Ονοματεπώνυμο: Ανδρέας Στάμος (03120***)	**Ομάδα:** 1
Όνομα PC/ΛΣ: linux / Ubuntu 22.04.2 LTS (με VPN στο δίκτυο του Πολυτεχνείου)	**Ημερομηνία:** 09/01/2024
Διεύθυνση IP: 147.102.131.218	**Διεύθυνση MAC:** DE–3F–DC–B2–E0–D0

Εργαστηριακή Άσκηση 12
Ασφάλεια

Απαντήστε στα ερωτήματα στον χώρο που σας δίνεται παρακάτω και στην πίσω σελίδα εάν δεν επαρκεί. Το φυλλάδιο αυτό θα παραδοθεί στον επιβλέποντα.

Η εργασία υλοποιήθηκε με σύνδεση στο δίκτυο VPN του Πολυτεχνείου.

Άσκηση 1

1.1 401

1.2 `WWW-Authenticate` και υποδεικνύει την μέθοδο Basic.

1.3 `Authorization`

1.4 `Authorization: Basic ZWR1LWR5OnBhc3N3b3Jk`

1.5 `edu-dy:password` (χωρίς αλλαγή γραμμής)

Συμπλήρωση Για base64 αποκωδικοποίηση στο Linux μπορούμε να χρησιμοποιήσουμε την εντολή `base64 -d`

1.6 Δεν πρόσφερει καμία ασφάλεια διότι τα διαπιστευτήρια μπορούν να διαρρεύσουν σε κάποιον ενδιάμεσο κόμβο, που μόλις τα διαθέτει θα μπορεί να περάσει τον έλεγχο αυθεντικότητας.

Άσκηση 2

2.1 TCP

2.2 Θύρα πελάτη: 46486

Θύρα εξυπηρετητή: 22

2.3 22

2.4 `ssh`

2.5 Χρησιμοποιεί ssh 2.0 με το λογισμικό OpenSSH_8.9p1.

Στα σχόλια βρίσκεται το `Ubuntu-3ubuntu0.6`.

2.6 Χρησιμοποιεί ssh 2.0 με το λογισμικό OpenSSH_6.6.1_hpn13v11.

Στα σχόλια βρίσκεται το `FreeBSD-20140420`.

2.7 Έχει μήκος 305 bytes και περιλαμβάνει 12 αλγόριθμους με τους πρώτους δύο να είναι οι `curve25519-sha256` και `curve25519-sha256@libssh.org`.

2.8 16 αλγόριθμοι με τους δύο πρώτους να είναι οι `ssh-ed25519-cert-v01@openssh.com` και `ecdsa-sha2-nistp256-cert-v01@openssh.com`.

2.9 `chacha20-poly1305@openssh.com` και `aes128-ctr`

2.10 `umac-64-etm@openssh.com` και `umac-128-etm@openssh.com`

2.11 `none` και `zlib@openssh.com`

2.12 Ο αλγόριθμος επιλέγεται implicity. Επιλέγεται ο πρώτος αλγόριθμος από την λίστα του πελάτη τον οποίο υποστηρίζει και ο εξυπηρετητής, για τον οποίο στις λίστες server_host_key_algorithms και πελάτη και εξυπηρετητή υπάρχει τουλάχιστον έναν συμβάτος αλγόριθμος. Και ο εξυπηρετητής και ο πελάτης διαθέτουν τις λίστες πελάτη και εξυπηρετητή οπότε μπορούν να γνωρίζουν ποιος αλγόριθμος επιλέγεται.

Εδώ επιλέγεται ο `curve25519-sha256@libssh.org`.

Το Wireshark εφαρμοζεί την λογική αυτή και στα μηνύματα ανταλλαγής κλειδιών (μετά το Key Exchange Init) εμφανίζει τον αλγόριθμο που ακολουθείται.

2.13 `ssh-ed25519`

2.14 `chacha20-poly1305@openssh.com`

2.15 `umac-64-etm@openssh.com`

2.16 `none` — δηλαδή όχι συμπίεση

2.17 Παρατηρούμε τα `Elliptic Curve Diffie-Hellman Key Exchange Init` και `Elliptic Curve Diffie-Hellman Key Exchange Reply`.

Αφού ολοκληρωθεί η ανταλλαγή κλειδιών, κάθε πλευρά στέλνει ένα μήνυμα `New Keys` που σηματοδοτεί ότι απο εδώ και πέρα η απέναντι πλευρά θα πρέπει να το χρησιμοποιεί.

2.18 Ο αλγόριθμος παραγωγής κλειδιών εμφανίζεται στα μηνύματα για την ανταλλαγή κλειδιών που απαιτεί ο επιλεγμένος αλγόριθμος, ο αλγόριθμος κρυπτογράφησης και ο αλγόριθμος συμπίεσης εμφανίζεται σε όλα τα μήνυματα (παραδόξως και στα αρχικά, που δεν είναι κρυπτογραφημένα). Ο αλγόριθμος πιστοποίησης αυθεντικότητας μηνυμάτων δεν εμφανίζεται.

2.19 Όχι. Τα στοιχεία αυτά μεταδίδονται κρυπτογραφημένα και το Wireshark είναι ακριβώς ένας ενδιάμεσος που λαμβάνει τα μεταδιδόμενα μηνύματα. Αν το Wireshark μπορούσε να δει τα στοιχεία διαπίστευσης, αυτό θα μπορούσε να κάνει και κάθε ενδιάμεσος οπότε δεν θα υπήρχε ασφάλεια.

2.20 Το ssh σχεδιάστηκε να ικανοποιεί όλες αυτές τις προδιαγραφές και είναι ευέλικτο ως προς την επιλογή αλγορίθμων, ώστε αν υπάρξουν καλύτεροι, ασφαλέστεροι, ταχύτεροι αλγόριθμοι να μεταπηδά σε αυτούς χωρίς δυσκολία μετάβασης.

1 Άσκηση 3

3.1 `host www.noc.ntua.gr`

3.2 `tcp.flags.syn == 0 && tcp.flags.ack == 0`

3.3 80, 443

3.4 Στο HTTP αντιστοιχεί η 80 και στο HTTPS η 443.

3.5 6 συνδέσεις και στο HTTP και στο HTTPS.

3.6 57634, 57638, 57644, 57654, 57660, 57666

3.7 `Content Type` μήκους 1 byte, `Version` μήκους 2 bytes και `Length` μήκους 2 bytes.

3.8

- **20** Change Cipher Spec
- **21** Encrypted Alert
- **22** Handshake
- **23** Application Data

3.9

- **1** Client Hello
- **2** Server Hello
- **11** Certificate
- **12** Server Key Exchange

14 Server Hello Done

16 Client Key Exchange

3.10 6, ισάριθμα με το πλήθος TCP συνδέσεων καθώς κάθε TCP σύνδεση είναι και μια διαφορετική TLS σύνδεση που απαιτεί δική της διαπραγμάτευση.

3.11 TLS 1.0 με τιμή 0χ0301=769.

3.12 TLS 1.2 με τιμή 0x0303=771.

Διαφέρει από την εγγραφή TLS. Εφόσον πρόκειται για διαφορετικά επίπεδα αυτό φαίνεται λογικό να μπορεί να συμβεί.

3.13 32 bytes. Τα πρώτα 4 bytes αναπαριστούν, θεωρητικά, την χρονική στιγμή δημιουργίας του Random, όμως αυτό τελικά, φαίνεται να έχει εγκαταλειφθεί για λόγους ιδιωτικότητας. Ενδεικτικά, δηλαδή, εδώ έχει δοθεί το 1035731164 που αντιστοιχεί σε ημερομηνία το 2002!

3.14 17 σουίτες με τις 2 πρώτες να είναι `TLS_AES_128_GCM_SHA256` (0x1301) και `TLS_CHACHA20_POLY1305_SHA256` (0x1303).

3.15 0x0304

3.16 `h2` (HTTP/2) και `http/1.1`

3.17 TLS 1.2 (0x0303)

3.18 32 bytes. Ισχύει ότι αναφέρθηκε στο ερώτημα 3.13. Εδώ έχει τιμή 2327686762, που αντιστοιχεί σε ημερομηνία το 2043!

3.19 `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256` (0xc02f)

3.20

Αλγόριθμος ανταλλαγής κλειδιού ECDHE

Αλγόριθμος πιστοποίησης ταυτότητας RSA

Αλγόριθμος κρυπτογράφησης AES με μέγεθος κλειδιού 128 bits

Συνάρτηση κατακερματισμού SHA μεγέθους 256 bits

3.21 Όχι. Το πεδίο Compression Method έχει τιμή `null` (0).

3.22 6202 bytes

3.23 Μεταφέρονται 4 πιστοποιητικά με μήκη αντίστοιχα 1930, 1769, 1413 και 1078 bytes.

3.24 5

3.25 65 bytes αμφότερων των πελάτη και εξυπηρετητή.

Τα 5 πρώτα γράμματα του δημόσιου κλειδιού πελάτη είναι `a0cf1` και του εξυπηρετητή είναι `40cb1`.

3.26 1 byte

3.27 40 bytes

3.28 Ναι.

3.29 HTTP

3.30 Ναι και από τις δύο πλευρές.

3.31 Τα TLS Alert χρησιμοποιούνται για να σηματοδοτήσουν ότι η αποστολή δεδομένων τελείωσε και πως η σύνδεση μπορεί να κλείσει.

3.32 Στην περίπτωση του HTTP είναι πολύ εύκολο να το εντοπίσουμε με απλή αναζήτηση, αφού τα δεδομένα μεταδίδονται σε plain text.

Στην περίπτωση του HTTP είναι αδύνατο να το εντοπίσουμε, αφού τα δεδομένα μεταδίδονται κρυπτογραφημένα.

3.33

Πιστοποίηση Αυθεντικότητας

HTTP Δεν παρέχει μηχανισμό για την επαλήθευση της ταυτότητας του διακομιστή. Ένας χρήστης δεν μπορεί να είναι βέβαιος ότι επικοινωνεί με τον αυθεντικό διακομιστή και όχι με έναν ενδιάμεσο επιτιθέμενο.

HTTPS Χρησιμοποιεί πιστοποιητικά SSL/TLS για να επιβεβαιώσει την ταυτότητα του διακομιστή. Αυτό βοηθά τους χρήστες να είναι σίγουροι ότι επικοινωνούν με τον πραγματικό διακομιστή και όχι με έναν ενδιάμεσο.

Εμπιστευτικότητα

HTTP Όλα τα δεδομένα αποστέλλονται σε απλό κείμενο, κάτι που τα καθιστά ευάλωτα σε παρακολούθηση και παραβίαση από τρίτους.

HTTPS Χρησιμοποιεί κρυπτογράφηση για να προστατεύσει τα δεδομένα κατά τη μετάδοση. Αυτό δυσκολεύει τους ενδιάμεσους επιτιθέμενους να διαβάσουν ή να τροποποιήσουν τα δεδομένα που ανταλλάσσονται μεταξύ του χρήστη και του διακομιστή.

Ακεραιότητα Δεδομένων

HTTP Δεν παρέχει καμία εγγύηση ότι τα δεδομένα που λαμβάνονται δεν έχουν τροποποιηθεί κατά τη μετάδοση.

HTTPS Η κρυπτογράφηση που χρησιμοποιείται επιτρέπει τον έλεγχο της ακεραιότητας των δεδομένων, εξασφαλίζοντας ότι τα δεδομένα που φτάνουν στον παραλήπτη δεν έχουν τροποποιηθεί κατά τη διαδρομή τους.