

ΙΟΝΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ



ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Μελέτη και Ανάλυση του πρωτοκόλλου TLS 1.3 με Python

ΚΑΛΑΘΑΣ ΑΛΕΞΑΝΔΡΟΣ

Π2019017

Επιβλέπων: ΝΤΑΝΤΟΓΙΑΝ ΧΡΙΣΤΟΦΟΡΟΣ

Σεπτέμβριος 2023

Επιβλέπων

ΝΤΑΝΤΟΓΙΑΝ ΧΡΙΣΤΟΦΟΡΟΣ, Επίκουρος Καθηγητής
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Τριμελής Επιτροπή

ΝΤΑΝΤΟΓΙΑΝ ΧΡΙΣΤΟΦΟΡΟΣ, Επίκουρος Καθηγητής
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΧΡΙΣΤΟΠΟΥΛΟΥ ΕΛΕΝΗ, Εργαστηριακό Διδακτικό Προσωπικό
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΜΑΓΚΟΣ ΕΜΜΑΝΟΥΗΛ, Καθηγητής
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ	I
ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ ΚΑΙ ΣΧΗΜΑΤΩΝ	II
ΚΕΦΑΛΑΙΟ Α. ΕΙΣΑΓΩΓΗ.....	1
Α'1. ΠΕΡΙΛΗΨΗ.....	1
Α'2. ΠΡΟΛΟΓΟΣ	2
Α'3. ΕΠΙΠΕΔΑ TCP/IP ΚΑΙ ΔΟΜΗ ΠΑΚΕΤΟΥ.....	2
Α'3.1. <i>Payload</i>	2
Α'3.2. <i>Application Layer</i>	3
Α'3.3. <i>Transport Layer</i>	3
Α'3.4. <i>Internet Layer</i>	4
Α'3.5. <i>Link Layer</i>	4
ΚΕΦΑΛΑΙΟ Β. ΜΕΛΕΤΗ ΚΑΙ ΑΝΑΛΥΣΗ ΤΟΥ TLS 1.3	5
Β'1. ΠΩΣ ΒΟΗΘΑΕΙ Η ΕΦΑΡΜΟΓΗ ΤΟΥ TLS	5
Β'1.1. <i>Ο λόγος κυκλοφορίας μιας νέας έκδοσης</i>	5
Β'2. ΕΙΣΑΓΩΓΗ ΣΤΟ TLS 1.3	6
Β'2.1. <i>TLS 1.2 και 1.3 cipher suites</i>	6
Β'2.2. <i>Εξήγηση των cipher suites του TLS 1.3</i>	7
Β'2.3. <i>Αναπαράσταση του αλγορίθμου Diffie-Hellman για το TLS 1.3</i>	8
Β'3. ΕΡΓΑΣΙΑ ΚΑΙ ΜΕΘΟΔΟΙ ΠΟΥ ΥΛΟΠΟΙΗΘΗΚΑΝ ΓΙΑ ΤΗ ΣΥΛΛΗΨΗ ΜΙΑΣ ΧΕΙΡΑΨΙΑΣ	10
Β'3.1. <i>Χαρακτηριστικά περιβάλλοντος και προετοιμασία</i>	10
Β'3.2. <i>Κώδικας</i>	11
Β'3.3. <i>Output</i>	16
Β'4. ΕΙΣΑΓΩΓΗ ΣΤΗ ΧΕΙΡΑΨΙΑ	18
Β'4.1. <i>Διαφορές RTT μεταξύ 1.2 και 1.3 στη χειραψία</i>	19
Β'4.1.1. TLS 1.2.....	19
Β'4.1.2. TLS 1.3.....	20
Β'4.1.3. <i>Session Resumption και Early Data</i>	21
Β'5. ΑΝΑΛΥΣΗ ΤΗΣ ΧΕΙΡΑΨΙΑΣ ΜΕ ΚΩΔΙΚΑ	21
Β'5.1. <i>Πρώτο βήμα της χειραψίας, Client Hello</i>	22
Β'5.1.1. <i>Early Data</i>	32
Β'5.2. <i>Δεύτερο βήμα της χειραψίας, Server Hello, Server Finished</i>	33
Β'5.2.1 <i>Certificate Chain</i>	37
Β'5.2.2. <i>Certificate Verify, Finished</i>	48
Β'5.3. <i>Τρίτο βήμα της χειραψίας, Client Finished</i>	49
Β'5.4. <i>Encrypted Application Data</i>	50
ΚΕΦΑΛΑΙΟ Γ. ΣΥΜΠΕΡΑΣΜΑ	53
ΒΙΒΛΙΟΓΡΑΦΙΑ	54
ΠΕΡΙΕΧΟΜΕΝΑ	54
ΗΛΕΚΤΡΟΝΙΚΕΣ ΠΗΓΕΣ	55
ΣΥΝΤΜΗΣΕΙΣ.....	57
ΓΛΩΣΣΑΡΙ ΞΕΝΙΚΩΝ ΌΡΩΝ	58
ΕΥΡΕΤΗΡΙΟ	61

Κατάλογος Εικόνων και Σχημάτων

ΕΙΚΟΝΑ 1 ΑΛΓΟΡΙΘΜΟΣ ΑΝΤΑΛΛΑΓΗΣ ΚΛΕΙΔΙΩΝ DIFFIE-HELLMAN	9
ΕΙΚΟΝΑ 2 ΤΑ ΜΥΣΤΙΚΑ ΚΛΕΙΔΙΑ ΓΙΑ ΤΗ ΠΑΡΑΓΩΓΗ ΤΟΥ ΚΟΙΝΟΥ ΜΥΣΤΙΚΟΥ ΚΛΕΙΔΙΟΥ.....	11
ΕΙΚΟΝΑ 3 ΜΕΓΕΘΟΣ ΤΟΥ ΠΑΚΕΤΟΥ, ΠΟΤΕ ΣΥΛΛΗΦΘΗΚΕ ΑΠΟ ΤΟ TSHARK ΚΑΙ ΠΟΙΑ ΕΠΙΠΕΔΑ ΒΡΙΣΚΟΝΤΑΙ ΣΤΟ ΠΑΚΕΤΟ	14
ΕΙΚΟΝΑ 4 ΠΡΩΤΟ ΕΠΙΠΕΔΟ, ETHERNET. ΈΧΩ ΚΡΥΨΕΙ ΤΑ ΕΥΑΙΣΘΗΤΑ ΔΕΔΟΜΕΝΑ ΚΑΘΩΣ ΜΠΟΡΕΙ ΝΑ ΕΠΗΡΕΑΣΕΙ ΤΟ ΠΕΡΙΒΑΛΛΟΝ ΕΚΤΟΣ ΤΗΣ ΕΙΚΟΝΙΚΗΣ ΜΗΧΑΝΗΣ	14
ΕΙΚΟΝΑ 5 ΔΕΥΤΕΡΟ ΕΠΙΠΕΔΟ, INTERNET PROTOCOL. Η ΔΙΕΥΘΥΝΣΗ 10.0.2.11 ΕΙΝΑΙ ΤΟΥ VM ΚΑΙ Η 195.130.127.114 ΤΟΥ ΙΟΝΙΟ.GR. ΤΟ ΣΥΓΚΕΚΡΙΜΕΝΟ ΠΑΚΕΤΟ ΑΠΟΣΤΕΛΛΕΤΑΙ ΑΠΟ ΕΜΕΝΑ ΣΤΟ ΙΟΝΙΟ.GR (SPOILER: ΕΙΝΑΙ ΤΟ CLIENT HELLO).....	15
ΕΙΚΟΝΑ 6 ΤΡΙΤΟ ΕΠΙΠΕΔΟ, TCP. ΤΟ TCP PAYLOAD ΣΕ ΑΥΤΟ ΤΟ ΠΑΚΕΤΟ ΕΙΝΑΙ ΤΟ ΕΠΟΜΕΝΟ ΕΠΙΠΕΔΟ. 15	
ΕΙΚΟΝΑ 7 ΑΥΤΟ ΒΛΕΠΕΙ ΟΠΟΙΟΣΔΗΠΟΤΕ ΔΕΝ ΜΠΟΡΕΙ ΝΑ ΑΠΟΚΡΥΠΤΟΓΡΑΦΗΣΕΙ ΤΗ ΚΙΝΗΣΗ.	16
ΕΙΚΟΝΑ 8 ΑΠΟΚΡΥΠΤΟΓΡΑΦΗΜΕΝΗ ΚΙΝΗΣΗ. ΜΠΟΡΟΥΜΕ ΞΕΚΑΘΑΡΑ ΝΑ ΔΟΥΜΕ ΤΙ ΕΙΠΑΝ Ο CLIENT ΚΑΙ Ο SERVER	17
ΕΙΚΟΝΑ 9 ΑΚΟΛΟΥΘΟΥΝ ΠΟΛΛΑΠΛΕΣ ΧΕΙΡΑΨΙΕΣ ΚΑΙ ΔΗΜΙΟΥΡΓΙΑ SESSION TICKETS.....	17
ΕΙΚΟΝΑ 10 ΔΙΑΔΙΚΑΣΙΑ ΧΕΙΡΑΨΙΑΣ ΣΤΟ TLS 1.2 ΜΕ ΤΟΝ ΑΛΓΟΡΙΘΜΟ ΑΝΤΑΛΛΑΓΗΣ ΚΛΕΙΔΙΩΝ RSA [14]	19
ΕΙΚΟΝΑ 11 Η ΧΕΙΡΑΨΙΑ ΣΤΟ TLS 1.3 ΓΙΝΕΤΑΙ ΜΟΝΟ ΜΕ ΑΛΓΟΡΙΘΜΟ ΑΝΤΑΛΛΑΓΗΣ ΚΛΕΙΔΙΩΝ DIFFIE-HELLMAN [15].....	20
ΕΙΚΟΝΑ 12 ΤΟ TSHARK ΕΜΦΑΝΙΖΕΙ ΜΕΡΙΚΕΣ ΠΛΗΡΟΦΟΡΙΕΣ ΓΙΑ ΤΟ ΚΑΘΕ ΠΑΚΕΤΟ ΠΡΙΝ ΠΑΕΙ ΣΤΑ ΕΠΙΠΕΔΑ, ΟΠΩΣ Ο ΧΡΟΝΟΣ ΠΟΥ ΕΓΙΝΕ CAPTURED ΣΤΟ PCAP ΚΑΙ ΤΟ ΜΕΓΕΘΟΣ ΤΟΥ. ΣΤΟ PROTOCOLS IN FRAME ΦΑΙΝΟΝΤΑΙ ΕΝΑ-ΕΝΑ ΤΑ ΕΠΙΠΕΔΑ ΞΕΚΙΝΩΝΤΑΣ ΑΠΟ ΤΟ ΚΑΤΩΤΑΤΟ (ΠΟΥ ΒΡΙΣΚΕΤΑΙ ΠΙΟ ΚΟΝΤΑ ΣΤΟ DATA LINK), ETHERNET.	22
ΕΙΚΟΝΑ 13 ΕΜΦΑΝΙΣΗ TLS1.3 RECORD TYPE, PROTOCOL TYPE ΚΑΙ VERSION ΜΕΣΩ TSHARK.....	23
ΕΙΚΟΝΑ 14 ΕΜΦΑΝΙΣΗ ΤΟΥ RANDOM BYTES, SESSION ID ΚΑΙ ΤΗ ΜΕΤΑΜΦΙΕΣΗ ΤΟΥ TLS 1.3 ΩΣ TLS 1.2. Η ΚΩΔΙΚΟΠΟΙΗΣΗ ΤΟΥ 1.2 ΕΙΝΑΙ 0303 ΑΝΤΙ ΓΙΑ 0302, ΠΑΡΟΜΟΙΩΣ ΜΕ ΤΟ 1.0.....	23
ΕΙΚΟΝΑ 15 TLS 1.3 ΚΑΙ 1.2 CIPHER SUITES ΠΟΥ ΥΠΟΣΤΗΡΙΖΕΙ Ο CLIENT. Η ΟΝΟΜΑΣΙΑ ΤΩΝ ΚΑΘΕ CIPHER SUITE ΕΙΝΑΙ ΜΟΝΑΔΙΚΗ ΓΙΑ ΤΟ ΚΑΘΕΝΑ ΚΑΙ ΒΡΙΣΚΕΤΑΙ ΣΤΗ ΠΑΡΕΝΘΕΣΗ. ΑΥΤΟ ΕΙΝΑΙ ΤΟ RAW DATA ΠΟΥ ΜΕΤΑΦΕΡΕΙ ΕΝΑ ΠΑΚΕΤΟ ΚΑΙ ΜΠΟΡΕΙ ΝΑ ΣΥΝΕΝΝΟΗΘΕΙ ΜΕ ΤΟΝ SERVER ΓΙΑ ΤΟ ΤΙ ΘΑ ΧΡΗΣΙΜΟΠΟΙΗΘΕΙ.....	24
ΕΙΚΟΝΑ 16 Η ΜΕΘΟΔΟΣ ΣΥΜΠΙΕΣΗΣ ΔΕΝ ΧΡΗΣΙΜΟΠΟΙΕΙΤΑΙ ΣΤΗΝ ΕΚΔΟΣΗ 1.3 ΚΑΙ Η ΤΙΜΗ ΤΗΣ ΕΙΝΑΙ ΠΑΝΤΑ NULL	25
ΕΙΚΟΝΑ 17 ΤΟ ΟΝΟΜΑ ΤΟΥ SERVER ΑΠΟΣΤΕΛΛΕΤΑΙ ΣΑΝ PLAINTEXT, ΧΩΡΙΣ ΚΩΔΙΚΟΠΟΙΗΣΗ.....	25
ΕΙΚΟΝΑ 18 ΟΙ ΜΕΘΟΔΟΙ ΔΗΜΙΟΥΡΓΙΑΣ ΚΛΕΙΔΙΩΝ ΑΠΟ ΕΛΛΕΙΠΤΙΚΗ ΚΑΜΠΥΛΗ ΦΕΡΟΥΝ ΣΥΓΚΕΚΡΙΜΕΝΗ ΟΝΟΜΑΣΙΑ ΣΕ ΠΑΡΕΝΘΕΣΗ. ΓΙΑ ΠΑΡΑΔΕΙΓΜΑ Η x25519 ΜΕΤΑΦΕΡΕΤΑΙ ΩΣ 001D. Η ΜΟΡΦΗ ΤΟΥ ΣΗΜΕΙΟΥ ΠΟΥ ΘΑ ΕΠΙΛΕΧΤΕΙ ΟΡΙΖΕΤΑΙ ΣΤΟ ΑΜΕΣΩΣ ΕΠΟΜΕΝΟ ΠΕΔΙΟ	27
ΕΙΚΟΝΑ 19 ΔΕΝ ΥΠΑΡΧΕΙ SESSION TICKET ΟΠΟΤΕ ΣΤΕΛΝΕΤΑΙ ΜΟΝΟ Η ΚΕΦΑΛΙΔΑ ΓΙΑ ΤΗΝ ΕΠΕΚΤΑΣΗ ΧΩΡΙΣ ΚΑΠΟΙΑ ΤΙΜΗ. Η ΕΠΕΚΤΑΣΗ ALPN ΕΙΝΑΙ 14 BYTES	27
ΕΙΚΟΝΑ 20 Η ΕΠΕΚΤΑΣΗ ΓΙΑ ΤΟ OCSP ΠΙΑΝΕΙ 5 BYTES	28
ΕΙΚΟΝΑ 21 Ο ΚΑΘΕ ΑΛΓΟΡΙΘΜΟΣ ΦΕΡΕΙ ΜΙΑ ΣΥΓΚΕΚΡΙΜΕΝΗ ΟΝΟΜΑΣΙΑ ΣΤΗΝ ΚΕΦΑΛΙΔΑ ΑΥΤΗ ΚΑΙ ΕΜΦΑΝΙΖΕΤΑΙ ΑΠΟ ΤΟ TSHARK ΣΤΙΣ ΠΑΡΕΝΘΕΣΕΙΣ. ΓΙΑ ΠΑΡΑΔΕΙΓΜΑ, Ο ECDSA_SECP256R1_SHA256 ΕΧΕΙ ΟΝΟΜΑΣΙΑ 0403	29
ΕΙΚΟΝΑ 22 ΕΑΝ Ο SERVER ΥΠΟΣΤΗΡΙΖΕΙ ΤΑ GROUP x25519 ΚΑΙ SECP256R1, ΘΑ ΕΠΙΛΕΞΕΙ ΤΟ ΠΡΩΤΟ ΚΑΙ ΘΑ ΠΑΡΕΙ ΤΗ ΤΙΜΗ ΤΟΥ KEY EXCHANGE ΩΣ ΤΟ ΔΗΜΟΣΙΟ ΚΛΕΙΔΙ ΤΟΥ CLIENT	29
ΕΙΚΟΝΑ 23 Ο CLIENT ΕΝΗΜΕΡΩΝΕΙ ΤΟΝ SERVER ΣΕ 5 BYTES ΠΟΙΕΣ ΕΚΔΟΣΕΙΣ ΥΠΟΣΤΗΡΙΖΕΙ. Η ΕΚΔΟΣΗ 1.3 ΕΧΕΙ ΚΩΔΙΚΟΠΟΙΗΣΗ 0304	30
ΕΙΚΟΝΑ 24 ΠΕΔΙΑ ΜΕ "UNKNOWN" ΔΕΝ ΜΠΟΡΕΣΑΝ ΝΑ ΒΡΕΘΟΥΝ ΑΠΟ ΤΟ TSHARK. Ο ΑΛΓΟΡΙΘΜΟΣ HASH ΕΙΝΑΙ ΠΑΝΤΑ SHA(#ΕΚΔΟΣΗΣ Η #BITS KEY).....	31
ΕΙΚΟΝΑ 25 Ο CLIENT ΕΝΗΜΕΡΩΝΕΙ ΤΟΝ SERVER ΟΤΙ ΥΠΟΣΤΗΡΙΖΕΙ ΤΗ ΔΗΜΙΟΥΡΓΙΑ PRE-SHARED KEY ΜΕ ΤΟΝ (EC)DHE. ΤΟ ΜΕΓΕΘΟΣ ΤΟΥ TLS RECORD ΔΕΝ ΜΠΟΡΕΙ ΝΑ ΞΕΠΕΡΝΑ ΤΑ 16385 BYTES.	32
ΕΙΚΟΝΑ 26 ΤΟ ΑΠΟΤΥΠΩΜΑ ΠΟΥ ΑΦΗΝΟΥΜΕ ΣΤΟ ΔΙΑΔΙΚΤΥΟ	32
ΕΙΚΟΝΑ 27 ΕΠΙΠΕΔΑ ΣΤΟ SERVER HELLO ΕΙΝΑΙ ΙΔΙΑ ΜΕ ΤΟ CLIENT HELLO.	33

ΕΙΚΟΝΑ 28 Ο SERVER ΧΡΗΣΙΜΟΠΟΙΕΙ TLS 1.3 ΚΑΙ ΤΗΝ ΜΕΤΑΜΦΙΕΖΕΙ ΩΣ 1.2 ΓΙΑ ΝΑ ΠΕΡΑΣΕΙ ΤΟΥΣ ΕΛΕΓΧΟΥΣ ΣΥΜΒΑΤΟΤΗΤΑΣ. ΈΧΕΙ ΠΕΡΙΕΧΟΜΕΝΟ 118 BYTES.....	34
ΕΙΚΟΝΑ 29 Η ΤΙΜΗ ΤΟΥ RANDOM ΠΟΥ ΔΗΜΙΟΥΡΓΕΙ ΚΑΙ ΤΟ ΕΧΟ ΤΟΥ SESSION ID ΠΟΥ ΕΣΤΕΙΛΕ Ο CLIENT	34
ΕΙΚΟΝΑ 30 ΑΠΟΦΑΣΙΣΤΗΚΕ ΑΠΟ ΤΟΝ SERVER ΝΑ ΧΡΗΣΙΜΟΠΟΙΗΘΕΙ Ο ΑΛΓΟΡΙΘΜΟΣ AES ΜΕ 128 BITS ΚΛΕΙΔΙ ΚΡΥΠΤΟΓΡΑΦΗΣΗΣ ΣΕ GCM MODE ΚΑΙ ΑΥΘΕΝΤΙΚΟΠΟΙΗΣΗ ΤΩΝ ΜΗΝΥΜΑΤΩΝ ΜΕ SHA256.	35
ΕΙΚΟΝΑ 31 ΟΙ ΕΠΕΚΤΑΣΕΙΣ ΠΟΥ ΧΡΕΙΑΖΕΤΑΙ ΝΑ ΣΤΕΙΛΕΙ ΕΙΝΑΙ Η ΕΚΔΟΣΗ TLS ΠΟΥ ΘΑ ΣΥΜΦΩΝΗΣΟΥΝ ΚΑΙ ΤΟ ΔΗΜΟΣΙΟ ΚΛΕΙΔΙ ΤΟΥ SERVER ΜΕ ΤΟ GROUP ΠΟΥ ΤΟ ΠΑΡΗΓΑΓΕ.	35
ΕΙΚΟΝΑ 32 ΤΟ ΑΠΟΤΥΠΩΜΑ ΤΟΥ SERVER ΕΙΝΑΙ ΠΑΝΤΑ ΜΙΚΡΟΤΕΡΟ ΑΠΟ ΤΟΥ CLIENT.....	35
ΕΙΚΟΝΑ 33 Η ΕΚΔΟΣΗ ΕΙΝΑΙ ΚΡΥΠΤΟΓΡΑΦΗΜΕΝΗ ΜΕ ΤΟ SHARED SECRET KEY ΚΑΙ ΤΟ ΜΗΝΥΜΑ ΣΤΕΛΝΕΤΑΙ ΩΣ CHANGE CIPHER SPEC ENCRYPTED.....	36
ΕΙΚΟΝΑ 34 ΠΡΙΝ ΤΗ ΧΡΗΣΗ ΤΟΥ SHARED SECRET KEY, ΑΝΤΙ ΓΙΑ ΚΑΝΟΝΙΚΕΣ ΟΝΟΜΑΣΙΕΣ ΣΤΙΣ ΕΠΕΚΤΑΣΕΙΣ ΕΜΦΑΝΙΖΕΙ ΜΟΝΟ ENCRYPTED EXTENSIONS.	37
ΕΙΚΟΝΑ 35 ΤΟ ΠΑΚΕΤΟ ΠΟΥ ΣΤΕΛΝΕΤΑΙ ΕΙΝΑΙ ΑΜΕΣΩΣ ΕΠΟΜΕΝΟ ΑΠΟ ΤΟ ΠΡΟΗΓΟΥΜΕΝΟ. ΔΕΝ ΜΕΣΟΛΑΒΕΙ ΚΑΝΕΝΑ ΑΛΛΟ ΚΑΙ ΕΧΕΙ ΜΕΓΕΘΟΣ 1209 BYTES.....	37
ΕΙΚΟΝΑ 36 ΛΙΓΟ ΠΙΟ ΜΕΤΑ, ΣΤΟ ΙΔΙΟ ΠΑΚΕΤΟ, ΣΤΟ TCP LAYER ΠΑΡΑΤΗΡΟΥΜΕ ΟΤΙ ΥΠΗΡΧΑΝ 4 PAYLOADS ΤΑ ΟΠΟΙΑ ΣΥΝΑΡΜΟΛΟΓΗΘΗΚΑΝ ΚΑΙ ΤΟ ΜΕΓΕΘΟΣ ΑΥΞΗΘΗΚΕ ΣΤΑ 6187 BYTES.	38
ΕΙΚΟΝΑ 37 Η ΑΛΥΣΙΔΑ ΕΙΝΑΙ 6161 BYTES. ΣΤΟ ΠΕΔΙΟ CERTIFICATE ΦΑΙΝΕΤΑΙ ΤΟ ΠΙΣΤΟΠΟΙΗΤΙΚΟ ΤΟΥ ΙΟΝΙΟ.GR ΣΕ ΔΕΚΑΕΞΑΔΙΚΗ ΜΟΡΦΗ ΚΑΙ ΣΕ PLAINTEXT	38
ΕΙΚΟΝΑ 38 ΤΑ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΤΟΥ CA ΕΜΦΑΝΙΖΟΝΤΑΙ ΣΤΟ ΠΙΣΤΟΠΟΙΗΤΙΚΟ ΤΟΥ ΙΟΝΙΟ.GR.....	39
ΕΙΚΟΝΑ 39 ΟΛΑ ΑΥΤΑ ΘΑ ΕΛΕΓΧΘΟΥΝ ΑΠΟ ΤΟΝ CLIENT ΓΙΑ ΝΑ ΕΠΙΒΕΒΑΙΩΘΕΙ Η ΤΑΥΤΟΤΗΤΑ ΤΟΥ SERVER	39
ΕΙΚΟΝΑ 40 ΤΟ ΔΗΜΟΣΙΟ ΚΛΕΙΔΙ ΤΟΥ ΙΟΝΙΟ.GR ΜΠΑΙΝΕΙ ΣΤΟ ΠΕΔΙΟ ΓΙΑ ΤΟ SUBJECTPUBLICKEY.....	40
ΕΙΚΟΝΑ 41 EXTENSIONS ΤΟΥ ΠΙΣΤΟΠΟΙΗΤΙΚΟΥ	40
ΕΙΚΟΝΑ 42 Η ΤΕΛΕΥΤΑΙΑ ΕΠΕΚΤΑΣΗ ΔΕΙΧΝΕΙ ΓΙΑ ΠΟΙΑ DOMAIN ΕΠΑΛΛΗΘΕΥΕΙ ΤΟ ΠΙΣΤΟΠΟΙΗΤΙΚΟ. ΠΡΟΣΟΧΗ ΟΤΙ ΥΠΑΡΧΟΥΝ ΔΥΟ ΟΝΟΜΑΤΑ. ΚΑΘΩΣ ΤΟ WWW ΜΠΟΡΕΙ ΝΑ ΠΑΡΑΛΗΦΘΕΙ ΟΤΑΝ ΨΑΧΝΟΥΜΕ ΜΙΑ DOMAIN, ΚΑΤΑΓΡΑΦΟΝΤΑΙ ΚΑΙ ΟΙ ΔΥΟ ΤΡΟΠΟΙ ΠΟΥ ΜΠΟΡΕΙ ΝΑ ΕΜΦΑΝΙΣΤΕΙ	41
ΕΙΚΟΝΑ 43 Ο SERVER ΑΠΑΝΤΗΣΕ ΣΤΗ ΠΡΩΤΗ ΕΠΙΛΟΓΗ ΤΟΥ CLIENT (ΕΙΚΟΝΑ 14) ΝΑ ΤΟΥ ΣΤΕΙΛΕΙ ΤΟ ΠΙΣΤΟΠΟΙΗΤΙΚΟ ΜΕ ΤΟΝ ΑΛΓΟΡΙΘΜΟ ΥΠΟΓΡΑΦΗΣ ECDSA_SECP256R1_SHA256. ΤΟ ΠΕΔΙΟ ENCRYPTED ΕΜΦΑΝΙΖΕΙ ΤΟ ΚΡΥΠΤΟΓΡΑΦΗΜΕΝΟ ΠΙΣΤΟΠΟΙΗΤΙΚΟ.....	42
ΕΙΚΟΝΑ 44 Η USERTRUST ΕΧΕΙ ΤΗΝ ΕΔΡΑ ΤΗΣ ΣΤΙΣ ΗΝΩΜΕΝΕΣ ΠΟΛΙΤΕΙΑΣ ΑΜΕΡΙΚΗΣ, ΣΤΟ NEW JERSEY.....	43
ΕΙΚΟΝΑ 45 ΣΗΜΕΙΩΣΗ ΟΤΙ ΤΟ ΠΙΣΤΟΠΟΙΗΤΙΚΟ ΤΗΣ GEANT ΕΧΕΙ ΙΣΧΥ ΑΠΟ ΤΟ 2020 ΕΩΣ ΤΟ 2033, ΣΕ ΣΧΕΣΗ ΜΕ ΤΗ ΤΕΛΙΚΗ DOMAIN ΙΟΝΙΟ.GR ΠΟΥ ΠΡΕΠΕΙ ΝΑ ΑΝΑΝΕΩΝΕΤΑΙ ΚΑΘΕ ΧΡΟΝΟ.	43
ΕΙΚΟΝΑ 46 ΔΗΜΟΣΙΟ ΚΛΕΙΔΙ ΤΗΣ GEANT	43
ΕΙΚΟΝΑ 47 ΠΛΗΡΟΦΟΡΙΕΣ ΓΙΑ ΤΟ ΠΙΣΤΟΠΟΙΗΤΙΚΟ ΤΗΣ GEANT	44
ΕΙΚΟΝΑ 48 ΠΛΗΡΟΦΟΡΙΕΣ ΓΙΑ ΤΟ ΠΙΣΤΟΠΟΙΗΤΙΚΟ ΤΗΣ GEANT	44
ΕΙΚΟΝΑ 49 Η AAA CERTIFICATE SERVICES ΑΝΗΚΕΙ ΣΤΟΝ ΟΡΓΑΝΙΣΜΟ COMODO CA LIMITED, ΜΕ ΕΔΡΑ ΤΗΝ ΕΠΑΡΧΙΑ ΤΟΥ SALFORD, Η ΟΠΟΙΑ ΒΡΙΣΚΕΤΑΙ ΣΤΗ ΠΟΛΗ ΤΗΣ GREATER MANCHESTER ΤΗΣ ΜΕΓΑΛΗΣ ΒΡΕΤΑΝΙΑΣ.	45
ΕΙΚΟΝΑ 50 ΠΙΣΤΟΠΟΙΗΤΙΚΟ ΤΗΣ USERTRUST.....	45
ΕΙΚΟΝΑ 51 ΔΗΜΟΣΙΟ ΚΛΕΙΔΙ ΤΟΥ ΠΙΣΤΟΠΟΙΗΤΙΚΟΥ.....	46
ΕΙΚΟΝΑ 52 ΠΛΗΡΟΦΟΡΙΕΣ ΓΙΑ ΤΟ ΠΙΣΤΟΠΟΙΗΤΙΚΟ ΤΗΣ USERTRUST	46
ΕΙΚΟΝΑ 53 ΠΙΣΤΟΠΟΙΗΤΙΚΟ ΤΗΣ AAA CERTIFICATE SERVICES ΚΑΙ ΙΣΧΥ	47
ΕΙΚΟΝΑ 54 ΟΤΑΝ ΦΤΑΣΟΥΜΕ ΣΤΗΝ ΠΡΩΤΗ CA ΣΤΗΝ ΑΛΥΣΙΔΑ, ΑΥΤΗ ΠΑΝΤΑ ΠΙΣΤΟΠΟΙΕΙ ΤΟΝ ΕΑΥΤΟ ΤΗΣ ΜΕ SELF-SIGNED CERTIFICATE	47
ΕΙΚΟΝΑ 55 ΔΗΜΟΣΙΟ ΚΛΕΙΔΙ ΤΗΣ CA.....	47
ΕΙΚΟΝΑ 56 ΛΕΠΤΟΜΕΡΕΙΕΣ ΤΟΥ SELF-SIGNED CERTIFICATE ΓΙΑ ΤΗΝ AAA CERTIFICATE SERVICES	48
ΕΙΚΟΝΑ 57 ΣΤΟ CERTIFICATE VERIFY ΜΠΑΙΝΕΙ Ο ΑΛΓΟΡΙΘΜΟΣ ΥΠΟΓΡΑΦΗΣ ΤΟΥ, ΤΟ ΜΕΓΕΘΟΣ ΤΗΣ ΥΠΟΓΡΑΦΗΣ ΚΑΙ Η ΥΠΟΓΡΑΦΗ ΤΟΥ. ΑΠΟ ΑΥΤΟ ΤΟ ΠΕΔΙΟ Ο CLIENT ΧΡΕΙΑΖΕΤΑΙ ΤΟ ΟΝΟΜΑ ΤΟΥ ΑΛΓΟΡΙΘΜΟΥ ΚΑΙ ΤΗΝ ΥΠΟΓΡΑΦΗ ΓΙΑ ΝΑ ΕΠΑΛΛΗΘΕΥΣΕΙ ΤΗ ΤΑΥΤΟΤΗΤΑ ΤΟΥ ΙΟΝΙΟ.GR.....	49
ΕΙΚΟΝΑ 58 ΤΟ FINISHED ΜΗΝΥΜΑ ΕΙΝΑΙ ΠΑΝΤΑ ΚΡΥΠΤΟΓΡΑΦΗΜΕΝΟ ΚΑΙ ΕΜΦΑΝΙΖΕΤΑΙ ΩΣ «ENCRYPTED APPLICATION DATA».....	49

ΕΙΚΟΝΑ 59 Ο CLIENT ΣΤΕΛΝΕΙ ΤΟ ΜΗΝΥΜΑ CHANGE CIPHER SPEC ΚΑΙ FINISHED ΩΣ ΤΑ ΤΕΛΕΥΤΑΙΑ ΜΗΝΥΜΑΤΑ ΓΙΑ ΤΟ CLIENT HELLO.....	50
ΕΙΚΟΝΑ 60 Ο ΣΥΝΔΕΣΜΟΣ ΠΟΥ ΖΗΤΑΕΙ ΕΙΝΑΙ HTTPS://IONIO.GR/. ΣΤΟ HTTP ΜΗΝΥΜΑ ΦΑΙΝΟΝΤΑΙ ΤΑ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΤΗΣ ΕΦΑΡΜΟΓΗΣ ΑΠΟ ΤΗΝ ΟΠΟΙΑ ΣΥΝΔΕΕΤΑΙ Ο CLIENT ΚΑΙ ΠΕΡΑΙΤΕΡΩ. MOZILLA FIREFOX ΑΠΟ UBUNTU LINUX.	51
ΕΙΚΟΝΑ 61 ΤΑ ΣΥΓΓΕΚΡΙΜΕΝΑ SESSION TICKETS ΕΧΟΥΝ ΙΣΧΥ ΓΙΑ 5 ΛΕΠΤΑ.	52
ΕΙΚΟΝΑ 62 ΤΟ TSHARK ΔΕΝ ΔΕΙΧΝΕΙ ΠΟΙΟ ΕΙΝΑΙ ΤΟ ΠΕΡΙΕΧΟΜΕΝΟ ΤΟΥ HTTP ΚΑΙ ΤΟ ΚΡΑΤΑΕΙ ΚΡΥΠΤΟΓΡΑΦΗΜΕΝΟ. ΣΥΜΦΩΝΑ ΜΕ ΤΟ ΠΕΔΙΟ LENGTH, ΤΟ ΠΡΩΤΟ RECORD LAYER ΠΕΡΙΕΧΕΙ 8209 BYTES ΑΠΟ ΔΕΔΟΜΕΝΑ, ΤΟ ΟΠΟΙΟ ΜΑΛΛΟΝ ΕΙΝΑΙ Η ΑΡΧΙΚΗ ΣΕΛΙΔΑ ΠΟΥ ΕΠΕΛΕΞΕ ΝΑ ΑΝΟΙΞΕΙ Ο CLIENT.....	52

Κεφάλαιο Α. Εισαγωγή

Α'1. Περίληψη

Η συγκεκριμένη πτυχιακή εργασία θα ασχοληθεί με τη μελέτη και ανάλυση της πιο πρόσφατης έκδοσης του TLS (1.3) και την παρουσίαση πραγματικών δεδομένων μέσω κώδικα Python 3. Στην αρχή της εκπόνησης της πτυχιακής έγινε μελέτη πάνω στο αντικείμενο του πρωτοκόλλου TLS και τι προσφέρει η νέα έκδοση. Με μια γενική κατανόηση που υπήρξε, είχα προχωρήσει στην γραφή του κώδικα που θα έπιανε την κίνηση όταν πλοηγούμε σε ιστοσελίδες και θα έβρισκε δεδομένα για το TLS. Έπειτα θα έκανε τις απαραίτητες ενέργειες για να μπορώ να διαβάσω αυτά τα δεδομένα χωρίς κρυπτογράφηση. Στο πρώτο κεφάλαιο γίνεται εισαγωγή σε βασικές έννοιες που θα χρησιμοποιούνται σε επόμενα κεφάλαια, όπως ο λόγος ύπαρξης του πρωτοκόλλου και γιατί κυκλοφόρησε η έκδοση 1.3. Τα επίπεδα στο TCP/IP [a] και το φορτίο που κουβαλούν από τη μια μεριά του κόσμου στην άλλη για να μεταφέρουν το μήνυμα ενός χρήστη αναφέρονται στο κείμενο καθώς το TLS βρίσκεται εδώ. Η δομή ενός πακέτου είναι προκαθορισμένη με τα επίπεδα του TCP/IP να το καθοδηγούν. Στο κύριο σώμα αναπτύσσονται ορολογίες που θα χρησιμεύσουν στη κατανόηση της λειτουργίας του πρωτοκόλλου TLS και στην ανάλυσή του μέσω κώδικα. Θα αναλυθεί η συσχέτιση του TLS με το TCP/IP και από τί αποτελείται. Τα cipher suites είναι ο κύριος άξονας που κάνει το TLS 1.3 να επιτυγχάνει απόλυτη μελλοντική ασφάλεια με χρήση ενός μόνο αλγορίθμου, του Diffie-Hellman. Η νέα έκδοση έρχεται με δύο νέα χαρακτηριστικά, την αποστολή δεδομένων χωρίς χειραψία (handshake) για ταχύτητα και την ασφάλεια εναντίον επιθέσεων με μυστικά κλειδιά που διέρρευσαν. Υπάρχουν ανησυχίες για την ασφάλεια της αποστολής δεδομένων πριν τη χειραψία, καθώς ανταλλάσσεται για τη ταχύτερη εγκαθίδρυση σύνδεσης και, έτσι, πολλοί πάροχοι δεν την επιτρέπουν. Εμείς θα περιγράψουμε τι ακριβώς είναι αυτό το χαρακτηριστικό, ποια είναι τα μειονεκτήματα και ποια είναι τα οφέλη από τη χρήση τους σε πιο λεπτομερή βαθμό στο κεφάλαιο Β. Στο τελευταίο κεφάλαιο παρουσιάζονται τα συμπεράσματα της έρευνας και γίνεται ένα κλείσιμο του θέματος κάνοντας αναφορά σε μελλοντικά χαρακτηριστικά και στόχους για το πρωτόκολλο TLS.

Α'2. Πρόλογος και Ευχαριστίες

Η συγκεκριμένη πτυχιακή εργασία είναι μια προσπάθεια να αναλύσει το πρωτόκολλο του TLS μέσω κώδικα Python και όχι με το παραδοσιακό εργαλείο του Wireshark. Πιστεύω ότι με αυτό το τρόπο γίνεται πιο ευνόητο τι ενέργειες κάνει το TLS για να ασφαλίσει μια σύνδεση.

Θα ήθελα να ευχαριστήσω τη μητέρα μου που με στήριξε όλα αυτά τα χρόνια για να φτάσω στο σημείο να σπουδάζω και να κάνω πτυχιακή εργασία και τους καθηγητές του τμήματός μου που μου δίδαξαν όσα χρησίμευσαν στη πτυχιακή και στη ζωή.

Θεσσαλονίκη, Σεπτέμβριος 2023

Καλαθάς Αλέξανδρος.

Α'3. Επίπεδα TCP/IP και δομή πακέτου

Ενα πακέτο απαρτίζεται από το *φορτίο* (payload) και *επίπεδα* (layers) τα οποία περιέχουν *μεταδεδομένα* (metadata) και εξηγούν στον παραλήπτη σε ποιον προορίζεται το πακέτο (διευθύνσεις IP και MAC), από ποια επίπεδα αποτελείται, τι είδους payload μεταφέρεται, ποια εφαρμογή θα το αναλάβει και άλλα που θα αναλυθούν αργότερα στο κεφάλαιο. Τα επίπεδα στη *σουίτα πρωτοκόλλου διαδικτύου* (Internet Protocol Suite - TCP/IP) [1] είναι τέσσερα και παρουσιάζουν τη διαδικασία «*τυλίγματος*» (wrapping) του payload με metadata, τη δημιουργία του πακέτου και τη διαδρομή του από το ανώτατο *επίπεδο εφαρμογής* (Application Layer) στο κατώτερο, το *επίπεδο σύνδεσης* (Link Layer) και έπειτα την αποσυναρμολόγηση του κάθε επιπέδου από τον παραλήπτη, ξεκινώντας από το κατώτερο επίπεδο και φτάνοντας στο ανώτατο. Το *φυσικό επίπεδο* (Physical Layer) στο οποίο μεταφέρονται ροές από bits περιγράφεται στο θεωρητικό *μοντέλο διασύνδεσης ανοικτών συστημάτων* (Open Systems Interconnection model - OSI) [b], αλλά δεν θα χρειαστεί για το θέμα αυτής της πτυχιακής και θα χρησιμοποιήσουμε το TCP/IP.

Α'3.1. Payload

Ένα payload που είναι έτοιμο να σταλθεί είναι το πραγματικό περιεχόμενο που θέλει να στείλει ο χρήστης. Για παράδειγμα ένα *απλό κείμενο* (plaintext) ή μια αίτηση στον *εξυπηρετητή* (server) για να συνδεθεί σε *μέσο κοινωνικής δικτύωσης* (social media). Το payload τυλίγεται με επίπεδα που περιέχουν metadata, χρήσιμες πληροφορίες τις οποίες χρειάζονται για να καταλάβει ο παραλήπτης με ποιο τρόπο θα

αποσυναρμολογηθεί το πακέτο ώστε να φτάσει στο περιεχόμενο που τον ενδιαφέρει (αυτό που του ζήτησε ο αποστολέας).

A'3.2. Application Layer

Η διαδικασία ξεκινάει από το ανώτατο επίπεδο, το Application Layer όπου βρίσκονται πολυχρησιμοποιημένα πρωτόκολλα όπως το *σύστημα ονομάτων περιοχής* (Domain Name System - DNS), *πρωτόκολλο μεταφοράς υπερκειμένου* (HyperText Transfer Protocol – HTTP) και *ασφαλές πρωτόκολλο μεταφοράς υπερκειμένου* (HyperText Transfer Protocol Secure – HTTPS) με χρήση του *ασφαλούς επιπέδου υποδοχής* (Secure Sockets Layer – SSL) ή *ασφάλειας επιπέδου διακίνησης* (Transport Layer Security - TLS) [c] ανάλογα την έκδοση (περισσότερα για αυτό σε επόμενα κεφάλαια) και άλλα. Το DNS είναι υπεύθυνο για τη παραλαβή αιτήματος από έναν χρήστη για να βρει ένα συγκεκριμένο server (π.χ. ionio.gr), τη μετάφραση αυτού σε μια διεύθυνση IP (Internet Protocol) και την αναζήτηση σε μια λίστα από πολλές *περιοχές* (domains) Το HTTP περιέχει το περιεχόμενο που βλέπουμε σε μια σελίδα όταν την επισκεπτόμαστε, όπως *υπερσυνδέσμους* (hyperlinks) οι οποίοι μπορούν να πατηθούν με το ποντίκι για να ανακατευθύνουν τον χρήστη σε άλλη σελίδα.

A'3.3. Transport Layer

Μετά το Application Layer έρχεται το *επίπεδο μεταφοράς* (Transport Layer) το οποίο ορίζει το πρωτόκολλο που θα χρησιμοποιηθεί για τη μεταφορά των πακέτων στο διαδίκτυο. Το πιο χρησιμοποιημένο είναι το *πρωτόκολλο ελέγχου μετάδοσης* (Transmission Control Protocol – TCP). Πάνω στο TCP βρίσκεται και το TLS. Πριν φορτώσει μια σελίδα και πριν εφαρμοστεί το TLS, το TCP θα επικοινωνήσει με τον server για να δει εάν απαντάει (δηλαδή εάν λειτουργεί η σελίδα και δεν πετάει κάποιο σφάλμα) και έπειτα θα εγκαθιδρύσει μια ασφαλής σύνδεση μαζί του. Ένα άλλο γνωστό πρωτόκολλο είναι το *πρωτόκολλο διαγράμματος χρήστη* (User Datagram Protocol – UDP). Αυτό είναι πιο γρήγορο από το TCP γιατί δεν ελέγχει αν χάθηκαν πακέτα και δεν τα αριθμεί ώστε να διατηρήσει τον έλεγχο της ροής των πακέτων, το οποίο το κάνει λιγότερο αξιόπιστο. Πάνω σε αυτό βασίστηκε το πρωτόκολλο QUIC, το οποίο στοχεύει να αντικαταστήσει το TCP σε ευρεία χρήση του. Το QUIC είναι βασισμένο στη λογική του UDP να είναι λιτό, έχει ασφάλεια, χρησιμοποιεί HTTP/3 και είναι πιο γρήγορο από τη χρήση TCP και TLS μαζί. Στο μέλλον πρόκειται να γίνει

το κανονικό (standard) τρόπο μετάδοσης δεδομένων στο διαδίκτυο, αλλά προς το παρόν ας μιλήσουμε για το TLS 1.3.

A'3.4. Internet Layer

Πριν το τελευταίο επίπεδο υπάρχει αυτό που εμπεριέχει το πιο γνωστό πρωτόκολλο, αυτό του *διαδικτύου* (Internet Protocol - IP). Μαζί με το TCP δημιουργούν τη TCP/IP suite. Το Internet Layer είναι υπεύθυνο για τη ψηφιακή δρομολόγηση των πακέτων στη σωστή διεύθυνση IP. Αν η διεύθυνση δεν βρεθεί, πρέπει να χειριστεί το σφάλμα και το πακέτο αναλόγως. Όπως και με τα άλλα επίπεδα, όταν τυλίγει το πακέτο που του φτάνει τη δική του *κεφαλίδα* (header), το στέλνει στο αμέσως επόμενο επίπεδο, το *επίπεδο σύνδεσης* (Link Layer).

A'3.5. Link Layer

Μια ομάδα από πρωτόκολλα. Είναι το αντίστοιχο επίπεδο του Physical Layer στο μοντέλο OSI. Ως το κατώτερο επίπεδο στη TCP/IP suite, τυλίγει το πακέτο με τη δική του κεφαλίδα και είναι υπεύθυνο για τη φυσική δρομολόγηση του ανάλογα το πρωτόκολλο που χρησιμοποιείται. Αν είναι Ethernet, περνάει ως ροή bits μέσα από καλώδια ενώ αν είναι ασύρματο (π.χ. IEEE 802.11 WiFi), στέλνεται σε *κόμβους δικτύου* (network nodes).

Κεφάλαιο Β. Μελέτη και ανάλυση του TLS 1.3

Β'1. Πως βοηθάει η εφαρμογή του TLS

Στο μοντέλο του OSI υπάρχουν επίπεδα τα οποία περιγράφουν τη διαδρομή που κάνει ένα πακέτο για να δημιουργηθεί, σταλθεί, παραληφθεί και αποσυναρμολογηθεί σε μια συζήτηση μεταξύ πελάτη (client) και server. Το TLS εφαρμόζεται καθημερινά στις περισσότερες ιστοσελίδες του παγκοσμίου διαδικτύου. Πριν καν φορτώσει μια σελίδα και ο χρήστης δει το περιεχόμενο, συμφωνείται με ποιο τρόπο θα ασφαλισθεί η συζήτηση. Ποια έκδοση TLS θα χρησιμοποιηθεί, ποιος αλγόριθμος κρυπτογράφησης, πως θα επαληθευτεί το πιστοποιητικό του server και άλλα που αναλύονται σε επόμενο υποκεφάλαιο. Όταν ένας χρήστης μπαίνει σε μια ιστοσελίδα ή κάνει μια οποιαδήποτε σύνδεση, μεταφέρονται πακέτα. Χωρίς την εφαρμογή του TLS στη σύνδεση, οποιοσδήποτε ακούει τη συζήτηση μεταξύ του client και του server μπορεί να υποκλέψει ό,τι μεταφέρεται, συμπεριλαμβανομένων ευαίσθητων πληροφοριών όπως γενετικά και βιομετρικά στοιχεία, πολιτικές, φυλετικές και θρησκευτικές απόψεις και να διαβάσει το περιεχόμενο τους [2]. Το TLS είναι ο μηχανισμός που μετατρέπει μιας ιστοσελίδας το HTTP σε HTTPS, εξασφαλίζοντας στο χρήστη ότι προσφέρεται ένα (σχετικά) ασφαλές περιβάλλον.

Β'1.1. Ο λόγος κυκλοφορίας μιας νέας έκδοσης

Το TLS έχει διαδεχτεί το πρωτόκολλο SSL καθώς με τη πάροδο του χρόνου η τεχνολογία είχε φτάσει σε σημείο να διαπερνά τις άμυνες του SSL και χρειάστηκε μια νέα λογική για την προστασία της σύνδεσης. Η κυκλοφορία της έκδοσης 1.3 προήλθε από την ανάγκη να εξαλείψουμε το πρόβλημα μιας μελλοντικής επίθεσης σε συζητήσεις που *συλλήφθηκαν* (captured) στο παρελθόν και για τις οποίες το μυστικό κρυπτογράφησης τους (δηλαδή το ιδιωτικό κλειδί του server) είχε γνωστοποιηθεί. Η ανταλλαγή κλειδιών σε εκδόσεις πριν το 1.3 μπορούσε να γίνει με την τεχνική του αλγορίθμου RSA, στον οποίο ο client παράγει ένα κλειδί (premaster key – PSK), ζητάει από τον server το δημόσιο κλειδί του και κρυπτογραφεί το PSK. Το στέλνει στον server ο οποίος είναι ο μόνος που θα πρέπει να έχει το ιδιωτικό κλειδί του ώστε

να το αποκρυπτογραφήσει. Ο client και ο server καταλήγουν σε ένα κοινό μυστικό κλειδί. Στη περίπτωση που το ιδιωτικό κλειδί διαρρεύσει, ένας επιτιθέμενος χρησιμοποιεί το ιδιωτικό κλειδί του server για να αποκρυπτογραφήσει το PSK και να βρει το κοινό μυστικό για να διαβάσει παλιές συζητήσεις ή να υπογράψει το πιστοποιητικό του server και να τον προσποιείται. Επομένως καταλήγει στην έκθεση μεγάλων ποσοτήτων προσωπικών δεδομένων των clients με ανυπολόγιστη ζημιά στη φήμη και τα οικονομικά του ιδιοκτήτη του server. Σε αντίθεση με τις προηγούμενες εκδόσεις, το TLS 1.3 προσφέρει πάντα *τέλεια μελλοντική μυστικότητα* (Perfect Forward Secrecy – PFS). Αυτό γίνεται με τη δημιουργία ενός συμμετρικού κλειδιού από κλειδιά που αλλάζουν συχνά, και τα γνωρίζουν μόνο ο client και ο server. Αρχικά δημιουργούνται *κλειδιά συνόδου* (session keys) για κάθε *σύνοδο* (session) και με αυτά υπολογίζεται ένα κοινό μυστικό κλειδί για την κρυπτογράφηση και αποκρυπτογράφηση δεδομένων [3]. Μια σύνοδος ορίζεται από τον server και μπορεί να είναι ‘συμβάν ή χρόνος από τη στιγμή που έφυγε ο χρήστης από την ιστοσελίδα’.

B’2. Εισαγωγή στο TLS 1.3

Το TLS 1.3 προτάθηκε από την Internet Engineering Task Force (IETF), ένας οργανισμός εθελοντών που έχει αναλάβει την ανάπτυξη και εκσυγχρόνιση προτύπων για το παγκόσμιο διαδίκτυο, όπως το TCP/IP στο οποίο βαδίζει το TLS [4]. Τα χαρακτηριστικά και διαφορές της κάθε έκδοσης με τη προηγούμενη αναγράφονται σε έγγραφο και είναι δημόσια διαθέσιμο για ανάγνωση. Οι προδιαγραφές του κάθε πρωτοκόλλου φέρουν συγκεκριμένο μέρος κώδικα και κανόνες που είτε πρέπει είτε είναι συνετό να υλοποιηθεί από τον server. Περιγράφει το τρόπο που θα γίνει η χειραγία και είναι αυτό που αποτελεί τη νέα έκδοση. Το έγγραφο για την έκδοση 1.3 δημοσιεύτηκε τον Αύγουστο του 2018 και εκτός από τον συγγραφέα, Eric Rescorla ο οποίος ήταν συνδεδεμένος με τη Mozilla εκείνη τη στιγμή, αναφέρει και τους συνεισφέροντες σε αυτό το έργο στο τέλος του εγγράφου [5]. Σύμφωνα με την F5 Labs [6], αυτή τη στιγμή το 63% του ενός εκατομμυρίου πιο διαδεδομένων server προτιμούν την έκδοση 1.3 και το 95% όλων των περιηγητών υποστηρίζουν τη νέα έκδοση.

B’2.1. TLS 1.2 και 1.3 cipher suites

Στην έκδοση TLS 1.2 υπάρχει η δυνατότητα να χρησιμοποιηθούν 37 *κρυπτογραφικές ακολουθίες* (cipher suites), για τις οποίες δεν προσφέρουν όλες PFS λόγω της χρήσης

του αλγορίθμου RSA (Rivest–Shamir–Adleman) για ανταλλαγή και συμφωνία συμμετρικού κλειδιού κρυπτογράφησης [7]. Για αυτό περίπου οι μισές μόνο που χρησιμοποιούν (Elliptic Curve) ephemeral Diffie-Hellman ή (EC)DHE έχουν PFS. Για να κρατήσουμε τη λίστα μικρή και να βεβαιωθούμε ότι θα γίνεται πάντα χρήση ενός cipher suite που προσφέρει PFS, στην έκδοση 1.3 καταργήθηκαν και υποστηρίζονται πλέον 5 ως η standard μέθοδος με τις οποίες καθιερώνει την ασφάλεια της σύνδεσης. Η συμφωνία μεταξύ client και server γίνεται πιο γρήγορα.

1. TLS_AES_128_CCM_SHA256
2. TLS_AES_128_CCM_8_SHA256
3. TLS_AES_128_GCM_SHA256
4. TLS_AES_256_GCM_SHA384
5. TLS_CHACHA20_POLY1305_SHA256

B'2.2. Εξήγηση των cipher suites του TLS 1.3

Ανάλογα την επιλογή του server, η cipher suite που θα χρησιμοποιηθεί αλλάζει. Ο αλγόριθμος κρυπτογράφησης των μηνυμάτων είναι είτε ο AES (Advanced Encryption Standard) για μηνύματα με δεδομένα σε block είτε ο ChaCha20-Poly1305 για *ροή δεδομένων* (data stream). Για τον AES χρησιμοποιείται κλειδί μεγέθους είτε 128 είτε 256 bits και κρυπτογραφεί δεδομένα τα οποία χωρίζονται και αποστέλλονται σε σταθερού μεγέθους μπλοκ . Ο ChaCha20-Poly1305 συνδυάζει τον αλγόριθμο ChaCha20 με τον αυθεντικοποιητή Poly1305 και κρυπτογραφεί μια ροή (και για αυτό λέγεται stream cipher) αντί για ξεχωριστά μπλοκ, χρησιμοποιώντας ένα συμμετρικό κλειδί, ή αλλιώς *κλειδί ροής* (keystream). Φορτώνεται ένα plaintext ως μια συνεχή σειρά από bits στον αλγόριθμο και κρυπτογραφούνται ένα-ένα. Αν και η μέθοδος είναι γρήγορη, το κάθε bit αντιστοιχεί σε ένα συγκεκριμένο *κρυπτογραφημένο κείμενο* (ciphertext). Αυτό σημαίνει ότι εάν ένας επιτιθέμενος πάρει πρόσβαση στο plaintext και το αντίστοιχο ciphertext, μπορεί να συμπεράνει το keystream [8]. Για την αυθεντικοποίηση και εμπιστευτικότητα των μηνυμάτων (κρυπτογράφηση/αποκρυπτογράφηση) στον AES χρησιμοποιείται ο *τρόπος λειτουργίας* (mode of operation) είτε GCM (Galois/Counter Mode) είτε CCM (Counter with Cipher Block Chaining-Message Authentication Code). Η *συνάρτηση κατακερματισμού* (hash function) SHA256 και SHA384 χρησιμοποιείται για την

μετατροπή ενός απλού μηνύματος σε γραμματοσειρά που είναι διαφορετική από το μήνυμα. Το κάθε μήνυμα έχει πάντα το ίδιο αποτέλεσμα hash οπότε αυτή η συνάρτηση χρησιμοποιείται για να βεβαιωθεί ότι το μήνυμα δεν έχει πειραχτεί μέχρι να φτάσει στον παραλήπτη (δηλαδή αν κάποιος επιτιθέμενος υπέκλεψε το μήνυμα και το άλλαξε). Επιπλέον, καθώς η γραμματοσειρά που δημιουργείται είναι πάντα σταθερού μεγέθους (είτε 256 είτε 384 bits), είναι χρήσιμη για εξοικονόμηση δικτύου και χρόνου για μεγάλα μηνύματα. Η συνάρτηση είναι μιας διαδρομής μόνο, που σημαίνει ότι είναι υπερβολικά δύσκολο να αντιστραφεί το κατακερματισμένο μήνυμα στη κανονική του μορφή. Για αυτό πρέπει να χρησιμοποιείται για να ελεγχτεί η ακεραιότητα του μηνύματος και να μην παρεξηγείται ως μέθοδο κρυπτογράφησης.

Β'2.3. Αναπαράσταση του αλγορίθμου Diffie-Hellman για το TLS 1.3

Και οι 5 φέρουν μια κοινή επιλογή. Κάνοντας χρήση του TLS 1.3 χρησιμοποιείται πάντα ο αλγόριθμος Diffie-Hellman (με τη δυνατότητα χρήσης του Elliptic Curve αν υποστηρίζεται, ECDHE) για την συμφωνία σε ένα κοινό μυστικό κλειδί. Αυτό το κλειδί είναι συμμετρικό και παράγεται από τα κλειδιά συνόδου. Χρησιμοποιείται για την κρυπτογράφηση και αποκρυπτογράφηση των δεδομένων στο Application Layer. Τα κλειδιά συνόδου είναι διαφορετικά για κάθε σύνοδο. Στην αρχή της συζήτησης, πριν τη χειραψία, ο client και ο server συμφωνούν σε δυο τυχαία νούμερα (έστω x, y). Αυτά τα νούμερα στέλνονται στο δίκτυο χωρίς κρυπτογράφηση, καθώς δεν μας ενδιαφέρει αν τα μάθει κάποιος. Ταυτόχρονα ο client και ο server παράγουν δυο τυχαία κλειδιά τα οποία θα είναι ιδιωτικά και διαφορετικά για τον καθένα (έστω C_{priv} και S_{priv} αντίστοιχα). Επιλέγουν και οι δύο το ίδιο τυχαίο νούμερο από αυτά που ανταλλάχθηκαν προηγουμένως (είτε x είτε y), το υψώνουν με εκθέτη το ιδιωτικό κλειδί τους (C_{priv} για τον client και S_{priv} για τον server) και κάνοντας modulo το άλλο τυχαίο νούμερο (είτε y είτε x αντίστοιχα, ανάλογα τι επιλέχθηκε σαν βάση). Από αυτή τη πράξη παράγεται ένα τυχαίο δημόσιο κλειδί για τον client και ένα διαφορετικό για τον server (έστω C_{pub} και S_{pub} αντίστοιχα), τα οποία ανταλλάσσουν μεταξύ τους πάλι σε μη κρυπτογραφημένη μορφή.

Σαν πράξη φαίνεται ως εξής:

$$C_{pub} = (x^{C_{priv}}) \bmod y,$$

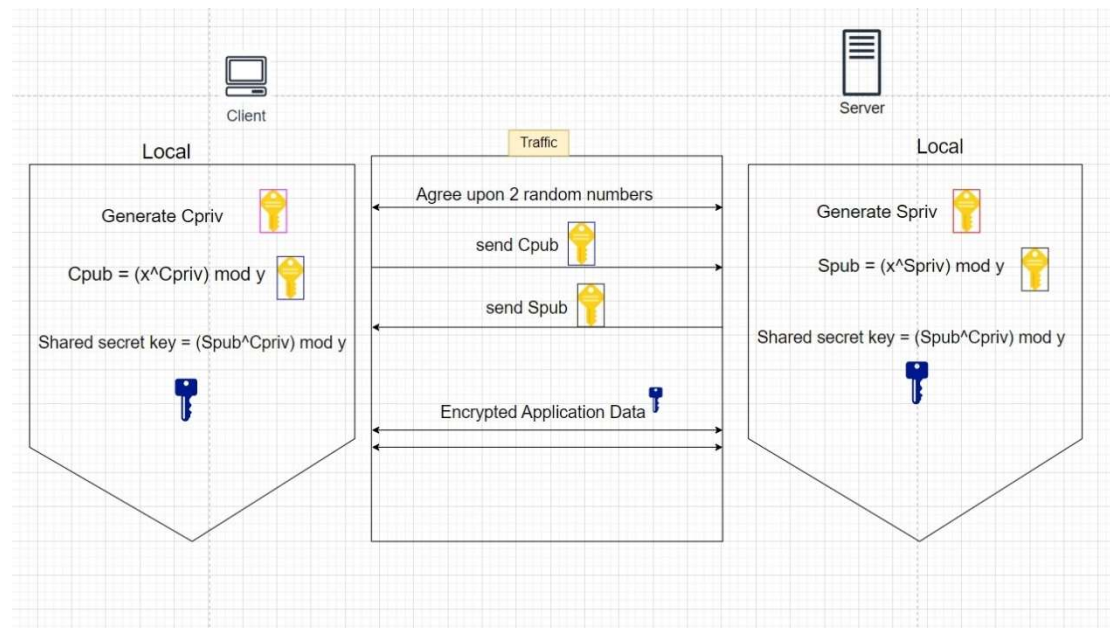
$$S_{pub} = (x^{S_{priv}}) \bmod y$$

Το κάθε μέλος της συζήτησης λαμβάνοντας το δημόσιο κλειδί του άλλου θα κάνει πάλι την ίδια πράξη, αλλά στη θέση του τυχαίου αριθμού ως βάση αυτή τη φορά θα μπει το δημόσιο κλειδί που έλαβε ως εξής:

$$\text{Shared secret key} = (\text{Spub}^{\text{Cpriv}}) \bmod y$$

$$\text{Shared secret key} = (\text{Cpub}^{\text{Spriv}}) \bmod y$$

Το κοινό μυστικό κλειδί που παράγεται είναι το ίδιο.



Εικόνα 1 Αλγόριθμος ανταλλαγής κλειδιών Diffie-Hellman

Αυτό είναι υπολογιστικά ανέφικτο να βρεθεί με τη τεχνολογία που έχουμε σήμερα. Επιπλέον, σε κάθε σύνοδο δημιουργούνται νέα κλειδιά. Επομένως ακόμη και αν ένας επιτιθέμενος ακούσει τη συζήτηση μεταξύ client και server και διαρρεύσει το ιδιωτικό κλειδί του ενός, θα μπορέσει να αποκρυπτογραφήσει μόνο τη συζήτηση της συγκεκριμένης συνοδού.

Με χρήση της μεθόδου της *ελλειπτικής καμπύλης* (Elliptic Curve) και ενός σημείου πάνω στη καμπύλη το οποίο συμφωνείται στη χειραψία, είναι εφικτό να παράγεται το δημόσιο και το ιδιωτικό κλειδί. Έτσι «κόβεται» ένα βήμα από τον Diffie-Hellman αλγόριθμο. Η χρήση της Elliptic Curve θεωρείται πιο ασφαλής και γρήγορη μέθοδος παραγωγής δημοσίων και ιδιωτικών κλειδιών από τη χρήση τυχαίων αριθμών. Να σημειωθεί ότι στο TLS 1.2 υπήρχαν δύο επιλογές ανταλλαγής και συμφωνίας κοινών κλειδιών κρυπτογράφησης και αποκρυπτογράφησης. Ο (EC)DHE και ο RSA. Λόγω της έλλειψης του PFS, ο RSA καταργήθηκε και ο (EC)DHE ορίστηκε ως ο μοναδικός

τρόπος ανταλλαγής και συμφωνίας κλειδιών. Για αυτό στους αλγορίθμους του TLS 1.3 παραλείπεται στην ονομασία. Ο RSA υπάρχει πλέον μόνο σαν επιλογή για την αυθεντικοποίηση του πιστοποιητικού του server, αν τον υποστηρίζει ο client.

B'3. Εργαλεία και μέθοδοι που υλοποιήθηκαν για τη σύλληψη μιας χειραψίας

B'3.1. Χαρακτηριστικά περιβάλλοντος και προετοιμασία

Το capture των πακέτων έγινε μέσα στην *εικονική μηχανή* (Virtual Machine – VM) VirtualBox 7.0 σε *λειτουργικό σύστημα* (Operating System) Ubuntu 22.04. Μέσα από ένα μικρό *σενάριο* (script) που δημιούργησα σε Python 3, καταφέρνω να ακούσω τη κίνηση (sniffing) των πακέτων που μεταφέρονται από εμένα στο ionio.gr και αντίστροφα, να εξάγω τα session keys και να αποκρυπτογραφήσω τα δεδομένα ώστε να δω με μεγαλύτερη λεπτομέρεια όλη τη διαδικασία του TLS 1.3. Έχω φιλτράρει ώστε να πιάνω μόνο TLS πακέτα και αφού τα αποκρυπτογραφήσω με το tshark [20] μέσα από τον ίδιο κώδικα python, τυπώνει το καθαρό κείμενο σε ένα αρχείο log. Το sniffing έγινε με τη βιβλιοθήκη του εργαλείου Scapy [9], ένα λογισμικό ανεπτυγμένο σε Python κώδικα και διαθέσιμο σαν Python module. Η εξαγωγή των κλειδιών έγινε με την *μεταβλητή περιβάλλοντος* (Environmental Variable) SSLKEYLOGFILE [10] σε μορφή NSS [11]. Όταν μπαίνω στο Firefox, τα κλειδιά συνόδου εξάγονται στο αρχείο που δείχνει η μεταβλητή και αυτά μπορούν να χρησιμοποιηθούν για την παραγωγή του κοινού μυστικού κλειδιού. Το tshark είναι εργαλείο με ίδιο κώδικα με το Wireshark αλλά για τη *γραμμή εντολών* (command line - cmd), πάνω στο οποίο συνεισφέρουν οι ίδιοι και για το Wireshark. Μπορεί να πάρει τα κλειδιά με την εντολή ``-o tls.keylog_file: <name of keylogfile>`` και να τα χρησιμοποιήσει για να αποκρυπτογραφήσει ένα αρχείο pcap. Ολόκληρη η εντολή που καλέστηκε είναι ``sudo tshark -o 'tls.keylog_file:sslkeyfile' -r captured_traffic.cap -R 'tls' -2 -Y 'tls' -V >'logfile.log'``. Η επιλογή `-r` διαβάζει ένα αρχείο pcap το οποίο έχει δημιουργηθεί προηγουμένως με το Scapy. Η επιλογή `-R` φιλτράρει τα πακέτα που θα λάβει υπόψη το tshark. Στη περίπτωσή μου, διαλέγω μόνο τη χειραψία του TLS και ό,τι application data κρυπτογραφούν με TLS. Η επιλογή `-2` είναι πάντα μαζί με την προηγούμενη επιλογή και σημαίνει 2-pass. Το tshark διαβάζει όλο το pcap αρχείο και μετά πάει από την αρχή να κάνει το φιλτράρισμα. Η επιλογή `-Y` δεν χρειάζεται αλλά την έβαλα για

εκπαιδευτικούς σκοπούς. Σε περίπτωση που ήθελα να έχω δεύτερο φίλτρο, χρησιμοποιώ αυτό. Εγώ έβαλα το ίδιο φίλτρο. Η επιλογή -V εμφανίζει τις λεπτομέρειες του πακέτου. Για να εξάγω όλη αυτή τη πληροφορία σε ένα αρχείο, γίνεται με το μαθηματικό σύμβολο που ακολουθεί και το όνομα του αρχείου σε «μονά αυτάκια» [12]. Το sniffing ξεκινάει με μια εντολή στην οθόνη τερματικού: `firefox & sudo ./pythonscript.py`. Αυτή εκτελεί το firefox και έπειτα το python script που γράφουμε (με δικαιώματα διαχειριστή sudo, καθώς το Scapy δεν επιτρέπει τη χρήση του interface σε απλούς χρήστες). Ανοίγει ένα παράθυρο firefox και, αφού γράψουμε το κωδικό για το sudo που μας ζητάει στην οθόνη τερματικού, ξεκινάει το sniffing.

B'3.2. Κώδικας

```
#!/usr/bin/env python3
```

```
from scapy import *
```

```
import sslkeylog
```

Ορίζουμε το αρχείο στο οποίο δείχνει (και εξάγει τα κλειδιά) η μεταβλητή SSLKEYLOGFILE και το ανοίγουμε σε ένα αρχείο για ανάγνωση μόνο. Το αρχείο είναι σε μορφή που καθορίζεται από την *υπηρεσία ασφάλειας δικτύων* (Network Security Services – NSS) και έχει μορφή <Label> (Στο 1.3 υπάρχουν 7 διαφορετικές ετικέτες) <ClientRandom> (μεγέθους 32 bytes) <Secret> (μεγέθους 32 bytes) [13].

Το αρχείο έχει την εξής μορφή. Καθώς τα κλειδιά συνόδου αλλάζουν συνέχεια και το μόνο που έκανα ήταν να ανοίγω την πρώτη σελίδα του ionio.gr, ακόμη και να άκουσε κάποιος τη κίνηση την ώρα του πειράματος δεν θα μάθαινε κάτι σημαντικό.

```
# SSL/TLS secrets log file, generated by NSS
CLIENT_HANDSHAKE_TRAFFIC_SECRET 0ee6b5d70994491b34699421affbe5bed62565dadd110fd18791a6edf01cb096 8b77345b552df47be6086db248bd22f63d98e786c88d6ffd72fe6463b671e9b
SERVER_HANDSHAKE_TRAFFIC_SECRET 0ee6b5d70994491b34699421affbe5bed62565dadd110fd18791a6edf01cb096 11e8079ef2b32d37ddc2ad6e47dafb4712b1be1eeb488ce99a01fd45d08b923
CLIENT_TRAFFIC_SECRET_0 0ee6b5d70994491b34699421affbe5bed62565dadd110fd18791a6edf01cb096 2d3fedca6f3fb895191319bf4d443bd10c02178fa6f1b496cc3724825e8235a9
SERVER_TRAFFIC_SECRET_0 0ee6b5d70994491b34699421affbe5bed62565dadd110fd18791a6edf01cb096 0e93776e63b63972b6ebbd489723c46f9e1d8e127f70be10cabf3c886f847758
EXPORTER_SECRET 0ee6b5d70994491b34699421affbe5bed62565dadd110fd18791a6edf01cb096 702f547146e258b3052a11351316ab06fa6421fae6d70f0425824c2913aa1b5d
```

Εικόνα 2 Τα μυστικά κλειδιά για τη παραγωγή του κοινού μυστικού κλειδιού.

```
key_path = "/path/to/sslkeyfile"
```

```
text_file = open(key_path, "r")
```

```
def packet_handler(packet):
```

```
# Φιλτράρουμε όλα τα TCP πακέτα που πιάνει το Scapy με τη συνάρτηση
haslayer(<layer>). Αυτή ελέγχει εάν υπάρχει το επίπεδο στο πακέτο και συνεχίζει
ανάλογα.
```

if packet.haslayer(TLS):

Ένα μήνυμα οθόνης για να ενημερώσει τον προγραμματιστή ότι συλλήφθηκε
ένα νέο πακέτο που περιέχει το επίπεδο TLS

print("\n-----NEW PACKET-----")

Το packet καλείται από το Scapy για να εμφανίσει τα επίπεδα του πακέτου με
τη σειρά που τυλίχθηκε το payload, τη διεύθυνση IP από την οποία ήρθε και σε ποια
προορίζεται το μήνυμα και το τύπο δεδομένων που είναι (για παράδειγμα Application
Data).

print("Packet Layers, source and destination, Type of data = ", packet)

Έξω από την packet_handler φορτώνουμε το tls layer package του Scapy για να
λειτουργήσει το φιλτράρισμα

load_layer('tls')

Χρησιμοποιούμε το sniff() του Scapy για να πιάσουμε πακέτα. Μπορούμε να
περιορίσουμε τον αριθμό πακέτων με τη παράμετρο count=n (όπου n είναι ο
επιθυμητός αριθμός). Η παράμετρος iface (interface) είναι απαραίτητη και
προσδιορίζει τη διεπαφή από την οποία θα κοιτάζει το Scapy για να πιάσει πακέτα.
Για να βρεθεί ο τύπος της διεπαφής σε Linux, γράφουμε την εντολή `ip addr` στο
τερματικό (μέρος για το input στο command line, η οθόνη τερματικού). Το φίλτρο tcp
περιορίζει τα πακέτα τα οποία θα αποθηκεύσει το Scapy σε μόνο TCP πακέτα.
Θυμόμαστε ότι το επίπεδο tls βρίσκεται πάνω στο TCP. Αν δεν υπάρχει Transport
Layer, δεν υπάρχει TLS. Υπάρχει η περίπτωση να χρησιμοποιηθεί το πρωτόκολλο
QUIC για HTTP/3, αλλά καθώς γνώριζα ότι το ionio.gr είναι σε HTTP/1.1, δεν
χάσαμε κάποιο πακέτο QUIC.

Με τη παράμετρο prn καλείται μια ψευδοσυνάρτηση lambda ή μια συνάρτηση της
επιλογής μας. Εμείς καλούμε την packet_handler που ορίσαμε προηγουμένως. Η
συνάρτηση καλείται πάντα όταν ένα πακέτο έχει συλληφθεί και πριν περάσει στο
επόμενο πακέτο. Έτσι μπορούμε να διαχειριστούμε το κάθε πακέτο ξεχωριστά. Μια
λίστα πακέτων αποθηκεύεται σε μεταβλητή της επιλογής μας captured_packets, η
οποία χρειάζεται για να γίνει η εξαγωγή τους σε pcap αρχείο.

**captured_packets = sniff(iface='enp0s3', filter="tcp", prn=packet_handler,
count=50)**

Η συνάρτηση wrpcap() ανήκει στη βιβλιοθήκη του Scapy. Σαν παραμέτρους παίρνει το όνομα του αρχείου σε εισαγωγικά και το όνομα του πακέτου που θα γράψει. Εμείς του δίνουμε μια λίστα από πακέτα.

```
wrpcap("captured_traffic.cap",captured_packets)
```

#Τυπώνεται περιληπτικά ο αριθμός πακέτων που έχουν συλληφθεί και τι είδους (TCP, UDP, ICMP, Other) και ένα μήνυμα που ενημερώνει για το τέλος της σύλληψης πακέτων.

```
print("Type and amount of captured packets = ", captured_packets)
```

```
print("-----END OF CAPTURING-----")
```

Ετοιμάζουμε μια εντολή για να τρέξουμε το tshark μέσα από το script. Το εργαλείο αυτό μπορεί να αλληλοεπιδράσει μόνο με το τερματικό. Του δίνουμε σαν input τα κλειδιά που εξήγαγε η μεταβλητή SSLKEYLOGFILE και το αρχείο pcap. Το tshark υπολογίζει το κοινό μυστικό κλειδί από τα μυστικά του αρχείου sslkeyfile και αποκρυπτογραφεί ένα-ένα τα πακέτα. Έπειτα, το προκαθορισμένο output είναι στο τερματικό, αν δεν περαστεί μια συγκεκριμένη παράμετρος (που βλέπουμε λίγο αργότερα στο κώδικα). Εμφανίζεται μια γραμμή για το κάθε πακέτο που περιγράφει τον αριθμό πακέτου στο pcap, το χρόνο στον οποίο λήφθηκε από τη sniff(), διευθύνσεις IP του παραλήπτη και αποστολέα και το είδος μηνύματος (TCP reassembled segment ή Client hello ή Certificate κλπ).

```
print("Running tshark to output the decrypted packets...\n")
```

```
comm="sudo tshark -o 'tls.keylog_file:sslkeyfile' -r captured_traffic.cap"
```

Η subprocess.run() τρέχει μια δεύτερη διεργασία στο παρασκήνιο και παίρνει σαν παράμετρο μια εντολή και δύο Boolean τιμές για το shell και το check. Η shell είναι για το αν το αποτέλεσμα θα εμφανιστεί στο τερματικό και η check αν θα γίνει έλεγχος σφαλμάτων.

```
subprocess.run(comm,shell=True,check=True)
```

Ίδια λογική με τη προηγούμενη εντολή, αλλά αυτή τη φορά θέλουμε το output να γραφεί σε ένα αρχείο logfile.log για να μπορέσουμε να το διαβάσουμε. Η παράμετρος

-R -2 περνάει ένα φίλτρο από όλα τα πακέτα και διαβάζει το αρχείο pcap δύο φορές. Η παράμετρος -Y βάζει ως δεύτερο φίλτρο στη δεύτερη αναζήτηση πάλι tls. Εδώ δεν έχει διαφορά καθώς το φίλτρο είναι ίδιο αλλά είναι απαραίτητο να επιλεγθεί κάποιο φίλτρο με την -Y, οπότε βάζουμε το ίδιο. Η παράμετρος -V εμφανίζει με πολύ λεπτομέρεια τα πακέτα και από τι αποτελούνται. Το tshark βάζει επιπλέον πληροφορία που έχει διερμηνεύσει, η οποία δεν εμφανίζεται στα πακέτα. Με το μαθηματικό σύμβολο > το output καταγράφεται σε ένα αρχείο logfile.log αντί να εμφανιστεί στο τερματικό.

```
command="sudo tshark -o 'tls.keylog_file:sslkeyfile' -r captured_traffic.cap -R 'tls' -2 -Y 'tls' -V >'logfile.log'"
```

```
subprocess.run(command,shell=True,check=True)
```

Το κάθε πακέτο ονομάζεται frame στο output του tshark. Ένα frame έχει την εξής μορφή:

```
Frame 4: 710 bytes on wire (5680 bits), 710 bytes captured (5680 bits) on interface 0
    Encapsulation type: Ethernet (1)
    Arrival Time: Sep 22, 2023 19:56:02.324940000 EEST
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1695401762.324940000 seconds
    [Time delta from previous captured frame: 9.193748000 seconds]
    [Time delta from previous displayed frame: 9.193748000 seconds]
    [Time since reference or first frame: 9.240640000 seconds]
    Frame Number: 4
    Frame Length: 710 bytes (5680 bits)
    Capture Length: 710 bytes (5680 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: eth:ethertype:ip:tcp:tls]
```

Εικόνα 3 μέγεθος του πακέτου, πότε συλλήφθηκε από το tshark και ποια επίπεδα βρίσκονται στο πακέτο

```
Ethernet II, Src: [redacted] ( [redacted] ), Dst: [redacted] ( [redacted] )
  Destination: [redacted] ( [redacted] )
    Address: [redacted] ( [redacted] )
      ....1. .... = LG bit: Locally administered address (this is NOT the factory default)
      ....0. .... = IG bit: Individual address (unicast)
  Source: [redacted] ( [redacted] )
    Address: [redacted] ( [redacted] )
      ....0. .... = LG bit: Globally unique address (factory default)
      ....0. .... = IG bit: Individual address (unicast)
  Type: IPv4 (0x0800)
```

Εικόνα 4 Πρώτο επίπεδο, Ethernet. Έχω κρύψει τα ευαίσθητα δεδομένα καθώς μπορεί να επηρεάσει το περιβάλλον εκτός της εικονικής μηχανής

```

Internet Protocol Version 4, Src: 10.0.2.11, Dst: 195.130.127.114
 0100 .... = Version: 4
 .... 0101 = Header Length: 20 bytes (5)
 Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
   0000 00.. = Differentiated Services Codepoint: Default (0)
   .... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
 Total Length: 696
 Identification: 0xf8af (63663)
 010. .... = Flags: 0x2, Don't fragment
   0... .... = Reserved bit: Not set
   .1.. .... = Don't fragment: Set
   ..0. .... = More fragments: Not set
 ...0 0000 0000 0000 = Fragment Offset: 0
 Time to Live: 64
 Protocol: TCP (6)
 Header Checksum: 0xf090 [validation disabled]
 [Header checksum status: Unverified]
 Source Address: 10.0.2.11
 Destination Address: 195.130.127.114

```

Εικόνα 5 Δεύτερο επίπεδο, Internet Protocol. Η διεύθυνση 10.0.2.11 είναι του VM και η 195.130.127.114 του ionio.gr. Το συγκεκριμένο πακέτο αποστέλλεται από εμένα στο ionio.gr (spoiler: είναι το client hello)

```

Transmission Control Protocol, Src Port: 33022, Dst Port: 443, Seq: 1, Ack: 1, Len: 656
 Source Port: 33022
 Destination Port: 443
 [Stream index: 4]
 [Conversation completeness: Incomplete, DATA (15)]
 [TCP Segment Len: 656]
 Sequence Number: 1 (relative sequence number)
 Sequence Number (raw): 1263247985
 [Next Sequence Number: 657 (relative sequence number)]
 Acknowledgment Number: 1 (relative ack number)
 Acknowledgment number (raw): 18507
 0101 .... = Header Length: 20 bytes (5)
 Flags: 0x018 (PSH, ACK)
   000. .... = Reserved: Not set
   ...0 .... = Accurate ECN: Not set
   .... 0... = Congestion Window Reduced: Not set
   .... .0.. = ECN-Echo: Not set
   .... ..0. = Urgent: Not set
   .... ...1 = Acknowledgment: Set
   .... .... 1.. = Push: Set
   .... .... .0.. = Reset: Not set
   .... .... ..0. = Syn: Not set
   .... .... ...0 = Fin: Not set
 [TCP Flags: .....AP...]
 Window: 64240
 [Calculated window size: 64240]
 [Window size scaling factor: -2 (no window scaling used)]
 Checksum: 0x51aa [unverified]
 [Checksum Status: Unverified]
 Urgent Pointer: 0
 [Timestamps]
   [Time since first frame in this TCP stream: 0.029119000 seconds]
   [Time since previous frame in this TCP stream: 0.002020000 seconds]
 [SEQ/ACK analysis]
   [This is an ACK to the segment in frame: 4]
   [The RTT to ACK the segment was: 0.000107000 seconds]
   [iRTT: 0.027099000 seconds]
   [Bytes in flight: 656]
   [Bytes sent since last PSH flag: 656]
 TCP payload (656 bytes)

```

Εικόνα 6 Τρίτο επίπεδο, TCP. Το TCP payload σε αυτό το πακέτο είναι το επόμενο επίπεδο.

Το επόμενο επίπεδο είναι το TLS το οποίο θα αναλυθεί στο υποκεφάλαιο Β'5.

Κλείνουμε το αρχείο sslkeyfile και ενημερώνουμε το χρήστη ότι το tshark τελείωσε με την αποκρυπτογράφηση του pcap

```
text_file.close()
```

```
print("-----END OF DECRYPTING-----")
```

B'3.3. Output

Τι εμφανίζεται στο τερματικό όταν εκτελέσουμε το script και επισκεφτούμε μια σελίδα:

```
-----NEW PACKET-----
Packet Layers, source and destination, Type of data = Ether / IP / TCP / TLS 10.0.2.11:51172 > 195.130.127.114:443 / TLS / TLS Handshake - Client Hello
-----NEW PACKET-----
Packet Layers, source and destination, Type of data = Ether / IP / TCP / TLS / TLS 195.130.127.114:443 > 10.0.2.11:51172 / TLS / TLS ChangeCipherSpec / Raw
-----NEW PACKET-----
Packet Layers, source and destination, Type of data = Ether / IP / TCP / TLS 10.0.2.11:41574 > 34.120.208.123:443 / TLS / TLS Application Data
-----NEW PACKET-----
Packet Layers, source and destination, Type of data = Ether / IP / TCP / TLS 10.0.2.11:41574 > 34.120.208.123:443 / TLS / TLS Application Data
-----NEW PACKET-----
Packet Layers, source and destination, Type of data = Ether / IP / TCP / TLS 10.0.2.11:51186 > 195.130.127.114:443 / TLS / TLS Handshake - Client Hello
-----NEW PACKET-----
Packet Layers, source and destination, Type of data = Ether / IP / TCP / TLS 10.0.2.11:51198 > 195.130.127.114:443 / TLS / TLS Handshake - Client Hello
-----NEW PACKET-----
Packet Layers, source and destination, Type of data = Ether / IP / TCP / TLS / TLS 195.130.127.114:443 > 10.0.2.11:51186 / TLS / TLS ChangeCipherSpec / Raw
-----NEW PACKET-----
Packet Layers, source and destination, Type of data = Ether / IP / TCP / TLS / TLS 195.130.127.114:443 > 10.0.2.11:51198 / TLS / TLS ChangeCipherSpec / Raw
-----NEW PACKET-----
Packet Layers, source and destination, Type of data = Ether / IP / TCP / TLS / TLS 34.120.208.123:443 > 10.0.2.11:41574 / TLS / TLS Application Data
-----NEW PACKET-----
Packet Layers, source and destination, Type of data = Ether / IP / TCP / TLS / TLS 10.0.2.11:51186 > 195.130.127.114:443 / TLS / TLS Application Data
-----NEW PACKET-----
Packet Layers, source and destination, Type of data = Ether / IP / TCP / TLS 10.0.2.11:51186 > 195.130.127.114:443 / TLS / TLS Application Data
-----NEW PACKET-----
Packet Layers, source and destination, Type of data = Ether / IP / TCP / TLS / TLS 10.0.2.11:51198 > 195.130.127.114:443 / TLS / TLS Application Data
-----NEW PACKET-----
Packet Layers, source and destination, Type of data = Ether / IP / TCP / TLS 195.130.127.114:443 > 10.0.2.11:51186 / TLS / TLS Application Data
-----NEW PACKET-----
Packet Layers, source and destination, Type of data = Ether / IP / TCP / TLS 195.130.127.114:443 > 10.0.2.11:51186 / TLS / TLS Application Data
```

Εικόνα 7 Αυτό βλέπει οποιοσδήποτε δεν μπορεί να αποκρυπτογραφήσει τη κίνηση.

Φαίνεται ότι ξεκινάει μια χειραψία και ενημερώνονται οι παράμετροι του server. Τα κρυπτογραφημένα πεδία εμφανίζονται ως Application Data και περιέχουν σημαντική πληροφορία για τη χειραψία και τα δεδομένα εφαρμογής (π.χ. HTTP) που μεταφέρονται. Δεν μπορούμε να δούμε τα επίπεδα μετά το TLS καθώς είναι κρυπτογραφημένα.

```

1 0.000000 10.0.2.11 → 195.130.127.114 TCP 74 51172 → 443 [SYN] Seq=0 Wln=64240 Len=0 MSS=1460 SACK_PERM TSval=218590143 TSecr=0 WS=128
2 0.024948 195.130.127.114 → 10.0.2.11 TCP 60 443 → 51172 [SYN, ACK] Seq=0 Ack=1 Wln=32768 Len=0 MSS=1460
3 0.025102 10.0.2.11 → 195.130.127.114 TCP 54 51172 → 443 [ACK] Seq=1 Ack=1 Wln=64240 Len=0
4 0.027538 10.0.2.11 → 195.130.127.114 TLSv1 710 Client Hello
5 0.062429 195.130.127.114 → 10.0.2.11 TLSv1.3 2974 Server Hello, Change Cipher Spec, Encrypted Extensions
6 0.062575 10.0.2.11 → 195.130.127.114 TCP 54 51172 → 443 [ACK] Seq=657 Ack=2921 Wln=62780 Len=0
7 0.063001 195.130.127.114 → 10.0.2.11 TCP 1230 443 → 51172 [PSH, ACK] Seq=2921 Ack=657 Wln=32112 Len=1176 [TCP segment of a reassembled PDU]
8 0.063020 10.0.2.11 → 195.130.127.114 TCP 54 51172 → 443 [ACK] Seq=657 Ack=4097 Wln=61604 Len=0
9 0.068058 195.130.127.114 → 10.0.2.11 TCP 1506 443 → 51172 [PSH, ACK] Seq=4097 Ack=657 Wln=32112 Len=1452 [TCP segment of a reassembled PDU]
10 0.068088 10.0.2.11 → 195.130.127.114 TCP 54 51172 → 443 [ACK] Seq=657 Ack=5549 Wln=62780 Len=0
11 0.068542 195.130.127.114 → 10.0.2.11 TLSv1.3 1217 Certificate, Certificate Verify, Finished
12 0.068612 10.0.2.11 → 195.130.127.114 TCP 54 51172 → 443 [ACK] Seq=657 Ack=6712 Wln=62780 Len=0
13 0.080120 10.0.2.11 → 34.120.208.123 TLSv1.2 202 Application Data
14 0.080308 10.0.2.11 → 34.120.208.123 TLSv1.2 567 Application Data
15 0.080618 34.120.208.123 → 10.0.2.11 TCP 60 443 → 41574 [ACK] Seq=1 Ack=662 Wln=32768 Len=0
16 0.080802 10.0.2.11 → 195.130.127.114 TCP 74 51186 → 443 [SYN] Seq=0 Wln=64240 Len=0 MSS=1460 SACK_PERM TSval=218590241 TSecr=0 WS=128
17 0.100999 10.0.2.11 → 195.130.127.114 TCP 74 51198 → 443 [SYN] Seq=0 Wln=64240 Len=0 MSS=1460 SACK_PERM TSval=218590244 TSecr=0 WS=128
18 0.123212 10.0.2.11 → 104.18.15.101 TCP 74 38622 → 80 [SYN] Seq=0 Wln=64240 Len=0 MSS=1460 SACK_PERM TSval=3375025081 TSecr=0 WS=128
19 0.125626 195.130.127.114 → 10.0.2.11 TCP 60 443 → 51186 [SYN, ACK] Seq=0 Ack=1 Wln=32768 Len=0 MSS=1460
20 0.125670 10.0.2.11 → 195.130.127.114 TCP 54 51186 → 443 [ACK] Seq=1 Ack=1 Wln=64240 Len=0
21 0.124627 195.130.127.114 → 10.0.2.11 TCP 60 443 → 51198 [SYN, ACK] Seq=0 Ack=1 Wln=32768 Len=0 MSS=1460
22 0.125761 10.0.2.11 → 195.130.127.114 TCP 54 51198 → 443 [ACK] Seq=1 Ack=1 Wln=64240 Len=0
23 0.128342 10.0.2.11 → 195.130.127.114 TLSv1 710 Client Hello
24 0.135902 10.0.2.11 → 195.130.127.114 TLSv1.3 2974 Server Hello, Change Cipher Spec, Encrypted Extensions
25 0.141927 104.18.15.101 → 10.0.2.11 TCP 60 80 → 38622 [SYN, ACK] Seq=0 Ack=1 Wln=32768 Len=0 MSS=1460
26 0.142064 10.0.2.11 → 104.18.15.101 TCP 54 38622 → 80 [ACK] Seq=1 Ack=1 Wln=64240 Len=0
27 0.142261 10.0.2.11 → 104.18.15.101 OCSP 484 Request
28 0.158013 104.18.15.101 → 10.0.2.11 TCP 60 80 → 38622 [ACK] Seq=1 Ack=431 Wln=32338 Len=0
29 0.158013 195.130.127.114 → 10.0.2.11 TCP 60 443 → 51198 [ACK] Seq=1 Ack=657 Wln=32112 Len=0
30 0.158013 195.130.127.114 → 10.0.2.11 TCP 60 443 → 51186 [ACK] Seq=1 Ack=657 Wln=32112 Len=0
31 0.166722 195.130.127.114 → 10.0.2.11 TLSv1.3 1506 Server Hello, Change Cipher Spec, Encrypted Extensions
32 0.166920 10.0.2.11 → 195.130.127.114 TCP 54 51186 → 443 [ACK] Seq=657 Ack=1453 Wln=63888 Len=0
33 0.166722 195.130.127.114 → 10.0.2.11 TCP 1514 443 → 51186 [PSH, ACK] Seq=1453 Ack=657 Wln=32112 Len=1460 [TCP segment of a reassembled PDU]
34 0.167001 10.0.2.11 → 195.130.127.114 TCP 54 51186 → 443 [ACK] Seq=657 Ack=2913 Wln=62780 Len=0
35 0.168944 195.130.127.114 → 10.0.2.11 TCP 1230 443 → 51198 [PSH, ACK] Seq=2913 Ack=657 Wln=32112 Len=1184 [TCP segment of a reassembled PDU]
36 0.168967 10.0.2.11 → 195.130.127.114 TCP 54 51186 → 443 [ACK] Seq=657 Ack=4097 Wln=61596 Len=0
37 0.175649 195.130.127.114 → 10.0.2.11 TCP 1514 443 → 51186 [ACK] Seq=4097 Ack=657 Wln=32112 Len=1460 [TCP segment of a reassembled PDU]
38 0.175802 10.0.2.11 → 195.130.127.114 TCP 54 51186 → 443 [ACK] Seq=657 Ack=5557 Wln=62780 Len=0
39 0.175650 195.130.127.114 → 10.0.2.11 TLSv1.3 2974 Server Hello, Change Cipher Spec, Encrypted Extensions
40 0.175886 10.0.2.11 → 195.130.127.114 TCP 54 51198 → 443 [ACK] Seq=657 Ack=2921 Wln=62780 Len=0
41 0.176354 195.130.127.114 → 10.0.2.11 TLSv1.3 1209 Certificate, Certificate Verify, Finished
42 0.176372 10.0.2.11 → 195.130.127.114 TCP 54 51186 → 443 [ACK] Seq=657 Ack=6712 Wln=62780 Len=0
43 0.176354 195.130.127.114 → 10.0.2.11 TCP 1230 443 → 51198 [PSH, ACK] Seq=2921 Ack=657 Wln=32112 Len=1176 [TCP segment of a reassembled PDU]
44 0.176483 10.0.2.11 → 195.130.127.114 TCP 54 51198 → 443 [ACK] Seq=657 Ack=4097 Wln=62780 Len=0
45 0.179142 195.130.127.114 → 10.0.2.11 TCP 1514 443 → 51198 [ACK] Seq=4097 Ack=657 Wln=32112 Len=1460 [TCP segment of a reassembled PDU]
46 0.179167 10.0.2.11 → 195.130.127.114 TCP 54 51198 → 443 [ACK] Seq=657 Ack=5557 Wln=62780 Len=0

```

Εικόνα 8 Αποκρυπτογραφημένη κίνηση. Μπορούμε ξεκάθαρα να δούμε τι είπαν ο client και ο server

Το πακέτο 4 είναι ένα Client Hello που στέλνω εγώ (client) στο ionio.gr (server). Το Server Hello είναι το αμέσως επόμενο (TLS) πακέτο από το server στον client.

Ακολουθούν διάφορα TCP SYN και ACK πακέτα και παρατηρούμε ότι δεν έχει σταλθεί ακόμη το πιστοποιητικό. Μέχρι που ο client αποφασίζει να στείλει αλλά 2 client hello στο server και ένα αίτημα OCSP (αναλύεται στο υποκεφάλαιο Β'5.1.). Ο server θα απαντήσει με δυο server hello και θα στείλει το πιστοποιητικό του.

```

48 0.179617 10.0.2.11 → 195.130.127.114 TCP 54 51198 → 443 [ACK] Seq=657 Ack=6712 Wln=62780 Len=0
49 0.192861 10.0.2.11 → 104.18.14.101 TCP 74 42158 → 80 [SYN] Seq=0 Wln=64240 Len=0 MSS=1460 SACK_PERM TSval=1577770921 TSecr=0 WS=128
50 0.208327 104.18.14.101 → 10.0.2.11 TCP 60 80 → 42158 [SYN, ACK] Seq=0 Ack=1 Wln=32768 Len=0 MSS=1460
51 0.208502 10.0.2.11 → 104.18.14.101 TCP 54 42158 → 80 [ACK] Seq=1 Ack=1 Wln=64240 Len=0
52 0.209942 10.0.2.11 → 104.18.14.101 OCSP 484 Request
53 0.210170 10.0.2.11 → 104.18.14.101 TCP 74 42108 → 80 [SYN] Seq=0 Wln=64240 Len=0 MSS=1460 SACK_PERM TSval=1577770938 TSecr=0 WS=128
54 0.226777 104.18.14.101 → 10.0.2.11 TCP 60 80 → 42168 [SYN, ACK] Seq=0 Ack=1 Wln=32768 Len=0 MSS=1460
55 0.226827 10.0.2.11 → 104.18.14.101 TCP 54 42168 → 80 [ACK] Seq=1 Ack=1 Wln=64240 Len=0
56 0.232464 10.0.2.11 → 104.18.14.101 OCSP 484 Request
57 0.306224 34.120.208.123 → 10.0.2.11 TLSv1.2 410 Application Data, Application Data
58 0.316817 104.18.14.101 → 10.0.2.11 OCSP 1276 Response
59 0.316915 10.0.2.11 → 104.18.14.101 TCP 54 42158 → 80 [ACK] Seq=431 Ack=1223 Wln=63544 Len=0
60 0.319439 10.0.2.11 → 195.130.127.114 TLSv1.3 118 Change Cipher Spec, Finished
61 0.322855 10.0.2.11 → 195.130.127.114 HTTP 514 GET / HTTP/1.1
62 0.323552 195.130.127.114 → 10.0.2.11 TCP 60 443 → 51186 [ACK] Seq=6712 Ack=1181 Wln=31588 Len=0
63 0.335233 104.18.14.101 → 10.0.2.11 OCSP 1276 Response
64 0.335284 10.0.2.11 → 104.18.14.101 TCP 54 42168 → 80 [ACK] Seq=431 Ack=1223 Wln=63544 Len=0
65 0.338683 10.0.2.11 → 195.130.127.114 TLSv1.3 118 Change Cipher Spec, Finished
66 0.344479 195.130.127.114 → 10.0.2.11 TLSv1.3 341 New Session Ticket
67 0.344628 10.0.2.11 → 195.130.127.114 TCP 54 51186 → 443 [ACK] Seq=1181 Ack=6999 Wln=62780 Len=0
68 0.345034 195.130.127.114 → 10.0.2.11 TLSv1.3 341 New Session Ticket
69 0.345051 10.0.2.11 → 195.130.127.114 TCP 54 51186 → 443 [ACK] Seq=1181 Ack=7286 Wln=62780 Len=0
70 0.350463 10.0.2.11 → 34.120.208.123 TCP 54 41574 → 443 [ACK] Seq=662 Ack=357 Wln=62780 Len=0
71 0.350788 34.120.208.123 → 10.0.2.11 TLSv1.2 382 Application Data, Application Data
72 0.350813 10.0.2.11 → 34.120.208.123 TCP 54 41574 → 443 [ACK] Seq=662 Ack=685 Wln=62780 Len=0
73 0.352416 10.0.2.11 → 34.120.208.123 TLSv1.2 100 Application Data
74 0.363426 195.130.127.114 → 10.0.2.11 TLSv1.3 341 New Session Ticket
75 0.363683 10.0.2.11 → 195.130.127.114 TCP 54 51198 → 443 [ACK] Seq=721 Ack=6999 Wln=62780 Len=0
76 0.364512 195.130.127.114 → 10.0.2.11 TLSv1.3 341 New Session Ticket
77 0.364582 10.0.2.11 → 195.130.127.114 TCP 54 51198 → 443 [ACK] Seq=721 Ack=7286 Wln=62780 Len=0
78 0.411316 34.120.208.123 → 10.0.2.11 TCP 60 443 → 41574 [ACK] Seq=685 Ack=708 Wln=32722 Len=0
79 0.415422 104.18.15.101 → 10.0.2.11 OCSP 1276 Response
80 0.415453 10.0.2.11 → 104.18.15.101 TCP 54 38622 → 80 [ACK] Seq=431 Ack=1223 Wln=63544 Len=0
81 0.416908 10.0.2.11 → 195.130.127.114 TLSv1.3 118 Change Cipher Spec, Finished
82 0.442750 195.130.127.114 → 10.0.2.11 TLSv1.3 628 New Session Ticket, New Session Ticket
83 0.442792 10.0.2.11 → 195.130.127.114 TCP 54 51172 → 443 [ACK] Seq=721 Ack=7286 Wln=62780 Len=0

```

Εικόνα 9 Ακολουθούν πολλαπλές χειραψίες και δημιουργία session tickets

Ο client συνδέεται με περισσότερους server όσο ανοίγει η σελίδα που ζήτησε (αρχική του ionio.gr) και ξεκινάει ξεχωριστές χειραψίας για το καθένα. Ο ionio.gr δημιουργεί

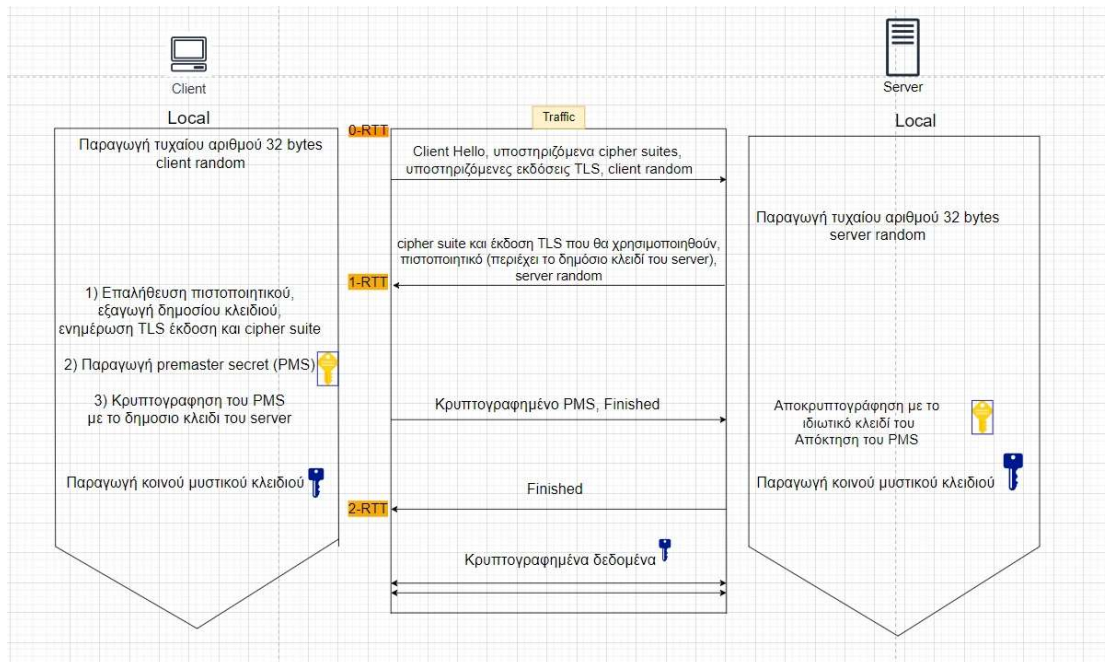
νέα session tickets στα πακέτα 66, 68, 76 και 82 και τα στέλνει στον client σε περίπτωση που θελήσει να συνεχίσει τα συγκεκριμένο sessions αργότερα. Περισσότερα για τα session, session resumption και session tickets στο υποκεφάλαιο B'4.1.2.

B'4. Εισαγωγή στη χειραψία

Όταν προηγούμαστε σε μια ιστοσελίδα, κάνουμε μια αίτηση στον server όπου ζητάμε το περιεχόμενο της σελίδας. Ο server μας απαντάει με το κατάλληλο μήνυμα (μας δείχνει τη σελίδα ή μας πετάει κάποιο σφάλμα). Πριν εμφανιστεί η σελίδα, γίνεται η χειραψία όπου γνωρίζονται client και server και συμφωνούν σε κάποια πράγματα για την σύνδεση που πρόκειται να πραγματοποιηθεί. Η χειραψία είναι μια συμφωνία μεταξύ client και server για την αυθεντικοποίηση των δυο μελών, τον αλγόριθμο κρυπτογράφησης που θα χρησιμοποιηθεί, το μυστικό κοινό κλειδί για τη κρυπτογράφηση, την έκδοση TLS που θα επιλεγεί και πολλά άλλα που θα αναλυθούν σε επόμενο υποκεφάλαιο. Συνήθως η αυθεντικοποίηση γίνεται μόνο από τη μεριά του server, καθώς ο client είναι ένας χρήστης χωρίς πιστοποιητικό (μπορεί αφού τελειώσει η χειραψία και φορτώσει η σελίδα να αυθεντικοποιείται στο HTTP με κωδικούς πρόσβασης). Η χειραψία παίρνει χρόνο, το οποίο μετريέται σε Round-trip time (RTT). Το RTT ορίζεται ως ο χρόνος που καταλαμβάνεται στο δίκτυο από τη στιγμή που θα σταλθεί μια *αίτηση* (request) έως και την παραλαβή απάντησης (response). Συγκεκριμένα, στην έκδοση TLS 1.2 χρειάζεται 2-RTT ενώ το 1.3 συγχωνεύει κάποια βήματα και τελειώνει σε 1-RTT. Υπάρχει η δυνατότητα στο TLS 1.3 και για 0-RTT αλλά έχει επιπλοκές ασφάλειας. Περισσότερα για αυτό σε επόμενο υποκεφάλαιο.

Β'4.1. Διαφορές RTT μεταξύ 1.2 και 1.3 στη χειραψία

Β'4.1.1. TLS 1.2

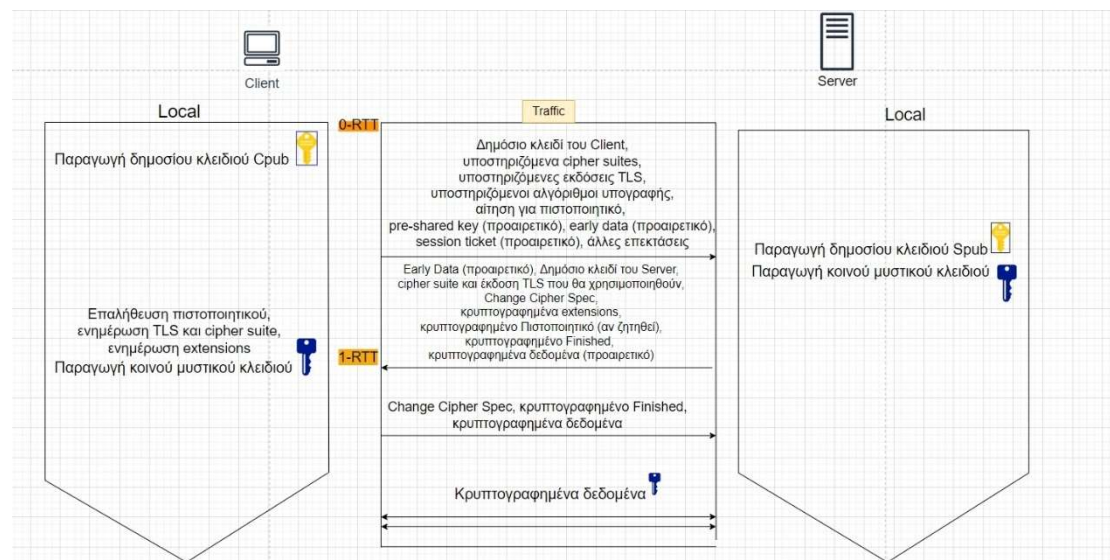


Εικόνα 10 Διαδικασία χειραψίας στο TLS 1.2 με τον αλγόριθμο ανταλλαγής κλειδιών RSA [14]

Στο TLS 1.2 ο Client χαιρετάει τον server με ένα Client Hello με το οποίο στέλνει μια λίστα με τα cipher suites και τις εκδόσεις TLS που υποστηρίζει και ένα τυχαίο νούμερο (ή το Random πεδίο, ένα τυχαίο νούμερο για την δημιουργία κλειδιών στον αλγόριθμο ανταλλαγής κλειδιών Diffie-Hellman) και αναμένει ανταπόκριση. Ο server απαντάει με το Server hello του το οποίο περιέχει το πιστοποιητικό του, την υπογραφή του πιστοποιητικού και ένα τυχαίο νούμερο (ενώ στον Diffie-Hellman παράγεται το δημόσιο κλειδί με το cipher suite της επιλογής του από τη λίστα που του έδωσε ο client για να ξεκινήσουν την ανταλλαγή ενός κοινού μυστικού κλειδιού). Επιπλέον γεμίζει και το δικό του Random πεδίο και τελειώνει με “Server Hello Done”. Όλο αυτό διήρκεσε 1-RTT. Ο client παίρνει το πιστοποιητικό, επιβεβαιώνει ότι ισχύει και παράγει το premaster secret key (PMS). Κρυπτογραφεί το PMS με το δημόσιο κλειδί του server και τα στέλνει με το Finished μήνυμα. Παράλληλα, υπολογίζει το κοινό μυστικό κλειδί στο οποίο θα φτάσουν και οι δύο. Υπολογίζεται από τα δύο τυχαία νούμερα που πλέον έχει ο καθένας και το PMS. Στο Finished περνάει όλη τη χειραψία μέχρι στιγμής από συνάρτηση κατακερματισμού hash ώστε να βεβαιωθεί ότι δεν έχει πειραχτεί κάποιο μήνυμα από κακόβουλο χρήστη στη σύνδεση. Ο server με τη σειρά του αποκρυπτογραφεί το PMS με το ιδιωτικό κλειδί

του και παράγει το κοινό μυστικό κλειδί με τον ίδιο τρόπο. Στέλνει ένα μήνυμα Finished που έχει κάνει hash όλη τη χειραψία. Τώρα που φτάσαμε στα 2-RTT, ολοκληρώνεται η διαδικασία και τα δύο μέλη μπορούν να στείλουν ο ένας στον άλλο κρυπτογραφημένα μηνύματα με το κοινό μυστικό τους.

B'4.1.2. TLS 1.3



Εικόνα 11 Η χειραψία στο TLS 1.3 γίνεται μόνο με αλγόριθμο ανταλλαγής κλειδιών Diffie-Hellman [15]

Το TLS 1.3 μπαίνει δυναμικά με 1-RTT κόβοντας πολλά ανώφελα βήματα και συγχωνεύοντας άλλα σε ένα βήμα (πολλαπλά πακέτα που στέλνονται διαδοχικά). Επιπλέον απλουστεύει τη διαδικασία επιλογής cipher suite με τη μείωσή της λίστας σε συνολικά 5 επιλογές. Ο Client ξεκινάει με το Client Hello με το οποίο στέλνει όλα τα πεδία που υποστηρίζει και θα ήθελε να χρησιμοποιήσει (ταξινομημένα από μεγαλύτερη προτίμηση σε μικρότερη), όπως τα cipher suites, TLS supported versions, αίτηση για πιστοποιητικό του server, το δημόσιο κλειδί που υπολόγισε και άλλα που αναλύονται σε επόμενο υποκεφάλαιο. Ο server στέλνει το cipher suite και έκδοση TLS που θα χρησιμοποιήσουν, το πιστοποιητικό του (αν του ζητηθεί) και το δημόσιο κλειδί του. Υπολογίζει το κοινό μυστικό κλειδί και αποστέλλει κρυπτογραφημένες επεκτάσεις (extensions) (όπως τι έκδοση HTTP θα δείξει το περιεχόμενο στον client) και κρυπτογραφημένο Finished (με ίδιο περιεχόμενο όπως στο TLS 1.2) τα οποία εμφανίζονται ως "Encrypted" σε όσους ακούν τη κίνηση του δικτύου (ακόμη και τον client). Έχει τη δυνατότητα πλέον να στέλνει στον client κρυπτογραφημένα

μηνύματα, πριν καν τελειώσει. Αυτό διήρκησε 1-RTT.

Ο client λαμβάνει το μήνυμα, επιβεβαιώνει την ταυτότητα του server, υπολογίζει το κοινό μυστικό κλειδί, αποκρυπτογραφεί τα μηνύματα, ενημερώνει τον server για την έκδοση TLS που θα χρησιμοποιήσει και αυτός με κάποια κόλπα (ανάλυση σε επόμενο υποκεφάλαιο όπου τεμαχίζουμε και εξηγούμε το περιεχόμενο του κάθε πακέτου) και στέλνει κρυπτογραφημένο το μήνυμα Finished του. Απευθείας στέλνει ότι δεδομένα θέλει στον server σε κρυπτογραφημένη μορφή και για αυτό δεν υπολογίζεται αυτό το βήμα στον χρόνο RTT. Πρακτικά σε χρόνο 2-RTT είναι ικανοί και οι δυο να τελειώσουν τη χειραψία και να στείλουν δεδομένα, ενώ στο TLS 1.2 ήθελε 2-RTT για να τελειώσει η χειραψία και τουλάχιστον 1-RTT για αποστολή δεδομένων αμφίδρομα.

B'4.1.3. Session Resumption και Early Data

Εκτός από ταχύτητα και ασφάλεια στη διαδικασία της χειραψίας, η νέα έκδοση προσφέρει και τη δυνατότητα για αποστολή *πρώιμων δεδομένων* (Early Data). Αυτά αποστέλλονται πριν τη χειραψία και είναι εφικτό όταν ο client έχει είτε ένα *εισιτήριο συνόδου* (session ticket) για να συνεχίσει μια προηγούμενη σύνοδο (session resumption) και το εισιτήριο δεν έχει λήξει (έχει χρόνο «ζωής», όπως τα πιστοποιητικά των server) είτε ένα pre-shared key (PSK). Το session ticket δημιουργείται από τον server όταν γίνεται μια χειραψία και το στέλνει ο client κάποια άλλη στιγμή όταν θελήσει να εγκαθιδρύσει ένα νέο session και θέλει να στείλει δεδομένα πριν τη χειραψία. Καθώς χρησιμοποιείται παλιό κλειδί (προηγούμενου session) για την κρυπτογράφηση των Early Data και δεν υπάρχει χειραψία, επιτυγχάνεται το 0-RTT. Το πρόβλημα με αυτό το χαρακτηριστικό είναι ότι δεν προσφέρει PFS. Εάν το session ticket διαρρεύσει, ένας κακόβουλος χρήστης μπορεί να κάνει μια *επίθεση επανάληψης* (replay attack) και να δει όλη τη κίνηση για εκείνο το session. Ο server δεν γνωρίζει ότι του στέλνει μήνυμα κάποιος άλλος και επαναλαμβάνει (η ονομασία της επίθεσης) την απάντηση που είχε στείλει. Με τις κατάλληλες άμυνες μπορεί να αποφευχθεί αυτό (περισσότερα σε επόμενο κεφάλαιο). Η άλλη επιλογή είναι η χρήση του PSK για τη θεμελίωση μιας ασφαλούς σύνδεσης.

B'5. Ανάλυση της χειραψίας με κώδικα

Αφού δημιουργηθεί το αρχείο, περνάω χειροκίνητα και το διαβάζω για να καταλάβω

τι ειπώθηκε στη χειραψία και αμέσως μετά σαν application data. Η χειραψία γίνεται με πακέτα και αυτά τα πακέτα περιέχουν επίπεδα. Το TLS θα εμφανίζεται μετά το TCP επίπεδο και πριν το Application (HTTP) επίπεδο. Τα πρώτα 3 πακέτα που στάλθηκαν είναι κρυπτογραφημένα σε TLS 1.2 και μεταφέρουν HTTP δεδομένα. Δεν επηρεάζουν το πείραμα και για αυτό αγνοούνται. Το πρώτο (frame 4) πακέτο είναι το Client Hello.

B'5.1. Πρώτο βήμα της χειραψίας, Client Hello

```
Frame 4: 710 bytes on wire (5680 bits), 710 bytes captured (5680 bits)
  Encapsulation type: Ethernet (1)
  Arrival Time: Sep 22, 2023 19:56:02.324940000 EEST
  [Time shift for this packet: 0.000000000 seconds]
  Epoch Time: 1695401762.324940000 seconds
  [Time delta from previous captured frame: 9.193748000 seconds]
  [Time delta from previous displayed frame: 9.193748000 seconds]
  [Time since reference or first frame: 9.240640000 seconds]
  Frame Number: 4
  Frame Length: 710 bytes (5680 bits)
  Capture Length: 710 bytes (5680 bits)
  [Frame is marked: False]
  [Frame is ignored: False]
  [Protocols in frame: eth:ethertype:ip:tcp:tls]
```

Εικόνα 12 Το tshark εμφανίζει μερικές πληροφορίες για το κάθε πακέτο πριν πάει στα επίπεδα, όπως ο χρόνος που έγινε captured στο pcap και το μέγεθός του. Στο protocols in frame φαίνονται ένα-ένα τα επίπεδα ξεκινώντας από το κατώτατο (που βρίσκεται πιο κοντά στο Data Link), Ethernet.

Το πρώτο μήνυμα στη χειραψία του TLS 1.3 στέλνεται από τον client, δηλαδή αυτόν που ξεκινάει τη σύνδεση. Όταν επισκεπτόμαστε μια ιστοσελίδα, εμείς στέλνουμε το πακέτο με το Client Hello. Το Client Hello βρίσκεται στο *επίπεδο καταγραφής του TLS* (TLS record layer) για να δείξει για τι είδους TLS πακέτο πρόκειται και απαρτίζεται από τα εξής πεδία:

- 1) Record Type. Το Client Hello στέλνεται πάντα σε μη κρυπτογραφημένη μορφή και για λόγους συμβατότητας επιλέχθηκε να είναι σε έκδοση 1.0, ώστε να το δέχεται πάντα ο server και να μην χαθεί. Στο Record Type γράφεται «Client Hello» και στο πεδίο length, το μέγεθος του client hello σε bytes.
- 2) Protocol Type. Ορίζεται ως Handshake.
- 3) Version. Για λόγους συμβατότητας με συσκευές που δεν θα καταφέρνανε να αναγνωρίσουν το πρωτόκολλο 1.3, η έκδοση αυτή μεταμφιέζεται ως TLS 1.2 και έτσι σε αυτό το πεδίο γράφει πάντα TLS 1.2. Υπάρχει μόνο για να περάσει αυτόν

τον έλεγχο. Η έκδοση που διαπραγματεύεται υπάρχει στο πεδίο Supported Versions. Το tshark εξάγει τα δεδομένα ένα-ένα και πριν φτάσει να δει τι είδους TLS version χρησιμοποιεί το client hello, το βάζει προσωρινά ως TLS 1.0. Επειδή η έκδοση TLS 1.0 ήταν μια μικρή αναθεώρηση του SSLv3.0, η κωδικοποίηση της έκδοσης είναι 0301 αντί για 0300.

```
a) Transport Layer Security
    TLSv1.3 Record Layer: Handshake Protocol: Client Hello
      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 651
```

Εικόνα 13 Εμφάνιση TLS1.3 record type, protocol type και version μέσω tshark

- 4) Random. Έμεινε από την έκδοση TLS 1.2 και συνεχίζεται να παράγεται αλλά δεν χρησιμοποιείται κάπου. Υπάρχει για λόγους συμβατότητας σε περίπτωση που ο server απαντήσει να χρησιμοποιηθεί η έκδοση 1.2. Δημιουργεί μια τυχαία γραμματοσειρά από 32 bytes.
- 5) Session ID. Έμεινε από την έκδοση TLS 1.2 και συνεχίζεται να παράγεται αλλά δεν χρησιμοποιείται κάπου. Σε περίπτωση που ο server απαντήσει να χρησιμοποιηθεί η έκδοση 1.2, το πεδίο αυτό θα ελεγχτεί για τιμή και (αν υπάρχει) θα συνεχιστεί μια παλιά σύννοδος (session resumption) χωρίς να χρειαστεί πάλι η χειραγία. Αποτελείται από 32 bytes string. Στην έκδοση 1.3 χρησιμοποιείται το PSK.

```
a) Handshake Protocol: Client Hello
    Handshake Type: Client Hello (1)
    Length: 647
    Version: TLS 1.2 (0x0303)
    Random: bea45b6e255950516eadaac2685f0eb6a8400e7f3beeaf4257b3154280d346cd
    Session ID Length: 32
    Session ID: 18bc1785aa0353a412bf692b93f200b65e9eac2b665403459c4f64a626f93f0c
```

Εικόνα 14 Εμφάνιση του Random bytes, Session ID και τη μεταμπίηση του TLS 1.3 ως TLS 1.2. Η κωδικοποίηση του 1.2 είναι 0303 αντί για 0302, παρομοίως με το 1.0.

- 6) Cipher suites (#NumberOf suites). Παρατίθενται τα cipher suites που υποστηρίζει ο client και ρωτάει τον server ποιο να επιλέξουν. Στην έκδοση 1.3 υπάρχουν 5 cipher suites αλλά συνήθως οι clients εμφανίζουν τους 3 πιο σημαντικούς για το TLS 1.3 και κάποιους για το TLS 1.2 (σε περίπτωση που ο server δεν υποστηρίζει TLS 1.3). Από αυτούς του 1.3 είναι οι TLS_AES_128_GCM_SHA256, TLS_CHACHA20_POLY1305_SHA256 και TLS_AES_256_GCM_SHA384. Ο πρώτος είναι το ελάχιστο που πρέπει να υποστηρίζεται για τη χρήση του TLS 1.3.

Οι άλλες δύο επιλογές είναι, αντίστοιχα, ο stream cipher ChaCha20-Poly1305 με συνάρτηση κατακερματισμού SHA256 και δεύτερη επιλογή τον αλγόριθμο AES με κλειδί 256 bits σε GCM mode of operation και με συνάρτηση κατακερματισμού SHA384. Στο παράδειγμα, βλέπουμε τα εξής cipher suites, που υποστηρίζονται για τον TLS 1.2. Μόνο αυτοί που χρησιμοποιούν ECDHE για την ανταλλαγή κλειδιών προσφέρουν PFS:

a) TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256

TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256

TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256

TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384

TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA

TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA

TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA

TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA

TLS_RSA_WITH_AES_128_GCM_SHA256

TLS_RSA_WITH_AES_256_GCM_SHA384

TLS_RSA_WITH_AES_128_CBC_SHA

TLS_RSA_WITH_AES_256_CBC_SHA

Cipher Suites Length: 34

Cipher Suites (17 suites)

Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)

Cipher Suite: TLS_CHACHA20_POLY1305_SHA256 (0x1303)

Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)

Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)

Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)

Cipher Suite: TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca9)

Cipher Suite: TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca8)

Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c)

Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)

Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xc00a)

Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xc009)

Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)

Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)

Cipher Suite: TLS_RSA_WITH_AES_128_GCM_SHA256 (0x009c)

Cipher Suite: TLS_RSA_WITH_AES_256_GCM_SHA384 (0x009d)

Cipher Suite: TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)

Cipher Suite: TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)

b)

Εικόνα 15 TLS 1.3 και 1.2 cipher suites που υποστηρίζει ο client. Η ονομασία των κάθε cipher suite είναι μοναδική για το καθένα και βρίσκεται στη παρένθεση. Αυτό είναι το raw data που μεταφέρει ένα πακέτο και μπορεί να συνεννοηθεί με τον server για το τι θα χρησιμοποιηθεί.

- 7) Compression Method. Λόγω συμβατότητας με την έκδοση 1.2, το πεδίο διατηρείται ακόμη αλλά η τιμή του είναι NULL. Δεν υπάρχει συμπίεση του μηνύματος στην έκδοση 1.3 λόγω της ευπάθειας CRIME η οποία επέτρεπε την διάρρηξη χρησίμων πληροφοριών (προς τον επιτιθέμενο) για τα κρυπτογραφημένα δεδομένα.

```
Compression Methods Length: 1
Compression Methods (1 method)
a) Compression Method: null (0)
```

Εικόνα 16 Η μέθοδος συμπίεσης δεν χρησιμοποιείται στην έκδοση 1.3 και η τιμή της είναι πάντα null

- 8) Extensions length. Το μέγεθος των επεκτάσεων που ακολουθούν. Οι επεκτάσεις είναι προαιρετικές και δεν απαιτείται να πληρούνται από τον server. Είναι πληροφορία την οποία στέλνει ο client στον server για όφελος του (π.χ. SNI) ή ζητάει κάτι από αυτόν (επιλογή group για τη ανταλλαγή κλειδίων συνόδου με ECDHE). Στο παράδειγμα έχει έκταση 540 bytes.
- 9) Extension: SNI (server name indicator). Γράφεται το όνομα του server που ζητάει ο client (στο παράδειγμά μας, ionio.gr). Αυτό το πεδίο είναι απαραίτητο για τον client ώστε να γνωρίζει ο παραλήπτης ποιο όνομα ψάχνει, σε περίπτωση που φιλοξενεί πολλαπλούς server.

```
Extensions Length: 540
Extension: server_name (len=13)
  Type: server_name (0)
  Length: 13
  Server Name Indication extension
    Server Name list length: 11
    Server Name Type: host_name (0)
    Server Name length: 8
a) Server Name: ionio.gr
```

Εικόνα 17 Το όνομα του server αποστέλλεται σαν plaintext, χωρίς κωδικοποίηση.

- 10) Extension: Extended Master Secret. Ο Client ενημερώνει τον server ότι υποστηρίζει επιπλέον κρυπτογραφικές μεθόδους για την κάλυψη ευπαθειών σε προηγούμενες εκδόσεις, ακόμη και που γινόταν χρήση του ECDHE. Η τιμή είναι 0 για το TLS 1.3 καθώς καλύπτει αυτά τα κενά και δεν χρειάζεται αυτό το πεδίο

πλέον. Αν συμφωνηθεί η χρήση της έκδοσης 1.2, το πεδίο θα γεμίσει με την κατάλληλη τιμή.

- 11) Extension: Renegotiation info. Στο TLS 1.3 δεν επιτρέπεται η αναδιαπραγμάτευση του πρωτοκόλλου TLS και της συμφωνίας τους και για αυτό το πεδίο μένει κενό με τιμή length = 0. Σε περίπτωση που έχει γίνει σύνδεση με παλαιότερη έκδοση TLS και ο client ζητήσει αναδιαπραγμάτευση για TLS 1.3, κάτι το οποίο γίνεται με παλαιότερες εκδόσεις, η σύνδεση πρέπει να παραμείνει με την παλιά έκδοση και να μην αλλάξει. Το ίδιο ισχύει και αν ο client ζητήσει αναδιαπραγμάτευση σε άλλη έκδοση εκτός του 1.3 και ο server απαντήσει για αλλαγή της έκδοσης σε 1.3. Ο client διακόπτει την επικοινωνία και στέλνει το κατάλληλο μήνυμα στον server.

```
Extension: extended_master_secret (len=0)
  Type: extended_master_secret (23)
  Length: 0
Extension: renegotiation_info (len=1)
  Type: renegotiation_info (65281)
  Length: 1
Renegotiation Info extension
a) Renegotiation info extension length: 0
```

- 12) Extension: Supported Groups. Το Length ορίζει το μέγεθος της λίστας και τα groups ξεκινάνε ταξινομημένα από αυτό που προτιμάει ο client. Χρησιμοποιείται για την χρήση του Elliptic Curve στον Diffie-Hellman.
- 13) Extension: EC point formats. Στις προηγούμενες εκδόσεις αυτό το πεδίο ήταν για να συμφωνηθεί μια μορφή σημείου (από πολλές) που θα χρησιμοποιηθεί στον elliptic curve του Diffie-Hellman για να βρεθούν τα δημόσια και ιδιωτικά κλειδιά. Στην έκδοση 1.3 υπάρχει μόνο μια μορφή οπότε το πεδίο αυτό έχει πάντα την ίδια τιμή. Ο client μπορεί να ενημερώσει τον server και για παραπάνω μορφές αν υποστηρίζει σε περίπτωση που δεν μπορεί να υποστηρίξει την έκδοση TLS 1.3. Ο server στο δικό του Server Hello θα στείλει μόνο μια τιμή, την “uncompressed” μορφή.

```

a) Extension: supported_groups (len=14)
    Type: supported_groups (10)
    Length: 14
    Supported Groups List Length: 12
    Supported Groups (6 groups)
        Supported Group: x25519 (0x001d)
        Supported Group: secp256r1 (0x0017)
        Supported Group: secp384r1 (0x0018)
        Supported Group: secp521r1 (0x0019)
        Supported Group: ffdhe2048 (0x0100)
        Supported Group: ffdhe3072 (0x0101)
    Extension: ec_point_formats (len=2)
        Type: ec_point_formats (11)
        Length: 2
        EC point formats Length: 1
        Elliptic curves point formats (1)
            EC point format: uncompressed (0)

```

Εικόνα 18 Οι μέθοδοι δημιουργίας κλειδιών από ελλειπτική καμπύλη φέρουν συγκεκριμένη ονομασία σε παρένθεση. Για παράδειγμα η x25519 μεταφέρεται ως 001d. Η μορφή του σημείου που θα επιλεγεί ορίζεται στο αμέσως επόμενο πεδίο

14) Extension: Session ticket. Αν έχει ξαναδημιουργηθεί σύνδεση συνόδου με session ticket το οποίο ισχύει ακόμη, αυτό το πεδίο θα το περιέχει για να χρησιμοποιηθεί από τον server αντί για τη διαδικασία της χειραψίας.

15) Extension: Application layer protocol negotiation (ALPN). Αποστέλλεται μια λίστα με τις εκδόσεις HTTP που υποστηρίζει ο client. Στο παράδειγμα ενημερώνει τον server ότι υποστηρίζει και 1.1 και 2.

```

a) Extension: session_ticket (len=0)
    Type: session_ticket (35)
    Length: 0
    Data (0 bytes)
    Extension: application_layer_protocol_negotiation (len=14)
        Type: application_layer_protocol_negotiation (16)
        Length: 14
        ALPN Extension Length: 12
        ALPN Protocol
            ALPN string length: 2
            ALPN Next Protocol: h2
            ALPN string length: 8
            ALPN Next Protocol: http/1.1

```

Εικόνα 19 Δεν υπάρχει session ticket οπότε στέλνεται μόνο η κεφαλίδα για την επέκταση χωρίς κάποια τιμή. Η επέκταση ALPN είναι 14 bytes

16) Extension: Status request. Στο TLS 1.3 επιτρέπεται να στέλνονται πολλαπλά αιτήματα για Online Certificate Status Protocol (OCSP), εξαλείφοντας το περιορισμό που υπήρχε σε προηγούμενες εκδόσεις όπου στέλνονταν μόνο ένα. Το OCSP χρησιμοποιείται για να ελεγχτεί εάν ένα X.509 πιστοποιητικό έχει ανακληθεί. Γεμίζοντας αυτό το πεδίο με τη τιμή που υποστηρίζει ο client, κάνει αίτημα στον server να στείλει μια *χρονοσφραγίδα* (timestamp) από την *αρχή έκδοσης πιστοποιητικού* (Certificate Authority - CA) η οποία επιβεβαιώνει την υπογραφή του πιστοποιητικού. Στο Server Hello, μαζί με το πιστοποιητικό, ο server στέλνει το OCSP response για να εκπληρώσει το αίτημα του client. Στο παράδειγμά μας ο client έχει βάλει τιμή 0 στο πεδίο καθώς δεν ζητάει κάτι τέτοιο.

a)

```
Extension: status_request (len=5)
      Type: status_request (5)
      Length: 5
      Certificate Status Type: OCSP (1)
      Responder ID list Length: 0
      Request Extensions Length: 0
```

Εικόνα 20 Η επέκταση για το OCSP πιάνει 5 bytes

17) Extension: Delegated credentials [16]. Ένας τρόπος πιστοποίησης του server, αντί για τον παραδοσιακό με Certificate. Αντί να στέλνεται το πιστοποιητικό του server και να αποθηκεύεται στον client μακροχρόνια, δημιουργείται μια πιστοποίηση μικρότερης διάρκειας. Βοηθάει servers που έχουν μεγάλη κίνηση στο δίκτυο τους για την ελάττωση μιας πιθανής κλοπής του ιδιωτικού κλειδιού. Στο παράδειγμα φαίνονται 4 αλγόριθμοι που υποστηρίζει ο client. Αλγόριθμος υπογραφής ECDSA με συνάρτηση κατακερματισμού SHA256 και supported group SECP256R1, ο ECDSA_SECP384R1_SHA384, ο ECDSA_SECP521R1_SHA512 και ο ECDSA_SHA1.

```

a) Extension: delegated_credentials (len=10)
    Type: delegated_credentials (34)
    Length: 10
    Signature Hash Algorithms Length: 8
    Signature Hash Algorithms (4 algorithms)
        Signature Algorithm: ecdsa_secp256r1_sha256 (0x0403)
            Signature Hash Algorithm Hash: SHA256 (4)
            Signature Hash Algorithm Signature: ECDSA (3)
        Signature Algorithm: ecdsa_secp384r1_sha384 (0x0503)
            Signature Hash Algorithm Hash: SHA384 (5)
            Signature Hash Algorithm Signature: ECDSA (3)
        Signature Algorithm: ecdsa_secp521r1_sha512 (0x0603)
            Signature Hash Algorithm Hash: SHA512 (6)
            Signature Hash Algorithm Signature: ECDSA (3)
        Signature Algorithm: ecdsa_sha1 (0x0203)
            Signature Hash Algorithm Hash: SHA1 (2)
            Signature Hash Algorithm Signature: ECDSA (3)

```

Εικόνα 21 Ο κάθε αλγόριθμος φέρει μια συγκεκριμένη ονομασία στην κεφαλίδα αυτή και εμφανίζεται από το *tshark* στις παρενθέσεις. Για παράδειγμα, ο *ECDSA_SECP256R1_SHA256* έχει ονομασία *0403*

18) Extension: Key share. Σε αυτό το πεδίο υπάρχει το μέγεθος των δημοσίων κλειδιών που δημιουργήθηκαν και αμέσως μετά η τιμή του κλειδιού. Στο παράδειγμα ο client δημιούργησε δυο δημόσια κλειδιά με δυο διαφορετικά supported groups, το *x25519* και το *secp256r1*, ταξινομημένα ανά μεγαλύτερη προτίμηση.

```

a) Extension: key_share (len=107)
    Type: key_share (51)
    Length: 107
    Key Share extension
        Client Key Share Length: 105
        Key Share Entry: Group: x25519, Key Exchange length: 32
            Group: x25519 (29)
            Key Exchange Length: 32
            Key Exchange: 295e18505813bbae156de841707f13d039c170310f4564fa89f23e30106b5535
        Key Share Entry: Group: secp256r1, Key Exchange length: 65
            Group: secp256r1 (23)
            Key Exchange Length: 65
            Key Exchange: 045ef645a8c7ed6a5ebe3bf00e5d470715ef0d9a3e24960b69769eb96631f8eef126af5a...

```

Εικόνα 22 Εάν ο server υποστηρίζει τα group *x25519* και *secp256r1*, θα επιλέξει το πρώτο και θα πάρει τη τιμή του key exchange ως το δημόσιο κλειδί του client

19) Extension: Supported versions. Σε αυτό το πεδίο ο client γράφει ποιες TLS εκδόσεις υποστηρίζει και ρωτάει τον server να επιλέξει. Στο παράδειγμα είναι οι εκδόσεις 1.2 και 1.3.

a) `Extension: supported_versions (len=5)
Type: supported_versions (43)
Length: 5
Supported Versions length: 4
Supported Version: TLS 1.3 (0x0304)
Supported Version: TLS 1.2 (0x0303)`

Εικόνα 23 Ο client ενημερώνει τον server σε 5 bytes ποιες εκδόσεις υποστηρίζει. Η έκδοση 1.3 έχει κωδικοποίηση 0304

20) Extension: Signature Algorithms. Αλγόριθμοι υπογραφής κατακερματισμού που υποστηρίζει ο client και ζητάει από το server για το record layer CertificateVerify (περισσότερες λεπτομέρειες στο Server Hello). Στο παράδειγμα αναφέρει 11 αλγορίθμους.

```

Extension: signature_algorithms (len=24)
  Type: signature_algorithms (13)
  Length: 24
  Signature Hash Algorithms Length: 22
  Signature Hash Algorithms (11 algorithms)
    Signature Algorithm: ecdsa_secp256r1_sha256 (0x0403)
      Signature Hash Algorithm Hash: SHA256 (4)
      Signature Hash Algorithm Signature: ECDSA (3)
    Signature Algorithm: ecdsa_secp384r1_sha384 (0x0503)
      Signature Hash Algorithm Hash: SHA384 (5)
      Signature Hash Algorithm Signature: ECDSA (3)
    Signature Algorithm: ecdsa_secp521r1_sha512 (0x0603)
      Signature Hash Algorithm Hash: SHA512 (6)
      Signature Hash Algorithm Signature: ECDSA (3)
    Signature Algorithm: rsa_pss_rsae_sha256 (0x0804)
      Signature Hash Algorithm Hash: Unknown (8)
      Signature Hash Algorithm Signature: SM2 (4)
    Signature Algorithm: rsa_pss_rsae_sha384 (0x0805)
      Signature Hash Algorithm Hash: Unknown (8)
      Signature Hash Algorithm Signature: Unknown (5)
    Signature Algorithm: rsa_pss_rsae_sha512 (0x0806)
      Signature Hash Algorithm Hash: Unknown (8)
      Signature Hash Algorithm Signature: Unknown (6)
    Signature Algorithm: rsa_pkcs1_sha256 (0x0401)
      Signature Hash Algorithm Hash: SHA256 (4)
      Signature Hash Algorithm Signature: RSA (1)
    Signature Algorithm: rsa_pkcs1_sha384 (0x0501)
      Signature Hash Algorithm Hash: SHA384 (5)
      Signature Hash Algorithm Signature: RSA (1)
    Signature Algorithm: rsa_pkcs1_sha512 (0x0601)
      Signature Hash Algorithm Hash: SHA512 (6)
      Signature Hash Algorithm Signature: RSA (1)
    Signature Algorithm: ecdsa_sha1 (0x0203)
      Signature Hash Algorithm Hash: SHA1 (2)
      Signature Hash Algorithm Signature: ECDSA (3)
    Signature Algorithm: rsa_pkcs1_sha1 (0x0201)
      Signature Hash Algorithm Hash: SHA1 (2)
      Signature Hash Algorithm Signature: RSA (1)

```

a)

Εικόνα 24 Πεδία με "Unknown" δεν μπόρεσαν να βρεθούν από το tshark. Ο αλγόριθμος hash είναι πάντα SHA(#Έκδοσης ή #bits key)

- 21) PSK key exchange modes. Σε περίπτωση που υπάρχει ένα PSK, αναγράφεται ο αλγόριθμος που υποστηρίζει ο client για τη δημιουργία κλειδιών από αυτό. Στο παράδειγμα είναι ο ephemeral Diffie-Hellman και προαιρετικά με elliptic curve.
- 22) Extension: Record size limit. Το μέγεθος του TLS record layer, δεν μπορεί να ξεπερνά τα 2^{14} bytes (πρακτικά $2^{14} + 1$ οκτάδες) [17]. Είναι το νούμερο που ενημερώνει στον server ότι μπορεί να υποστηρίξει (περί το standard του TLS 1.3).

```

Extension: psk_key_exchange_modes (len=2)
  Type: psk_key_exchange_modes (45)
  Length: 2
  PSK Key Exchange Modes Length: 1
  PSK Key Exchange Mode: PSK with (EC)DHE key establishment (psk_dhe_ke) (1)
Extension: record_size_limit (len=2)
  Type: record_size_limit (28)
  Length: 2
  Record Size Limit: 16385

```

a)

Εικόνα 25 Ο client ενημερώνει τον server ότι υποστηρίζει τη δημιουργία pre-shared key με τον (EC)DHE. Το μέγεθος του TLS record δεν μπορεί να ξεπερνά τα 16385 bytes.

23) JA3 Fullstring. Αποτύπωμα του μηνύματος στο διαδίκτυο. Αναγράφεται με νούμερα και χωρίζεται με κόμματα για τα πεδία. Τα πεδία για το Client Hello και το Server Hello διαφέρουν. Στο Client Hello είναι τα Version, Cipher Suites, Extensions (το μέγεθος της λίστας), Elliptic Curves και τα EC point formats τους. Το κάθε πεδίο έχει τις τιμές του χωρισμένες με παύλες. Το TLS version που χρησιμοποιείται για την σύνδεση (1.2 για το Client Hello), ονομαστικά το κάθε cipher suite και επέκταση, τα supported groups του Elliptic Curve και τα point formats του [18].

24) JA3. Κάνοντας χρήση του MD5, παράγεται μια γραμματοσειρά από 32 χαρακτήρες που είναι το hashed JA3 Fullstring [18].

```

[JA3 Fullstring:
771,4865-4867-4866-49195-49199-52393-52392-49196-49200-49162-49161-49171-49172-156-157-47-53,0-23-652
81-10-11-35-16-5-34-51-43-13-45-28-65037,29-23-24-25-256-257,0]
[JA3: b5001237acdf006056b409cc433726b0]

```

a)

Εικόνα 26 Το αποτύπωμα που αφήνουμε στο διαδίκτυο

B'5.1.1. Early Data

Όλο το Client Hello στέλνεται μη κρυπτογραφημένο. Δεν υπάρχει κάποια πληροφορία που θα μπορούσε να χρησιμοποιήσει ένας επιτιθέμενος για να μάθει το κοινό μυστικό κλειδί που παράγεται αργότερα. Παρόλα αυτά, υπάρχει η δυνατότητα στην έκδοση 1.3 να στείλει ο client πιο νωρίς δεδομένα, αναφερόμενα ως early data, όπως για παράδειγμα να ζητήσει μια σελίδα πριν γίνει η χειραψία. Σε αυτή τη περίπτωση είτε χρησιμοποιείται ένα session ticket είτε δημιουργείται ένα νέο ζευγάρι κλειδιών (από το PSK που μάθαμε νωρίτερα) με τον server για την κρυπτογράφηση και αποκρυπτογράφηση των early data. Αυτά τα δεδομένα στέλνονται πριν την χειραψία (και για αυτό είναι 0-RTT) και ο server απαντάει στον client μετά το Server Hello. Τα early data είναι ευάλωτα σε replay attack όπου ένας κακόβουλος χρήστης πιάνει αυτό το μήνυμα και το επαναλαμβάνει στον server. Καθώς ο server πιστεύει ότι ο χρήστης είναι αυτός που ξαναστέλνει το αίτημα, το δέχεται και προωθούνται οι

ανάλογες ενέργειες. Έστω ότι ένας χρήστης (Α) επισκέπτεται ένα ξενοδοχείο στο οποίο υπάρχει WiFi. Συνδέεται, κάνει τη δουλειά του, δημιουργούνται κλειδιά και σύνδεση με TLS 1.3 και αποσυνδέεται. Την επόμενη μέρα εισέρχεται ξανά στο χώρο του ξενοδοχείου και συνδέεται στο WiFi. Ζητάει να επισκεφθεί την ίδια σελίδα και χρησιμοποιείται το PSK για να σταλθούν δεδομένα (π.χ. αγορά ενός ακριβού γιλέκου από ηλεκτρονικό κατάστημα). Ένας άλλος χρήστης (Β) βρίσκεται στον ίδιο χώρο και είναι συνδεδεμένος και αυτός στο WiFi. Ακούει τη κίνηση που γίνεται στο δίκτυο. Ο χρήστης Α αποσυνδέεται και αποχωρεί από το χώρο. Ο χρήστης Β επαναλαμβάνει το μήνυμα που έστειλε ο χρήστης Α, πιστεύοντας ότι θα καταφέρει να βρει ένα μήνυμα ευάλωτο σε replay attack, κάτι το οποίο βρίσκει στο early data που στάλθηκε. Με αυτή την επίθεση ζητάει από τον server να επαναληφθεί το αίτημα και ο server το προωθεί. Ο χρήστης Α σε μια εβδομάδα θα δει δύο γιλέκα στη πόρτα του και το λογαριασμό του χρεωμένο διπλά. Για αυτό καλό είναι να υλοποιηθούν άμυνες εναντίον του replay attack ή να ελέγχεται εάν το αίτημα που λήφθηκε είναι όντως 0-RTT ή όχι. Καθώς δεν έχει βρεθεί προς το παρόν τρόπος για να καλυφθεί αυτό το κενό, μεγάλοι πάροχοι υπηρεσιών δικτύου όπως η Cloudflare [19] αποφάσισαν προς το παρόν να μην επιτρέπουν (τα περισσότερα) 0-RTT αιτήματα.

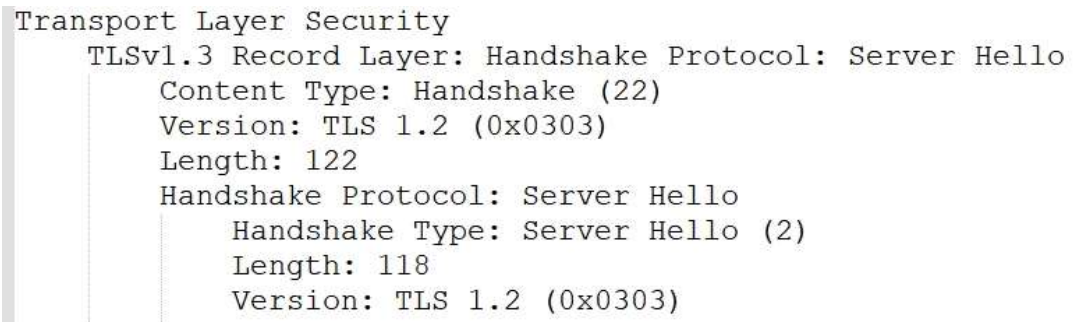
B'5.2. Δεύτερο βήμα της χειραψίας, Server Hello, Server Finished

```
Frame 7: 2974 bytes on wire (23792 bits), 2974 bytes captured (23792 bits)
  Encapsulation type: Ethernet (1)
  Arrival Time: Sep 22, 2023 19:56:02.363938000 EEST
  [Time shift for this packet: 0.000000000 seconds]
  Epoch Time: 1695401762.363938000 seconds
  [Time delta from previous captured frame: 0.018524000 seconds]
  [Time delta from previous displayed frame: 0.018524000 seconds]
  [Time since reference or first frame: 9.279638000 seconds]
  Frame Number: 7
  Frame Length: 2974 bytes (23792 bits)
  Capture Length: 2974 bytes (23792 bits)
  [Frame is marked: False]
  [Frame is ignored: False]
  [Protocols in frame: eth:ethertype:ip:tcp:tls]
```

Εικόνα 27 Επίπεδα στο Server hello είναι ίδια με το Client Hello.

- 1) Record Type. Το μισό Server Hello στέλνεται πάντα σε μη κρυπτογραφημένη μορφή μέχρι τη στιγμή που θα σταλθεί το δημόσιο κλειδί του server. Εδώ γράφεται «Server Hello» και στο length το μέγεθος του Server Hello σε bytes.
- 2) Protocol Type. Ορίζεται ως Handshake.
- 3) Version. Για λόγους συμβατότητας με συσκευές που δεν θα καταφέρνανε να αναγνωρίσουν το πρωτόκολλο 1.3, η έκδοση αυτή μεταμφιέζεται ως TLS 1.2 και

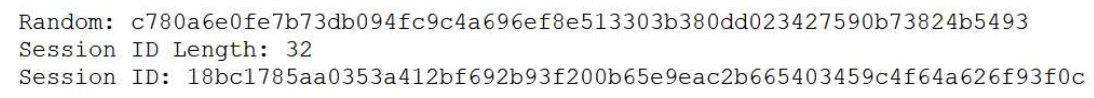
έτσι σε αυτό το πεδίο γράφει πάντα TLS 1.2. Υπάρχει μόνο για να περάσει αυτόν τον έλεγχο. Η έκδοση που θα χρησιμοποιηθεί υπάρχει στο πεδίο Supported Versions, όπου αναγράφεται μόνο το TLS 1.3.

a) 

Εικόνα 28 Ο Server χρησιμοποιεί TLS 1.3 και την μεταμφιέζει ως 1.2 για να περάσει τους ελέγχους συμβατότητας. Έχει περιεχόμενο 118 bytes.

4) Random. Έμεινε από την έκδοση TLS 1.2 και συνεχίζεται να παράγεται αλλά δεν χρησιμοποιείται κάπου. Υπάρχει για λόγους συμβατότητας σε περίπτωση που ο server απαντήσει να χρησιμοποιηθεί η έκδοση 1.2. Δημιουργεί μια τυχαία γραμματοσειρά από 32 bytes.

5) Session ID. Έμεινε από την έκδοση TLS 1.2. Καθώς στην έκδοση 1.3 δεν χρησιμοποιείται, ο server αντιγράφει το session ID που του έστειλε ο client και το βάζει σαν τιμή στο πεδίο του. Σε περίπτωση που ο server απαντούσε να χρησιμοποιηθεί η έκδοση 1.2, θα έλεγχε το session ID του client και (αν υπήρχε) θα συνεχιζόταν μια παλιά σύνοδος χωρίς να χρειαστεί πάλι η χειραψία. Αποτελείται από 32 bytes string. Αν ο client ήθελε να συνεχίσει μια σύνοδο με TLS 1.3, θα χρησιμοποιούταν το PSK.

a) 

Εικόνα 29 Η τιμή του random που δημιουργεί και το echo του Session ID που έστειλε ο client

6) Cipher suite. Παρατίθεται το cipher suite για το TLS 1.3 που αποφάσισε ο server να χρησιμοποιήσει, βάση αυτών που υποστηρίζει ο client, δηλαδή TLS_AES_128_GCM_SHA256.

7) Compression method. Η τεχνική συμπίεσης που θα χρησιμοποιηθεί (NULL στο TLS 1.3).

Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
a) Compression Method: null (0)

Εικόνα 30 Αποφασίστηκε από τον server να χρησιμοποιηθεί ο αλγόριθμος AES με 128 bits κλειδί κρυπτογράφησης σε GCM mode και αυθεντικοποίηση των μηνυμάτων με SHA256.

- 8) Extensions length. Ίδια τιμή με τον client. Το πεδίο αυτό δείχνει στον client πόση είναι η έκταση των επεκτάσεων που θα στείλει, η οποία είναι πάντα (πολύ) μικρότερη από αυτή του client. Ο server στο δικό του Hello στέλνει μόνο τις ρυθμίσεις που θα εφαρμοστούν για τη συμφωνία τους (TLS 1.3), όχι τι υποστηρίζει.
- 9) Extension: Supported versions. Εδώ αναφέρεται ποια έκδοση θα χρησιμοποιηθεί και είναι το μοναδικό πεδίο που αναφέρεται η έκδοση 1.3. Ο client θα διαβάσει τη τιμή της επέκτασης και θα γνωρίζει ποια έκδοση επιλέχθηκε.
- 10) Extension: Key share. Αφού παραχθεί το δημόσιο κλειδί του server, μπαίνει σε αυτό το πεδίο (μαζί με τη πρώτη προτίμηση του supported group που επιλέχθηκε για τον elliptic curve αν υποστηρίζεται από τον client και τον server) και αποστέλλεται στον client ώστε να υπολογίσουν το κοινό μυστικό κλειδί.

```
Extensions Length: 46
Extension: supported_versions (len=2)
  Type: supported_versions (43)
  Length: 2
  Supported Version: TLS 1.3 (0x0304)
Extension: key_share (len=36)
  Type: key_share (51)
  Length: 36
  Key Share extension
    Key Share Entry: Group: x25519, Key Exchange length: 32
    Group: x25519 (29)
    Key Exchange Length: 32
    Key Exchange: d4647537ea8cbd9fa82bae3da690b9c15781c2d8ca97505e24244eca28c76e37
```

a)

Εικόνα 31 Οι επεκτάσεις που χρειάζεται να στείλει είναι η έκδοση TLS που θα συμφωνήσουν και το δημόσιο κλειδί του server με το group που το παρήγαγε.

- 11) JA3S Fullstring. Ίδια λογική με το αποτύπωμα του client, αλλά περιέχει πολύ λιγότερα πεδία και τιμές. Είναι το πεδίο για την έκδοση TLS (TLS version) του μηνύματος, το αλγόριθμο κρυπτογράφησης που επιλέχθηκε (Cipher) και ονομαστικά τις επεκτάσεις (Extensions).
- 12) JA3S. Hashed string του αποτύπωματος.

```
[JA3S Fullstring: 771,4865,43-51]
a) [JA3S: f4febc55ea12b31ae17cfb7e614afda8]
```

Εικόνα 32 Το αποτύπωμα του server είναι πάντα μικρότερο από του client

Πριν τελειώσει το Server hello, γίνεται ο υπολογισμός του shared secret key, καθώς έχει το δημόσιο κλειδί του client. Έπειτα συνεχίζεται αλλά όλα τα πεδία είναι πλέον κρυπτογραφημένα με το κλειδί (το οποίο θα το μάθει ο client όταν λάβει το server hello). Σε αντίθεση με το TLS 1.2 όπου η χειραψία ήταν μη κρυπτογραφημένη, μπορούμε στο TLS 1.3 να στέλνουμε δεδομένα με ασφάλεια κατευθείαν μετά το key_share extension.

Δημιουργείται ένα νέο (κρυπτογραφημένο) record layer ονομαζόμενο Change Cipher Spec. Αυτό λέει στον client ότι έχει ενημερώσει τις παραμέτρους της συμφωνίας (cipher, tls version κλπ.) με καινούργιες τιμές. Το μήνυμα Change Cipher Spec έχει κληρονομηθεί από τη προηγούμενη έκδοση. Δεν χρειάζεται πλέον αλλά υπάρχει για λόγους συμβατότητας και εκμεταλλεύεται από τον server για να μεταμφιέσει το TLS 1.3 ως 1.2.

```
[0A33: 141E0C03Ea12D31aE17C1D7E014a1a00]
TLSv1.3 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
Content Type: Change Cipher Spec (20)
Version: TLS 1.2 (0x0303)
Length: 1
Change Cipher Spec Message
```

Εικόνα 33 Η έκδοση είναι κρυπτογραφημένη με το shared secret key και το μήνυμα στέλνεται ως Change Cipher Spec Encrypted

Στο δεύτερο (κρυπτογραφημένο) record layer είναι οι επεκτάσεις που στέλνει όπως το SNI (το οποίο επιστρέφει με τιμή 0) και η επιλογή ALPN για να δείξει στον client τι

έκδοση HTTP θα παρουσιαστεί (1.1).

```

TLSv1.3 Record Layer: Handshake Protocol: Encrypted Extensions
  Opaque Type: Application Data (23)
  Version: TLS 1.2 (0x0303)
  Length: 42
  [Content Type: Handshake (22)]
  Handshake Protocol: Encrypted Extensions
    Handshake Type: Encrypted Extensions (8)
    Length: 21
    Extensions Length: 19
    Extension: server_name (len=0)
      Type: server_name (0)
      Length: 0
    Extension: application_layer_protocol_negotiation (len=11)
      Type: application_layer_protocol_negotiation (16)
      Length: 11
      ALPN Extension Length: 9
      ALPN Protocol
        ALPN string length: 8
        ALPN Next Protocol: http/1.1

```

Εικόνα 34 Πριν τη χρήση του shared secret key, αντί για κανονικές ονομασίες στις επεκτάσεις εμφανίζει μόνο Encrypted Extensions.

B'5.2.1 Certificate Chain

Κάποιες φορές το πιστοποιητικό του server αποστέλλεται στο ίδιο frame, ενώ άλλες φορές σε ξεχωριστό. Στο παράδειγμα έγινε στο αμέσως επόμενο frame, όπου στέλνεται η αλυσίδα πιστοποιητικών. Η αλυσίδα πιστοποιεί το `ionio.gr`, πιστοποιεί την αρχή που πιστοποιεί το `ionio.gr` και ούτω καθεξής μέχρι να φτάσουμε στην ανώτατη CA η οποία πιστοποιεί τον εαυτό της.

Όταν τα δεδομένα είναι κρυπτογραφημένα, τα επίπεδα που εμφανίζονται τελειώνουν πάντα στο `tls`, αλλά εδώ μπορούμε να δούμε τι ακολουθεί (η αλυσίδα πιστοποιητικών).

```

Frame 8: 1209 bytes on wire (9672 bits), 1209 bytes captured (9672 bits)
  Encapsulation type: Ethernet (1)
    Arrival Time: Sep 22, 2023 19:56:02.371793000 EEST
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1695401762.371793000 seconds
    [Time delta from previous captured frame: 0.007855000 seconds]
    [Time delta from previous displayed frame: 0.007855000 seconds]
    [Time since reference or first frame: 9.287493000 seconds]
    Frame Number: 8
    Frame Length: 1209 bytes (9672 bits)
    Capture Length: 1209 bytes (9672 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame [truncated]]:
    eth:ethertype:ip:tcp:tls:x509sat:x509sat:x509sat:x509sat:x509sat:x509sat:x509sat:x509ce:x509ce:x509ce:
    x509ce:x509ce:x509ce:pkixexplicit:x509ce:pkiximplicit:x509ce:tls:x509sat:x509sat:x509sat:x509sat:x50
    9s1

```

Εικόνα 35 Το πακέτο που στέλνεται είναι αμέσως επόμενο από το προηγούμενο. Δεν μεσολαβεί κανένα άλλο και έχει μέγεθος 1209 bytes.

```
[4 Reassembled TCP Segments (6187 bytes): #7(2740), #8(1176), #8(1460), #8(811)]
[Frame: 7, payload: 0-2739 (2740 bytes)]
[Frame: 8, payload: 2740-3915 (1176 bytes)]
[Frame: 8, payload: 3916-5375 (1460 bytes)]
[Frame: 8, payload: 5376-6186 (811 bytes)]
[Segment count: 4]
[Reassembled TCP length: 6187]
[Reassembled TCP Data: 17030318267eda338eb3a0951efb6daa4d9574ec64f26a143387ed2445dcd14c299c5415...]
Transport Layer Security
  TLSv1.3 Record Layer: Handshake Protocol: Certificate
```

Εικόνα 36 Λίγο πιο μετά, στο ίδιο πακέτο, στο TCP layer παρατηρούμε ότι υπήρχαν 4 payloads τα οποία συναρμολογήθηκαν και το μέγεθος αυξήθηκε στα 6187 bytes.

Αρχικά φαίνεται για τι είδους Handshake πρόκειται. Handshake Protocol: Certificate. Αυτό μας ενημερώνει ότι η χειραψία βρίσκεται ακόμη στο Server Hello και τώρα στέλνει ο server το πιστοποιητικό του (ή καλύτερα, αυτό και την αλυσίδα αυτού). Επειδή το string είναι πολύ μεγάλο, το tshark σταματάει να το εμφανίζει και βάζει τρεις τελείες στο τέλος για να δείξει ότι συνεχίζεται. Στο raw data είναι κρυπτογραφημένο και μικρότερο αλλά μπορούμε να το δούμε ολόκληρο.

```
Transport Layer Security
  TLSv1.3 Record Layer: Handshake Protocol: Certificate
    Opaque Type: Application Data (23)
    Version: TLS 1.2 (0x0303)
    Length: 6182
    [Content Type: Handshake (22)]
    Handshake Protocol: Certificate
      Handshake Type: Certificate (11)
      Length: 6161
      Certificate Request Context Length: 0
      Certificates Length: 6157
      Certificates (6157 bytes)
        Certificate Length: 1877
        Certificate: 3082075130820539a003020102021100d66eccc4fb4e7e11eed2bcda2d1744f6300d0609...
        (id-at-commonName=www.ionio.gr,id-at-organizationName=Ionian University,id-at-stateOrProvinceName=Ionian
        Nisia,id-at-countryName=GR)
```

Εικόνα 37 Η αλυσίδα είναι 6161 bytes. Στο πεδίο Certificate φαίνεται το πιστοποιητικό του ionio.gr σε δεκαεξαδική μορφή και σε plaintext

Το πιστοποιητικό είναι υπογεγραμμένο με μοναδικό σειριακό αριθμό `00d66eccc4fb4e7e11eed2bcda2d1744f6` και συνάρτηση κατακερματισμού SHA384 με κρυπτογράφηση RSA. Όπως είχαμε πει νωρίτερα, ο RSA μπορεί να χρησιμοποιείται για την αυθεντικοποίηση του πιστοποιητικού. Η CA που το υπέγραψε είναι η GEANT OV RSA CA 4 (ανήκει στην GEANT Vereniging), με έδρα

την Ολλανδία.

```
signedCertificate
  version: v3 (2)
  serialNumber: 0x00d66eccc4fb4e7e11eed2bcda2d1744f6
  signature (sha384WithRSAEncryption)
    Algorithm Id: 1.2.840.113549.1.1.12 (sha384WithRSAEncryption)
  issuer: rdnSequence (0)
    rdnSequence: 3 items (id-at-commonName=GEANT OV RSA CA 4,id-at-organizationName=GEANT
    Vereniging,id-at-countryName=NL)
      RDNSSequence item: 1 item (id-at-countryName=NL)
        RelativeDistinguishedName item (id-at-countryName=NL)
          Object Id: 2.5.4.6 (id-at-countryName)
          CountryName: NL
      RDNSSequence item: 1 item (id-at-organizationName=GEANT Vereniging)
        RelativeDistinguishedName item (id-at-organizationName=GEANT Vereniging)
          Object Id: 2.5.4.10 (id-at-organizationName)
          DirectoryString: printableString (1)
            printableString: GEANT Vereniging
      RDNSSequence item: 1 item (id-at-commonName=GEANT OV RSA CA 4)
        RelativeDistinguishedName item (id-at-commonName=GEANT OV RSA CA 4)
          Object Id: 2.5.4.3 (id-at-commonName)
          DirectoryString: printableString (1)
            printableString: GEANT OV RSA CA 4
```

Εικόνα 38 Τα χαρακτηριστικά του CA εμφανίζονται στο πιστοποιητικό του ionio.gr

Στο πιστοποιητικό υπάρχει φυσικά ένα πεδίο που προσδιορίζει την ώρα και χρονολογία από την οποία ισχύει το πιστοποιητικό και τότε λήγει. Το επόμενο πεδίο γράφει ποιον πιστοποιεί η CA. Πρώτα εμφανίζεται η χώρα (GR για Ελλάδα), έπειτα η επαρχία (Ionía Nisia), μετά ο οργανισμός στον οποίο ανήκει το πιστοποιητικό (Ionian University) και τέλος ποια domain πιστοποιεί (το www.ionio.gr)

```
validity
  notBefore: utcTime (0)
    utcTime: 2023-04-06 00:00:00 (UTC)
  notAfter: utcTime (0)
    utcTime: 2024-04-05 23:59:59 (UTC)
subject: rdnSequence (0)
  rdnSequence: 4 items (id-at-commonName=www.ionio.gr,id-at-organizationName=Ionian
  University,id-at-stateOrProvinceName=Ionía Nisia,id-at-countryName=GR)
    RDNSSequence item: 1 item (id-at-countryName=GR)
      RelativeDistinguishedName item (id-at-countryName=GR)
        Object Id: 2.5.4.6 (id-at-countryName)
        CountryName: GR
    RDNSSequence item: 1 item (id-at-stateOrProvinceName=Ionía Nisia)
      RelativeDistinguishedName item (id-at-stateOrProvinceName=Ionía Nisia)
        Object Id: 2.5.4.8 (id-at-stateOrProvinceName)
        DirectoryString: uTF8String (4)
          uTF8String: Ionía Nisia
    RDNSSequence item: 1 item (id-at-organizationName=Ionian University)
      RelativeDistinguishedName item (id-at-organizationName=Ionian University)
        Object Id: 2.5.4.10 (id-at-organizationName)
        DirectoryString: printableString (1)
          printableString: Ionian University
    RDNSSequence item: 1 item (id-at-commonName=www.ionio.gr)
      RelativeDistinguishedName item (id-at-commonName=www.ionio.gr)
        Object Id: 2.5.4.3 (id-at-commonName)
        DirectoryString: printableString (1)
          printableString: www.ionio.gr
```

Εικόνα 39 Όλα αυτά θα ελεγχθούν από τον client για να επιβεβαιωθεί η ταυτότητα του server

Μαζί με το πιστοποιητικό αποστέλλεται και το δημόσιο κλειδί του server. Αυτό θα χρησιμοποιηθεί από τον client για την αποκρυπτογράφηση του πιστοποιητικού.

Περισσότερα στο επόμενο Client handshake μήνυμα.

```
subjectPublicKeyInfo
  algorithm (rsaEncryption)
    Algorithm Id: 1.2.840.113549.1.1.1 (rsaEncryption)
  subjectPublicKey: 3082010a028201010095388e811eb603d33304206379113374c63a8dd861038813064d76...
    modulus: 0x0095388e811eb603d33304206379113374c63a8dd861038813064d7603c5d3e639b90fd1...
    publicExponent: 65537
```

Εικόνα 40 Το δημόσιο κλειδί του *ionio.gr* μπαίνει στο πεδίο για το *subjectPublicKey*

Ακολουθούν λεπτομερείς πληροφορίες για το πιστοποιητικό που εκδόθηκε από την CA προς το *ionio.gr* μέσα σε επεκτάσεις.

```
extensions: 10 items
  Extension (id-ce-authorityKeyIdentifier)
    Extension Id: 2.5.29.35 (id-ce-authorityKeyIdentifier)
    AuthorityKeyIdentifier
      keyIdentifier: 6f1d3549106c32fa59a09ebc8ae81f95be717a0c
  Extension (id-ce-subjectKeyIdentifier)
    Extension Id: 2.5.29.14 (id-ce-subjectKeyIdentifier)
    SubjectKeyIdentifier: 7bed14adfc6f51868b0f05d2fff811088691d9aa
  Extension (id-ce-keyUsage)
    Extension Id: 2.5.29.15 (id-ce-keyUsage)
    critical: True
    Padding: 5
    KeyUsage: a0
      1... .... = digitalSignature: True
      .0.. .... = contentCommitment: False
      ..1. .... = keyEncipherment: True
      ...0 .... = dataEncipherment: False
      .... 0... = keyAgreement: False
      .... .0.. = keyCertSign: False
      .... ..0. = cRLSign: False
      .... ...0 = encipherOnly: False
      0... .... = decipherOnly: False
  Extension (id-ce-basicConstraints)
    Extension Id: 2.5.29.19 (id-ce-basicConstraints)
    critical: True
    BasicConstraintsSyntax [0 length]
  Extension (id-ce-extKeyUsage)
    Extension Id: 2.5.29.37 (id-ce-extKeyUsage)
    KeyPurposeIDs: 2 items
      KeyPurposeId: 1.3.6.1.5.5.7.3.1 (id-kp-serverAuth)
      KeyPurposeId: 1.3.6.1.5.5.7.3.2 (id-kp-clientAuth)
  Extension (id-ce-certificatePolicies)
    Extension Id: 2.5.29.32 (id-ce-certificatePolicies)
    CertificatePoliciesSyntax: 2 items
      PolicyInformation
        policyIdentifier: 1.3.6.1.4.1.6449.1.2.2.79 (iso.3.6.1.4.1.6449.1.2.2.79)
        policyQualifiers: 1 item
          PolicyQualifierInfo
            Id: 1.3.6.1.5.5.7.2.1 (id-qt-cps)
            DirectoryString: https://sectigo.com/CPS
      PolicyInformation
        policyIdentifier: 2.23.140.1.2.2 (joint-iso-itu-t.23.140.1.2.2)
```

Εικόνα 41 Extensions του πιστοποιητικού

```

Extension (id-ce-cRLDistributionPoints)
  Extension Id: 2.5.29.31 (id-ce-cRLDistributionPoints)
  CRLDistPointsSyntax: 1 item
    DistributionPoint
      distributionPoint: fullName (0)
        fullName: 1 item
          GeneralName: uniformResourceIdentifier (6)
            uniformResourceIdentifier: http://GEANT.crl.sectigo.com/GEANTOVRSACA4.crl
Extension (id-pe-authorityInfoAccess)
  Extension Id: 1.3.6.1.5.5.7.1.1 (id-pe-authorityInfoAccess)
  AuthorityInfoAccessSyntax: 2 items
    AccessDescription
      accessMethod: 1.3.6.1.5.5.7.48.2 (id-ad-caIssuers)
      accessLocation: 6
        uniformResourceIdentifier: http://GEANT.crt.sectigo.com/GEANTOVRSACA4.crt
    AccessDescription
      accessMethod: 1.3.6.1.5.5.7.48.1 (id-ad-ocsp)
      accessLocation: 6
        uniformResourceIdentifier: http://GEANT.ocsp.sectigo.com
Extension (id-ce-subjectAltName)
  Extension Id: 2.5.29.17 (id-ce-subjectAltName)
  GeneralNames: 2 items
    GeneralName: dNSName (2)
      dNSName: www.ionio.gr
    GeneralName: dNSName (2)
      dNSName: ionio.gr

```

Εικόνα 42 Η τελευταία επέκταση δείχνει για ποια domain επαληθεύει το πιστοποιητικό. Προσοχή ότι υπάρχουν δύο ονόματα. Καθώς το www μπορεί να παραληφθεί όταν ψάχνουμε μια domain, καταγράφονται και οι δύο τρόποι που μπορεί να εμφανιστεί

Η χρονοσφραγίδα έρχεται μαζί με το πιστοποιητικό και επιβεβαιώνει τη μέρα και χρονολογία που υπογράφηκε και το log αρχείο στο οποίο καταγράφηκε η διαδικασία αυτή. Το συγκεκριμένο φέρει τρεις χρονοσφραγίδες, δυο από τη Google και μια από τη Cloudflare. Ο αλγόριθμος που χρησιμοποιήθηκε για την υπογραφή είναι ο ECDSA_SECP256R1_SHA256.

```

Extension (SignedCertificateTimestampList)
  Extension Id: 1.3.6.1.4.1.11129.2.4.2 (SignedCertificateTimestampList)
  Serialized SCT List Length: 359
  Signed Certificate Timestamp (Google 'Xenon2024' log)
    Serialized SCT Length: 118
    SCT Version: 0
    Log ID: 76ff883f0ab6fb9551c261ccf587ba34b4a4cddb29dc68420a9fe6674c5a3a74
    Timestamp: Apr 6, 2023 00:03:39.751000000 UTC
    Extensions length: 0
    Signature Algorithm: ecdsa_secp256r1_sha256 (0x0403)
      Signature Hash Algorithm Hash: SHA256 (4)
      Signature Hash Algorithm Signature: ECDSA (3)
    Signature Length: 71
    Signature: 304502205fblaf5e664ff21ea80a773fb2e24ba2596f23b97d17b66ed258199c8d30539c...
  Signed Certificate Timestamp (Cloudflare 'Nimbus2024' Log)
    Serialized SCT Length: 118
    SCT Version: 0
    Log ID: dab6bf6b3fb5b6229f9bc2bb5c6be87091716cbb51848534bda43d3048d7fbab
    Timestamp: Apr 6, 2023 00:03:39.834000000 UTC
    Extensions length: 0
    Signature Algorithm: ecdsa_secp256r1_sha256 (0x0403)
      Signature Hash Algorithm Hash: SHA256 (4)
      Signature Hash Algorithm Signature: ECDSA (3)
    Signature Length: 71
    Signature: 3045022012cc6589ae8b44c7368279043b0f158c61bcl633b6afbdd7albbfb603450b526...
  Signed Certificate Timestamp (Google 'Argon2024' log)
    Serialized SCT Length: 117
    SCT Version: 0
    Log ID: eecdd064d5dblacec55cb79db4cd13a23287467cbcecdec351485946711fb59b
    Timestamp: Apr 6, 2023 00:03:39.792000000 UTC
    Extensions length: 0
    Signature Algorithm: ecdsa_secp256r1_sha256 (0x0403)
      Signature Hash Algorithm Hash: SHA256 (4)
      Signature Hash Algorithm Signature: ECDSA (3)
    Signature Length: 70
    Signature: 304402204bcb5a8d21fd352b723e6b0dald9c01baaf2e301aad0f4f380f6ed2164ae41fc...
algorithmIdentifier (sha384WithRSAEncryption)
  Algorithm Id: 1.2.840.113549.1.1.12 (sha384WithRSAEncryption)
  Padding: 0
  encrypted: 50d17a3a8e60decbb582a16b7f34ce30d3dale745a702a2c710ce8573d81cba395b9c7fa7...

```

Εικόνα 43 Ο server απάντησε στη πρώτη επιλογή του client (εικόνα 14) να του στείλει το πιστοποιητικό με τον αλγόριθμο υπογραφής ECDSA_SECP256R1_SHA256. Το πεδίο encrypted εμφανίζει το κρυπτογραφημένο πιστοποιητικό.

Ακολουθούν τα υπόλοιπα πιστοποιητικά της αλυσίδας.

Πρώτα πιστοποιείται η CA του ionio.gr, η GEANT OV RSA CA 4. Η CA που την πιστοποιεί είναι η USERTrust RSA Certification Authority που ανήκει στον οργανισμό The USERTRUST Network και στην εικόνα 34 φαίνονται πληροφορίες για το πιστοποιητικό της.

```

Certificate Length: 1769
Certificate: 308206e5308204cda03020102021100da43bd139bd258bb4dd61cacc4f3dbe0300d0609... (id-at-commonName=GEANT OV RSA CA 4,id-at-organizationName=GEANT
Vereniging,id-at-countryName=NL)
signedCertificate
  version: v3 (2)
  serialNumber: 0x00da43bd139bd258bb4dd61cacc4f3dbe0
  signature (sha384WithRSAEncryption)
    Algorithm Id: 1.2.840.113549.1.1.12 (sha384WithRSAEncryption)
  issuer: rdnSequence (0)
  rdnSequence: 5 items (id-at-commonName=USERTrust RSA Certification Authority,id-at-organizationName=The USERTRUST Network,id-at-localityName=Jersey
City,id-at-stateOrProvinceName=New Jersey,id-at-countryName=US)
    RDNSequence item: 1 item (id-at-countryName=US)
      RelativeDistinguishedName item (id-at-countryName=US)
        Object Id: 2.5.4.6 (id-at-countryName)
        CountryName: US
    RDNSequence item: 1 item (id-at-stateOrProvinceName=New Jersey)
      RelativeDistinguishedName item (id-at-stateOrProvinceName=New Jersey)
        Object Id: 2.5.4.8 (id-at-stateOrProvinceName)
        DirectoryString: printableString (1)
          printableString: New Jersey
    RDNSequence item: 1 item (id-at-localityName=Jersey City)
      RelativeDistinguishedName item (id-at-localityName=Jersey City)
        Object Id: 2.5.4.7 (id-at-localityName)
        DirectoryString: printableString (1)
          printableString: Jersey City
    RDNSequence item: 1 item (id-at-organizationName=The USERTRUST Network)
      RelativeDistinguishedName item (id-at-organizationName=The USERTRUST Network)
        Object Id: 2.5.4.10 (id-at-organizationName)
        DirectoryString: printableString (1)
          printableString: The USERTRUST Network
    RDNSequence item: 1 item (id-at-commonName=USERTrust RSA Certification Authority)
      RelativeDistinguishedName item (id-at-commonName=USERTrust RSA Certification Authority)
        Object Id: 2.5.4.3 (id-at-commonName)
        DirectoryString: printableString (1)
          printableString: USERTrust RSA Certification Authority

```

Εικόνα 44 Η USERTrust έχει την έδρα της στις Ηνωμένες Πολιτείες Αμερικής, στο New Jersey.

Ίδια διαδικασία με τη πιστοποίηση του ionio.gr, η CA πιστοποιεί την GEANT OV RSA CA 4.

```

validity
  notBefore: utcTime (0)
    utcTime: 2020-02-18 00:00:00 (UTC)
  notAfter: utcTime (0)
    utcTime: 2033-05-01 23:59:59 (UTC)
subject: rdnSequence (0)
  rdnSequence: 3 items (id-at-commonName=GEANT OV RSA CA 4,id-at-organizationName=GEANT Vereniging,id-at-countryName=NL)
    RDNSequence item: 1 item (id-at-countryName=NL)
      RelativeDistinguishedName item (id-at-countryName=NL)
        Object Id: 2.5.4.6 (id-at-countryName)
        CountryName: NL
    RDNSequence item: 1 item (id-at-organizationName=GEANT Vereniging)
      RelativeDistinguishedName item (id-at-organizationName=GEANT Vereniging)
        Object Id: 2.5.4.10 (id-at-organizationName)
        DirectoryString: printableString (1)
          printableString: GEANT Vereniging
    RDNSequence item: 1 item (id-at-commonName=GEANT OV RSA CA 4)
      RelativeDistinguishedName item (id-at-commonName=GEANT OV RSA CA 4)
        Object Id: 2.5.4.3 (id-at-commonName)
        DirectoryString: printableString (1)
          printableString: GEANT OV RSA CA 4

```

Εικόνα 45 Σημείωση ότι το πιστοποιητικό της GEANT έχει ισχύ από το 2020 έως το 2033, σε σχέση με τη τελική domain ionio.gr που πρέπει να ανανεώνεται κάθε χρόνο.

Το δημόσιο κλειδί της εμφανίζεται στο αντίστοιχο πεδίο subjectPublicKey και ο αλγόριθμος που χρησιμοποιήθηκε (RSA).

```

subjectPublicKeyInfo
  algorithm (rsaEncryption)
    Algorithm Id: 1.2.840.113549.1.1.1 (rsaEncryption)
  subjectPublicKey: 3082020a0282020100a58862d5a1223ec83d64a44c030f50afc32d86cafa476d1549f15e...
    modulus: 0x00a58862d5a1223ec83d64a44c030f50afc32d86cafa476d1549f15e87b4e0c2d2d08ba4...
    publicExponent: 65537

```

Εικόνα 46 Δημόσιο κλειδί της GEANT

```

extensions: 8 items
  Extension (id-ce-authorityKeyIdentifier)
    Extension Id: 2.5.29.35 (id-ce-authorityKeyIdentifier)
    AuthorityKeyIdentifier
      keyIdentifier: 5379bf5aaa2b4acf5480e1d89bc09df2b20366cb
  Extension (id-ce-subjectKeyIdentifier)
    Extension Id: 2.5.29.14 (id-ce-subjectKeyIdentifier)
    SubjectKeyIdentifier: 6f1d3549106c32fa59a09ebc8ae81f95be717a0c
  Extension (id-ce-keyUsage)
    Extension Id: 2.5.29.15 (id-ce-keyUsage)
    critical: True
    Padding: 1
    KeyUsage: 86
      1... .... = digitalSignature: True
      .0.. .... = contentCommitment: False
      ..0. .... = keyEncipherment: False
      ...0 .... = dataEncipherment: False
      .... 0... = keyAgreement: False
      .... .1.. = keyCertSign: True
      .... ..1. = cRLSign: True
      .... ...0 = encipherOnly: False
      0... .... = decipherOnly: False
  Extension (id-ce-basicConstraints)
    Extension Id: 2.5.29.19 (id-ce-basicConstraints)
    critical: True
    BasicConstraintsSyntax
      cA: True
      pathLenConstraint: 0
  Extension (id-ce-extKeyUsage)
    Extension Id: 2.5.29.37 (id-ce-extKeyUsage)
    KeyPurposeIDs: 2 items
      KeyPurposeId: 1.3.6.1.5.5.7.3.1 (id-kp-serverAuth)
      KeyPurposeId: 1.3.6.1.5.5.7.3.2 (id-kp-clientAuth)
  Extension (id-ce-certificatePolicies)
    Extension Id: 2.5.29.32 (id-ce-certificatePolicies)
    CertificatePoliciesSyntax: 1 item
      PolicyInformation
        policyIdentifier: 2.5.29.32.0 (anyPolicy)
        policyQualifiers: 1 item
          PolicyQualifierInfo
            Id: 1.3.6.1.5.5.7.2.1 (id-qt-cps)
            DirectoryString: https://sectigo.com/CPS

```

Εικόνα 47 Πληροφορίες για το πιστοποιητικό της GEANT

```

  Extension (id-ce-cRLDistributionPoints)
    Extension Id: 2.5.29.31 (id-ce-cRLDistributionPoints)
    CRLDistPointsSyntax: 1 item
      DistributionPoint
        distributionPoint: fullName (0)
          fullName: 1 item
            GeneralName: uniformResourceIdentifier (6)
              uniformResourceIdentifier: http://crl.usertrust.com/USERTrustRSACertificationAuthority.crl
  Extension (id-pe-authorityInfoAccess)
    Extension Id: 1.3.6.1.5.5.7.1.1 (id-pe-authorityInfoAccess)
    AuthorityInfoAccessSyntax: 2 items
      AccessDescription
        accessMethod: 1.3.6.1.5.5.7.48.2 (id-ad-caIssuers)
        accessLocation: 6
          uniformResourceIdentifier: http://crt.usertrust.com/USERTrustRSAAddTrustCA.crt
      AccessDescription
        accessMethod: 1.3.6.1.5.5.7.48.1 (id-ad-ocsp)
        accessLocation: 6
          uniformResourceIdentifier: http://ocsp.usertrust.com
  algorithmIdentifier (sha384WithRSAEncryption)
    Algorithm Id: 1.2.840.113549.1.1.12 (sha384WithRSAEncryption)
    Padding: 0
    encrypted: 52d942dded318ffd4131f3e17508545de2e36f4c69f14136f24cd6139c43cc629b7c6827...

```

Εικόνα 48 Πληροφορίες για το πιστοποιητικό της GEANT

Αυτό ήταν το πιστοποιητικό της USERTrust για την GEANT, το οποίο αποστέλλεται στον client για να πιστοποιήσει την CA που εγγυάται για την GEANT. Το τρίτο στην αλυσίδα είναι για την USERTrust από την AAA Certificate Services.

```
Certificate Length: 1413
Certificate: 3082058130820469a00302010202103972443af922b751d7d36c10dd313595300d06092a... (id-at-commonName=USERTrust RSA Certification
Authority,id-at-organizationName=The USERTRUST Network,id-at-localityName=Jersey City,id-at-stateOrProvi
signedCertificate
  version: v3 (2)
  serialNumber: 0x3972443af922b751d7d36c10dd313595
  signature (sha384WithRSAEncryption)
    Algorithm Id: 1.2.840.113549.1.1.12 (sha384WithRSAEncryption)
  issuer: rdnSequence (0)
    rdnSequence: 5 items (id-at-commonName=AAA Certificate Services,id-at-organizationName=Comodo CA
    Limited,id-at-localityName=Salford,id-at-stateOrProvinceName=Greater Manchester,id-at-countryName=GB)
      RDNSequence item: 1 item (id-at-countryName=GB)
        RelativeDistinguishedName item (id-at-countryName=GB)
          Object Id: 2.5.4.6 (id-at-countryName)
          CountryName: GB
      RDNSequence item: 1 item (id-at-stateOrProvinceName=Greater Manchester)
        RelativeDistinguishedName item (id-at-stateOrProvinceName=Greater Manchester)
          Object Id: 2.5.4.8 (id-at-stateOrProvinceName)
          DirectoryString: UTF8String (4)
            UTF8String: Greater Manchester
      RDNSequence item: 1 item (id-at-localityName=Salford)
        RelativeDistinguishedName item (id-at-localityName=Salford)
          Object Id: 2.5.4.7 (id-at-localityName)
          DirectoryString: UTF8String (4)
            UTF8String: Salford
      RDNSequence item: 1 item (id-at-organizationName=Comodo CA Limited)
        RelativeDistinguishedName item (id-at-organizationName=Comodo CA Limited)
          Object Id: 2.5.4.10 (id-at-organizationName)
          DirectoryString: UTF8String (4)
            UTF8String: Comodo CA Limited
      RDNSequence item: 1 item (id-at-commonName=AAA Certificate Services)
        RelativeDistinguishedName item (id-at-commonName=AAA Certificate Services)
          Object Id: 2.5.4.3 (id-at-commonName)
          DirectoryString: UTF8String (4)
            UTF8String: AAA Certificate Services
```

Εικόνα 49 Η AAA Certificate Services ανήκει στον οργανισμό Comodo CA Limited, με έδρα την επαρχία του Salford, η οποία βρίσκεται στη πόλη της Greater Manchester της Μεγάλης Βρετανίας.

Το πιστοποιητικό είναι για την USERTrust RSA Certification Authority και έχει ισχύ από το 2019 ως το 2028.

```
validity
  notBefore: utcTime (0)
    utcTime: 2019-03-12 00:00:00 (UTC)
  notAfter: utcTime (0)
    utcTime: 2028-12-31 23:59:59 (UTC)
subject: rdnSequence (0)
  rdnSequence: 5 items (id-at-commonName=USERTrust RSA Certification Authority,id-at-organizationName=The USERTRUST
  Network,id-at-localityName=Jersey City,id-at-stateOrProvinceName=New Jersey,id-at-countryName=US)
    RDNSequence item: 1 item (id-at-countryName=US)
      RelativeDistinguishedName item (id-at-countryName=US)
        Object Id: 2.5.4.6 (id-at-countryName)
        CountryName: US
    RDNSequence item: 1 item (id-at-stateOrProvinceName=New Jersey)
      RelativeDistinguishedName item (id-at-stateOrProvinceName=New Jersey)
        Object Id: 2.5.4.8 (id-at-stateOrProvinceName)
        DirectoryString: printableString (1)
          printableString: New Jersey
    RDNSequence item: 1 item (id-at-localityName=Jersey City)
      RelativeDistinguishedName item (id-at-localityName=Jersey City)
        Object Id: 2.5.4.7 (id-at-localityName)
        DirectoryString: printableString (1)
          printableString: Jersey City
    RDNSequence item: 1 item (id-at-organizationName=The USERTRUST Network)
      RelativeDistinguishedName item (id-at-organizationName=The USERTRUST Network)
        Object Id: 2.5.4.10 (id-at-organizationName)
        DirectoryString: printableString (1)
          printableString: The USERTRUST Network
    RDNSequence item: 1 item (id-at-commonName=USERTrust RSA Certification Authority)
      RelativeDistinguishedName item (id-at-commonName=USERTrust RSA Certification Authority)
        Object Id: 2.5.4.3 (id-at-commonName)
        DirectoryString: printableString (1)
          printableString: USERTrust RSA Certification Authority
```

Εικόνα 50 Πιστοποιητικό της USERTrust

```

subjectPublicKeyInfo
  algorithm (rsaEncryption)
    Algorithm Id: 1.2.840.113549.1.1.1 (rsaEncryption)
  subjectPublicKey: 3082020a028202010080126517360ec3db08b3d0ac570d76edcd27d34cad508361e2aa20...
    modulus: 0x0080126517360ec3db08b3d0ac570d76edcd27d34cad508361e2aa204d092d6409dcce89...
    publicExponent: 65537

```

Εικόνα 51 Δημόσιο κλειδί του πιστοποιητικού

```

extensions: 7 items
  Extension (id-ce-authorityKeyIdentifier)
    Extension Id: 2.5.29.35 (id-ce-authorityKeyIdentifier)
    AuthorityKeyIdentifier
      keyIdentifier: a0110a233e96f107ece2af29ef82a57fd030a4b4
  Extension (id-ce-subjectKeyIdentifier)
    Extension Id: 2.5.29.14 (id-ce-subjectKeyIdentifier)
    SubjectKeyIdentifier: 5379bf5aaa2b4acf5480eld89bc09df2b20366cb
  Extension (id-ce-keyUsage)
    Extension Id: 2.5.29.15 (id-ce-keyUsage)
    critical: True
    Padding: 1
    KeyUsage: 86
    1... .... = digitalSignature: True
    .0... .... = contentCommitment: False
    ..0... .... = keyEncipherment: False
    ...0... .... = dataEncipherment: False
    .... 0... = keyAgreement: False
    .... 1... = keyCertSign: True
    .... 1... = cRLSign: True
    .... ...0 = encipherOnly: False
    0... .... = decipherOnly: False
  Extension (id-ce-basicConstraints)
    Extension Id: 2.5.29.19 (id-ce-basicConstraints)
    critical: True
    BasicConstraintsSyntax
      cA: True
  Extension (id-ce-certificatePolicies)
    Extension Id: 2.5.29.32 (id-ce-certificatePolicies)
    CertificatePoliciesSyntax: 1 item
      PolicyInformation
        policyIdentifier: 2.5.29.32.0 (anyPolicy)
  Extension (id-ce-cRLDistributionPoints)
    Extension Id: 2.5.29.31 (id-ce-cRLDistributionPoints)
    CRLDistPointsSyntax: 1 item
      DistributionPoint
        distributionPoint: fullName (0)
        fullName: 1 item
          GeneralName: uniformResourceIdentifier (6)
            uniformResourceIdentifier: http://crl.comodoca.com/AAACertificateServices.crl
  Extension (id-pe-authorityInfoAccess)
    Extension Id: 1.3.6.1.5.5.7.1.1 (id-pe-authorityInfoAccess)
    AuthorityInfoAccessSyntax: 1 item
      AccessDescription
        accessMethod: 1.3.6.1.5.5.7.48.1 (id-ad-ocsp)
        accessLocation: 6
          uniformResourceIdentifier: http://ocsp.comodoca.com
  algorithmIdentifier (sha384WithRSAEncryption)
    Algorithm Id: 1.2.840.113549.1.1.12 (sha384WithRSAEncryption)
  Padding: 0
  encrypted: 188751dc74213d9c8ae027b733d02eccecf0e6cb5e11de226f9b758e9e72fee4d6feaa1f...

```

Εικόνα 52 Πληροφορίες για το πιστοποιητικό της USERTrust

Επόμενο στη λίστα είναι το πιστοποιητικό της AAA Certificate Services και έχει ισχύ από το 2004 ως το 2028.

```

Certificate Length: 1078
Certificate: 308204323092031aa003020102020101300406092a864896f70d0101050500307b310b30... (id-at-commonName=AAA Certificate Services,id-at-organizationName=Comodo CA Limited,id-at-localityName=Salford,id-at-stateOrProvinceName=Greater Manch
signedCertificate
  version: v3 (2)
  serialNumber: 0x01
  signature (sha1WithRSAEncryption)
  Algorithm Id: 1.2.840.113549.1.1.5 (sha1WithRSAEncryption)
  issuer: rdnSequence (0)
    rdnSequence: 5 items (id-at-commonName=AAA Certificate Services,id-at-organizationName=Comodo CA Limited,id-at-localityName=Salford,id-at-stateOrProvinceName=Greater Manchester,id-at-countryName=GB)
      RDNSequence item: 1 item (id-at-countryName=GB)
        RelativeDistinguishedName item (id-at-countryName=GB)
          Object Id: 2.5.4.6 (id-at-countryName)
          CountryName: GB
      RDNSequence item: 1 item (id-at-stateOrProvinceName=Greater Manchester)
        RelativeDistinguishedName item (id-at-stateOrProvinceName=Greater Manchester)
          Object Id: 2.5.4.8 (id-at-stateOrProvinceName)
          DirectoryString: UTF8String (4)
          UTF8String: Greater Manchester
      RDNSequence item: 1 item (id-at-localityName=Salford)
        RelativeDistinguishedName item (id-at-localityName=Salford)
          Object Id: 2.5.4.7 (id-at-localityName)
          DirectoryString: UTF8String (4)
          UTF8String: Salford
      RDNSequence item: 1 item (id-at-organizationName=Comodo CA Limited)
        RelativeDistinguishedName item (id-at-organizationName=Comodo CA Limited)
          Object Id: 2.5.4.10 (id-at-organizationName)
          DirectoryString: UTF8String (4)
          UTF8String: Comodo CA Limited
      RDNSequence item: 1 item (id-at-commonName=AAA Certificate Services)
        RelativeDistinguishedName item (id-at-commonName=AAA Certificate Services)
          Object Id: 2.5.4.3 (id-at-commonName)
          DirectoryString: UTF8String (4)
          UTF8String: AAA Certificate Services
  validity
    notBefore: utcTime (0)
    utcTime: 2004-01-01 00:00:00 (UTC)
    notAfter: utcTime (0)
    utcTime: 2028-12-31 23:59:59 (UTC)

```

Εικόνα 53 Πιστοποιητικό της AAA Certificate Services και ισχύ

Η συγκεκριμένη CA πιστοποιεί τον εαυτό της (δηλαδή χρησιμοποιεί ένα self-signed certificate) και για αυτό το πιστοποιητικό της είναι ακριβώς το ίδιο που χρησιμοποιεί για την εγκυρότητα της ταυτότητας της USERTrust RSA Certificate Authority.

```

subject: rdnSequence (0)
  rdnSequence: 5 items (id-at-commonName=AAA Certificate Services,id-at-organizationName=Comodo CA Limited,id-at-localityName=Salford,id-at-stateOrProvinceName=Greater Manchester,id-at-countryName=GB)
    RDNSequence item: 1 item (id-at-countryName=GB)
      RelativeDistinguishedName item (id-at-countryName=GB)
        Object Id: 2.5.4.6 (id-at-countryName)
        CountryName: GB
    RDNSequence item: 1 item (id-at-stateOrProvinceName=Greater Manchester)
      RelativeDistinguishedName item (id-at-stateOrProvinceName=Greater Manchester)
        Object Id: 2.5.4.8 (id-at-stateOrProvinceName)
        DirectoryString: UTF8String (4)
        UTF8String: Greater Manchester
    RDNSequence item: 1 item (id-at-localityName=Salford)
      RelativeDistinguishedName item (id-at-localityName=Salford)
        Object Id: 2.5.4.7 (id-at-localityName)
        DirectoryString: UTF8String (4)
        UTF8String: Salford
    RDNSequence item: 1 item (id-at-organizationName=Comodo CA Limited)
      RelativeDistinguishedName item (id-at-organizationName=Comodo CA Limited)
        Object Id: 2.5.4.10 (id-at-organizationName)
        DirectoryString: UTF8String (4)
        UTF8String: Comodo CA Limited
    RDNSequence item: 1 item (id-at-commonName=AAA Certificate Services)
      RelativeDistinguishedName item (id-at-commonName=AAA Certificate Services)
        Object Id: 2.5.4.3 (id-at-commonName)
        DirectoryString: UTF8String (4)
        UTF8String: AAA Certificate Services

```

Εικόνα 54 Όταν φτάσουμε στην πρώτη CA στην αλυσίδα, αυτή πάντα πιστοποιεί τον εαυτό της με self-signed certificate

```

subjectPublicKeyInfo
  algorithm (rsaEncryption)
    Algorithm Id: 1.2.840.113549.1.1.1 (rsaEncryption)
  subjectPublicKey: 3082010a0282010100be409df46eelea76871c4d45448ebe46c883069dc12afe181f8ee4...
  modulus: 0x00be409df46eelea76871c4d45448ebe46c883069dc12afe181f8ee402faf3ab5d508a16...
  publicExponent: 65537

```

Εικόνα 55 Δημόσιο κλειδί της CA

Δεν υπάρχει ψηφιακή υπογραφή για αυτό το πιστοποιητικό καθώς πιστοποιείται από το ίδιο. Για αυτό στο πεδίο `digitalSignature`, παίρνει `False` (τιμή = 0).

```
extensions: 4 items
  Extension (id-ce-subjectKeyIdentifier)
    Extension Id: 2.5.29.14 (id-ce-subjectKeyIdentifier)
    SubjectKeyIdentifier: a0110a233e96f107ece2af29ef82a57fd030a4b4
  Extension (id-ce-keyUsage)
    Extension Id: 2.5.29.15 (id-ce-keyUsage)
    critical: True
    Padding: 1
    KeyUsage: 06
    0... .. = digitalSignature: False
    ..0... .. = contentCommitment: False
    ...0... .. = keyEncipherment: False
    ....0... .. = dataEncipherment: False
    ....0... .. = keyAgreement: False
    ....1... .. = keyCertSign: True
    ....1... .. = cRLSign: True
    ....0... .. = encipherOnly: False
    0... .. = decipherOnly: False
  Extension (id-ce-basicConstraints)
    Extension Id: 2.5.29.19 (id-ce-basicConstraints)
    critical: True
    BasicConstraintsSyntax
      CA: True
  Extension (id-ce-cRLDistributionPoints)
    Extension Id: 2.5.29.31 (id-ce-cRLDistributionPoints)
    CRLDistPointsSyntax: 2 items
      DistributionPoint
        distributionPoint: fullName (0)
        fullName: 1 item
          GeneralName: uniformResourceIdentifier (6)
            uniformResourceIdentifier: http://crl.comodoca.com/AAACertificateServices.crl
      DistributionPoint
        distributionPoint: fullName (0)
        fullName: 1 item
          GeneralName: uniformResourceIdentifier (6)
            uniformResourceIdentifier: http://crl.comodo.net/AAACertificateServices.crl
algorithmIdentifier (sha1WithRSAEncryption)
  Algorithm Id: 1.2.840.113549.1.1.5 (sha1WithRSAEncryption)
  Padding: 0
  encrypted: 0856fc02f09be8ffa4fad67bc64480ce4fc4c5f60058cca6b6bc1449680476e8e6ee5dec...
```

Εικόνα 56 Λεπτομέρειες του *self-signed certificate* για την *AAA Certificate Services*

Τα *self-signed certificates* δεν τα δέχεται ο client παρά μόνο αν τα εμπιστεύεται ήδη. Για να μπορέσει να δεχτεί ο client αυτό το πιστοποιητικό, έχει ήδη αποθηκεύσει τα πιστοποιητικά μεγάλων CA οι οποίοι πιστοποιούν τον εαυτό τους και άλλες CA. Αυτά ορίζονται ως «έμπιστα» πριν καν γίνει η σύνδεση με τη domain, καθώς τα έχει ήδη «κατεβάσει» (download) και αποθηκεύσει μόνιμα. Πριν λήξει το κάθε πιστοποιητικό των CA, γίνεται ανανέωση τους από την αντίστοιχη CA όταν γίνει μια σύνδεση μαζί της.

B'5.2.2. Certificate Verify, Finished

Αμέσως μετά αυτό το TLS layer υπάρχει άλλο ένα TLS layer με δυο record layers. Το Certificate Verify (πληροφορίες στον client που θα χρειαστεί για να επιβεβαιώσει το πιστοποιητικό του *ionio.gr*) και το μήνυμα Finished, υποδεικνύοντας το τέλος της

χειραψίας από τη μεριά του server.

```
Transport Layer Security
  TLSv1.3 Record Layer: Handshake Protocol: Certificate Verify
    Opaque Type: Application Data (23)
    Version: TLS 1.2 (0x0303)
    Length: 281
    [Content Type: Handshake (22)]
    Handshake Protocol: Certificate Verify
      Handshake Type: Certificate Verify (15)
      Length: 260
      Signature Algorithm: rsa_pss_rsae_sha256 (0x0804)
        Signature Hash Algorithm Hash: Unknown (8)
        Signature Hash Algorithm Signature: SM2 (4)
      Signature length: 256
      Signature: 83fcbcef6842d340a6cdb2c621b425683265e3ce6363f6d8c72f059b3ac306d0d9fcfd50...
```

Εικόνα 57 Στο *Certificate Verify* μπαίνει ο αλγόριθμος υπογραφής του, το μέγεθος της υπογραφής και η υπογραφή του. Από αυτό το πεδίο ο *client* χρειάζεται το όνομα του αλγορίθμου και την υπογραφή για να επαληθεύσει τη ταυτότητα του *ionio.gr*

Στο μήνυμα *finished* περιέχεται η χειραψία μέχρι στιγμής σε hash μορφή και είναι τύπου *Application Data*.

```
TLSv1.3 Record Layer: Handshake Protocol: Finished
  Opaque Type: Application Data (23)
  Version: TLS 1.2 (0x0303)
  Length: 53
  [Content Type: Handshake (22)]
  Handshake Protocol: Finished
    Handshake Type: Finished (20)
    Length: 32
    Verify Data
```

Εικόνα 58 Το *finished* μήνυμα είναι πάντα κρυπτογραφημένο και εμφανίζεται ως «*Encrypted Application Data*»

Με αυτό το μήνυμα ολοκληρώνεται ένα ολόκληρο RTT. Εάν το θελήσει ο server, μπορεί να στείλει μετά το *Finished* επιπλέον κρυπτογραφημένα *Application Data*. Εξαιτίας του μηχανισμού του TLS 1.3 να μπορεί να κρυπτογραφεί ο server στα μισά του *Server Hello*, στέλνει το πιστοποιητικό μαζί και το RTT μειώνεται σε 1 (από 2-RTT που ήταν το TLS 1.2).

B'5.3. Τρίτο βήμα της χειραψίας, Client Finished

Ο Client επαληθεύει το πιστοποιητικό του *ionio.gr* και την υπογραφή του. Αυτό γίνεται τοπικά στη συσκευή του client και όχι στο δίκτυο. Παίρνει το κρυπτογραφημένο (με το ιδιωτικό κλειδί του *ionio.gr*) πιστοποιητικό και χρησιμοποιεί το δημόσιο κλειδί του που του έστειλε το *ionio.gr* μαζί με το πακέτο του *Certificate Chain*. Διαβάζει τον αλγόριθμο που χρησιμοποιήθηκε για την υπογραφή του πιστοποιητικού και χρησιμοποιεί τον ίδιο πάνω στο πιστοποιητικό. Το

αποτέλεσμα μιας συνάρτησης κατακερματισμού είναι πάντα το ίδιο και ελέγχει εάν η υπογραφή είναι ίδια με το αποτέλεσμα της συνάρτησης. Έπειτα ελέγχει τα πεδία του ονόματος domain για να δει αν είναι αυτός που θέλει να συζητήσει και τότε λήγει για να δει αν ισχύει η ημερομηνία. Αν και όλα τα υπόλοιπα πεδία μπορεί να είναι σωστά, το πιστοποιητικό μπορεί να έληξε πριν χρόνια ή ακόμη και χθες. Σε αυτή τη περίπτωση διακόπτει τη σύνδεση με ένα μήνυμα ειδοποίησης προς τον server.

```
Transport Layer Security
  TLSv1.3 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
    Content Type: Change Cipher Spec (20)
    Version: TLS 1.2 (0x0303)
    Length: 1
    Change Cipher Spec Message
  TLSv1.3 Record Layer: Handshake Protocol: Finished
    Opaque Type: Application Data (23)
    Version: TLS 1.2 (0x0303)
    Length: 53
    [Content Type: Handshake (22)]
    Handshake Protocol: Finished
      Handshake Type: Finished (20)
      Length: 32
      Verify Data
```

Εικόνα 59 Ο Client στέλνει το μήνυμα Change Cipher Spec και Finished ως τα τελευταία μηνύματα για το Client Hello

Καθώς στο παράδειγμα το πιστοποιητικό είναι το σωστό, προχωράει στην αποστολή ενός Change Cipher Spec μηνύματος για να μεταμφιέσει το TLS 1.3 ως TLS 1.2 και να στείλει το δικό του Finished μήνυμα. Στο μήνυμα Finished περιέχεται ολόκληρη η χειραψία σε hash μορφή και είναι τύπου Application Data. Εφόσον αυτά τα δυο είναι πολύ μικρά σε μέγεθος και ο client μπορεί να στείλει απευθείας με αυτό το πακέτο και Encrypted Application Data, δεν υπολογίζονται στο χρόνο RTT. Με αυτό, ολοκληρώνεται η χειραψία και ξεκινάει η ασφαλής συζήτηση μεταξύ client και server.

Υπάρχει περίπτωση ο Client μέσα σε μια σύνοδο να στείλει ξανά στον server ένα client hello για ενημέρωση των κλειδιών τους. Η διαδικασία είναι ακριβώς η ίδια, με τη χειραψία να πραγματοποιείται κανονικά. Αν και το πιστοποιητικό δεν χρειάζεται να στέλνεται κάθε φορά, στο παράδειγμα ο server στέλνει ολόκληρη την αλυσίδα και ο client το επαληθεύει με τον ίδιο τρόπο όπως τη πρώτη χειραψία.

B'5.4. Encrypted Application Data

Το επόμενο πακέτο φαίνεται να στέλνει ο Client και να ζητάει να εμφανιστεί η σελίδα HTTP/1.1 του ionio.gr. Περιγράφει το σύστημα από το οποίο εισέρχεται (User Agent)

και τι επιλογές υποστηρίζει (πεδία Accept).

```
Transport Layer Security
  TLSv1.3 Record Layer: Application Data Protocol: Hypertext Transfer Protocol
    Opaque Type: Application Data (23)
    Version: TLS 1.2 (0x0303)
    Length: 455
    [Content Type: Application Data (23)]
    Encrypted Application Data: 4487163ea98b6cfb46a46e5f37dd2d1461ea2b72e395a0cb88e6b2aa782a2c00fc07d434...
    [Application Data Protocol: Hypertext Transfer Protocol]
Hypertext Transfer Protocol
  GET / HTTP/1.1\r\n
    [Expert Info (Chat/Sequence): GET / HTTP/1.1\r\n]
    [GET / HTTP/1.1\r\n]
    [Severity level: Chat]
    [Group: Sequence]
  Request Method: GET
  Request URI: /
  Request Version: HTTP/1.1
Host: ionio.gr\r\n
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/117.0\r\n
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8\r\n
Accept-Language: en-US,en;q=0.5\r\n
Accept-Encoding: gzip, deflate, br\r\n
Connection: keep-alive\r\n
Upgrade-Insecure-Requests: 1\r\n
Sec-Fetch-Dest: document\r\n
Sec-Fetch-Mode: navigate\r\n
Sec-Fetch-Site: none\r\n
Sec-Fetch-User: ?1\r\n
\r\n
[Full request URI: https://ionio.gr/]
[HTTP request 1/1]
```

Εικόνα 60 Ο σύνδεσμος που ζητάει είναι <https://ionio.gr/>. Στο HTTP μήνυμα φαίνονται τα χαρακτηριστικά της εφαρμογής από την οποία συνδέεται ο client και περαιτέρω. Mozilla Firefox απο Ubuntu Linux.

Ο server διαβιβάζει δύο νέα session tickets που παρήγαγε για τον Client και μπορεί να τα χρησιμοποιήσει στο μέλλον για να εγκαθιδρύσει συνόδους 0-RTT και να στείλει Early Data. Το session ticket απαρτίζεται από την έκδοση TLS για την οποία ισχύει, τη διάρκειά ισχύς του, ένα nonce (τυχαίο νούμερο χωρίς νόημα για την αυθεντικοποίηση ενός μηνύματος) και τη γραμματοσειρά του. Η γραμματοσειρά είναι πληροφορία που εκμεταλλεύεται ο server για να δει αν μπορεί να συνεχίσει το session. Όταν ο client το χρησιμοποιήσει και το στείλει στο server, αυτός με τη σειρά του προσθέτει το Age add πεδίο σε milliseconds στο timestamp του πακέτου. Αυτό γίνεται για να μην μπορεί ένας επιτιθέμενος να καταλάβει πότε δημιουργήθηκε το

session ticket.

```
Transport Layer Security
  TLSv1.3 Record Layer: Handshake Protocol: New Session Ticket
    Opaque Type: Application Data (23)
    Version: TLS 1.2 (0x0303)
    Length: 282
    [Content Type: Handshake (22)]
    Handshake Protocol: New Session Ticket
      Handshake Type: New Session Ticket (4)
      Length: 261
      TLS Session Ticket
        Session Ticket Lifetime Hint: 300 seconds (5 minutes)
        Session Ticket Age Add: 2673374910
        Session Ticket Nonce Length: 8
        Session Ticket Nonce: 0000000000000000
        Session Ticket Length: 240
        Session Ticket: 700fe031743dlcee87196950443117c7f2699e730054ddb969f4199dd2ba0093c012a6af...
        Extensions Length: 0
  TLSv1.3 Record Layer: Handshake Protocol: New Session Ticket
    Opaque Type: Application Data (23)
    Version: TLS 1.2 (0x0303)
    Length: 282
    [Content Type: Handshake (22)]
    Handshake Protocol: New Session Ticket
      Handshake Type: New Session Ticket (4)
      Length: 261
      TLS Session Ticket
        Session Ticket Lifetime Hint: 300 seconds (5 minutes)
        Session Ticket Age Add: 1705335816
        Session Ticket Nonce Length: 8
        Session Ticket Nonce: 00000000000000001
        Session Ticket Length: 240
        Session Ticket: 700fe031743dlcee87196950443117c7a9f71607a99f1bea94b47eb7f85b4baec40724ba...
        Extensions Length: 0
```

Εικόνα 61 Τα συγκεκριμένα session tickets έχουν ισχύ για 5 λεπτά.

Στο επόμενο πακέτο ο server στέλνει στον client ένα HTTP response σε αυτό που του ζήτησε. Το πιο πιθανό είναι να φορτώνει τη πρώτη σελίδα που βλέπει ο χρήστης.

```
Transport Layer Security
  TLSv1.3 Record Layer: Application Data Protocol: Hypertext Transfer Protocol
    Opaque Type: Application Data (23)
    Version: TLS 1.2 (0x0303)
    Length: 8209
    [Content Type: Application Data (23)]
    Encrypted Application Data: e1bd25d988a5ce625eedf441df7c82ec43fd85ea5574010a337328a051ae62c66dba91ba...
    [Application Data Protocol: Hypertext Transfer Protocol]
  TLS segment data (8192 bytes)
Transport Layer Security
  TLSv1.3 Record Layer: Application Data Protocol: Hypertext Transfer Protocol
    Opaque Type: Application Data (23)
    Version: TLS 1.2 (0x0303)
    Length: 337
    [Content Type: Application Data (23)]
    Encrypted Application Data: ea17d924199ad548174db44b09cac8c386e54d6f1blac48f6645534ab86927b0afffd125...
    [Application Data Protocol: Hypertext Transfer Protocol]
  TLS segment data (320 bytes)
```

Εικόνα 62 Το tshark δεν δείχνει ποιο είναι το περιεχόμενο του HTTP και το κρατάει κρυπτογραφημένο. Σύμφωνα με το πεδίο length, το πρώτο record layer περιέχει 8209 bytes από δεδομένα, το οποίο μάλλον είναι η αρχική σελίδα που επέλεξε να ανοίξει ο client

Κεφάλαιο Γ. Συμπέρασμα

Tο πρωτόκολλο του TLS 1.2 ήταν μια καλή έκδοση ασφάλειας που υλοποιείται ακόμη και σήμερα. Από την ανάγκη για να εξασφαλίσουμε ότι η διαρροή του ιδιωτικού κλειδιού του server δεν θα επηρεάσει τις μυριάδες συζητήσεις κάθε μέρα, ιδρύθηκε το TLS 1.3 το οποίο εφαρμόζεται πλέον σε μεγάλη κλίμακα στο διαδίκτυο χάρη στην PFS που προσφέρει και την ταχύτερη καθιέρωση μιας ασφαλούς σύνδεσης. Η ανάλυση του πρωτοκόλλου με κώδικα χρησίμευσε στη διχοτόμηση του κάθε πεδίου και τιμής των πακέτων το οποίο μας οδήγησε στη καλύτερη κατανόηση του πρωτοκόλλου στη πράξη. Αν και ο παραδοσιακός τρόπος να διαβάσουμε την αποκρυπτογραφημένη κίνηση παραμένει το εργαλείο Wireshark, η συγκεκριμένη μέθοδος αναλύει ένα μέρος της διαδικασίας που γίνεται στο παρασκήνιο του Wireshark για να μας εμφανίσει το τελικό αποτέλεσμα. Αν και το tshark μετέφρασε ό,τι του εισήγαγα σαν δεδομένα, παρατηρήσαμε ότι τα πακέτα μεταμορφώνονται μέσα από διάφορες μεθόδους για να μην είναι εύκολα αναγνώσιμα από αδιάκριτα μάτια. Μάθαμε για τις δυνατότητες και ευπάθειες του 0-RTT, το οποίο αναμένεται να βελτιωθεί σε επόμενη έκδοση του TLS. Καθώς η τεχνολογία προχωράει σε *κβαντικό υπολογισμό* (quantum computing), απειλείται η PFS και τα θεμέλια του TLS 1.3. Στα επόμενα χρόνια θα χρειαστεί να περάσουμε σε μια νέα έκδοση που θα αντιμετωπίζει αυτό το πρόβλημα εφόσον δεν θα έχουμε ασφάλεια με τη χρήση του ECDHE. Για την ώρα είμαστε ασφαλείς με αυτά που έχουμε.

Βιβλιογραφία

Περιεχόμενα

[a] Στέφανου Γκρίτζαλη – Σωκράτη Κ. Κάτσικα, Δημήτρη Γκρίτζαλη, εκδόσεις Παπασωτηρίου, Αθήνα Οκτώβριος 2003, κεφάλαιο 1.2.3. Το μοντέλο του Internet

[b] Στέφανου Γκρίτζαλη – Σωκράτη Κ. Κάτσικα, Δημήτρη Γκρίτζαλη, εκδόσεις Παπασωτηρίου, Αθήνα Οκτώβριος 2003, κεφάλαιο 1.3.2. Επίπεδα του μοντέλου αναφοράς OSI

[c] Charles P. Pfleeger, Shari Lawrence Pfleeger, Jonathan Margulies, Ασφάλεια Πληροφοριακών συστημάτων (μετάφραση Παύλος Γωγούσης, εκδόσεις Τζιολα), 5^η έκδοση, κεφάλαιο 6 Οι κρυπτογραφίες SSL και TLS

Ηλεκτρονικές πηγές

- [1] Unknown Author, ibm.com, TLS Protocol Overview, Μια γενική ματιά στο πρωτόκολλο TLS, <https://www.ibm.com/docs/en/sdk-java-technology/8?topic=provider-tls-protocol-overview>
- [2] Unknown Author, gdpr-info.eu, GDPR Personal Data, Ονομαστικά τα ευαίσθητα προσωπικά δεδομένα, <https://gdpr-info.eu/issues/personal-data/>
- [3] Unknown Author, cloudflare.com, What is a session key? Session keys and TLS handshakes, <https://www.cloudflare.com/el-gr/learning/ssl/what-is-a-session-key/>
- [4] Unknown author, IETF, Introduction to the IETF, Unknown Date, <https://www.ietf.org/about/introduction/>
- [5] E. Rescorla, datatracker.ietf.org, The Transport Layer Security (TLS) Protocol Version 1.3, August 2018, <https://datatracker.ietf.org/doc/html/rfc8446>
- [6] David Warburton and Sander Vinberg, F5 Labs, The 2021 TLS Telemetry Report, October 2021, <https://www.f5.com/labs/articles/threat-intelligence/the-2021-tls-telemetry-report>
- [7] T. Dierks and E. Rescorla, ietf.org, The Transport Layer Security (TLS) Protocol Version 1.2, August 2008, <https://www.ietf.org/rfc/rfc5246.txt>
- [8] Y. Nir, A. Langley, datatracker.ietf.org, ChaCha20 and Poly1305 for IETF Protocols, May 2015, <https://datatracker.ietf.org/doc/html/rfc7539>
- [9] Philippe Biondi et.al., scapy.readthedocs.io, Usage, September 2023, This document is under a Creative Commons Attribution - Non-Commercial - Share Alike 2.5 license, <https://scapy.readthedocs.io/en/latest/usage.html>
- [10] Aaron Philips, comparitech.com, How to Decrypt SSL with Wireshark – HTTPS Decryption Guide, July 2023, <https://www.comparitech.com/net-admin/decrypt-ssl-with-wireshark/>
- [11] Martin Thomson, datatracker.ietf.org, The SSLKEYLOGFILE Format for TLS, July 2023, <https://datatracker.ietf.org/doc/draft-thomson-tls-keylogfile/>

- [12] Ross Jacobs, tshark.dev, DISPLAY FILTERS, July 2019, https://tshark.dev/analyze/packet_hunting/packet_hunting/
- [13] Unknown Author, UDN Web Docs, NSS Key Log Format, Unknown date, https://udn.realityripple.com/docs/Mozilla/Projects/NSS/Key_Log_Format
- [14] Dierks and Rescorla, ietf, TLS, August 2008, page 35, <https://www.ietf.org/rfc/rfc5246.txt>
- [15] Unknown Author, ibm.com, The TLS 1.3 Protocol, March 2023, <https://www.ibm.com/docs/en/sdk-java-technology/8?topic=handshake-tls-13-protocol>
- [16] Anastasios Arampatzis, venafi.com, What are Delegated Credentials?, June 2023, <https://venafi.com/blog/what-are-delegated-credentials/>
- [17] M. Thomson, datatracker.ietf.org, Record Size Limit Extension for TLS, August 2018, <https://datatracker.ietf.org/doc/html/rfc8449>
- [18] John Althouse, engineering.salesforce.com, TLS Fingerprinting with JA3 and JA3S, January 2015, <https://engineering.salesforce.com/tls-fingerprinting-with-ja3-and-ja3s-247362855967/>
- [19] Nick Sullivan, blog.cloudflare.com, Introducing Zero Round Trip Time Resumption (0-RTT), July 2017, <https://blog.cloudflare.com/introducing-0-rtt/#whatsthecatch>
- [20] Gerald Combs et.al., linux.die.net, tshark(1) - Linux man page, Unknown Date, <https://linux.die.net/man/1/tshark>
- Michael Driscoll, tls13.xargs.org, The Illustrated TLS 1.3 Connection, April 2022, <https://tls13.xargs.org/>

Συντμήσεις

- AES = Advanced Encryption Standard
- CA = Certificate Authority
- CCM = Counter with Cipher Block Chaining-Message Authentication Code
- cmd = command line
- DNS = Domain Name System
- GCM = Galois/Counter Mode
- HTTP = HyperText Transfer Protocol
- HTTPS = HyperText Transfer Protocol Secure
- IETF = Internet Engineering Task Force
- IP = Internet Protocol
- NSS = Network Security Services
- TCP/IP = Internet Protocol Suite
- OCSP = Online Certificate Status Protocol
- OSI = Open Systems Interconnection model
- PFS = Perfect Forward Secrecy
- PSK = Pre-Shared Key
- RTT = Round-trip time
- SSL = Secure Sockets Layer
- TCP = Transport Control Protocol
- TLS = Transport Layer Security
- UDP = User Datagram Protocol
- VM = Virtual Machine

Γλωσσάρι Ξενικών Όρων

- **Bit.** Ένα bit μπορεί να βρίσκεται σε μία από δύο καταστάσεις. Αυτές αντιστοιχούν στο εάν εκείνη τη στιγμή περνάει ρεύμα στο μηχανήμα (1) ή όχι (0). Είναι η μικρότερη μονάδα μέτρησης στη πληροφορική.
- **Boolean.** Τιμή για λογικές μεταβλητές. Μπορεί είτε να είναι True είτε False και χρησιμοποιείται συχνά σε συνθήκες if-else για να ελεγχτεί κάτι πριν ξεκινήσουν άλλες εντολές.
- **Byte.** Οκτάδα αποτελούμενη από bits (0 και 1). Ένα byte είναι μονάδα μέτρησης στη πληροφορική.
- **Client.** Πελάτης σε ένα δίκτυο είναι μια ηλεκτρονική συσκευή που κάνει αιτήματα σε έναν εξυπηρετητή. Παραδείγματα αποτελούν το να ρωτήσει εάν είναι το domain που ψάχνει και να ζητήσει τη φόρτωση μιας σελίδας HTTP.
- **Check.** Παράμετρος της συνάρτησης subprocess.run().
- **Domain.** Περιοχή καθορίζεται ένας εξυπηρετητής που έχει διεύθυνση IP στο διαδίκτυο. Το όνομα που συσχετίζεται με τη διεύθυνση είναι το domain name.
- **Header.** Κεφαλίδα είναι το πρώτο πράγμα που φαίνεται σε ένα επίπεδο όταν το διαβάζει ο παραλήπτης. Μέσα σε λίγα bytes εξηγεί για τι πρόκειται το πακέτο που θα ξετυλίξει και σε ποιο επίπεδο να σταλθεί. Το επίπεδο μεταφοράς αφαιρέσει μόνο την κεφαλίδα στην οποία του απευθύνεται (και επομένως το περιεχόμενο προορίζεται για το ίδιο) και όχι το υπόλοιπο για το επίπεδο εφαρμογής. Εκείνο θα σταλθεί στο επόμενο επίπεδο, της εφαρμογής.
- **Input.** Είσοδος. Είτε είναι στην οθόνη τερματικού ως μια εντολή, είτε ένα κείμενο txt το οποίο θα διαβαστεί από μια συνάρτηση ενός script για να εκτελεστούν πράξεις, το input γίνεται με το πληκτρολόγιο και αφορά δεδομένα που θέλει να περάσει ο χρήστης στον υπολογιστή για να πάρει ένα αποτέλεσμα ακόμη και αν είναι σφάλμα. Το αποτέλεσμα ονομάζεται output.
- **Layer.** Ένα επίπεδο υπάρχει για να μεταφέρει το περιεχόμενο που λαμβάνει (είτε είναι το payload στο ανώτατο επίπεδο, είτε wrapped payload στο κατώτατο) στο επόμενο επίπεδο ή στον αποστολέα. Χρησιμοποιεί μια συγκεκριμένη επικεφαλίδα.
- **Metadata.** Τα μεταδεδομένα δεν βρίσκονται στο πακέτο αλλά είναι ο τρόπος που καταλαβαίνει κάποιο επίπεδο ή εργαλείο (στο επίπεδο εφαρμογής)

περισσότερες πληροφορίες για το πακέτο, χωρίς αυτές όμως να είναι παρόν στο περιεχόμενο.

- **Operating System.** Το λειτουργικό σύστημα είναι ένα περιβάλλον στο οποίο αλληλοεπιδρά ο χρήστης με τη συσκευή. Τα Windows και τα Ubuntu (Unix-like, οικογένεια των Linux) είναι λειτουργικά συστήματα.
- **Output.** Έξοδος. Το αποτέλεσμα μιας ενέργειας (πράξης, εντολής, script και άλλες) που εξάγεται στην οθόνη ή σε κάποιο αρχείο. Για να υπάρχει output χρειάζεται πάντα ένα ή περισσότερα input.
- **Payload.** Το περιεχόμενο που ζητάει ο client. Είναι το μήνυμα ενός πακέτου και οι επικεφαλίδες που το καλύπτουν είναι τα γραμματόσημα.
- **Plaintext.** Το απλό κείμενο είναι σαν αυτό που διαβάζεται αυτή τη στιγμή. Κατανοητό κείμενο σε μια γλώσσα (οποιαδήποτε). Το κρυπτογραφημένο κείμενο δεν μπορεί να γίνει κατανοητό χωρίς τις απαραίτητες ενέργειες και ονομάζεται ciphertext.
- **Python.** Γλώσσα προγραμματισμού υψηλού επιπέδου. Σε αντίθεση με την C και Java, στη Python δεν ορίζεται ο τύπος της μεταβλητής. Επιπλέον αφαιρεί την χρήση ερωτηματικών στο τέλος κάθε γραμμής. Η Python προσφέρει τη δυνατότητα δημιουργίας και εκτέλεσης μικρών προγραμμάτων, αποκαλούμενα script. Τα αρχεία σε python βρίσκονται με τη κατάληξη `.py`.
- **Script.** Κώδικας ο οποίος μέσα από τη χρήση βιβλιοθηκών και άλλων εργαλείων φτάνει σε ένα αποτέλεσμα. Διαφορετικό από ένα πρόγραμμα που γράφεται με βιβλιοθήκες από το μηδέν και παράγει πολλά αρχεία όταν συντάσσεται (compiled), όπως στην γλώσσα C++.
- **Server.** Ο εξυπηρετητής είναι μια συσκευή που αναμένει αιτήματα να ολοκληρώσει. Μπορεί να είναι μια ιστοσελίδα `ionio.gr` ή μια τοπική συσκευή LAN που εξυπηρετεί πολλούς άλλους υπολογιστές (εργαστήριο πληροφορικής ενός Πανεπιστημίου).
- **Session resumption.** Η συνέχιση μια παλιάς συνόδου ώστε να παραληφθεί η χειραψία. Γίνεται χρήση του κλειδιού κρυπτογράφησης και παραμέτρων TLS 1.3 που είχαν συμφωνήσει στη συγκεκριμένη σύνοδος. Το session resumption επιτυγχάνεται με ένα session ticket.
- **Shell.** Παράμετρος της συνάρτησης `subprocess.run()`.

- Social Media. Τα μέσα κοινωνικής δικτύωσης συνδέουν πολλούς χρήστες μεταξύ τους για την ενημέρωση μέσω ειδήσεων ή να ανταλλάξουν απόψεις σε μορφή κειμένου, φωτογραφίας ή βίντεο. Η μονάδα μέτρησης είναι η «σύνδεση». Για να υπολογιστεί πόσες συνδέσεις έχει κάποιος, καθορίζεται από κοινές τιμές παραμέτρων. Για παράδειγμα στο Facebook είναι οι κοινοί φίλοι, ενώ στο Twitter οι ακόλουθοι.
- Wrap. Όταν ένα επίπεδο βάζει τα δικά του αναγνωριστικά (π.χ. κεφαλίδα) στο payload ή πακέτο που λαμβάνει, το «τυλίγει». Αυτό γίνεται από τον αποστολέα του μηνύματος. Ονομάζεται και encapsulation. Το ξετύλιγμα γίνεται από τον παραλήπτη.

Ευρετήριο

Κεφάλαιο Α'3.1. Payload

Plaintext, σελίδα 2

Payload, σελίδα 2

Server, σελίδα 2

Κεφάλαιο Α'3.2. Application Layer

HTTP, σελίδα 2

DNS, σελίδα 2

Κεφάλαιο Α'3.3. Transport Layer

TCP, σελίδα 2

standard, σελίδα 3

Κεφάλαιο Β'1. Πως βοηθάει η εφαρμογή του TLS

TLS, σελίδα 5

Client, σελίδα 5

Κεφάλαιο Β'1.1. Ο λόγος κυκλοφορίας μιας νέας έκδοσης

Perfect Forward Secrecy, σελίδα 6

Κεφάλαιο Β'2.1. TLS 1.2 και 1.3 cipher suites

RSA, σελίδα 7

Κεφάλαιο Β'2.2. Εξήγηση των cipher suites του TLS 1.3.

AES, σελίδα 7

Ciphertext, σελίδα 7

GCM, σελίδα 7

CCM, σελίδα 7

Hash function, σελίδα 7

Κεφάλαιο Β'2.3. Αναπαράσταση του αλγορίθμου Diffie-Hellman για το TLS 1.3

Diffie-Hellman, σελίδα 8

Κεφάλαιο Β'3.1. Χαρακτηριστικά περιβάλλοντος και προετοιμασία

Scapy, σελίδα 10

tshark, σελίδα 10

Κεφάλαιο Β'3.2. Κώδικας

SSLKEYLOGFILE, σελίδα 11

Κεφάλαιο Β'4 Εισαγωγή στη χειραψία

Handshake, σελίδα 18

RTT, σελίδα 18

Κεφάλαιο Β'4.1. Διαφορές RTT μεταξύ 1.2 και 1.3 στη χειραψία

Κεφάλαιο Β'4.1.1. TLS 1.2

PMS, σελίδα 19

Κεφάλαιο Β'4.1.2. TLS 1.3

Εικόνα 11. Η χειραψία στο TLS 1.3 γίνεται μόνο με αλγόριθμο ανταλλαγής κλειδιών Diffie-Hellman [15]

Κεφάλαιο Β'4.1.3. Session Resumption και Early Data

Session ticket, σελίδα 21

Early data, σελίδα 21

Κεφάλαιο Β'5.1. Πρώτο βήμα της χειραψίας, Client Hello

Record layer, σελίδα 22

Extension: Key share, σελίδα 29

Extension: Supported versions, σελίδα 29

Extension: Signature Algorithms, σελίδα 30

Κεφάλαιο Β'5.2. Δεύτερο βήμα της χειραψίας, Server Hello, Server Finished

Κεφάλαιο 5'2.1. Certificate Chain

Εικόνα 37, σελίδα 38

Εικόνα 38, σελίδα 39

Εικόνα 39, σελίδα 39

Εικόνα 40, σελίδα 40

Κεφάλαιο 5'.2.2. Certificate Verify, Finished

Εικόνα 57, σελίδα 49

