201300180 Data & Information – Test 4 ANSWERS

17 June 2016, 13:45 – 15:15

Question 1 (Security) (40 points)

a) The code is vulnerable to (reflected) Cross-Site Scripting since user input is reflected back without sanitization.An attacker could trick a victim into visiting a crafted URL triggering the XSS in order to execute a malicious script allowing them to for example steal a user's cookie and hijack their session.

b) The code is vulnerable to SQL injection since user input to the query is not sanitized nor parameterized/prepared.
The above can be exploited in **multiple ways** to log in as *admin.* Examples are:

- By crafting a query which ignores the password via commenting: e.g. setting the *user* field to be
    - `admin' -- -`
- By crafting a completely tautological query: e.g., setting the *user* field to be
    - `(anything)' OR 1 = 1 -- -`
- …

SQL injection can be prevented by using prepared statements.

c) The approach is insufficiently secure because it stores credential hashes without salting them thus allowing an attacker to perform a cost-effective Time-Memory Tradeoff (TMTO) attack (e.g., using Rainbow Tables).
Ideally these credentials should be stored using a memory-hard key derivation function / password hashing scheme.

d) The function is insecure because it uses a non-cryptographically secure pseudorandom number generator (PRNG), namely the default Java PRNG which is then seeded with the user id and timestamp which are known to the attacker (or, in the case of the timestamp, can be easily bruteforced since we have approximate knowledge of the value with a session being created at most 24 hours ago), thus violating the secrecy of the PRNG seed.
An attacker can thus trivially obtain the session token for any user (provided they know the userid) by simply executing the above function with the ID.

Question 2 (From and to XML) (30 points)

WARNING: The answers below are not complete, i.e., there may be many more correct and almost correct answers. Furthermore, the assessment is not based on a number of points per sub-question, but on the level of mastery on a certain number of aspects.

a)

```
SELECT
   XMLELEMENT(NAME language,
      XMLATTRIBUTES(l.language AS name, COUNT(*) AS count)
   ) AS xml
FROM language AS l, movie AS m
WHERE l.mid = m.mid
GROUP BY l.language
```

b)

```
SELECT
   XMLELEMENT(NAME language,
      XMLATTRIBUTES(l.language AS name, COUNT(*) AS count),
      XMLAGG(
         XMLELEMENT(NAME movie,
            XMLATTRIBUTES(m.year AS year),
            m.name
         )
      )
   ) AS xml
FROM language AS l, movie AS m
WHERE l.mid = m.mid
GROUP BY l.language
```

c) Since there is a '*' inbetween languages and language in the DTD graph as well as between language and movie, there are 3 tables. name and count can be inlined into language and year into movie. Every table needs an ID. language and movie need a reference to the parent.

```
CREATE TABLE languages(
     id INT PRIMARY KEY,
);

CREATE TABLE language(
     id INT PRIMARY KEY,
     parent INT REFERENCES languages(id),
     name TEXT NOT NULL,
     count TEXT NOT NULL
);

CREATE TABLE movie(
     id INT PRIMARY KEY,
     parent INT REFERENCES language(ID),
     year TEXT NOT NULL
);
```

d) Note that text nodes are separate nodes, but that an attribute with its value is one node. You can also start counting from 0.

pre	post	level	kind	name	value
1	10	1	elem	languages	
2	9	2	elem	language	
3	1	3	attr	name	Spanish
4	2	3	attr	count	2
5	5	3	elem	movie	
6	3	4	attr	year	1999
7	4	4	text		Sixth Sense, The
8	8	3	elem	movie	
9	6	4	attr	year	1999
10	7	4	text		Todo sobre mi madre

e) The query is *//movie[@year="1999"]/parent::language*

```
SELECT DISTINCT l.pre
FROM edge r, edge m, edge y, edge l
WHERE r.pre = 1                            (root)
  AND m.pre > r.pre AND m.post < r.post    (descendant)
  AND m.kind = ‘elem’ AND m.name=’movie’
  AND y.pre > m.pre AND y.post < m.post
  AND y.level = m.level+1                  (child)
  AND y.kind = ‘attr’ AND y.name = ‘year’
  AND l.pre < m.pre AND l.post > m.post
  AND l.level = m.level-1                  (parent)
  AND l.kind = ‘attr’ AND l.name = ‘language’
ORDER BY l.pre
```

Question 3 (Information Retrieval & Full-Text search) (30 points)

a) YES. The full text search capabilities are case insensitive and remove punctuation. Also, they search through all descendants, not just the children.

b) idf(finds) = log (2/2) = log (1) = 0
idf(only) = log (2/1) = log (2) is approximately 0,3.

tf(776,finds) = 1; tf(776,only) = 1
tf(586,finds) = 1; tf(586,only) = 0

Rank(776) = tf(776,finds) * idf(finds) + tf(776,only) * idf(only) = 1*0 + 1*0,3 = 0,3
Rank(586) = tf(586,finds) * idf(finds) + tf(586,only) * idf(only) = 1*0 + 0*0,3 = 0

c) It is a document which should receive a higher rank if the rating is higher. Any factor based on the rating is correct, for example,

$$\text{Rank}(d,q) = \textstyle\sum_{t \in q} \text{tf}(d,t)\ \text{idf}(t)\ (d.\text{rating} / 10)$$

d) P(finds,586) = 1/22 (one occurrence from 22 words)

e) Most important here is to multiply the probabilities for both search terms.
P(D|T) = P(T|D) P(D) / P(T) … we assume P(D) and P(T) to not influence the ranking

P(D=776 | T=finds,only) = P(D=776 | T=finds) P(D=776 | T=only) = 1/21 * 1/22 = 1/462
P(D=586 | T=finds,only) = P(D=586 | T=finds) P(D=586 | T=only) = 1/21 * 0/22 = 0

f) Since we like to find some movies (= documents) higher than others, it means P(D) is not assumed uniform anymore and we should incorporate the rating in P(D). In other words, that our monkeys who are randomly picking movies, are more likely to pick a movie with high ranking.