Πανεπιστήμιο Αιγαίου

Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων

321-3404 Ασφάλεια Πληροφοριακών και Επικοινωνιακών Συστημάτων

Διδάσκων: Καρύδα Μαρία , Καμπουράκης Γεώργιος

Επίθεση σε Padding Oracle & Παραγωγή Αυθεντικοποιημένης Κρυπτογράφησης

Εργαστηριακοί Συνεργάτες : Νίκος Αλεξίου & Αλέξανδρος Φακής

Μέλη Ομάδας

Αντωνιάδης Χαράλαμπος icsd10011

Ευκαρπίδης Κωνσταντίνος icsd15051

Ζιώζας Γεώργιος icsd15058

Σάμος, 18/5, 2018

Αντωνιάδης Χαράλαμπος icsd10011 , Ευκαρπίδης Κωνσταντίνος icsd15051 , Ζιώζας Γεώργιος icsd15058

Περιεχόμενα

Αντωνιάδης Χαράλαμπος icsd10011 , Ευκαρπίδης Κωνσταντίνος icsd15051 , Ζιώζας Γεώργιος icsd15058

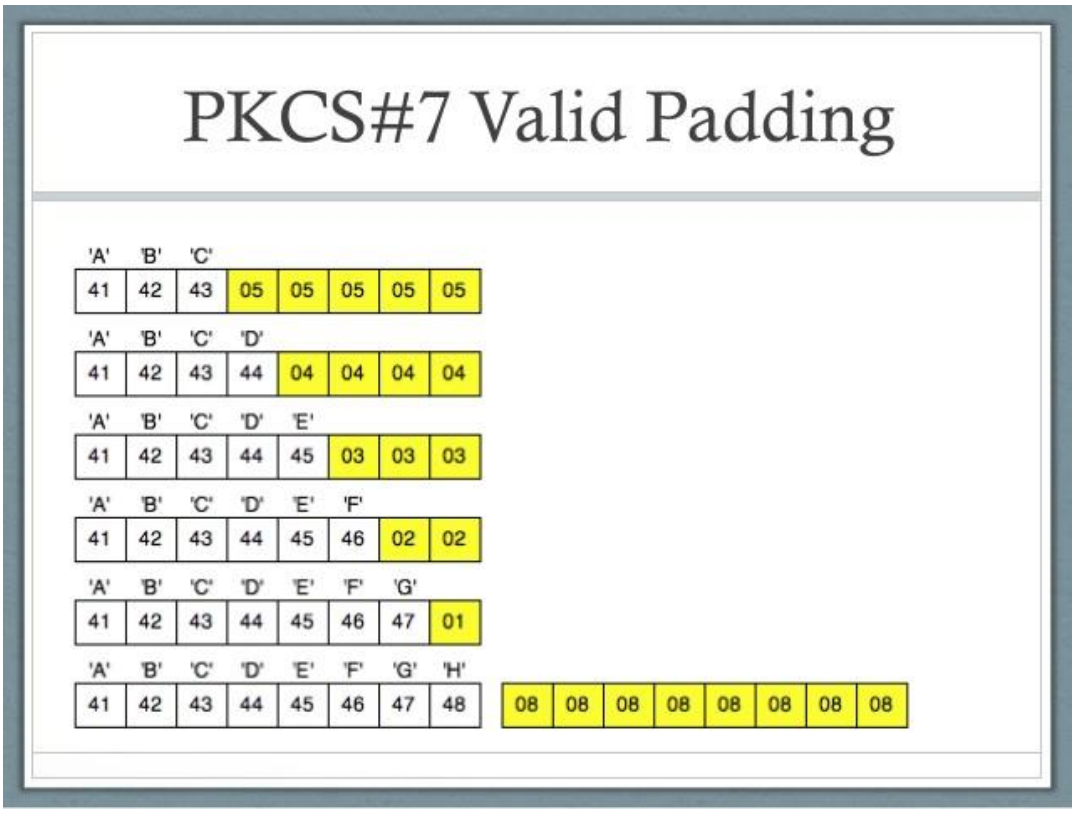
1 Εισαγωγή

Στην τρίτη εργαστηριακή άσκηση καλούμαστε να εξετάσουμε τι είναι ένας padding oracle server και πως θα επιτύχουμε επίθεση σαυτόν. Ο αλγόριθμος κρυπτογράφησης που χρησιμοποιείτε είναι AES σε CBC mode. Αφού επιτύχουμε την επίθεση και αποκρυπτογραφήσουμε το κρυπτοκείμενο , πρέπει να κάνουμε την αντίθετη διαδικασία και να ξανακρυπτογραφήσουμε το αρχικό κείμενο εξασφαλίζοντας όμως την αυθεντικότητα του. Συνεπώς σκοπός της εργασίας είναι η εξοικείωση με το crypto API της JAVA και του AES-CBC αλγορίθμου.

2 Επίθεση σε Padding Oracle

2.1 Padding (PKCS7)

Ο αλγόριθμος κρυπτογράφησης AES χωρίζει το προς κρυπτογράφηση περιεχόμενο σε block μεγέθους 16 bytes , δηλαδή 128 bits. Υπάρχει όμως και η περίπτωση όπου το περιεχόμενο δεν χωράει ακριβώς σε αυτά τα μπλοκ και στο τελευταίο απομένουν περίσσιες θέσεις. Κάπου εδώ εμφανίζεται και το padding το οποίου ο σκοπός είναι γεμίσει τις κενές θέσεις με τον αριθμό των κενών θέσεων. Δηλαδή αν για παράδειγμα είχαμε ένα κείμενο μήκους 140 bytes μας περισσεύουν 4 bytes. Συνεπώς η διαδικασία του padding θα είχε ως αποτέλεσμα τις τελευταίες τέσσερις θέσεις του τελευταίου block να περιείχαν την τιμή "**04**". Παρακάτω στην εικόνα φαίνεται η συμπλήρωση. Επίσης παρατηρούμε ότι στην περίπτωση που τμηματοποιείτε η συμβολοσειρά ακριβώς σε block των 16 bytes ,πρέπει πάλι να προστεθεί padding.



Εικόνα 2.1.1 : Αφορά αλγορίθμους με είσοδο 8 bytes

Αυτό που καθιστά έναν server ως Padding Oracle είναι ότι έχει την λειτουργία να απαντάει με δύο μηνύματα ανάλογα με το αν το padding που του στέλνουμε είναι έγκυρο ή μη. Στην περίπτωση όπου είναι άκυρο απαντάει με "**HTTP/1.1 403 Forbidden**" , ενώ σε αντίθετη περίπτωση με "**HTTP/1.1 404 Not Found**".

2.2 AES – CBC mode

Ως CBC ορίζεται Cipher Block Chaining. Αυτό σημαίνει αλυσίδα κρυπτογραφημένων μπλοκ. Συγκεκριμένα κάθε μπλοκ απλού κειμένου γίνεται XOR με το προηγούμενο κρυπτογραφημένο μπλοκ ,πριν αυτό κρυπτογραφηθεί. Συνεπώς καταλαβαίνουμε ότι κάθε "cyphertext" εξαρτάται από όλα τα μπλοκ απλού κειμένου που έχουν δουλευτεί μέχρι εκείνο το σημείο. Δηλαδή υπάρχει αλληλουχία μεταξύ των άλλων μπλοκ.

2.3 Υλοποίηση επίθεσης

Για το κομμάτι της επίθεσης στον Oracle Padding ολοκληρώσαμε μία κλάση με όνομα "**PaddingOracle**", η οποία αποτελείτε από τέσσερα τμήματα. Αρχικά μέσω της συνάρτησης "**createBlock()**" γίνεται η τμηματοποίηση του αρχικού κειμένου σε blocks, αφού μετατραπεί πρώτα , για κάθε μπλοκ, σε bytes. Αν το cyphertext δεν είναι ακέραιο πολλαπλάσιο του μεγέθους των block (16 bytes) , τότε έχουμε πρόβλημα. Διότι από την στιγμή που κάνουμε επίθεση για αποκρυπτογράφηση θεωρούμε ότι η κρυπτογράφηση ολοκληρώθηκε σωστά και έχει προστεθεί το padding. Αφού ολοκληρωθεί λοιπόν καλείτε η επόμενη συνάρτηση "**getSolution()**", της οποίας το πρώτο μέρος είναι με μία επανάληψη να διατρέξει τα διαθέσιμα μπλοκ και να τα χρησιμοποιήσει ως παραμέτρους για την κλήση της συνάρτησης "**crackBlock()**". Αυτή με την σειρά της θα τρέξει από το τέλος του μπλοκ που μπήκε ως $1^{η}$ παράμετρος στην συνάρτηση μας μέχρι το $1^{ο}$ στοιχείο το πολύ 256 φορές για το κάθε byte. Κάθε φορά μέσα σαυτήν την επανάληψη αλλοιώνει το byte προς αποκρυπτογράφηση και στέλνει ως αίτημα το λινκ του σέρβερ συν τα δύο μπλοκ με τις αλλαγές ως ένα string. Η αποστολή του αιτήματος έχει οριστεί από επιλογή μας να γίνεται μέσω της συνάρτησης "**getCode()**" (Οι απαντήσεις του εξυπηρετητή αναφέρονται στην Ενότητα 2.1) .Όσο παίρνει απάντηση 403 από τον σέρβερ η επανάληψη συνεχίζει. Αφού βρει το σωστό byte (Απάντηση 404) τότε κάνει την διαδικασία με τα **XOR** που περιγράφεται στην εκφώνηση της άσκησης και αποθηκεύει το κομμάτι της λύσης στον πίνακα **crypto_solution** και μετέπειτα τροποποιείται κατάλληλα το περιεχόμενο του **temp_array** προκειμένου να συνεχίσει στο επόμενο byte της αποκρυπτογράφησης. Μόλις τελειώσουν τα διαθέσιμα block προς αποκρυπτογράφηση θα τελειώσει η επανάληψη της συνάρτησης getSolution και απομένει ένα τελευταίο κομμάτι, η αφαίρεση του padding ώστε να εμφανιστεί το αρχικό μήνυμα. Για την διαδικασία αυτή έχουμε ορίσει ένα πίνακα με όνομα **paddings** ,περιέχει όλα τα πιθανά paddings, τον οποίο διατρέχουμε με μία επανάληψη ώστε να διασταυρώσουμε με ποιο από αυτά τελειώνει το τελικό string.

2.4 Στιγμιότυπα εκτέλεσης

2.4.1 Επίθεση στον Server

```
run:
128
f20bdba6ff29eed7b046d1df9fb70000
58b1ffb4210a580f748b4ac714c001bd
4a61044426fb515dad3f21f18aa577c0
bdf302936266926ff37dbf7035d5eeb4
inputBlock    58b1ffb4210a580f748b4ac714c001bd    previous    f20bdba6ff29eed7b046d1df9fb70000
```

Εικόνα 2.4.1.1 : Τμηματοποίηση κρυπτοκειμένου & block προς εξέταση

```
HTTP/1.1 403 Forbidden
HTTP/1.1 403 Forbidden
HTTP/1.1 403 Forbidden
HTTP/1.1 404 Not Found
[B@5c647e05
Bit found: 21 in 00000000000000000000000000002158b1ffb4210a580f748b4ac714c001bd
Crypto Solution: 00000000000000000000000000000020 /
Set almost valid padding to 2 in 00000000000000000000000000002258b1ffb4210a580f748b4ac714c001bd
```

Εικόνα 2.4.1.2 : Αρχή επίθεσης | Εύρεση τελευταίου byte 2ου block

```
HTTP/1.1 403 Forbidden
HTTP/1.1 403 Forbidden
HTTP/1.1 403 Forbidden
HTTP/1.1 404 Not Found
[B@5c647e05
Bit found: 71 in 00000000000000000000000000712258b1ffb4210a580f748b4ac714c001bd
Crypto Solution: 00000000000000000000000000007320 / s
Set almost valid padding to 3 in 00000000000000000000000000702358b1ffb4210a580f748b4ac714c001bd
```

Εικόνα 2.4.1.3 : Εύρεση προτελευταίου byte 2ου block

```
Bit found: 29 in 29187226578c2444b42638e893bc6ed9bdf30293626626926ff37dbf7035d5eeb4
Crypto Solution: 73696672616765090909090909090909 / sifrage
Decrypted message : The Magic Words are Squeamish Ossifrage
BUILD SUCCESSFUL (total time: 18 minutes 46 seconds)
```

Εικόνα 2.4.1.4 : Αποκρυπτογράφηση τελευταίου block & εμφάνιση plaintext

2.4.2 Παραγωγή αυθεντικοποιημένης κρυπτογράφησης & αδυναμία επαλήθευσης

Για την προβολή της λειτουργίας έχουμε βάλει να εμφανίζεται το κρυπτογραφημένο μήνυμα σε περίπτωση που οι MAC συμφωνούν ,ενώ σε άλλη περίπτωση εμφανίζεται κατάλληλο μήνυμα.

```
run:
Key1 : javax.crypto.spec.SecretKeySpec@fffe8c16
Key2 : javax.crypto.spec.SecretKeySpec@fffe873d
Encrypted text's length : 48
Final hash's length : 64
MAC's length : 32
Final's block size : 96
Decrypted message : The Magic Words are Squeamish Ossifrage
BUILD SUCCESSFUL (total time: 0 seconds)
```

Εικόνα 2.4.2.1 : Συμφωνία MAC

```
run:
Key1 : javax.crypto.spec.SecretKeySpec@fffe846f
Key2 : javax.crypto.spec.SecretKeySpec@fffe81d9
Encrypted text's length : 48
Final hash's length : 64
MAC's length : 32
Final's block size : 96
Sender MAC and calculated MAC are different. Message cannot be authenticated.
BUILD SUCCESSFUL (total time: 0 seconds)
```

Εικόνα 2.4.2.2 : Ασυμφωνία MAC

3 Ερωτήματα

3.1 Είναι απαραίτητη η αποκρυπτογράφηση του πρώτου block του κρυπτοκειμένου; Αν ναι, γιατί ;

Όχι δεν είναι απαραίτητη η αποκρυπτογράφηση του ,διότι δεν περιέχει πληροφορία σχετική με τον κείμενο μας ,απλά βοηθάει στην αποκρυπτογράφηση του δεύτερου μπλοκ όπου ξεκινάει και η πληροφορία. Από αναζήτηση που κάναμε στο διαδίκτυο[1] συμπεράναμε ότι στο πρώτο block εμπεριέχεται το IV (Initialization Vector).

3.2 Ποιος είναι ο μέγιστος αριθμός των «http requests» που χρειάζονται ώστε να αποκρυπτογραφηθεί ένα byte του plaintext στη συγκεκριμένη επίθεση;

Ο μέγιστος αριθμός των αιτημάτων προς απάντηση από τον server που χρειάζονται για την αποκρυπτογράφηση ενός byte είναι 256. Αυτό συμβαίνει επειδή κάθε φορά που στέλνει ένα αίτημα ελέγχει ένα byte του plaintext, το οποίο στο cyphertext αναπαρίσταται από δύο ψηφία(byte) δεκαεξαδικών αριθμών. Συνεπώς 16 x 16 = 256 (όλοι οι πιθανοί συνδυασμοί).

3.3 Θα ήταν δυνατή η επίθεση αν ο Server χειριζόταν με τον ίδιο τρόπο τα σφάλματα κατά την αποκρυπτογράφηση (μη έγκυρο Padding) με τα σφάλματα που προκύπτουν από ένα αλλοιωμένο plaintext;

Σε καμία περίπτωση δεν θα ήταν δυνατή η πραγματοποίηση της επίθεσης. Εάν ο εξυπηρετητής διαχειριζόταν όλα τα σφάλματα με τον ίδιο τρόπο , εμείς ως επιτιθέμενος δεν θα γνωρίζαμε ποτέ την εύρεση του σωστού αλλοιωμένου byte του plaintext. Συνεπώς να μην είναι δυνατή η εκκίνηση της αποκρυπτογράφησης

3.4 Θα ήταν δυνατή η επίθεση αν είχε χρησιμοποιηθεί AES σε CTR mode;

Το cyphertext που μας δίνεται έχει κρυπτογραφηθεί σε CBC mode. Αυτό σημαίνει ότι κάθε block εξαρτάται το προηγούμενό του για να αποκρυπτογραφηθεί από. Εκεί βασίζεται η επίθεση μας, καθώς εμείς έχουμε ολόκληρο το cyphertext και μπορούμε να βρούμε όλες τις συνιστώσες του XOR για να το αποκρυπτογραφήσουμε. Το CTR mode δεν είναι block chain και η αποκρυπτογράφηση του εξαρτάται από ένα διαφορετικό counter για κάθε block. Συνεπώς μας λείπει μια βασική παράμετρος για να γίνει δυνατή η αποκρυπτογράφηση του cyphertext.

3.5 Αν η αυθεντικοποίηση του κρυπτοκειμένου γινόταν με το σχήμα MAC-then-Encrypt κι όχι με το Encrypt-then-MAC που αναφέρθηκε στα ζητούμενα, θα μπορούσε ο Server να παρέχει ασφάλεια; Αν ναι, με ποιόν τρόπο;

Το μοντέλο MtE στην περίπτωση μας δεν είναι το ίδιο ασφαλές με το EtM, επειδή το πρώτο μοντέλο πρώτα αποκρυπτογραφεί το cyphertext και μετέπειτα ελέγχει την εγκυρότητα του αποστολέα. Αντίθετα το δικό μας μοντέλο θα ελέγξει αρχικά την εγκυρότητα του αποστολέα και αφού εγκριθεί θα ξεκινήσει η διαδικασία της αποκρυπτογράφησης. Παρόλα αυτά το MtE θα μπορούσε να παρέχει ασφάλεια εφόσον υπάρχει ένα «κρυμμένο» κρυπτογραφημένο block το οποίο θα αύξανε την εντροπία της κρυπτογράφησης,

3.6 Ποια σχέση συνδέει το plaintext με το κρυπτοσύστημα RSA ;

Η φράση που χρησιμοποιήθηκε στης άσκηση μας ως plaintext ήταν η λύση σε ένα από τους πρώτους διαγωνισμούς κρυπτογραφίας, τον οποίο διοργάνωσαν οι δημιουργοί του RSA κρυπτοσυστήματος. Το πρόβλημα λύθηκε αρκετά αργότερα χρησιμοποιώντας τεράστια υπολογιστική ισχύ για την τότε εποχή με συντονισμένη προσπάθεια μέσω του διαδικτύου.

4 Βιβλιογραφία

~T2d. “Implementation of Padding Oracle in Java”. 2016. Link

~Itarato. “AES CBC encryption”. 2014. Link

~Paŭlo Ebermann. “Conversations about CBC”. 5th November 2011. Link

~Bellare, M.; Namprempre, C. “Authenticated Encryption”. 2000. Link

~Wikipedia. “The Magic Words are Squeamish Ossifrage”. Link

~Kyle Brantley. “MtE or EtM ?”. 5th July 2017. Link

~Wikipedia. “Block cipher mode of operation”. Link