ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ - ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ
ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΑΣΦΑΛΕΙΑ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΤΕΥΧΟΣ ΕΡΓΑΣΤΗΡΙΑΚΩΝ ΑΣΚΗΣΕΩΝ

Γεώργιος Καμπουράκης

Αναπληρωτής Καθηγητής Τμήματος Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων

Μαρία Καρύδα

Επίκουρη Καθηγήτρια Τμήματος Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων

Αλέξανδρος Φακής, Νικόλαος Αλεξίου

Υποψήφιοι Διδάκτορες Τμήματος Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων

Άσκηση 3: Επίθεση σε Padding Oracle και παραγωγή Αυθεντικοποιημένης Κρυπτογράφησης.

Εισαγωγή

Σκοπός της εργασίας είναι αφενός να υποδείξει τη σωστή εφαρμογή των θεμελιακών στοιχείων (building blocks) της κρυπτογραφίας και αφετέρου να σας φέρει σε πρώτη επαφή με το κρυπτογραφικό API της Java.

Στα πλαίσια της εργασίας καλείστε αρχικά να εφαρμόσετε μια padding oracle επίθεση σε AES-CBC, έπειτα να αποκαλύψετε το αρχικό κείμενο (plaintext) από το κρυπτοκείμενο (ciphertext) και τέλος να κρυπτογραφήσετε το plaintext εξασφαλίζοντας όμως την αυθεντικότητά του.

Περιγραφή εργασίας

Η κρυπτογράφηση ενός μηνύματος δεν εξασφαλίζει την ακεραιότητα του παραγόμενου κρυπτοκειμένου. Ένας επιτιθέμενος έχει υπό συνθήκες τη δυνατότητα να εφαρμόσει μια chosen ciphertext επίθεση κατά την οποία, αλλοιώνοντας το κρυπτοκείμενο, πετυχαίνει την αποκρυπτογράφησή του χωρίς να έχει γνώση του κλειδιού.

Σενάριο

Στην περίπτωσή μας ο επιτιθέμενος έχει στόχο να υποκλέψει πληροφορίες από τον ιστότοπο crypto-class.appspot.com. Κατά τη διάρκεια μιας συνόδου του εξυπηρετητή με έναν πελάτη, ο επιτιθέμενος κατέγραψε το αίτημα http://crypto-class.appspot.com/po?er=f20bdba6ff29eed7b046d1df9fb7000058b1ffb4210a580f748b4ac714c001bd4a61044426fb515dad3f21f18aa577c0bdf302936266926ff37dbf7035d5eeb4 και υποψιάζεται ότι το τμήμα μετά το po?er= αποτελεί κρυπτογράφημα ενός κωδικοποιημένου σε δεκαεξαδικό αρχικού μηνύματος το οποίο έχει κρυπτογραφηθεί με AES σε CBC mode of operation. Έπειτα ο επιτιθέμενος, στέλνοντας το ίδιο αίτημα στον εξυπηρετητή με αλλοιωμένο κρυπτοκείμενο, διαπίστωσε ότι όταν η αποκρυπτογράφησή είχε ως αποτέλεσμα ένα μη έγκυρο padding, η απάντηση του server ήταν 403 (forbidden request). Όταν το padding ήταν έγκυρο αλλά το περιεχόμενο του μηνύματος ήταν αλλοιωμένο, ο server επέστρεφε 404 (URL not found).

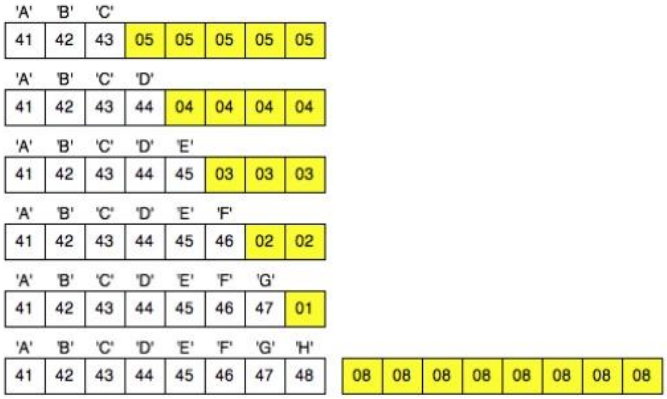
Στην παραπάνω περίπτωση ο server αποτελεί ένα padding oracle και καλείστε αλλάζοντας το κρυπτοκείμενο να αποκαλύψετε το αρχικό περιεχόμενο του μηνύματος. Με κάθε επιτυχημένο http request θα αποκαλύπτετε κι ένα byte από το plaintext. Το αρχικό μήνυμα είναι κωδικοποιημένο σε ASCII.

Ανάλυση της επίθεσης

Padding

Είναι γνωστό ότι ο AES ως block cipher κρυπτογραφεί τα δεδομένα σε τμήματα. Όταν το μήκος του plaintext δεν είναι ακέραιο πολλαπλάσιο του block (128 bit ή 16 bytes) τότε πρέπει να προστεθεί στο τέλος του το κατάλληλο padding. Ο τρόπος σχηματισμού του padding στον AES-CBC ορίζεται από το standard PKCS7 και είναι ουσιαστικά ίδιος με αυτόν που περιγράφεται στο PKCS5 (RFC2898). Ο τελευταίος αφορούσε block size των 8 bytes του DES. Συγκεκριμένα το padding αποτελείται από bytes τα οποία περιέχουν ως τιμή το πλήθος των bytes που χρησιμοποιούνται για το padding[i].

Ως παράδειγμα ας θεωρήσουμε το plaintext “Hello”. Κωδικοποιημένο σε ASCII (1 byte ανά χαρακτήρα) αντιστοιχεί στα 5 bytes 48656C6C6F (σε δεκαεξαδικό). Η κρυπτογράφηση του θα χρειαζόταν 11 bytes για padding. Αφού το 11 αντιστοιχεί σε *0x*0B στο δεκαεξαδικό, το block που θα χρησιμοποιούταν ως *είσοδο* στον AES-CBC θα είχε τη μορφή 48656C6C6F0B0B0B0B0B0B0B0B0B0B. (Η *έξοδος* του AES με 128-bit κλειδί *0x*A456B7A422C5145ABCF2B3CB206579A8 θα ήταν 42FF63CC06D53DA93F24389723E1611A)[ii].

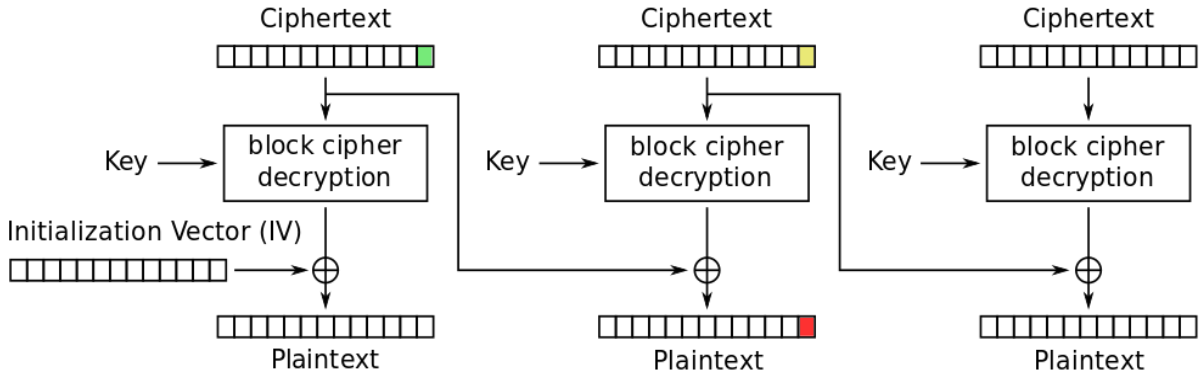


Εικόνα 1: PKCS7 padding. [Πηγή: Stack Overflow]

Padding Oracle Attack

Όταν ο εξυπηρετητής που θα χρησιμοποιήσετε για την εργασία σας αποκρυπτογραφεί το κρυπτοκείμενο ελέγχει αν το padding είναι έγκυρο και αποκρίνεται με διαφορετικό τρόπο αν είναι άκυρο. Αυτό τον καθιστά ένα padding oracle.

Για την περιγραφή της επίθεσης είναι απαραίτητη η γνώση του τρόπου με τον οποίο γίνεται η αποκρυπτογράφηση στον AES σε CBC mode. Όπως φαίνεται στην εικόνα 2, για να παραχθεί ένα block του plaintext, η αποκρυπτογράφηση του αντίστοιχου block του ciphertext γίνεται xor με το προηγούμενο block του ciphertext.



Εικόνα 2: Αποκρυπτογράφηση AES-CBC [Πηγή: Wikipedia]

Έστω ότι ο επιτιθέμενος στοχεύει στην αποκρυπτογράφηση του τελευταίου byte C2 του δεύτερου block που σημειώνεται με κίτρινο χρώμα. Αλλοιώνοντας το byte C1 που βρίσκεται στην αντίστοιχη θέση του ciphertext ένα block πριν (πράσινο) και δίνοντάς του την τιμή C1', καταφέρνει να ξεγελάσει το server παράγοντας το έγκυρο pad *0x*01 για εκείνη τη θέση. Ο επιτιθέμενος μαθαίνει μέσω της απόκρισης του server ότι το ciphertext κατά την αποκρυπτογράφηση έδωσε έγκυρο pad. Τότε μπορεί να υπολογίσει το plaintext ως εξής. Στην κανονική περίπτωση αποκρυπτογράφησης θα είχαμε:

$$D(C2) \oplus C1 = P \Leftrightarrow D(C2) = C1 \oplus P \ (1)$$

όπου με D εννοούμε decryption.

Με την αλλοίωση του ciphertext έχουμε:

$$D(C2) \oplus C1' = 0x01 \Rightarrow (1)\ C1 \oplus P \oplus C1' = 0x01 \Rightarrow P = C1 \oplus C1' \oplus 0x01$$

Με αυτόν τον τρόπο ο επιτιθέμενος αποκρυπτογραφεί το τελευταίο byte. Στη συνέχεια θα πρέπει να αλλοιώσει ξανά το κρυπτοκείμενο ώστε να παραχθεί κατά την αποκρυπτογράφηση το έγκυρο pad *0x*0202 και να αποκρυπτογραφήσει το προηγούμενο σε σειρά byte.

Ζητούμενα

I. Σκοπός σας είναι η *συγγραφή ενός προγράμματος σε Java* το οποίο θα υλοποιεί την επίθεση που περιγράφηκε στα προηγούμενα.
II. Αφού αποκαλύψετε το plaintext καλείστε να χρησιμοποιήσετε το crypto API της Java προκειμένου να δημιουργήσετε μια *αυθεντικοποιημένη κρυπτογράφηση* (Authenticated Encryption ή A.E.) του plaintext. Για να το πετύχετε δημιουργήστε δύο κλειδιά k1 και k2 με ασφαλή τρόπο. Κρυπτογραφήστε το plaintext με κλειδί k1 κι έπειτα χρησιμοποιήστε ξεχωριστό κλειδί k2 για δημιουργήστε ένα κώδικα αυθεντικότητας (MAC) του κρυπτοκειμένου. Το σχήμα αυτό ονομάζεται στη Encrypt-then-MAC.
III. Τέλος, δείξτε ότι αν ο server ήταν προγραμματισμένος ώστε να ελέγχει την αυθεντικότητα του ciphertext μέσω της επαλήθευσης του MAC τότε η padding oracle επίθεση δε θα μπορούσε να συμβεί.

Ερωτήσεις

1. Είναι απαραίτητη η αποκρυπτογράφηση του πρώτου block του κρυπτοκειμένου; Αν ναι, γιατί;
2. Ποιος είναι ο μέγιστος αριθμός των http requests που χρειάζονται ώστε να αποκρυπτογραφηθεί ένα byte του plaintext στη συγκεκριμένη επίθεση;
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;
4. Θα ήταν δυνατή η επίθεση αν είχε χρησιμοποιηθεί AES σε CTR mode;
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;
6. Τέλος, ποια σχέση συνδέει το plaintext με το κρυπτοσύστημα RSA;

Παραδοτέα

1. Ο κώδικας του προγράμματός σας σε Java. Πρέπει να περιέχει στοχευμένα σχόλια και να είναι αυτό-επεξηγηματικός όσον αφορά τη λειτουργία του, για παράδειγμα μέσω των ονομάτων των μεταβλητών που χρησιμοποιούνται ή/και της λειτουργικότητας των αντικειμένων που έχετε δημιουργήσει.
2. Αναφορά που περιέχει:
    α. Την αποκρυπτογράφηση του κρυπτοκειμένου.
    β. Στιγμιότυπα οθόνης τα οποία παρουσιάζουν
        i. την επίθεση στο server
        ii. την παραγωγή της αυθεντικοποιημένης κρυπτογράφησης
        iii. την αδυναμία επαλήθευσης του MAC όταν αλλοιώνεται το αυθεντικοποιημένο κρυπτοκείμενο
    γ. Απαντήσεις στα ερωτήματα που αναφέρθηκαν.
    δ. Σχόλια που αφορούν τις δυσκολίες που αντιμετωπίσατε.

Επισημάνσεις

Πρέπει να μετατρέψετε το plaintext από το δεκαεξαδικό σε ASCII προκειμένου να διαβάσετε το κρυμμένο μήνυμα.

Ενδεικτική Βιβλιογραφία

Understanding Cryptography, Paar & Pelzl, chapters 5.1, 12
Introduction to Modern Cryptography, Katz & Lindell, chapters: 4.8, 4.9
Java Security 2nd ed, Scott Oaks

[i] https://tools.ietf.org/html/rfc8018#appendix-B.2.5

[ii] https://www.di-mgt.com.au/cryptopad.html