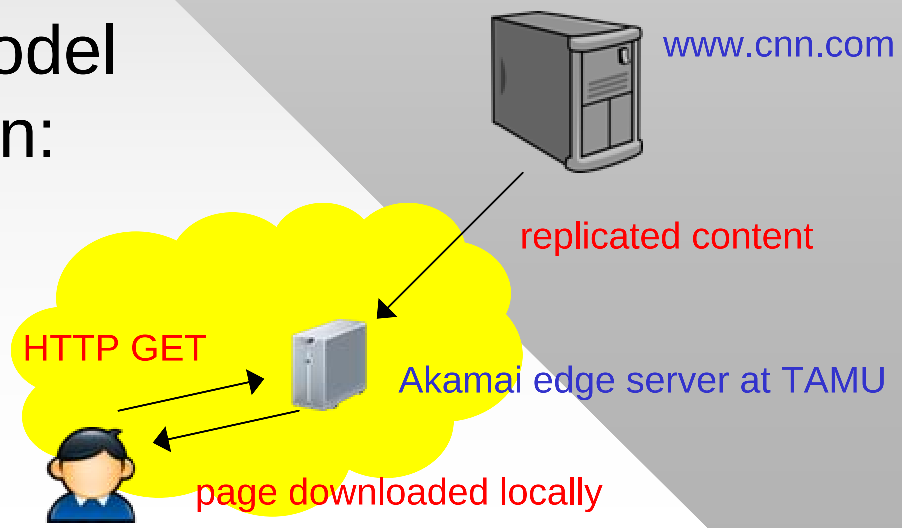
- 2.1 Principles of network applications
- 2.2 Web and HTTP
- 2.3 FTP
- 2.4 Electronic Mail
  - SMTP, POP3, IMAP
- 2.5 DNS (extras)
- 2.6 P2P file sharing
- 2.7 Socket programming with TCP
- 2.8 Socket programming with UDP
- 2.9 Building a Web server

DNS Reality

- DNS is an old protocol with seemingly simple operation
  - Standardized in 1987, mostly unchanged since then
  - Single-packet query, single-packet response
  - UDP-based operation, no congestion/flow control
  - Timeout-based retransmission
- In practice, DNS is rather complex
  - Many decisions go into writing a good resolver, some of which are still not well understood
  - Topic of ongoing research in security, distributed systems, Internet measurement, future network architecture
- Goal now is to understand the limitations of DNS, its vulnerabilities, and various uses

CDNs

- Content Distribution Networks (CDNs)
  - Push replicated content (files, video, images) towards edges
  - Distributed system of application-layer servers
- One of the pioneering CDNs is Akamai
  - J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman, and B. Weihl, “Globally Distributed Content Delivery,” IEEE Internet Computing, Sep/Oct 2002.
- Desired model of operation:



www.cnn.com

replicated content

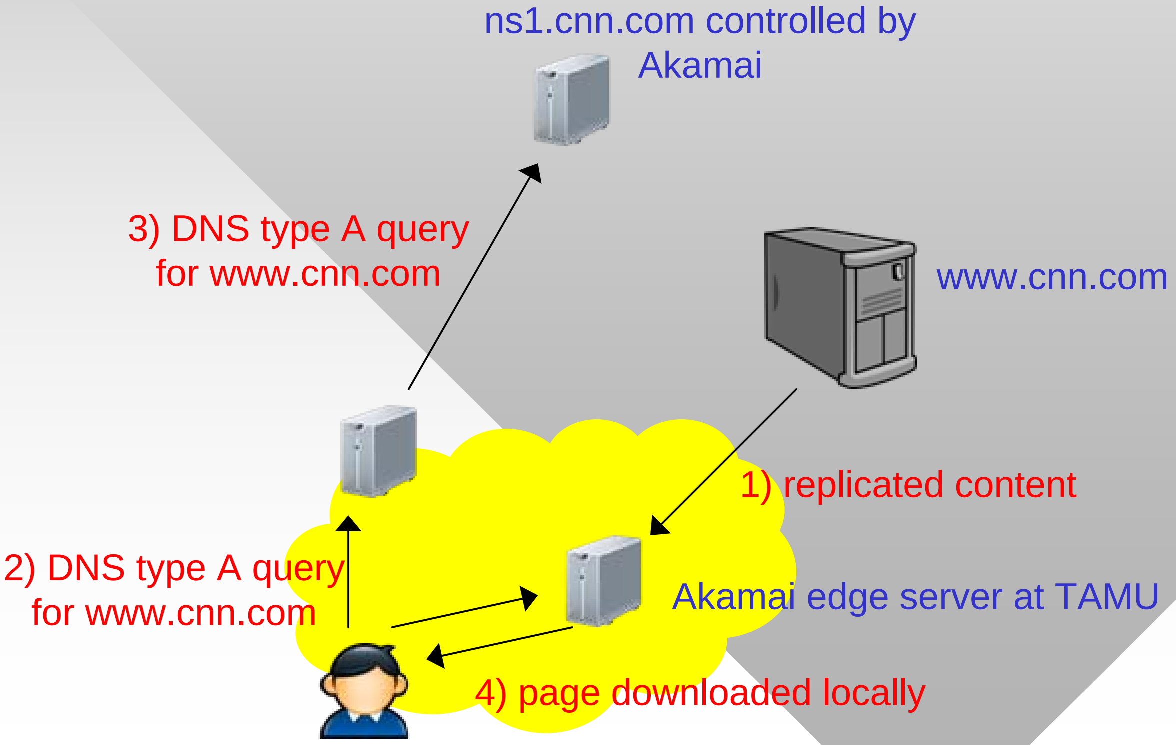
HTTP GET

Akamai edge server at TAMU

page downloaded locally

CDNs 2

- How to direct user to closest replica?
  - Akamai relies on DNS to bounce the user to the best server
  - Based on location of local resolver to find best server (e.g., using distance, load, latency, available bandwidth)



ns1.cnn.com controlled by Akamai

3) DNS type A query for www.cnn.com

www.cnn.com

1) replicated content

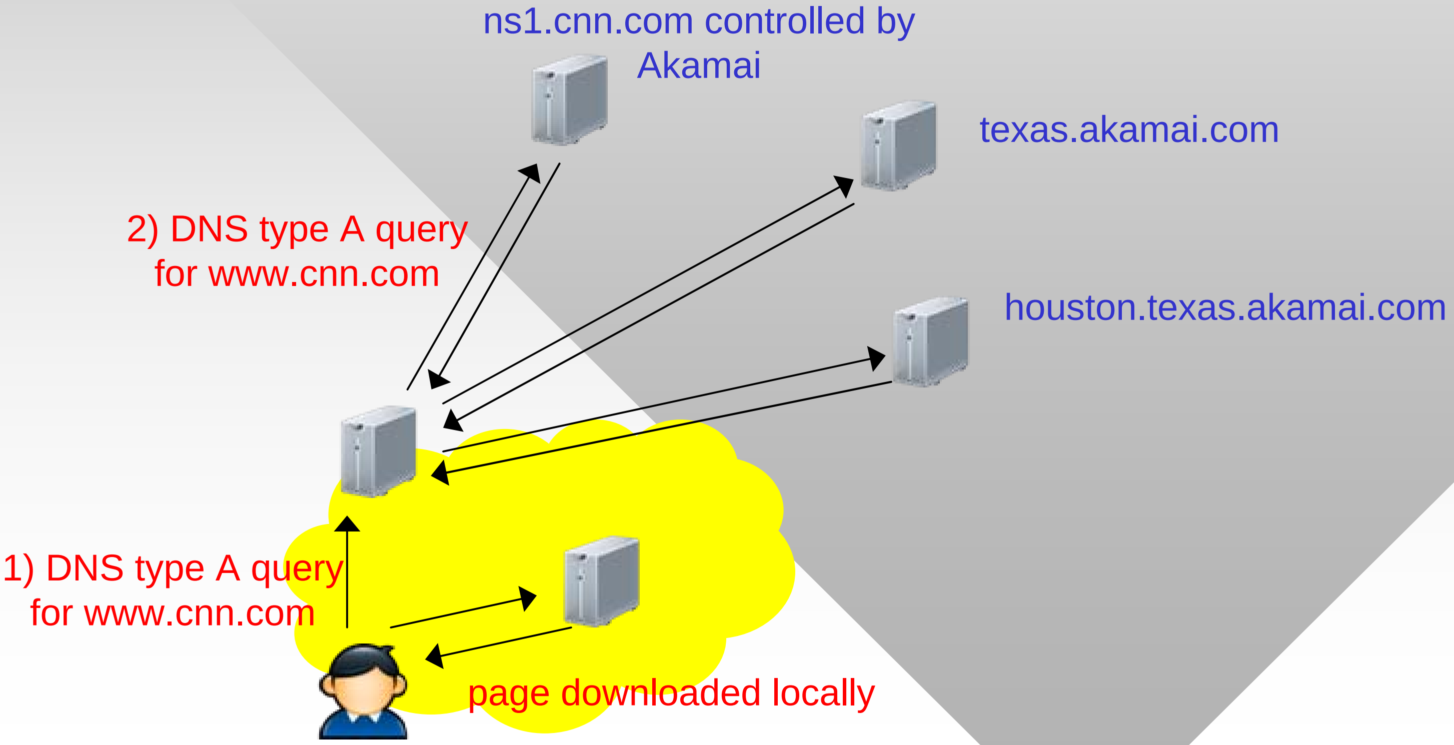
2) DNS type A query for www.cnn.com

Akamai edge server at TAMU

4) page downloaded locally

CDNs 3

- How many servers are there?
  - Over 200K in 120 countries and 1500 networks
- Often Akamai produces long redirect chains
  - Usually through CNAMEs based on the IP of local resolver



CDNs 4

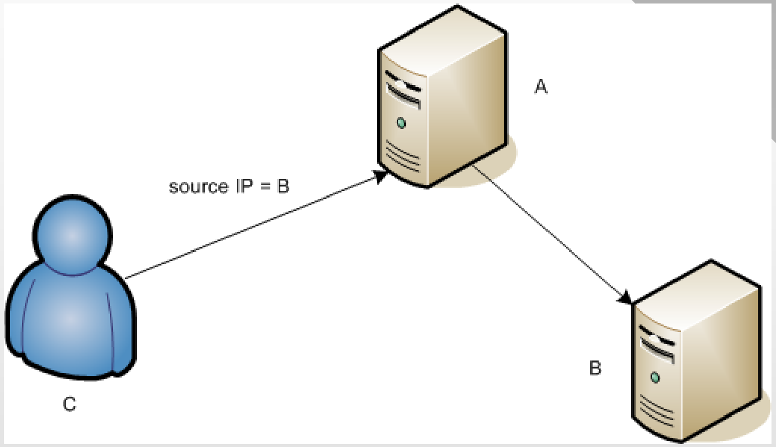
- One research problem in CDNs is how to determine best edge server for the user
  - If multiple options are present, which one is better?
  - What if closest server is overloaded?
  - Not all servers have every possible version of content
  - Need to account for ISP agreements on bandwidth
- Example:

```
www.dhs.gov CNAME www.dhs.gov.edgekey.net
www.dhs.gov.edgekey.net CNAME e4340.dscg.akamaiedge.net
e4340.dscg.akamaiedge.net A 23.45.237.161 (TTL 20 seconds)
```

CDNs 5

- One pitfall of CDNs is that distance from user to their local resolver is generally unknown
  - May lead to inaccuracies for large ISPs
- Another drawback is long resolution chains
  - 15 CNAMEs back-to-back is not just huge latency, but also prone to incorrect configuration, dead-ends, loops
- Caching helps with latency, but Akamai uses extremely small TTLs (e.g., 20 sec), so this is still an issue
- Useful online tools
  - IPgeotool.com tries to map IPs to country/city
  - ARIN, RIPE, APNIC are authorities that allocate subnets; their whois database can be used to map IPs to owners

DNS Vulnerabilities



- Terminology: IP spoofing
  - Packets with fake source IP
- For spoofing to work, ISP network of attacker must allow such packets to depart
  - Robert Beverly, Arthur Berger, Young Hyun, and K Claffy, “Understanding the Efficacy of Deployed Internet Source Address Validation Filtering,” ACM IMC, 2009
  - Of 12K IPs tested, 31% were able to spoof (18% across the US, 5% for edu and home networks)
- TCP spoofing is hard
  - Almost impossible to complete the handshake without knowing parameters of the response packet (only B sees that)
- However, UDP spoofing is easy

DNS Vulnerabilities 2

- Terminology: amplification attacks
  - Hacker transmits small packets to intermediate hosts, which then generate more traffic towards the victim
  - Relies on spoofing the IP of the victim
  - Difficult to trace as the attacker remains hidden
- DNS amplification (1999)
  - Short questions produce large replies, combined with spoofing
  - Large reply = many answers or additional records
- How much amplification can be achieved?
  - IP+UDP+DNS headers = 40 bytes, question $\approx$ 15 bytes
  - Maximum reply is 512 bytes over UDP, ratio 9.3:1
  - 1 Mbps upstream bandwidth per attacker host → 9.3 Mbps

DNS Vulnerabilities 3

- 1000 hijacked hosts → 9.3 Gbps
  - Even a tiny botnet (collection of infected computers under centralized control) can saturate 10 Gbps link
- Main problem: how to find DNS zone with large replies?
- 1) DNS TXT queries
  - Some text associated with a host/domain

```
C:\>nslookup -querytype=txt google.com
Server:  s18.irl.cs.tamu.edu
Address:  128.194.135.58

Non-authoritative answer:
google.com      text = "v=spf1 include:_netblocks.google.com ip4:216.73.93.70/31
ip4:216.73.93.72/31 ~all"
```

Sender Policy Framework (SPF) shows which IPs are authorized to send email on behalf of this domain

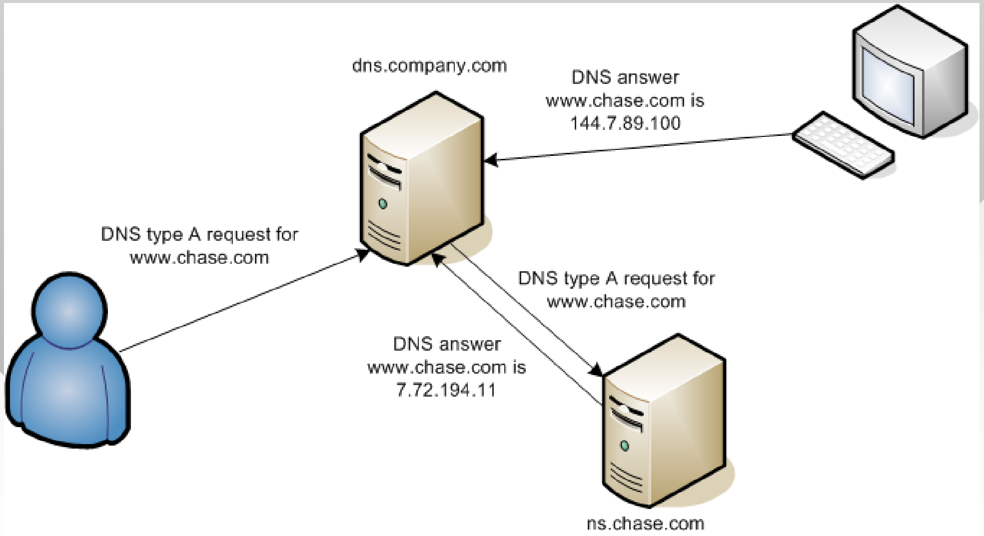
- Text may be large, which leads to easy amplification
  - Traditionally, TXT records were rare; however, new spam-related verification protocols are now actively using them

DNS Vulnerabilities 4

- 2) Domains with many A records/host
  - Google returns 11 IPs per query (212 bytes per packet)
- 3) IPv6 queries (type AAAA) and SOA
  - IPv6 addresses are 16 bytes, SOA contains lots of fields
- 4) DNS extensions (EDNS)
  - Extensions to DNS that support large packets
  - Necessary for signed replies (DNSSEC)
  - With 4KB packets, 134 attacker hosts can flood 10 Gbps
- Amplification falls under the umbrella of DDoS (Distributed Denial of Service) attacks
  - Goal is to overload target server with incoming traffic
- Terminology: insertion of falsified records into local DNS resolver is called cache poisoning

DNS Vulnerabilities 5

- Remote TXID Guess attack (1997)
  - DNS responses cannot be verified for authenticity
  - Possible for attacker to send fake replies to fool local resolver
  - With fake DNS replies, user may arrive to a phishing server and allow attackers to steal their login credentials
- 1) Attacker must know
  - Local DNS server's IP
  - Query string
- 2) Attacker must send fake reply quicker than the authoritative server
  - DNS servers use only the first reply they get, ignore all others

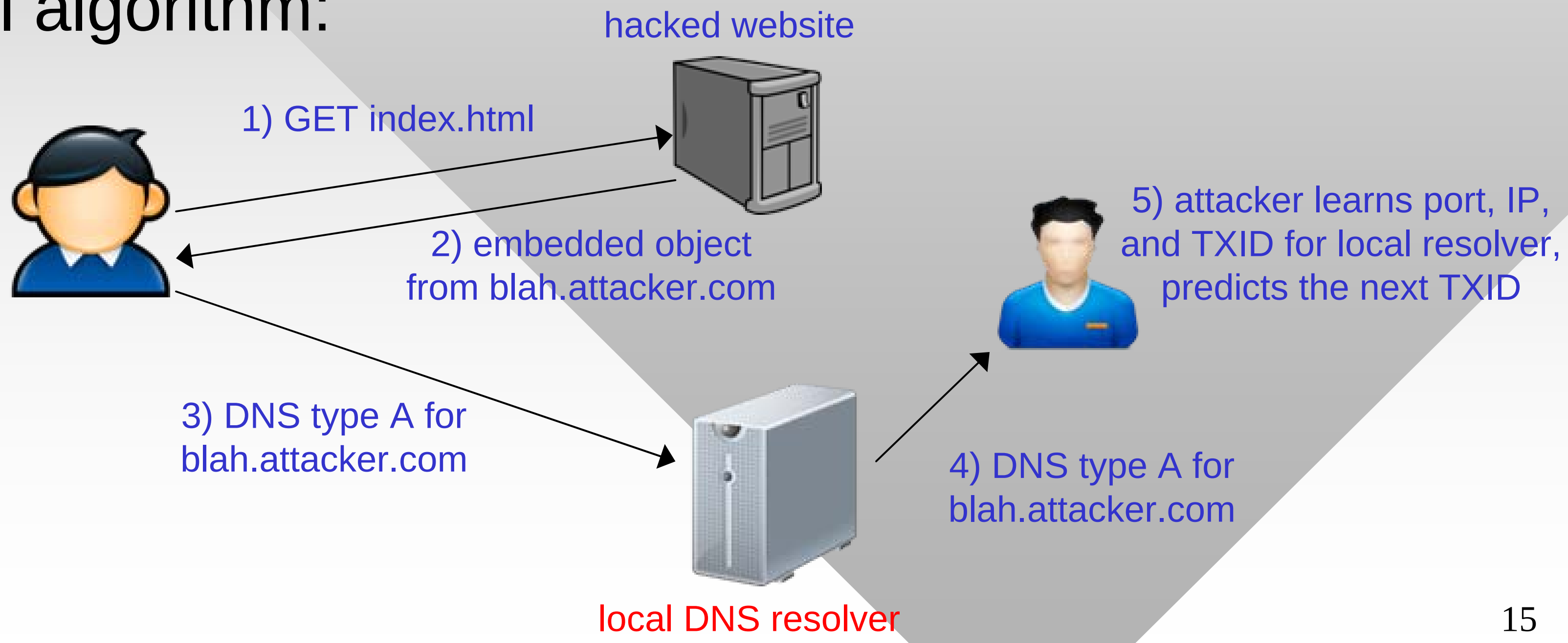


DNS Vulnerabilities 6

- Original DNS design provided protection mechanisms against this type of attack
- Recursive DNS resolver rejects answers unless:
  - Source IP of reply matches that of the authoritative server
  - Local port number is correct
  - TXID in DNS header matches that of the query
- 3) Attacker must spoof source IP of authoritative server
  - Not difficult if the lookup string (www.chase.com) is known
  - If multiple authoritative servers for chase.com, spoof them all
- 4) Attacker must guess local DNS port number
  - DNS servers used to pick a random port during boot and use it for all outgoing queries

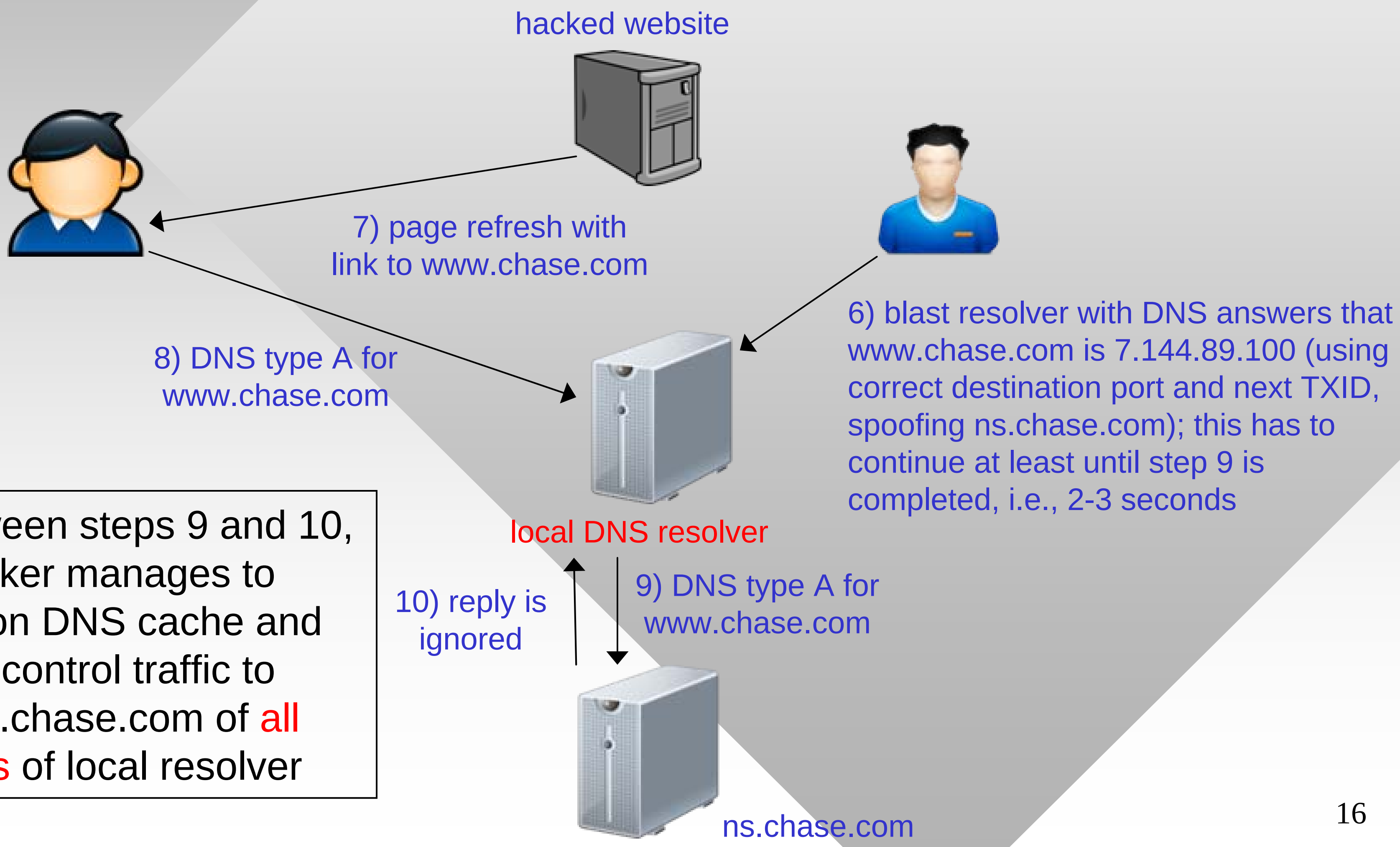
DNS Vulnerabilities 7

- 5) Attacker must guess the TXID of the query
  - Possible only if local resolver (LR) uses predictable TXIDs
  - Many older implementations simply incremented the TXID between the queries or used deterministic random number generators with a fixed seed
- Full algorithm:



DNS Vulnerabilities 8

- Full algorithm (cont’d):



hacked website

7) page refresh with link to www.chase.com

6) blast resolver with DNS answers that www.chase.com is 7.144.89.100 (using correct destination port and next TXID, spoofing ns.chase.com); this has to continue at least until step 9 is completed, i.e., 2-3 seconds

8) DNS type A for www.chase.com

local DNS resolver

10) reply is ignored

9) DNS type A for www.chase.com

ns.chase.com

Between steps 9 and 10, attacker manages to poison DNS cache and then control traffic to www.chase.com of all users of local resolver

Improvements

Birthday problem: in a group of N people, what is the probability that two of them have the same birthday (out of 366 possible birthdays)? → Paradox: 100% with 367 people, 99% with 57, and 50% with 23

- Remote TXID Guess attack is difficult
  - Getting user to visit hijacked website is non-trivial
  - Most modern DNS servers now use unpredictable TXIDs
- The next method works around these possibilities
- Suppose LR transmits each query to authoritative server, even if the same hostname is already pending
  - Each repeated query gets a new TXID
  - BIND 8.2 did this if questions came from unique source IPs
- Birthday paradox (2002) relies on rogue local users
  - Attacker forces local resolver to perform lookups for www.chase.com N times back-to-back
  - After N requests, attacker blasts N answers at LR with random TXIDs, spoofing ns.chase.com's IP address

Improvements 2

- Probability of success p(N) scales quadratically in N
  - Define $M = 2^{16}$ to be the size of TXID space
  - First guess is correct with probability N / M
  - Second with N / (M-1), third N / (M-2), etc.
  - Approximations are accurate for N << M

$$p(N) = 1 - \prod_{i=0}^{N-1} \left(1 - \frac{N}{M-i}\right) \approx 1 - \left(1 - \frac{N}{M}\right)^N \approx 1 - e^{-N^2/M} \approx \frac{N^2}{M}$$

- Examples
  - $p(1) = 2^{-16}$, p(10) = 0.15%, p(128) = 22%, p(512) = 98%
  - Note that p(N) = 100% for N > M / 2
- What if www.chase.com is already cached by LR?
  - Both Birthday Paradox and Remote TXID Guess fail

Improvements 3

- Attacker must wait until target expires, then pull off attack just before the host gets cached again
  - For popular websites, window of opportunity is small
- Kaminsky exploit (2008) works around this problem
  - Noticed a loophole: NS records override cached versions if *they come from an authoritative server*
  - LR’s outbound port is known, but all other bugs are fixed (i.e., TXID is unpredictable, one pending request per hostname)
- Local user issues request for hash1.chase.com
  - Sends K spoofed packets to LR with random TXIDs
  - Spoofed packets have no answers, only NS and additional records for domain chase.com
- Response manages to overwrite existing NS entries!

Improvements 4

- Modeling probability of success
  - First packet is a correct guess with probability 1/M
  - Second packet with probability 1/(M-1), third 1/(M-2), etc.
- If attack does not work, repeat with hash2.chase.com
  - Each attempt is independent, thus the probability to fail is the product of individual probabilities to fail in each attempt
- After N attempts (N*K packets), we have:

$$p(K,N) = 1 - \prod_{i=0}^{K-1} \left(1 - \frac{1}{M-i}\right)^{N} \approx 1 - \left(1 - \frac{1}{M}\right)^{KN} \approx 1 - e^{-KN/M}$$

- Kaminsky broke common DNS implementations (IIS, BIND) in about 10 seconds
  - p(100,10) = 1.5%, p(250,40) = 14%, p(500,200) = 78%

Improvements 5

- Why can’t K be equal to M?
  - May not have enough bandwidth before ns.chase.com replies
- Closing the Kaminsky loophole
  - Randomization of port numbers for each query (IIS, BIND)
  - Random capitalization of query strings (wWw.ChasE.coM) and case-sensitive comparison of answers (Pydig, Unbound)
  - Rejection of new NS records if already cached (not recommended in case domain needs to override old answers)
- With port randomization, $M = 2^{32}$ possibilities
  - Windows 7-10 has 16K (default) available ports, $M = 2^{30}$ = 1B
- Random capitalization adds 2^S options, S = host len
  - For the average Internet hostname, S = 20
  - This increases M by an additional factor of 1M