Introduction into Cyber Security

Chapter 12: DNS Security

WiSe 18/19

Chair of IT Security

Chapter Overview

- Overview on the operation of the DNS
- Attacks against DNS
- Securing DNS

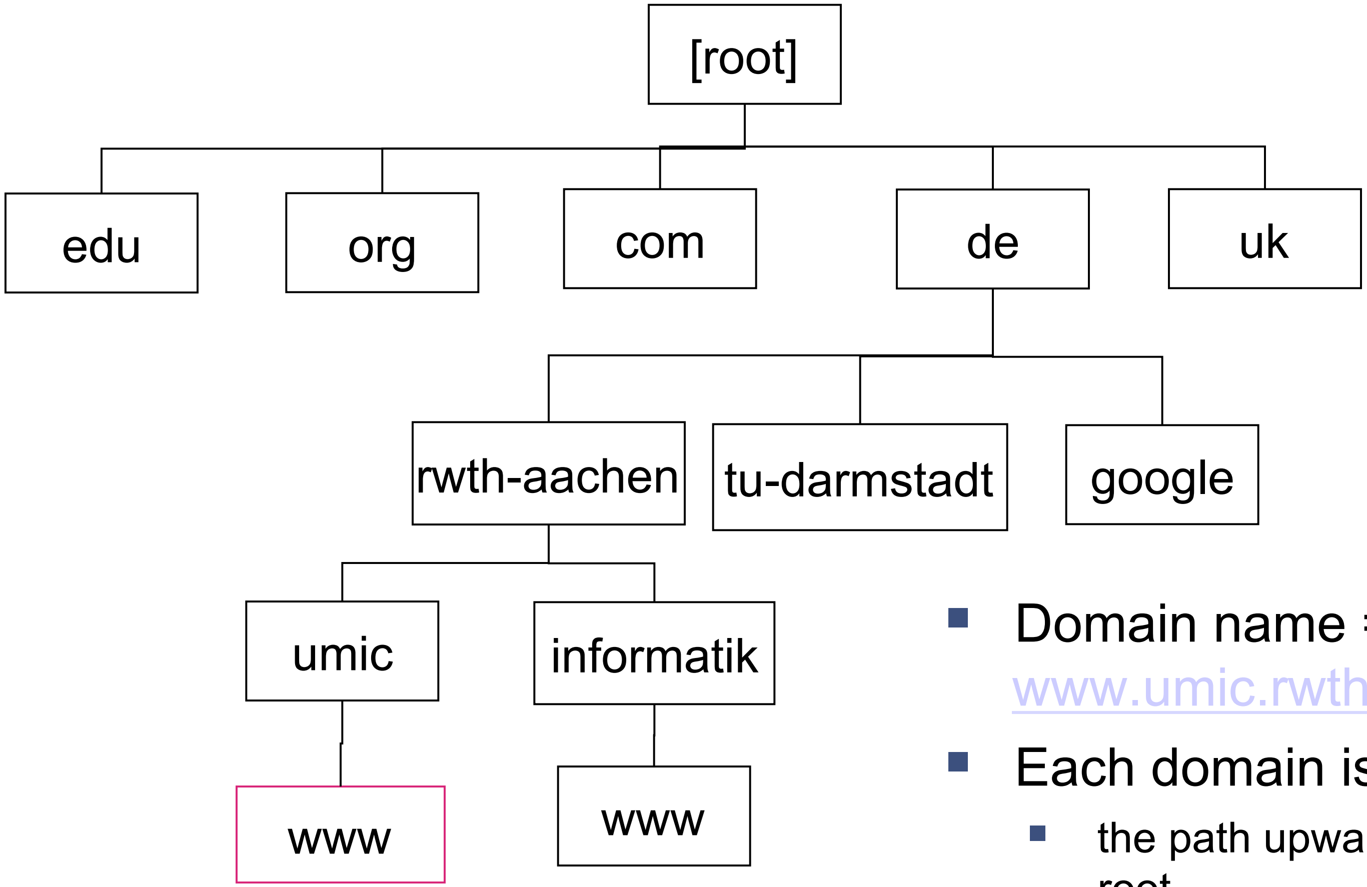
The Domain Name System

- Main purpose
  - Map human readable, hierarchically structured “domain name” to IP addresses
- The DNS system is specified in RFC 1034 and 1035
  - Updating RFCs have been published but non rendering the above RFCs obsolete
- Briefly:
  - To map an IP address to a domain name an application program calls resolver and passes it the domain name
  - Resolver sends a UDP packet to a local DNS server
  - Local DNS server looks up the name and returns it to the resolver
  - Resolver returns IP address to the application

DNS Name Space

- Is a tree structured name space starting from the unnamed root domain
- There are over 200 top level domains
  - Come in two flavors: generic and countries
    - Generic: e.g. org, com, edu, mil, net, pro, ….
    - Countries: e.g. de, es, cn, uk,…
- Each top-level domain is partitioned into subdomains which are partitioned into subdomains, ….
- A leaf domain in a tree may contain a single host or represent thousands of hosts

Some Part of the Domain Name Tree



- Domain name = www.umic.rwth-aachen.de
- Each domain is named by
  - the path upward from it to the unnamed root
  - The components are separated by dots

Adding New Subdomains

- Each domain controls the allocation of subdomains under it
- To create a new domain, permission is required of the parent domain in which it will be included
- Once a new domain is registered it can create subdomains without further permissions

Resource Records (1)

- Every domain can have a set of resource records associated with it
- The most common resource record is the “A” record which stores the IP address corresponding to a domain name
- The primary function of DNS is to map domain names to resource records
- Resource records have the format

Domain Name	Time-to-live	Class	Type	Value

Resource Records (2)

Domain Name	Time-to-live	Class	Type	Value

- Domain name - indicates the domain to which the record applies
- Time-to-live - indicates the number of seconds this record should be cached, i.e. how stable the record is
- Class – indicates the class of information
  - IN for Internet information
  - Other classes are barely ever seen
- Type field – indicates what kind of resource record this is
  - Most common type is the “A” type for an IP address
- Value – contains the actual value corresponding to the type
  - E.g. the IP address in case of an “A” type resource record

Types of Resource Records

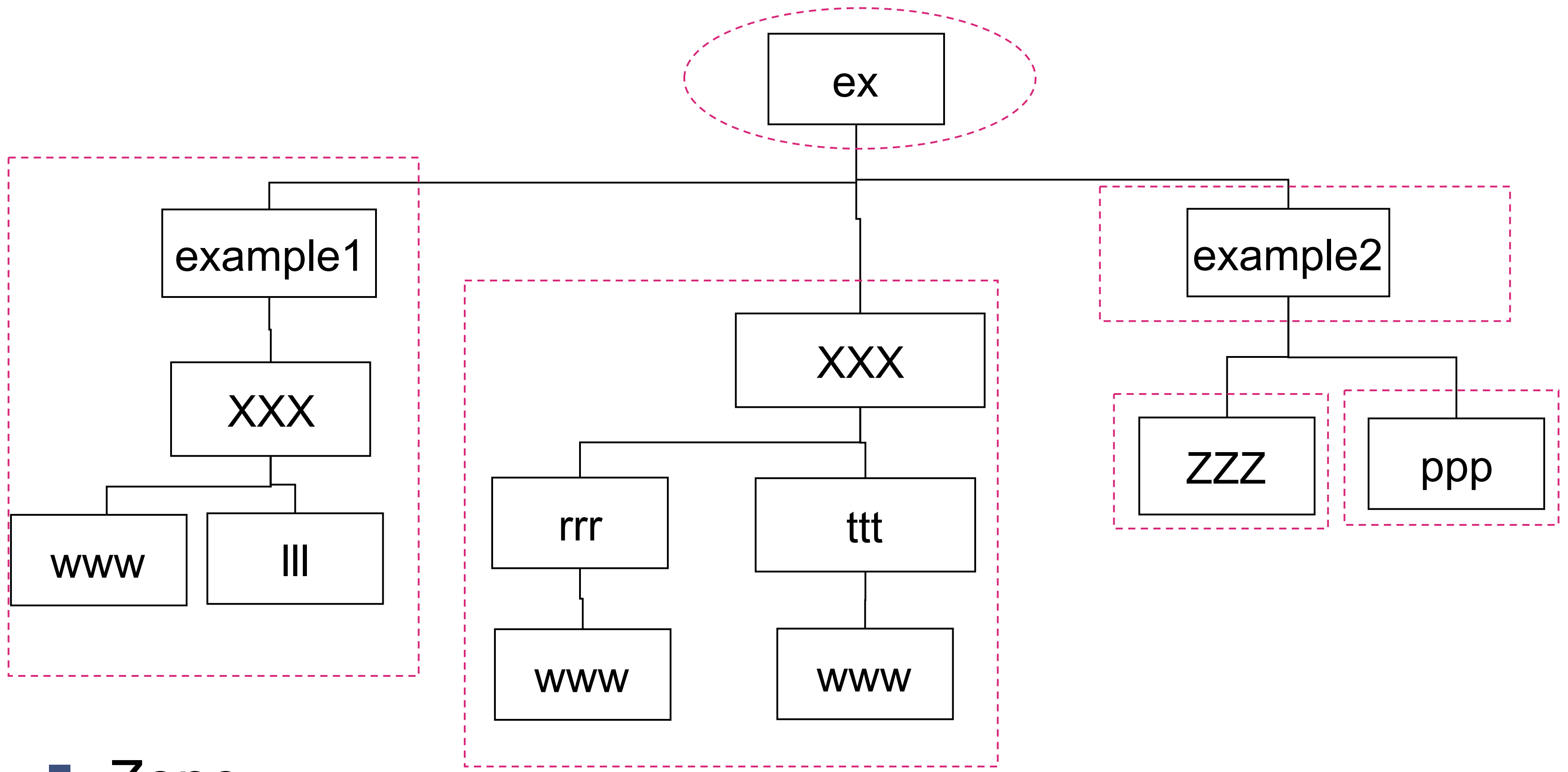
Type	Meaning	Value
SOA	Start of Authority	Parameters for this zone
A	IP address of a host	32 bit
AAAA	IPv6 address of a host	128 bit
MX	Mail exchange	Domain name of mail server accepting email for this domain
NS	Name Server	Domain name of a name server for this domain
CNAME	Canonical Name	Domain name, allow for creation of aliases
PTR	Pointer	Used mainly for reverse lookups
HINFO	Host information	Provides additional information on hosts, such as OS running on it

Structure of DNS Messages

Header
Question
Answer
Authority
Additional Information

- Header
  - Always included, indicates what other parts are present
  - Includes a query ID and indicates if message is query or response
- Question - Domain name, type of answer desired
- Answer - Contains the resource records that answer the question
- Authority - Resource record that points to the authority for this domain
- Additional information - Resource records with additional information

Zones



- Zone
  - Connected part of the domain name space
  - As opposed to domains, the children of a node in a zone do not have to belong to the same zone

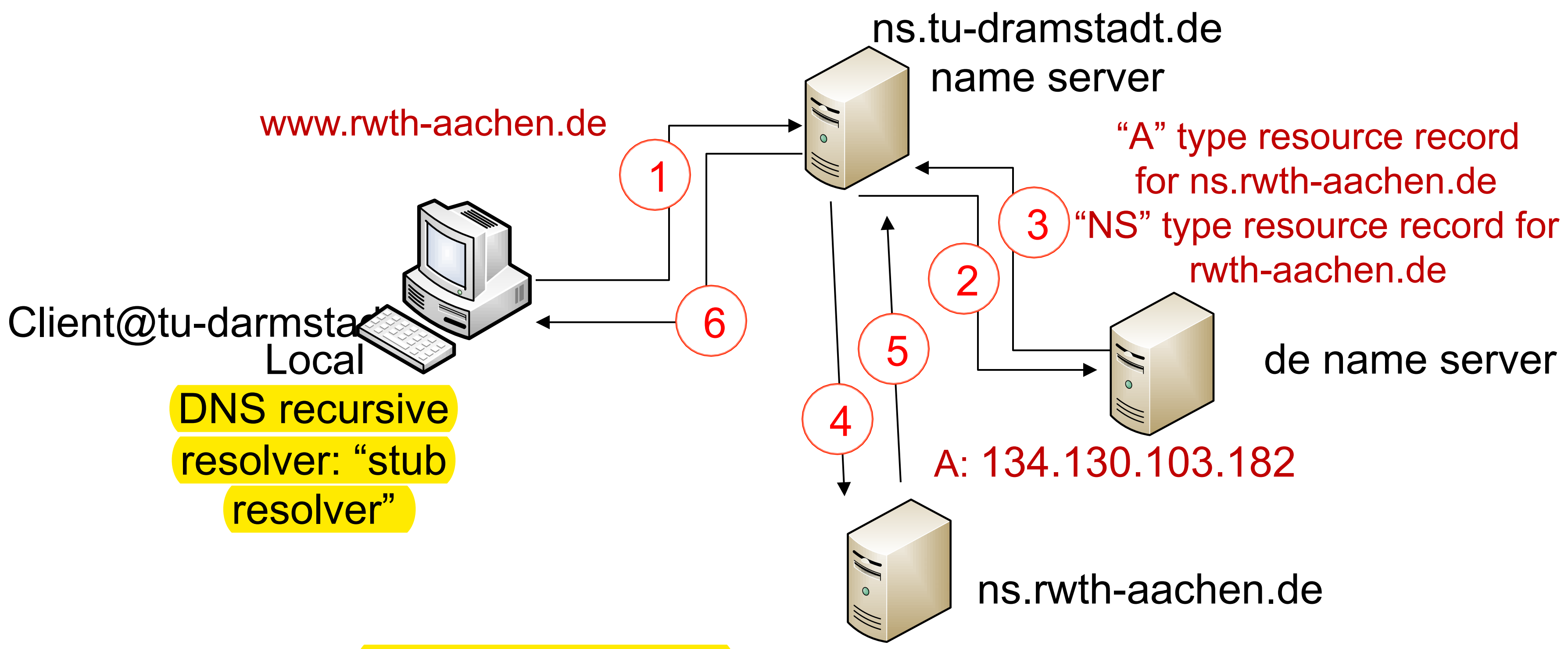
Authoritative Name Servers

- Each zone contains part of the domain name tree and name servers holding the information about this zone
- A name server is authoritative for a zone if it is the primary source of resource records for the domain names in the zone
- Data stored on a name server is typically divided into
  - Authoritative data for that name server
  - Cached data
- Each name server has resource records for all children nodes in its zone

Recursive vs Non-Recursive

- Non-Recursive resolution
  - The name server queried returns an answer corresponding to its own knowledge
  - Does not request information from other name servers
  - May respond with a suitable NS resource record containing the domain name of a suitable name server and a type A resource record containing the IP address of the name server
- Recursive resoution
  - Name server queries other name server if it does not have the answer to a query itself

Address Resolution

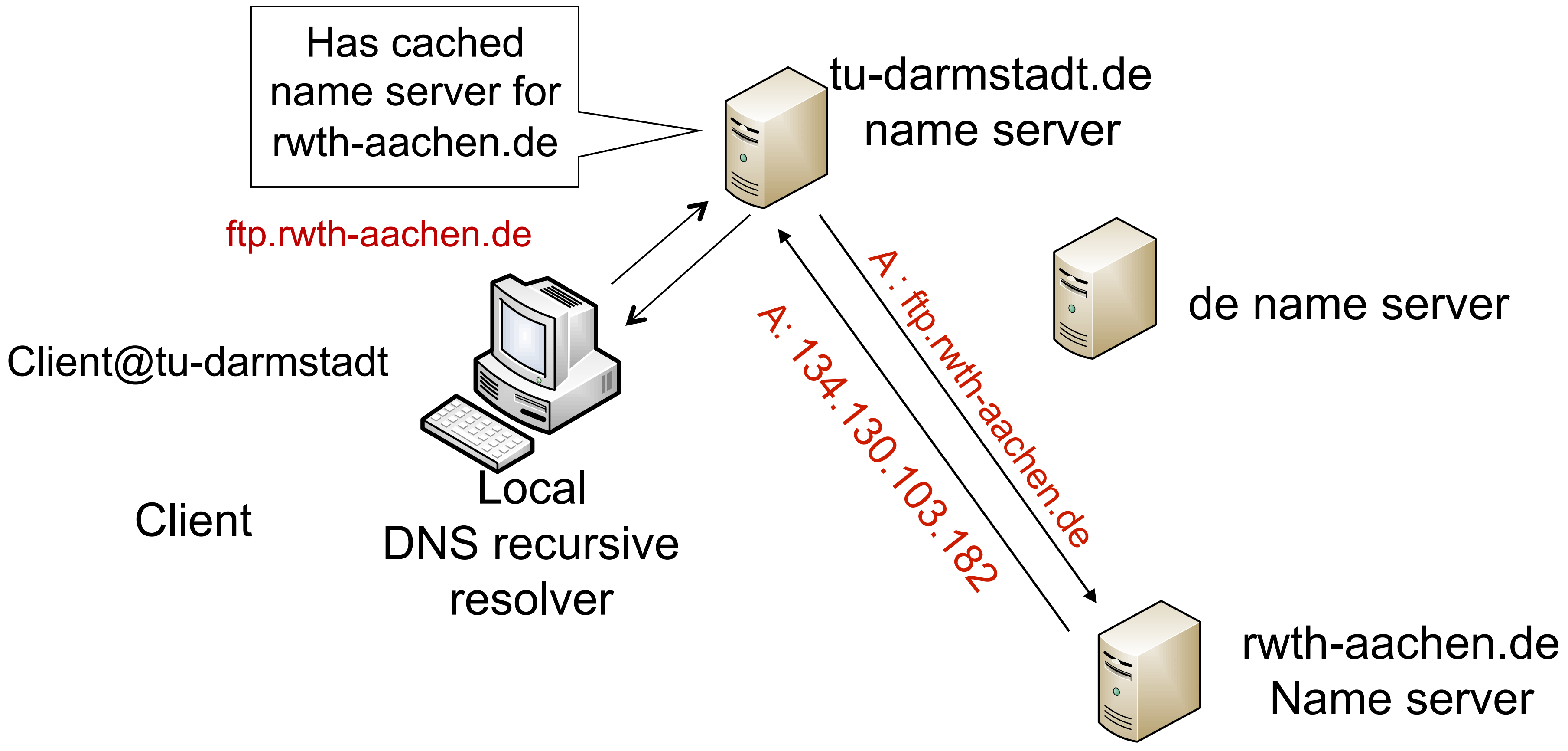


- The client's local stub resolver recursively queries the configured tu-darmstadt.de name server
- If the name server does not have the answer cached it queries the de name server which, e.g. responds with an "NS" type resource record for rwth-aachen.de and an "A" type resource record for the rwth-aachen.de name server

Caching

- DNS responses are cached
  - Quick response for repeated translations
  - Other queries may reuse some parts of lookup
    - NS records for domains
- DNS negative queries are cached
  - Don’t have to repeat past mistakes
    - For example, misspellings
- Cached data periodically times out
  - Time-to-live (TTL) of data controlled by owner of data
  - TTL passed with every resource record (see above)
  - But TTL not authenticated (see bellow)

Cached Lookup Example


Has cached name server for rwth-aachen.de
tu-darmstadt.de name server
ftp.rwth-aachen.de
de name server
Client@tu-darmstadt
A : ftp.rwth-aachen.de
A: 134.130.103.182
Client
Local DNS recursive resolver
rwth-aachen.de Name server

Resolution Paths

- Applications
  - Use their own small caches, query OS-integrated resolver for anything not in the cache
- Local Stub Resolver in operating system
  - Uses own cache, queries name server configured by user or via DHCP
- Other name servers on local network
- Root Servers, i.e. the ones authoritative for the top-level domains
  - Although caching is used, root servers receive ~ 12 000 queries per second (in 2012)

Attacking DNS

- Altering or replacing information stored by DNS name servers
  - Tricking a DNS server into caching or permanently storing a false mapping between an IP address and a domain name is called DNS spoofing
  - A cache that holds a false IP address / domain name mapping is called poisoned
    - Note that such entries stay in the cache as long as the TTL indicates
- Flooding DNS name servers with requests
  - If requests originate from different IP addresses
    - Valid queries are very hard to tell apart from those that are part of an attack

Motives for Attacking DNS

- Making hosts and their services unavailable
  - By forging “non-existent” replies or
  - Forging replies that contain wrong IP addresses
- With the goal of
  - Blackmailing companies or even countries
  - Sabotaging competing organizations
  - Vandalism
- Redirecting web traffic by answering to queries with IP addresses pointing to another web server
  - Phishing
  - Disinformation
  - Censorship via DNS

Attacks Against Stub Resolvers

- Attacker manipulates information kept in the stub resolver on a host, e.g. by
  - Intercepting and replacing responses to queries made by the stub resolver
  - Using a vulnerability of the implementation

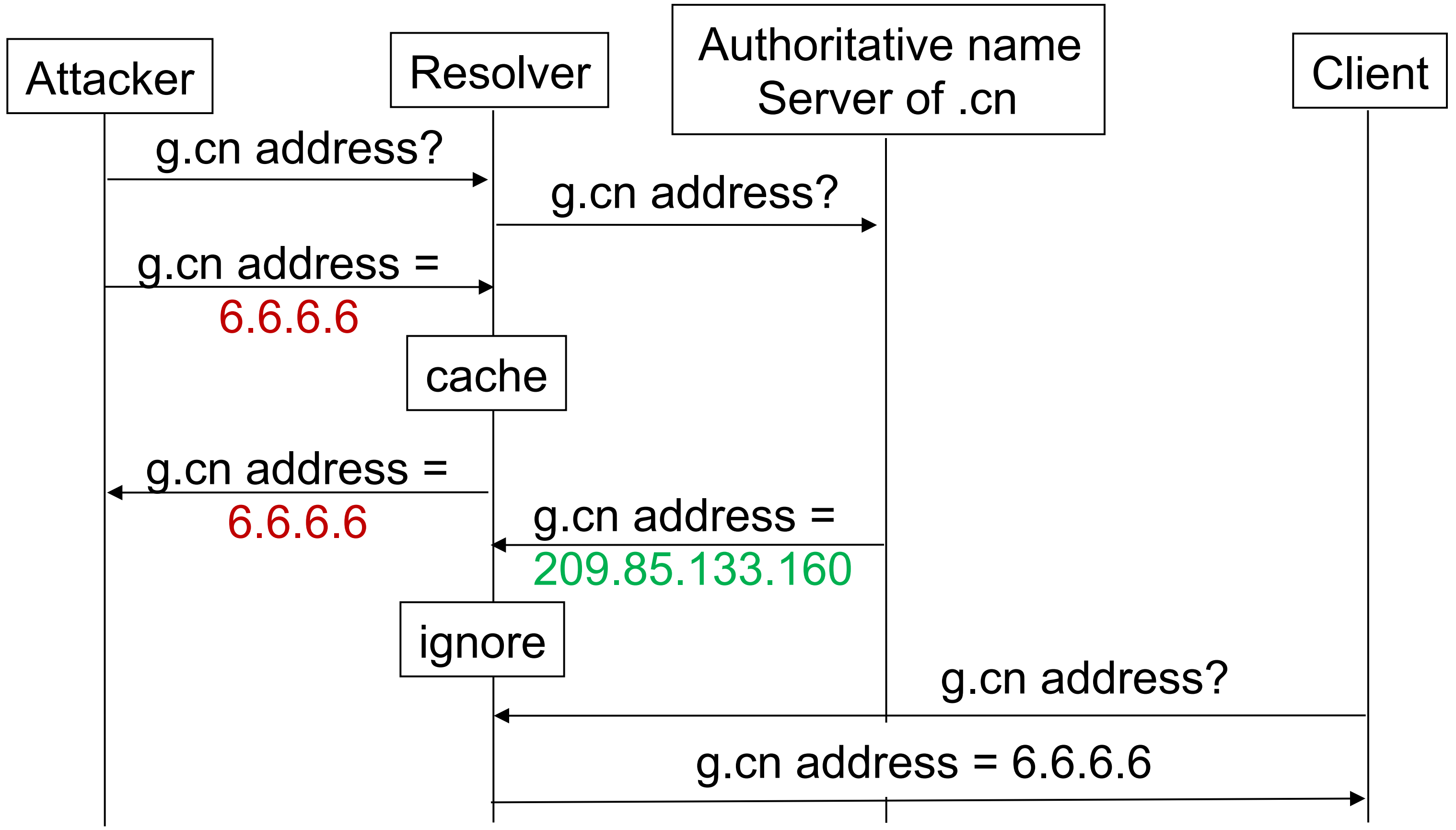
How are DNS responses checked today?

- Upon receipt of a response today’s resolvers will
  - Check that the query ID in query and response match
  - The IP address of the responding name server correspond to the server to which it sent the query
  - The response was sent to the destination port indicated as source port in the query

Cache Poisoning Attack

- Most serious attacks on DNS itself
- Targets a caching resolver
- Tries to make the resolver accept a faked reply to a query it sent out
- Requires attacker to
  - Predict or intercept
    - the query ID used by the resolver when sending out the query
    - the source port number of the query
    - the IP address of the name server the resolver has queried
  - And either
    - Suppress the response of the real name server
    - Or beat the real name server in a race condition
    - Or be the real name server in the first place

Cache Poisoning Example



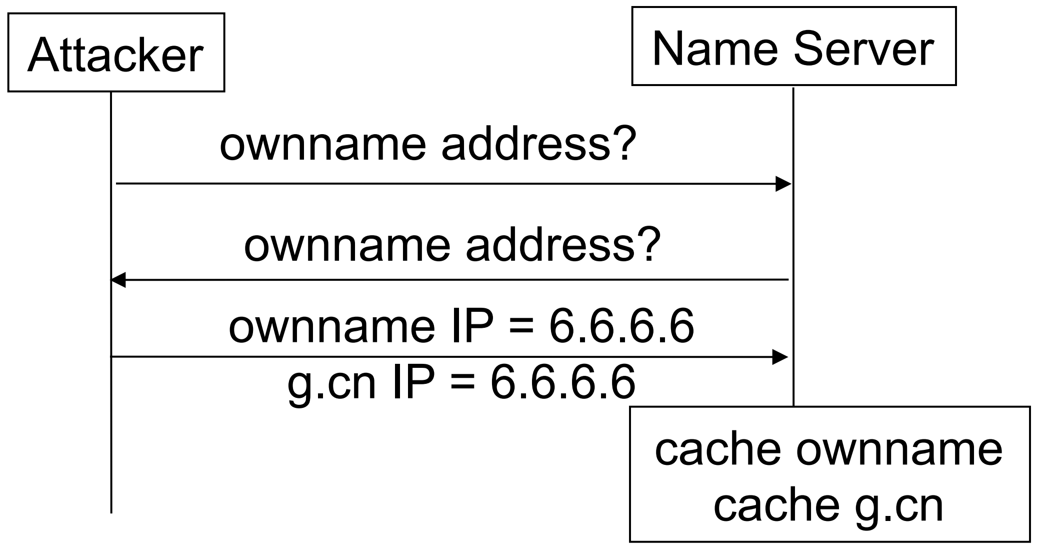
- Illustrates how attacker tries to trick resolver into accepting a false entry for g.cn

Early Example of a Cache Poisoning Attack (1)

- Responses may have an “additional information section”
  - Contain additional domain name / address mappings
- Originally, it was not checked if these mappings were in the zone of the name server providing them
  - Allowed also cached information to be added to the additional info section
- This enabled an attacker (compromised DNS server) to include fake mappings in responses to queries for domains in his authority
- This is not possible any more today
  - DNS servers today check that additional information section does not include information for which the responding name server is not authoritative

Early Example of a Cache Poisoning Attack (2)

- Attacker pretends to be a client and asks victim name server to resolve address under attacker's authority
- Name server asks (other name servers and ultimately) the attacker for the address in question
- Attacker includes other wrong information in the answer, here a false IP address for Google China



Why Query ID Should be Random

- Query IDs are 16 bit identifiers
  - Included in queries and copied into responses
- Used by the resolver to match queries and responses
- If an attacker can predict the next query ID used by a resolver he can
  - Send a query for a victim domain to the victim resolver
  - Immediately send a response to this query with the IP address of the name server asked by the resolver with the predicted query ID
  - This successfully poisons the cache of the resolver

➢ Query IDs are randomly selected to protect against this very simple variant of cache poisoning

DNS Attack by Kaminsky 2008 - Goal

- Also a cache poisoning attack
- Based on the observation that if a cache poisoning attack is not successful, the correct resource records for the victim will stay in the cache until the TTL expires, i.e. attacker has to wait until then before he is able to try again
- With Kaminsky's attack, the attacker never causes a correct resource record to be cached at the resolver

DNS Attack by Kaminsky 2008 - Operation

- Attacker sends queries for possibly non-existing other subdomains of the victim domain to the victim resolver
  - E.g. 1.example.com if the victim domain is www.example.com
- Attacker sends responses to resolver with
  - Varying query IDs for 1.example.com
  - From faked IP address of the authoritative name server of example.com domain
  - Including an address resource record for www.example.com in the additional information section
- If not successful attacker tries again with 2.example.com
- Attack will eventually work and address for www.example.com will also be accepted as it is in bailiwick of x.example.com
- This attack is particularly dangerous as
  - the attacker can immediately chose a new domain when the real name server responded faster, i.e. the attacker does not have to wait for the TTL of the real response to expire before being able to try again as the real name server answers with info about x.example.com but not for www.example.com

Source Port Randomization

- The 16 bit query ID needs to match in an attackers fake response to a resolvers query in cache poisoning attacks
- Successfully guessing a query ID before the authoritative response arrives is thus necessary
- Additionally randomizing the source port number of queries requires the attacker to successfully guess a pair of source port and query ID
- This is considered to make Kamininsky’s attack impossible

- However: this just makes the attack slower, it does not completely hinder it

Securing DNS with IPsec?

- Use of IPsec between each resolver and name server would allow for detecting responses with spoofed IP addresses
  - E.g. cache poisoning would not be possible anymore at all
- IPsec security associations would have to be dynamically established between all name servers and all resolvers
- Would require a PKI to be in place and nodes to be able to obtain public keys
- Deemed to be too much of an overload

DNSSEC

- Specified in RFC 4033
- Goals: authentication and integrity protection of resource records included in DNS responses with the help of digital signatures
- DNS server signs a hash of the resource records with the private key for the zone
- Signature provided to the resolver in a new resource record
- Resolvers learn public keys of zones by DNS queries
  - Public keys are signed by the parent domain
- The public keys of some zones must be preconfigured in the resolvers
- DNSSEC does not provide confidentiality or access control

Problems with DNSSEC

- Supported by many resolver implementations already but not in stub resolvers of operating systems
- Penetration of DNS servers with DNSSEC slow but steady
  - As of May 2010 the root domain is finally singed
  - Early 2013: about 35% of DNS servers supported DNSSec validation
  - Summer 2018: 90% top-level domains but only 3% of individual domains in general!

DNS Intrusion Detection

- Methods to detect DNS attacks include
  - Monitoring the number of packets received that contain non-matching query IDs
  - Monitoring the number of unusually large packets
  - Waiting for a second reply to potentially arrive
  - Monitor the number of non-existent responses and alert if very high
  - Re-resolving names after short delays to ensure consistency
- Measures that can be taken when under attack
  - Limit data rates for certain senders
  - Switch queries to TCP (additional 32 bit sequence number)
  - Limit data rate for whole server?

References and Reading

- DNS in general
  - Chapter 2.1 of Andrew Tanenbaum’s “Computer Networks”
  - Chapter 19 of Stallings “Cryptography and Network Security”
- RFCs
  - RFC 1034: Domain Names – Concepts and Facilities
  - RFC 1035: Domain Names – Implementation and Specification
  - RFC 4033: DNS Security Introduction and Requirements