ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

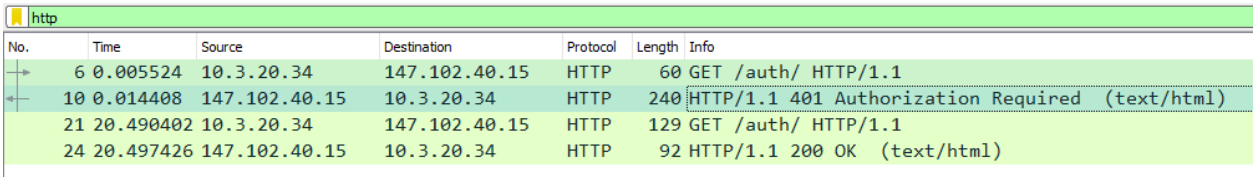
ΟΝΟΜΑΤΕΠΩΝΥΜΟ: ΓΚΟΥΜΕ ΛΑΟΥΡΕΝΤΙΑΝ
ΑΜ: 031 18 014
ΕΞΑΜΗΝΟ: 7^{ο}
ΟΜΑΔΑ: 4
MAC ADDRESS: B4-69-21-1B-6C-FF
IPv4: Άσκ1, Άσκ2, Άσκ3:10.3.20.34
ΌΝΟΜΑ ΥΠΟΛΟΓΙΣΤΗ: LAPTOP-B2DVAJKK
ΛΕΙΤΟΥΡΓΙΚΟ ΣΥΣΤΗΜΑ: WINDOWS 10

ΔΙΚΤΥΑ ΥΠΟΛΟΓΙΣΤΩΝ

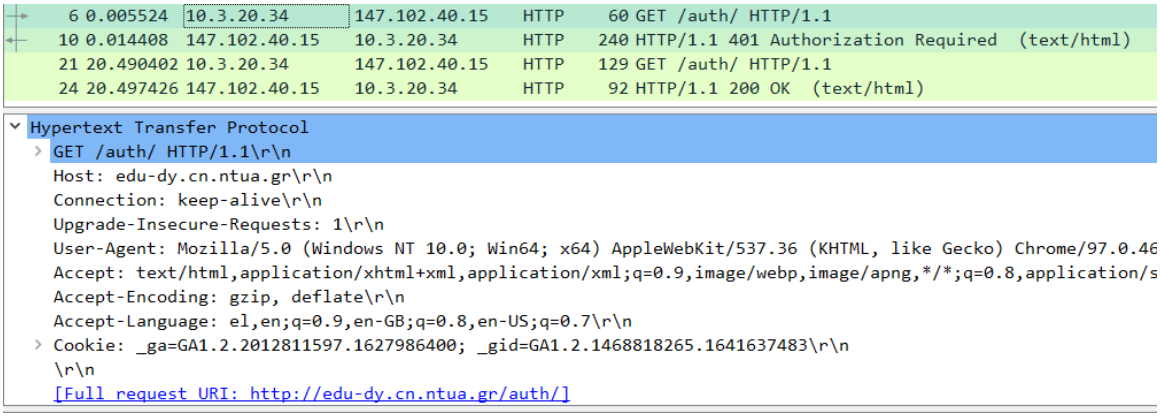
ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 12: Ασφάλεια

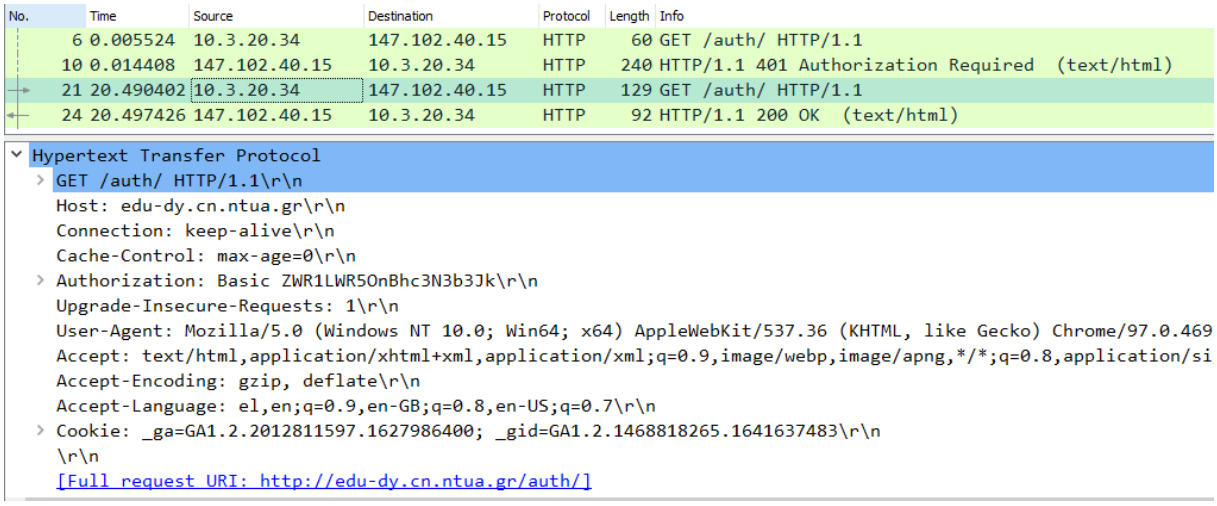
Άσκηση 1: Πιστοποίηση αυθεντικότητας στο πρωτόκολλο HTTP

1.1) Η απόκριση του εξυπηρετητή στο αρχικό μήνυμα HTTP τύπου GET, έχει **status code 401** και φράση **Authorization Required**.



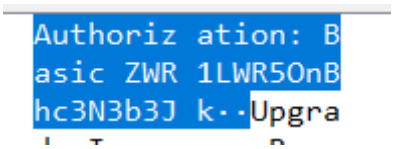
1.2) Βλέπουμε τα πεδία των 2 GET αιτημάτων κατά σειρά:





Παρατηρούμε πως υπάρχουν 2 επιπλέον πεδία, τα **Cache-Control και Authorization**.

1.3) Θα ασχοληθούμε με το πεδίο Authorization. Το περιεχόμενο του πεδίου αυτού σε μορφή ASCII φαίνεται παρακάτω: **(Authorization: Basic ZWR1LWR5OnBhc3N3b3Jk)**



Οι 2 κουκίδες στο τέλος αντιστοιχούν στο Carriage Return (/r) και Newline (/n).

1.4) Όπως αναμέναμε, το αποτέλεσμα της αποκωδικοποίησης είναι **edu-dy:password**.

1.5) Διαπιστώνουμε πως ο μηχανισμός πιστοποίησης αυθεντικότητας που παρέχει το HTTP και βασίζεται στην κωδικοποίηση Base64 **δεν είναι καθόλου ασφαλής** και αυτό διότι αρκεί κάποιος να καταφέρει να υποκλέπτει τα πακέτα της σύνδεσης μεταξύ των 2 άκρων, καθώς μετά μπορεί εύκολα να αποκωδικοποιήσει οποιαδήποτε ευαίσθητα δεδομένα εστάλησαν κατά την επικοινωνία αυτή. Συνεπώς, εφόσον ο αποστολέας και ο "κανονικός" παραλήπτης δεν είναι οι μόνοι που μπορούν να κατανοούν το περιερχόμενο της σύνδεσης **δεν υπάρχει εμπιστευτικότητα**.

Άσκηση 2: Υπηρεσία SSH-Secure Shell

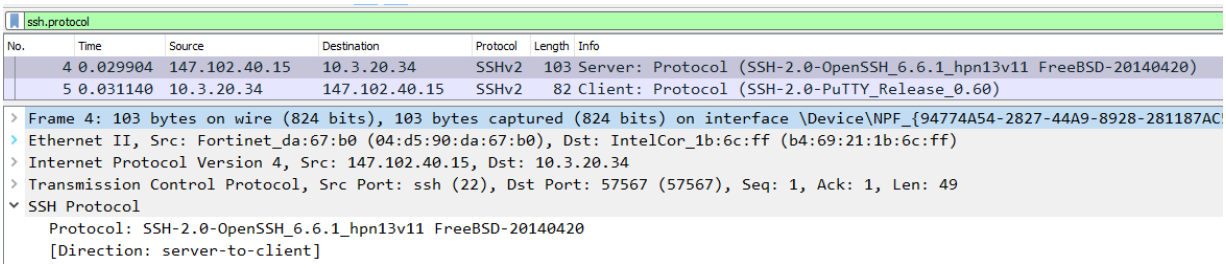
2.1) Το SSH χρησιμοποιεί το πρωτόκολλο μεταφοράς **TCP**.

2.2) Χρησιμοποιούνται οι **θύρες 22 (ssh) από τον εξυπηρετητή και 57.576 από εμάς**.

2.3) Η θύρα **22**.

2.4) Φίλτρο απεικόνισης: **ssh**.

2.5) Με το φίλτρο **ssh.protocol** εντοπίζουμε τα μηνύματα SSH τύπου Protocol, όπως φαίνονται παρακάτω:

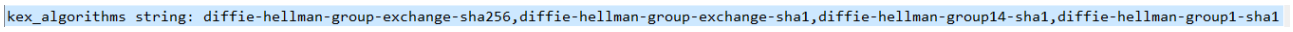


Συμπεραίνουμε πως από τον εξυπηρετητή χρησιμοποιείται η **έκδοση SSH-2.0**, η **έκδοση λογισμικού OpenSSH_6.6.1_hpn13v11** και στα **σχόλια εντοπίζουμε το FreeBSD-20140420**.

2.6) Με τον ίδιο τρόπο, για τον πελάτη βρίσκουμε:

- **Έκδοση: SSH-2.0**
- **Λογισμικό: PuTTY_Release_0.60**

2.7) Εντοπίζουμε τη λίστα με τους αλγορίθμους ανταλλαγής κλειδιών:



Όπως βλέπουμε, εμφανίζονται **4 kex αλγόριθμοι**. Οι 2 πρώτοι εξ αυτών είναι οι **diffie-hellman-group-exchange-sha256 και diffie-hellman-group-exchange-sha1**.

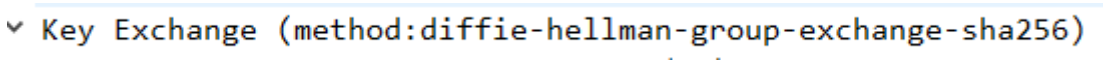
2.8) Στο ίδιο πακέτο με πριν, εντοπίζουμε τους **2 αλγορίθμους ssh-rsa και ssh-dss**.

2.9) Οι 2 πρώτοι αλγόριθμοι κρυπτογράφησης που υποστηρίζει ο πελάτης με κατεύθυνση client to server είναι **aes256-ctr, aes256-cbc**.

2.10) Αντίστοιχα, για τους αλγορίθμους πιστοποίησης αυθεντικότητας (mac), έχουμε τους **hmac-sha1 και hmac-sha1-96**.

2.11) Αντίστοιχα για τους αλγορίθμους συμπίεσης (compression), έχουμε τους **none και zlib**.

2.12) Βλέποντας τη λίστα αλγορίθμων ανταλλαγής κλειδιών του εξυπηρετητή και συγκρίνοντας με αυτή του πελάτη, αναμένουμε να χρησιμοποιηθεί ο αλγόριθμος **diffie-hellman-group-exchange-sha256**. Πράγματι, το επαληθεύουμε από το **πεδίο Key Exchange** όπως βλέπουμε παρακάτω.

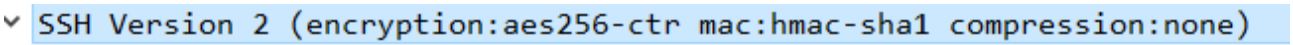


2.13) Στο πεδίο SSH Version 2 βλέπουμε το **encryption:aes256-ctr**. Παρατηρούμε πως είναι ο πρώτος από τη λίστα του εξυπηρετητή που υπάρχει και στη λίστα του πελάτη.

2.14) Χρησιμοποιείται ο αλγόριθμος πιστοποίησης αυθεντικότητας μηνυμάτων **hmac-sha1** κατά την κατεύθυνση πελάτης → εξυπηρετητής. Ισχύει η ίδια παρατήρηση με το 2.13.

2.15) Δε χρησιμοποιείται κανένας αλγόριθμος συμπίεσης **(none)**.

2.16) Επιλέγοντας ένα εκ των δύο πακέτων Key Exchange Init, βλέπουμε στο πεδίο **SSH Version 2** της επικεφαλίδας SSH Protocol τις ζητούμενες πληροφορίες.



2.17) Καταγράφηκαν επιπλέον οι εξής τύποι μηνυμάτων SSH:

- **Diffie-Hellman Group Exchange Request (Old)**
- **Diffie-Hellman Group Exchange Group**
- **Diffie-Hellman Group Exchange Init**
- **Server: Diffie-Hellman Group Exchange Reply, New Keys**
- **New Keys**
- **Encrypted Packet**

2.18) Παρατηρούμε πως **δε γίνεται αντιληπτό ποια πακέτα αφορούν το login και το password** στην περίπτωση του SSH και ο λόγος είναι πως τα πακέτα αυτά είναι κρυπτογραφημένα.

2.19) Αναφορικά με την ασφάλεια του SSH:

- **Πιστοποίηση αυθεντικότητας:** Έχουμε authentication μέσω public-private keys, από τις ασφαλέστερες δηλαδή μεθόδους.
- **Εμπιστευτικότητα:** Λόγω της κρυπτογράφησης, το περιεχόμενο γίνεται κατανοητό μόνο από τον εξυπηρετητή και τον πελάτη.
- **Ακεραιότητα δεδομένων:** Παρέχονται hashing αλγόριθμοι για data-integrity (MAC).

Κρίνεται, επομένως, ως μια ασφαλής επιλογή.

Άσκηση 3: Υπηρεσία HTTPS

3.1) Φίλτρο σύλληψης: **host bbb2.cn.ntua.gr**.

3.2) Με το φίλτρο απεικόνισης **tcp.len==0 and ((tcp.seq==0 and tcp.ack==0) or (tcp.seq==0 and tcp.ack==1) or (tcp.seq==1 and tcp.ack==1))** εμφανίζονται όλες οι τριπλές χειραψίες που έγιναν. Εμφανίζονται 24 πακέτα, επομένως συμπεραίνουμε πως έγιναν 8 TCP συνδέσεις (φαίνονται στην παρακάτω εικόνα).

3.3) Οι συνδέσεις έγιναν στις θύρες **80 (HTTP) και 443 (HTTPS)** του εξυπηρετητή.

3.4) 80-HTTP, 443-HTTPS

3.5) Ανοίχτηκαν **6 συνδέσεις HTTP και 2 συνδέσεις HTTPS**.

3.6) Χρησιμοποιούνται οι θύρες **55.498 και 55.499**.

3.7) Παρατηρούμε τα εξής πεδία:

- **Content Type (1 Byte)**
- **Version (2 Bytes)**
- **Length (2 Bytes)**

3.8) Καταγράφουμε τις παρακάτω τιμές:

- **Handshake (22)**
- **Change Cipher Spec (20)**
- **Application Data (23)**

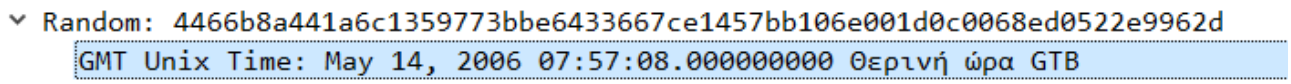
3.9) Καταγράφουμε τους εξής τύπους μηνυμάτων χειραψίας:

- **Client Hello**
- **Server Hello**
- **Certificate**
- **Server Key Exchange**
- **Server Hello Done**
- **Client Key Exchange**
- **Change Cipher Spec**
- **Encrypted Handshake Message**
- **New Session Ticket**

3.10) Ο πελάτης έστειλε **2 Client Hello, όσες και οι HTTPS συνδέσεις**.

3.11) Η μέγιστη υποστηριζόμενη έκδοση είναι η **TLS 1.0 (0x0301)**.

3.12) Το μήκος του τυχαίου αριθμού είναι **32 bytes**, με τα πρώτα 4 εξ αυτών να είναι τα **44 66 b8 a4**. Τα bytes αυτά δηλώνουν το **GMT Unix Time**.



3.13) Καταγράφονται **16 Cipher Suites**, ενώ οι 2 πρώτες εξ αυτών είναι οι **0x1a1a (Reserved (GREASE))** και **0x1301 (TLS_AES_128_GCM_SHA256)**. Επειδή το Reserved (GREASE) δεν είναι στην πραγματικότητα cipher, παραθέτουμε επιπλέον το **0x1302 (TLS_AES_256_GCM_SHA384)**.

3.14) Από τον εξυπηρετητή θα χρησιμοποιηθεί η έκδοση **TLS 1.2 (0x0303)**, ενώ επιλέχθηκε η σουίτα **TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)**.

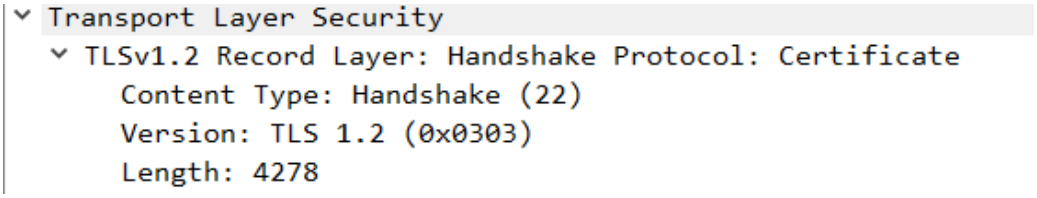
3.15) Και εδώ περιέχονται **32 bytes** στον τυχαίο αριθμό. Τα πρώτα 4 bytes του Random είναι τα **f0 c1 0b 80**.

3.16) Το πεδίο Compression Method έχει τιμή **null (0)**, επομένως δε χρησιμοποείται κάποια μέθοδος συμπίεσης.

3.17) Τα ζητούμενα βρίσκονται στο πεδίο Cipher Suite, το οποίο στην περίπτωσή μας έχει τιμή **TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256**. Ειδικότερα, από το όνομα αυτό εξάγουμε τα εξής:

- Αλγόριθμος ανταλλαγής κλειδιών: **ECDHE**
- Αλγόριθμος πιστοποίησης ταυτότητας: **RSA**
- Αλγόριθμος κρυπτογράφησης: **AES**(128bits)
- Αλγόριθμος συνάρτησης κατακερματισμού: **SHA**(256bits)

3.18) Είναι **4278 bytes**, όπως φαίνεται παρακάτω.



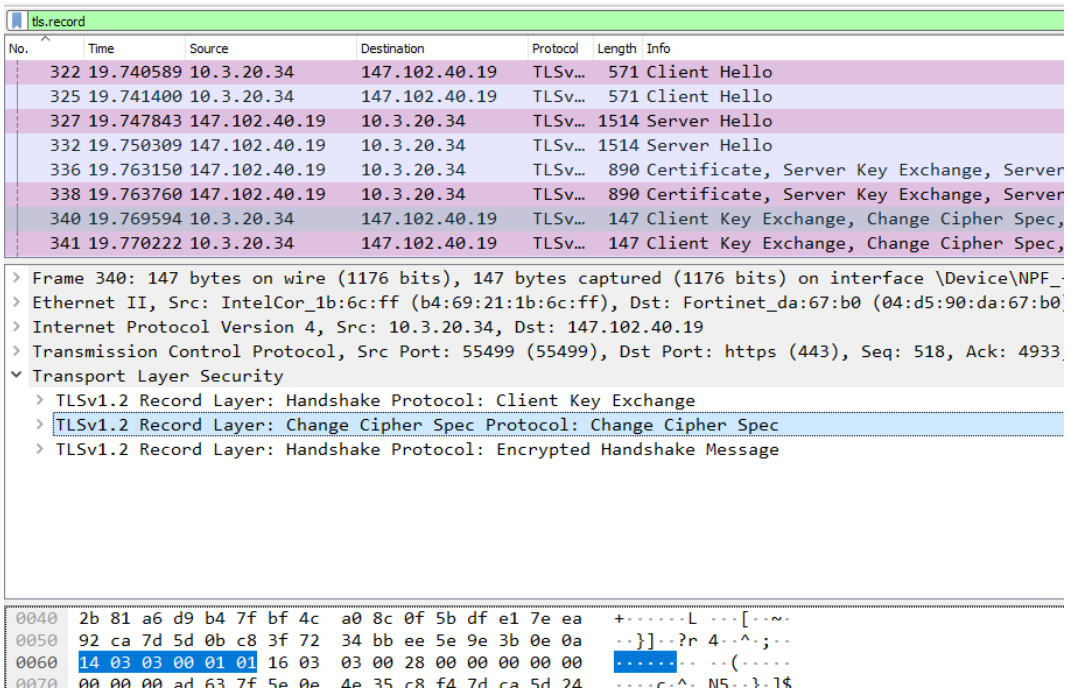
3.19) Μεταφέρονται **3 πιστοποιητικά**. Τα ονόματα τους είναι τα εξής:

1) id-at-commonName = **bbb2.cn.ntua.gr**
2) id-at-commonName = **R3**
3) id-at-commonName = **ISRG**

3.20) Χρειάστηκαν **4 πλαίσια Ethernet**, ώστε να μεταφερθεί η παραπάνω εγγραφή TLS.

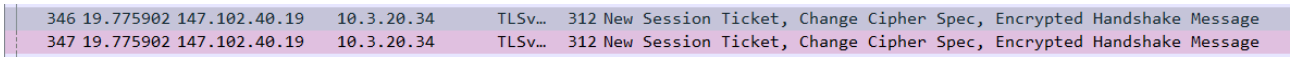
3.21) Ο πελάτης αποστέλλει δημόσιο κλειδί μήκους **32 bytes (5 αρχικά γράμματα: 2b81a)**, **όσα bytes αποστέλλει δηλαδή και ο εξυπηρετητής (5 πρότερα γράμματα: cc76b)**.

3.22) Είναι **6 bytes**, όπως φαίνεται παρακάτω.



3.23) Είναι **45 bytes (40 εκ των οποίων αποτελούν το Encrypted Handshake Message)**.

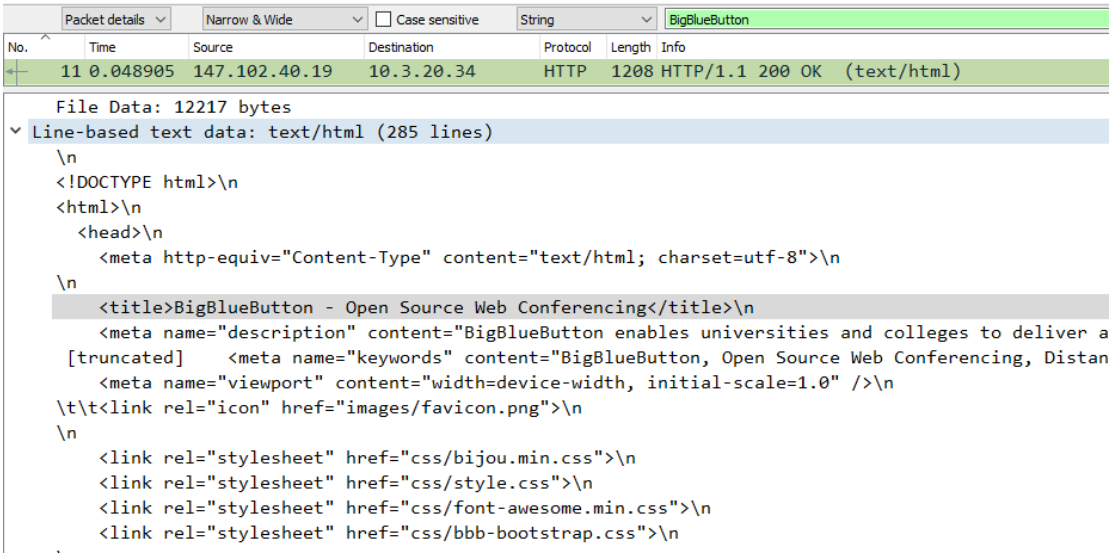
3.24) Ναι, παρατηρήσαμε.



3.25) Όχι, δε παρατηρήσαμε.

3.26) Ο λόγος που θα υπήρχαν, θα ήταν προκειμένου **ο πελάτης να ειδοποιήσει τον εξυπηρετητή για την απόλυση της TCP σύνδεσης**.

3.27) Στη περίπτωση του HTTP βρίσκουμε πακέτο που έχει ως περιεχόμενο το περιεχόμενο της ιστοσελίδας που ζητήσαμε σε μορφή html, όπως βλέπουμε παρακάτω. Αντίθετα, στα πακέτα HTTPS δε μπορούμε να βρούμε κάποιο πακέτο αναζητώντας το String BigBlueButton και αυτό διότι η **πληροφορία μεταφέρεται κρυπτογραφημένη στο https σε αντίθεση με το http**.



3.28) Συγκρίνοντας το HTTP με το HTTPS, μπορούμε να πούμε πως:

- **Πιστοποίηση αυθεντικότητας:** Στο HTTPS, όταν ένας client εκκινεί έναν “δίαυλο” επικοινωνίας με έναν εξυπηρετητή, ο εξυπηρετητής επαληθεύει τη γνησιότητα του αντιστοιχίζοντας το private key του με το public key στο SSL certificate (το οποίο είναι signed από μία έμπιστη αρχή) της σελίδας που επισκεπτόμαστε. Στο HTTP δεν υπάρχει κάποια αντίστοιχη διαδικασία που να εξασφαλίζει την πιστότητα του εξυπηρετητή.
- **Εμπιστευτικότητα:** Στο HTTP τα δεδομένα στέλνονται ως plaintext, επομένως είναι άμεσα αναγνώσιμα από κάποιον που θα καταφέρει να υποκλέψει κάποια πακέτα. Αντιθέτως, το περιεχόμενο στο HTTPS είναι κρυπτογραφημένο, με αποτέλεσμα ακόμα και αν κάποιος υποκλέψει πακέτα να διαβάσει κάτι που δε βγάζει νόημα και από το οποίο δε μπορεί να εξάγει κάτι χρήσιμο.
- **Ακεραιότητα:** Στο HTTPS είναι αδύνατον να μεταβληθούν τα δεδομένα χωρίς αυτό να γίνει αντιληπτό από τους συμμετέχοντες στη σύνδεση. Αντιθέτως, το HTTP είναι επιρρεπές σε επιθέσεις τύπου Man-In-The-Middle, οι οποίες θα μπορούσαν να αλλοιώσουν το περιεχόμενο των πακέτων.