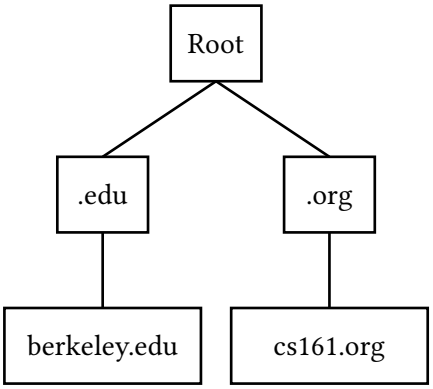
CS161 Fall 2026 — Introduction to Computer Security — Exam Prep 7

Q1 *Networking and Web: DNS* ❓ **(10 points)**

In this question, consider the DNS hierarchy to the right.

Unless otherwise stated, assume normal DNS is being used, without DNSSEC or DNS over HTTPS.

A student is performing a DNS lookup, and Mallory is an in-path attacker between the student and the recursive resolver.



Q1.1 (2 points) Which attacks could Mallory carry out against the student's DNS lookup for `cs161.berkeley.edu`? Select all that apply.

- ■ Observe which domain name the student is looking up.
- ■ Spoof a DNS reply that maps the domain to an attacker-controlled IP address.
- ☐ Decrypt the student's HTTPS traffic after the browser has verified the server's TLS certificate.
- ☐ Directly read cookies stored on the real `cs161.berkeley.edu` server.
- ○ None of the above

Solution: Normal DNS requests and replies are not encrypted or authenticated.

Therefore, an in-network attacker can see the domain being queried and can try to tamper with the response, for example by returning an attacker-controlled IP address.

However, spoofing DNS does not automatically break HTTPS. If the student visits an HTTPS URL and the browser correctly verifies the server certificate, Mallory still needs a valid certificate for the target domain to complete the TLS handshake successfully.

Mallory also does not compromise the real CS 161 server merely by tampering with DNS.

This content is protected and may not be shared, uploaded, or distributed.

Q1.2 (2 points) Now suppose DNS over HTTPS is in use. Which attacks could Mallory carry out against the student's DNS lookup for `cs161.berkeley.edu`? Select all that apply.

- ☐ Observe which domain name the student is looking up.
- ☐ Spoof a DNS reply that maps the domain to an attacker-controlled IP address.
- ☐ Decrypt the student's HTTPS traffic after the browser has verified the server's TLS certificate.
- ☐ Directly read cookies stored on the real `cs161.berkeley.edu` server.
- ● None of the above

Solution: DNS over HTTPS protects the DNS traffic between the client and the DNS-over-HTTPS resolver by sending the DNS query and response inside HTTPS.

This means an in-network attacker can no longer simply read or modify the DNS request or response in transit.

However, DNS over HTTPS does not hide the query from the DNS-over-HTTPS resolver itself. It also does not guarantee that the DNS records are morally or semantically safe. If the relevant authoritative DNS records have been maliciously changed, or if the DNS-over-HTTPS resolver receives malicious records from the DNS system, DNS over HTTPS does not magically fix that.

For the rest of the question, suppose Mallory has successfully executed an attack which gives her control of the `cs161.berkeley.edu` domain.

This content is protected and may not be shared, uploaded, or distributed.

Q1.3 (2 points) Suppose a student has the following five cookies in their browser, with `Path=/`, `Secure=false` and `HttpOnly=true` on all five cookies.

After Mallory's attack, which cookies can Mallory read? Select all that apply.

- ☐ `Domain=sp26.cs161.org`
- ☐ `Domain=cs161.org`
- ■ `Domain=www.mallory.com`
- ■ `Domain=cs161.berkeley.edu`
- ■ `Domain=berkeley.edu`
- ○ None of the above

Solution:

(A) and (B) are false. If the student's browser makes a request to `http://sp26.cs161.org`, per cookie policy, cookies with `Domain=sp26.cs161.org` or `Domain=cs161.org` will be included in the request.

The student's browser will then perform a DNS lookup for the IP address of `sp26.cs161.org`. Since Mallory's attack only gives her control over `cs161.berkeley.edu`, `sp26.cs161.org` will resolve to the correct domain, and Mallory cannot learn anything.

(C) was originally intended to be true, but we accepted both answers for credit. If the student's browser makes a request to `http://www.mallory.com`, per cookie policy, cookies with `Domain=www.mallory.com` will be included in the request.

- If you assumed, as staff did, that Mallory controls `www.mallory.com`, she is able to see the contents of the request, including the cookies.
- If you did not assume that Mallory controls `www.mallory.com`, she is not able to see anything.

In the future, we will do our best to clarify whether Mallory controls a domain.

(D) and (E) are true. If the student's browser makes a request to `http://cs161.berkeley.edu`, per cookie policy, cookies with `Domain=cs161.berkeley.edu` or `Domain=berkeley.edu` will be included in the request.

Then, after the HTTP request is fully-formed, with the relevant cookies attached (as seen above), the student's browser performs a DNS lookup for the IP address of `cs161.berkeley.edu`, which is mapped to Mallory's IP (as a result of her attack). Now, the cookies get sent to Mallory's IP.

This content is protected and may not be shared, uploaded, or distributed.

Q1.4 (2 points) After Mallory's attack, what strategies can Mallory use to read the cookies selected in the previous subpart? Consider each option independently.

- [■] Mallory tricks the student into clicking some links she supplies.
- [■] Mallory tricks the student into running some JavaScript she supplies.
- [☐] Mallory's attack allows her to directly read the cookies stored on the CS 161 server.
- ◯ None of the above

Solution:

(A) is true. As seen in the solution to the previous subpart, if the student makes some HTTP GET requests to URLs like `http://cs161.berkeley.edu` or `http://www.mallory.com` or `http://sp26.cs161.org`, the requests will have some cookies attached and get sent to Mallory.

(B) is true. JavaScript can be used to trigger HTTP GET requests. As in option (A), those requests will have cookies attached and be sent to Mallory.

(C) is false. Mallory's attack creates a malicious DNS record mapping `sp26.cs161.org` (and `cs161.berkeley.edu` by extension) to her own IP. However, Mallory does not actually compromise the CS 161 server in her attack.

Q1.5 (1 point) Suppose `Secure=true` is set on the cookie with `Domain=cs161.berkeley.edu` above.

After Mallory's attack, can Mallory read this cookie?

- ◯ Yes, even without a certificate mapping `cs161.berkeley.edu` to her public key.
- ● Yes, but only with a certificate mapping `cs161.berkeley.edu` to her public key.
- ◯ No, even with a certificate mapping `cs161.berkeley.edu` to her public key.

Solution:

If `Secure=true` is set, then a TLS connection is formed between the student's browser and `cs161.berkeley.edu` (which is mapped to Mallory's IP, per the DNS attack earlier).

In order for this TLS connection to succeed, Mallory must provide a certificate that maps `cs161.berkeley.edu` to her own public key. This allows the rest of the handshake to complete, since Mallory will be using her own private key later in the handshake (e.g. in the premaster secret exchange).

Note that Mallory has not compromised the CS 161 server, so she does not know the CS 161 server's private key. Therefore, if Mallory uses the original unmodified certificate (mapping `cs161.berkeley.edu` to the CS 161 server's private key) in the handshake, the handshake will fail because Mallory does not know the CS 161 server's private key.

This content is protected and may not be shared, uploaded, or distributed.

Q1.6 (1 point) After Mallory's attack, a student visits `http://cs161.berkeley.edu` in their browser and receives some JavaScript created by Mallory. Is this an XSS attack?

- ○ Yes, because the JavaScript was sent by the student and reflected back to the student.
- ● Yes, because the JavaScript runs with the origin of `cs161.berkeley.edu`.
- ○ No, because the JavaScript runs with the origin of `www.mallory.com`.
- ○ No, because this is a CSRF attack, not an XSS attack.

Solution:

An XSS attack occurs when JavaScript supplied by the attacker runs with the origin of somebody else (not the attacker).

Here, the JavaScript is supplied by Mallory, but runs with the origin of `cs161.berkeley.edu`, so an XSS attack has occurred.

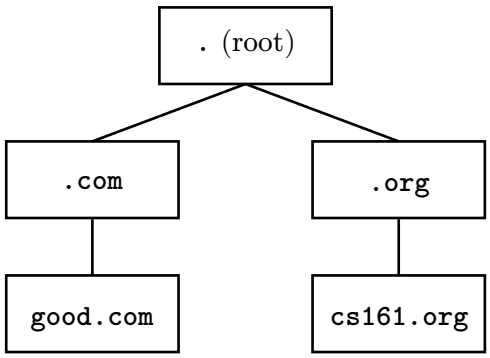
(A) is false because the student did not send any JavaScript. For a reflected XSS attack, the student would have supplied some malicious JavaScript in the URL, e.g. `http://cs161.berkeley.edu/search?q=<script>alert(1)</script>`.

(D) is false because a CSRF attack involves unintended authentication from the student's browser automatically attaching an authentication cookie. In this attack, no authentication is described, so a CSRF attack is not the best answer.

This content is protected and may not be shared, uploaded, or distributed.

Q2 *Networking: GooDNS* 🤫 **(9 points)**

Consider this DNS hierarchy, where each box represents a name server:



EvanBot has the following records cached:

Record 1:	`org.`	`NS`	`a.org-servers.net`
Record 2:	`a.org-servers.net`	`A`	`192.7.14.21`
Record 3:	`com.`	`NS`	`a.com-servers.net`
Record 4:	`a.com-servers.net`	`A`	`192.6.16.161`
Record 5:	`evil.com`	`A`	`192.5.55.555`

In Q2.1 to Q2.3, each subpart continues on from previous subparts, i.e. records received in one subpart can be cached for later subparts.

Q2.1 (1 point) How many DNS requests does EvanBot need to make to learn the IP address of `www.cs161.org`?

○ 0 ● 2 ○ 3

○ 1

Solution: One request to learn the IP address of the `cs161.org` name server, and another request to ask this name server the IP address of `www.cs161.org`.

Q2.2 (1 point) Record 1 is expired and removed from the cache.

How many DNS requests does the EvanBot need to make to learn the IP address of `www.cs161.org`?

● 0 ○ 2 ○ 3

○ 1

Solution: The `www.cs161.org` record is still stored within the cache, so it is unnecessary to make queries to try to obtain it again. As such, 0 queries are needed.

This content is protected and may not be shared, uploaded, or distributed.

Q2.3 (1 point) How many DNS requests does EvanBot need to make to learn the IP address of `not.good.com`?

○ 0 ● 2 ○ 3

○ 1

Solution: Two queries are needed. The resolver first goes to the .com name server address it has cached and makes a request, obtaining the IP address of the good.com name server. This is then accessed in a second request to obtain the IP address of not.good.com.

The rest of the question is independent of earlier subparts.

Q2.4 (1 point) Which of these best describes why an attacker would use the Kaminsky attack, instead of some other cache poisoning attack?

○ The attacker is on-path; the Kaminsky attack only works for on-path attackers.

○ Unlike other DNS attacks, the Kaminsky attack can poison a cache shared by many users.

○ The attacker is off-path; the Kaminsky attack guarantees that the attacker will guess correctly.

● The attacker is off-path; the Kaminsky allows the attacker to make more guesses.

Q2.5 (2 points) Suppose the attacker can place HTML on a website that the victim will visit.

Which HTML snippets can help the attacker poison the cache for `www.google.com` (using the Kaminsky attack)? Select all that apply.

■ `<img src="false1.google.com">`

■ `<img src="false1.google.com/image.png">`

☐ `<img src="false1.cs161.org">`

☐ `<img src="www.google.com">`

○ None of the above

Solution: The intended solution is listed above. However, we did not specify whether Bailiwick checking was enabled, so we awarded points whether or not option C was selected.

Q2.6 (1 point) When source port randomization is enabled, what is the approximate probability that an off-path attacker successfully spoofs a DNS response?

○ $1/2^{16}$ ○ $1/2^{64}$ ○ 1 ○ 0

● $1/2^{32}$

Solution: There are randomized 16-bit values that must be guessed, the ID and the source port. There is a $\frac{1}{2^{32}}$ chance of guessing both correctly.

This content is protected and may not be shared, uploaded, or distributed.

Q2.7 (1 point) When executing a Kaminsky attack, what should be the source IP in the attacker's spoofed DNS response?

- ○ Attacker's IP address
- ○ Resolver's IP address
- ● Name server's IP address
- ○ The source IP field can be left blank.

Q2.8 (1 point) What is the primary reason DNSSEC does not provide confidentiality?

- ○ The trust anchor model used by DNSSEC is incompatible with encryption protocols.
- ○ Confidentiality would make the DNS query process too slow.
- ● DNS data is considered public information.
- ○ Implementing encryption would require a new set of DNS record types, which is not feasible.

Solution: DNS is a public, distributed database. Its function is to provide public mappings of names to IP addresses, so encrypting this data would contradict its core purpose.

This content is protected and may not be shared, uploaded, or distributed.