Python 2.7.11 Secure Server - Client Application

ΑΠΟΣΤΟΛΟΣ - ΝΙΚΟΛΑΟΣ ΒΑΪΛΑΚΗΣ, Α.Μ. : 2014030174

1. ΕΙΣΑΓΩΓΗ

Το κείμενο αυτό συνοδεύει το 4ο project Ασφάλειας Πληροφοριακών Συστημάτων και αποτελεί αναφορά της εφαρμογής client-server ασφαλής επικοινωνίας που μας ζητήθηκε να αναπτύξουμε. Η εφαρμογή έχει ως σκοπό την επικοινωνία ανάμεσα σε δύο οντότητες (client και server) μέσω ενός συνδυασμού κρυπτογραφικών αλγορίθμων για την ασφάλεια των μηνυμάτων. Επίσης πέρα από την κρυπτογράφηση τους η εφαρμογή ελέγχει και για την ακεραιότητα των μηνυμάτων που ανταλλάσσονται με σκοπό την αποτροπή αλλοίωσης τους από τρίτους. Η λειτουργία της εφαρμογής περιορίζεται σε ένα μήνυμα το οποίο αποστέλλεται από τον Client στον Server, η απλή αυτή λειτουργία έχει ως σκοπό να εστιάσει κυρίως στις τεχνικές ασφαλούς επικοινωνίας οι οποίες με την σειρά τους μπορούν να εφαρμοστούν σε περιπλοκότερες* σχεδιάσεις.

Το zip που συνοδεύει αυτή η αναφορά αποτελεί project του PyCharm και εμπεριέχει :

- Final_A : Προσχέδιο Εφαρμογής Server-Client (Ερώτημα A_4).
- Final_B : Πλήρης Εφαρμογή Server-Client (Ερώτημα B).
  - -B_Server.py : ο Server της εφαρμογής.
  - -B_Client.py : ο Client της εφαρμογής.
- Functions.py : Μέθοδοι για τις αναγκαίες κρυπτογραφήσεις και αποκρυπτογραφήσεις του server

Η εφαρμογή χρησιμοποιεί τις εξής libraries :

PyCrypto : Για την Συμμετρική και Ασύμμετρη Κρυπτογράφηση/ Αποκρυπτογράφηση Μηνυμάτων

hashlib : Για τα κατάλληλα digest

os : Για την δημιουργία random αριθμών

socket : Για την επικοινωνία ανάμεσα σε Server και Client

* Και ποιο ρεαλιστικές

2. ΑΝΑΛΥΣΗ ΤΟΥ FUNCTIONS.PY

2.1 Padding and Un-padding

Οι συναρτήσεις pad() και unpad() χρησιμοποιούνται για την "ολοκλήρωση" των δεδομένων που πρόκειται να κρυπτογραφηθούν με αλγόριθμο PKCS1

2.2 Digest

Η μέθοδος digest() χρησιμοποιείται για την δημιουργία hash σε SHA-512

2.3 AES Symmetric Encryption and Decryption

Αρχίζουμε με την μέθοδο AESencrypt(). Αυτή η μέθοδος χρησιμοποιείται για την κρυπτογράφηση μηνυμάτων AES, σε mode ECB και padding PKCS5. Επίσης για την επιβεβαίωση ακεραιότητας των μηνυμάτων , προστίθενται σε αυτά το digest του κλειδιού κρυπτογράφησης, όπως και το μέγεθος του αρχικού μηνύματος. Πριν την τελική κρυπτογράφηση το εμπλουτισμένο plaintext που επρόκειτο να κρυπτογραφηθεί περνάει απο την μέθοδο pad() όπως και με την σειρά του το password της κρυπτογράφησης περνάει από το την ίδια διαδικασία[*]. Τέλος το κρυπτογραφημένο μήνυμα μετατρέπεται σε HEX για την ευκολότερη επεξεργασία του.
Για την αποκρυπτογράφηση του μηνύματος χρησιμοποιείται η μέθοδος *AESdecrypt()*. Αφού μετατρέψει και πάλι τα δεδομένα στην κατάλληλη μορφή τους, αποκρυπτογραφεί το μήνυμα ελέγχει αν τα δεδομένα που αποκρυπτογράφησε πληρούν τις κατάλληλες προδιαγραφές. Στην περίπτωση που βρεθεί λάθος στην ακεραιότητα των δεδομένων, η μέθοδος "σηκώνει" το exception 'Corrupted' με σκοπό την ενημέρωση της εφαρμογής για τυχόν διαστρέβλωση των δεδομένων.

2.4 Asymmetric Encryption and Decryption

Για την ασύμμετρη κρυπτογράφηση χρησιμοποιούνται τρεις συναρτήσεις. Οι RSAgenerate(), RSAencrypt() και RSAdecrypt().

2.4.1 RSAgenerate()

Η μεθοδος αυτή χρησιμοποιείται για την δημιουργία ενός private και public keypair RSA-2048 και ως ορίσματα δέχεται τα ονόματα των δύο αρχείων που επρόκειτο να στεγάσουν αυτά τα δύο κλειδιά.

2.4.2 RSAencrypt()

[*] Το κλειδί μπορει να είναι μέχρι 32 byte, οτιδήποτε μεγαλύτερο των 32byte περικόβεται στο κατάλληλο μέγεθος.

Εδώ τα παραπάνω κλειδιά* χρησιμοποιούνται για την κρυπτογράφηση ενός μηνύματος με τον αλγόριθμο RSA. Όπως και στις διαδικασίες συμμετρικής κρυπτογράφησης και αποκρυπτογράφησης αντίστοιχα, το αρχικό μήνυμα εμπλουτίζεται με το digest του κλειδιού κρυπτογράφησης, όπως και με το μέγεθος του αρχικού μηνύματος.

2.4.3 RSAdecrypt()

Όμοια με την συμμετρική αποκρυπτογράφηση, την αποκρυπτογράφηση των δεδομένων ακολουθεί ο έλεγχος της ακεραιότητας τους, σηκώνοντας το κατάλληλο exception στην περίπτωση προβλήματος με αυτούς τους ελέγχους

3. SERVER-CLIENT OPERATION

Ξεκινώντας τον Server, δημιουργείται ένα τυχαίο κλειδί από αυτόν. Έπειτα η λειτουργία της εφαρμογής χωρίζεται σε 3 διαφορετικές προσεγγίσεις ασφαλούς επικοινωνίας. Αυτές οι προσεγγίσεις είναι :

3.1 Χρήση Συμμετρικής Κρυπτογραφίας

Εδώ ο Client κρυπτογραφεί το μήνυμα μέσω συμμετρικής κρυπτογραφίας και το στέλνει στον Server

3.2 Χρήση Ασύμμετρης Κρυπτογραφίας

Εδώ ο Client κρυπτογραφεί το μήνυμα μέσω ασύμμετρης κρυπτογραφίας και το στέλνει στον Server

3.3 Χρήση Ασύμμετρης ΚΑΙ Συμμετρικής Κρυπτογραφίας

Εδώ ο Client κρυπτογραφεί αρχικά το μήνυμα μέσω συμμετρικής κρυπτογραφίας και το αποτέλεσμα κρυπτογραφείται με την σειρά του μέσω ασύμμετρης κρυπτογραφίας πριν το στείλει στον Server

-Αρχικά, αφού ο χρήστης έχει επιλέξει τον τρόπο κρυπτογράφησης που θέλει να χρησιμοποιήσει, ο Client ενημερώνει τον Server για την μέθοδο που πρόκειται να ακολουθήσει.

-Στις μεθόδους κρυπτογράφησης που εμπεριέχουν και συμμετρική, ο Server κρυπτογραφεί με ασύμμετρη κρυπτογράφηση το συμμετρικό κλειδί που θα χρησιμοποιήσει ο Client, και του το αποστέλλει.

* Για την ακρίβεια μόνο το public κομμάτι ενός keypair

-Και στις τρεις περιπτώσεις το μήνυμα προς αποστολή χωρίζεται σε μικρότερα κομμάτια για να επιτρέψει τον χρήστη να εισάγει απεριόριστου μήκους κείμενο.

-Οι server και client είναι σχεδιασμένοι να εκμεταλλεύονται τα πιθανά exceptions τόσο του Functions.py όσο και των PyCrypto και Socket για την διασφάλιση της ακεραιότητας των μηνυμάτων, φροντίζοντας να ενημερώνουν τον χρήστη για τυχόν προβλήματα στην επικοινωνία τους.

4. OPERATION NOTES

- Η εφαρμογή είναι σχεδιασμένη ώστε η εκτέλεση του Server να προηγείται της εκτέλεσης του Client.

- Στην περίπτωση ήδη κατειλημμένης PORT, στην αρχή των αρχείων Server.py και Client.py η σταθερά PORT μπορεί να αντικατασταθεί ανάλογα.

- Σε περίπτωση ανάγκης δημιουργίας νέων κλειδιών, κάντε uncomment τις δύο τελευταίες γραμμές του αρχείου Functions.py και εκτελέστε το.

5. TRANSMISSION

Για την μετάδοση του κρυπτογραφημένου μηνύματος, ο Client το αποδεκατίζει και κρυπτογραφεί τα επιμέρους κομμάτια του. Κάθε πακέτο μεταφοράς είναι μεγέθους 512 ή 1024 byte. Για το padding των κρυπτογραφημένων τμημάτων χρησιμοποιείται ή μέθοδος zfill() και για το unpadding από την μεριά του Server χρησιμοποιείται ως σημείο αναφοράς ο χαρακτήρας "(" που αρχίζει σε κάθε κρυπτογραφημένο μήνυμα.

6. CONCLUSION

Με το παραπάνω project προσομοιώσαμε την χρήση των γνώσεων μας περί κρυπτογράφησης σε ένα ποιο ρεαλιστικό σενάριο και ήρθαμε σε επαφή με τον άθλο της δημιουργίας ασφαλών εφαρμογών. Για την εξασφάλιση του δημιουργηθέντος συστήματος μεταπηδήσαμε αρκετές φορές από την θέση του προστάτη σε αυτή του επιτιθεμένου για να ανακαλύψουμε κάθε φορά πιθανές αδυναμίες στον κώδικα ώστε να τις εξαλείψουμε.

7. BIBLIOGRAPHY

-Python Exceptions Documentation
https://docs.python.org/2/tutorial/errors.html

-Python Network Programming
http://www.tutorialspoint.com/python/python_networking.htm

-Python Socket Documentation
https://docs.python.org/2/library/socket.html

-PyCrypto Documentation
https://pypi.python.org/pypi/pycrypto