ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Διαλειτουργικά Συστήματα Ταυτοποίησης και ελέγχου πρόσβασης: Μελέτη περίπτωσης σε Active Directory»

«Εικόνα»

Του φοιτητή:
Μανιατάκου Γεώργιου
Αρ. Μητρώου: 175118

Επιβλέπων:
Ηλιούδης Χρήστος
Βαθμίδα: Καθηγητής

Ημερομηνία: 13/01/2026

Τίτλος Δ.Ε.: Διαλειτουργικά Συστήματα Ταυτοποίησης και ελέγχου πρόσβασης: Μελέτη περίπτωσης σε Active Directory

Κωδικός Δ.Ε.: 24314

Ονοματεπώνυμο φοιτητή: Μανιατάκος Γεώργιος

Ονοματεπώνυμο εισηγητή: Χρήστος Ηλιούδης

Ημερομηνία ανάληψης Δ.Ε.: 03/11/2024

Ημερομηνία περάτωσης Δ.Ε.: 13/01/2026

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η παρούσα εργασία αποτελεί πνευματική ιδιοκτησία του φοιτητή Μανιατάκου Γεώργιου που την εκπόνησε. Στο πλαίσιο της πολιτικής ανοικτής πρόσβασης, ο συγγραφέας/δημιουργός εκχωρεί στο Διεθνές Πανεπιστήμιο της Ελλάδος άδεια χρήσης του δικαιώματος αναπαραγωγής, δανεισμού, παρουσίασης στο κοινό και ψηφιακής διάχυσης της εργασίας διεθνώς, σε ηλεκτρονική μορφή και σε οποιοδήποτε μέσο, για διδακτικούς και ερευνητικούς σκοπούς, άνευ ανταλλάγματος. Η ανοικτή πρόσβαση στο πλήρες κείμενο της εργασίας, δεν σημαίνει καθ' οιονδήποτε τρόπο παραχώρηση δικαιωμάτων διανοητικής ιδιοκτησίας του συγγραφέα/δημιουργού, ούτε επιτρέπει την αναπαραγωγή, αναδημοσίευση, αντιγραφή, πώληση, εμπορική χρήση, διανομή, έκδοση, μεταφόρτωση (downloading), ανάρτηση (uploading), μετάφραση, τροποποίηση με οποιονδήποτε τρόπο, τμηματικά ή περιληπτικά της εργασίας, χωρίς τη ρητή προηγούμενη έγγραφη συναίνεση του συγγραφέα/δημιουργού.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

«Αφιέρωση»

Πρόλογος

Η επιθυμία μου να εμβαθύνω τις γνώσεις μου στον τομέα της ασφάλειας πληροφοριακών συστημάτων, με απώτερο σκοπό την μελλοντική μου επαγγελματική ενασχόληση, ήταν ο πιο σημαντικός λόγος για τον οποίο επέλεξα την πτυχιακή αυτή. Καθοριστικό ρόλο στην απόφαση αυτή ήταν και η συνεργασία με τον κύριο Ηλιούδη, ο οποίος λόγω της εξειδίκευσης του στον κλάδο αυτό, γνώριζα πως θα μπορούσε να με βοηθήσει σε όλα τα στάδια της υλοποίησης. Επιπλέον, το θέμα μου φάνηκε ιδιαίτερα ενδιαφέρον από την αρχή, λόγω του ότι μελετά μια ευρύτερη κατηγορία συστημάτων, εμβαθύνοντας όμως σε μια συγκεκριμένη περίπτωση, κάνοντας το ταυτόχρονα γενικό και ειδικό, προσφέροντας πολυεπίπεδη γνώση.

Περίληψη

Στα πλαίσια αυτής της πτυχιακής εργασίας, παρουσιάζεται η επιτακτική ανάγκη για ενιαία πρόσβαση (SSO) σε ετερογενή περιβάλλοντα, στα οποία κυριαρχεί το Active Directory, έναντι των παραδοσιακών μοντέλων αυθεντικοποίησης, τα οποία βασίζονται αποκλειστικά σε username και password. Επιπλέον, παρουσιάζεται η εκτενής θεωρητική μελέτη που έγινε για τα πλέον διαδεδομένα διαλειτουργικά συστήματα αυθεντικοποίησης, καθώς και η συγκριτική αξιολόγηση τους, με σκοπό την ανάδειξη των πλεονεκτημάτων που τα καθιστούν κατάλληλα για διαφορετικά σενάρια υλοποίησης. Ταυτόχρονα με την θεωρητική μελέτη των παραπάνω, σχεδιάστηκε και υλοποιήθηκε ένα custom API σε .NET Core, το οποίο φιλοξενείται στον IIS του Windows Server που διαμορφώθηκε για τις ανάγκες της πτυχιακής. Απώτερος σκοπός ήταν η γεφύρωση του Active Directory με σύγχρονες εφαρμογές, όσο το API λειτουργεί ως backend για τον SSO Proxy που υλοποιήθηκε σε Node.js. Η προτεινόμενη αυτή υλοποίηση, δημιουργεί ένα επεκτάσιμο διαλειτουργικό σύστημα, το οποίο ενισχύει την ασφάλεια του όλου συστήματος, υιοθετώντας αρχές Zero Trust, χωρίς όμως να επιβαρύνει την εμπειρία του χρήστη.

«Διαλειτουργικά Συστήματα Ταυτοποίησης και ελέγχου πρόσβασης: Μελέτη περίπτωσης σε Active Directory»

(Interoperable Identification and Access Control Systems: A Case Study in Active Directory)

«Μανιατάκος Γεώργιος»

(Maniatakos Georgios)

Abstract

Within the framework of this thesis, the urgent need for single sign-on (SSO) in heterogeneous environments dominated by Active Directory is presented, as opposed to traditional authentication models, which rely exclusively on username and password. In addition, it presents an extensive theoretical study of the most widely used interoperable authentication systems, as well as a comparative evaluation of them, with the aim of highlighting the advantages that make them suitable for different implementation scenarios. Alongside the theoretical study of the above, a custom API was designed and implemented in .NET Core, which is hosted on the IIS of Windows Server, which was configured for the needs of the thesis. The ultimate goal was to bridge Active Directory with modern applications, as the API acts as a backend for the SSO Proxy implemented in Node.js. This proposed implementation creates an extensible interoperable system that enhances the security of the entire system by adopting Zero Trust principles, without compromising the user experience.

Ευχαριστίες

Για τη διεκπεραίωση της παρούσας ερευνητικής εργασίας, θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου, Χρήστο Ηλιούδη για τη συνεργασία και την πολύτιμη συμβολή του στην ολοκλήρωση της. Θα ήθελα επίσης να ευχαριστήσω την οικογένεια, τους φίλους και τη σχέση μου για την πολύτιμη υποστήριξη κατά τη διάρκεια της συγγραφής της και την συνεχή εμψύχωση τους.

Περιεχόμενα

Κατάλογος Πινάκων

Κατάλογος Εικόνων

Συντομογραφίες

Δ.Ε.	Διπλωματική Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Π.Ε.	Πτυχιακή Εργασία
IoT	Internet of the things
SSO	Single Sign-On
2FA	Two-Factor Authentication
MFA	Multi-factor authentication
JWT	JSON Web Token
ΔΠΕ	Διεπαφές Προγραμματισμού Εφαρμογής
API	Application Programming Interface
OWASP	Open Web Application Security Project
AD	Active Directory
XML	Extensible Markup Language
ΚΑΠΥ	Κατανεμημένη Άρνηση Παροχή Υπηρεσιών
DDoS	Distributed Denial of Services
ΟΑΠΥ	Οικονομικής Άρνησης Παροχής Υπηρεσιών
EDoS	Economic Denial of Services
IT	Information Technology
ΥΠΔ	Υπηρεσίες Πληροφοριών Διαδικτύου
IIS	Internet Information Services
TOTP	Time-Based One-Time Password
OTP	One-time password
URL	Uniform Resource Locator
URI	Uniform Resource Identifier
CSRF	Cross-Site Request Forgery

RO	Resource Owner
RP	Relying Party
IdP	Identity Provider
RS	Resource Server
PKCE	Proof Key for Code Exchange
OIDC	OpenID Connect
SAML	Security Assertion Markup Language
TGT	Ticket Granting Ticket
TGS	Ticket Granting Server
ST	Service Ticket
SS	Service Server
KDC	Key Distribution Center

Κεφάλαιο 1ο: Εισαγωγή

Στο πλαίσιο της ψηφιακής μετάβασης και της αυξανόμενης ζήτησης διαδικτυακών εφαρμογών και υπηρεσιών, η ανάγκη για αποτελεσματική και ασφαλή διαχείριση της ταυτότητας των χρηστών και των δεδομένων τους, είναι μεγάλη. Οργανισμοί, εταιρείες και φορείς όλων των μεγεθών που διαχειρίζονται χρήστες και ευαίσθητα δεδομένα, βρίσκονται αντιμέτωποι με πλήθος κυβερνοαπειλών και επιθέσεων που σχετίζονται με μην εξουσιοδοτημένη πρόσβαση, παραβίαση ευαίσθητων δεδομένων και κατάχρηση προνομίων. Εξαιτίας όλων των παραπάνω, χρειάζονται αξιόπιστα πληροφοριακά συστήματα αυθεντικοποίησης και υπηρεσίες, τα οποία αποτελούν την πρώτη γραμμή άμυνας, και διασφαλίζουν την ταυτότητα των χρηστών και την ελεγχόμενη παροχή δικαιωμάτων.

Τα διαλειτουργικά συστήματα αυθεντικοποίησης SSO χρησιμοποιούνται, ώστε “μεγάλοι οργανισμοί” που έχουν πολλές εφαρμογές ή συστήματα, να μειώσουν τον αριθμό των συνδέσεων που πρέπει να πραγματοποιήσει ένας χρήστης. Αντί δηλαδή να έχουν η κάθε εφαρμογή το δικό της τρόπο αυθεντικοποίησης και να αποθηκεύουν κωδικούς πρόσβασης στη βάση δεδομένων τους, ορίζεται ένας διακομιστής, ο οποίος είναι υπεύθυνος για τα παραπάνω. Κάνει την ταυτοποίηση, αποθηκεύει τους κωδικούς και με την ολοκλήρωση της σύνδεσης, ενημερώνει τους υπόλοιπους διακομιστές, [1] [2] [3] ή επιστρέφει στον χρήστη ένα διαπιστευτήριο, το οποίο επαληθεύει την ταυτότητα του. Με αυτόν τον τρόπο, μειώνεται η πολυπλοκότητα της αυθεντικοποίησης, με αποτέλεσμα να γίνεται πολύ πιο φιλική στον χρήστη, ο οποίος δε χρειάζεται να θυμάται πολλούς κωδικούς, προλαμβάνοντας έτσι την επαναλαμβανόμενη χρήση του ίδιου κωδικού σε πολλές εφαρμογές, γεγονός που υπονομεύει την ασφάλεια. Ταυτόχρονα, οι διαχειριστές επικεντρώνουν την προσοχή τους σε έναν server, με αποτέλεσμα να κάνουν πιο αποτελεσματικό monitoring και να μπορούν να δώσουν έμφαση στην ενίσχυση της ασφάλειας του. [1] [3] Τα SSO μπορούν να χρησιμοποιηθούν και σε συστήματα IoT.[4] Οι “μεγάλοι οργανισμοί” που αναφέρθηκαν παραπάνω, μπορεί να είναι πανεπιστήμια, δημόσιες υπηρεσίες, νοσοκομεία ή οι γενικότερα οργανισμοί με μικροϋπηρεσιακή αρχιτεκτονική.

Η ύπαρξη συστημάτων που καλύπτουν τις σύγχρονες απαιτήσεις ασφάλειας, όπως η πολλαπλή αυθεντικοποίηση (2FA – Two-Factor Authentication), η έκδοση tokens (JWT) και η διαχείριση ρόλων (RBAC – Role-Based Access Control), κάνουν το παραδοσιακό μοντέλο «όνομα χρήστη - κωδικός πρόσβασης» να φαίνεται και να είναι ανεπαρκές. Αναλυτικότερα, η διαχείριση ρόλων επιτρέπει τον λεπτομερή έλεγχο και τη δυναμική απόδοση δικαιωμάτων.

Στα σύγχρονα πληροφοριακά συστήματα, η ανταλλαγή δεδομένων και η διασύνδεση μεταξύ εφαρμογών, συστημάτων και συσκευών διευκολύνεται από τις Διεπαφές Προγραμματισμού Εφαρμογής (ΔΠΕ) ή αλλιώς από τα Application Programming Interfaces (APIs). Η αυξανόμενη χρήση τους και η συχνή έκθεση τους στο διαδίκτυο, έχει κάνει τα APIs βασικούς στόχους κακόβουλων ενεργειών. Κάθε τρία με τέσσερα χρόνια, ο οργανισμός OWASP δημοσιεύει λίστα με τις δέκα πιο συχνές ευπάθειες των APIs, κάνοντας γνωστές μερικές ευπάθειες που κάποιοι μπορεί να μην είχαν εντοπίσει, αλλά και βοηθώντας όσους τα χρησιμοποιούν να εντοπίσουν τα σημεία που χρειάζονται αλλαγές, ώστε να ενισχυθεί η ασφάλεια τους. Μερικά από τα κύρια τρωτά σημεία, είναι ο ανεπαρκής έλεγχος ταυτότητας, η κακή διαχείριση εξουσιοδοτήσεων και η υπερβολική έκθεση δεδομένων.

1.1 Στόχοι

Στόχος αυτής της πτυχιακής εργασίας ήταν η εμβάθυνση των γνώσεων όσων αφορά τα συστήματα διαλειτουργικής αυθεντικοποίησης και ελέγχου πρόσβασης, και πιο συγκεκριμένα η μελέτη της περίπτωσης των AD. Για να γίνει αυτό, αναγκαίο ήταν να προηγηθεί αναζήτηση πληροφοριών για τα πιο ευρέως χρησιμοποιούμενα συστήματα, ώστε να αποκτηθούν πληροφορίες για τον τρόπο λειτουργίας τους, τις υποδομές που χρειάζονται, τα πρωτόκολλα που χρησιμοποιούν, την απόδοση ταχύτητας που έχουν, το κόστος τους, τον αριθμό των χρηστών που μπορούν να εξυπηρετηθούν, τα προτερήματα και τις γνωστές μέχρι τώρα ευπάθειες που έχει το κάθε ένα. Μετά την αναζήτηση αυτή, στόχος ήταν να γίνει μια εικονική σύγκριση μεταξύ των συστημάτων για την επιλογή του καταλληλότερου για την κάθε περίπτωση και η δημιουργία ενός σύγχρονου συστήματος διαλειτουργικής αυθεντικοποίησης, το οποίο θα είναι ασφαλές και επεκτάσιμο, ενώ θα χρησιμοποιείται και για την αυθεντικοποίηση των χρηστών του AD.

1.2 Επιτεύγματα

Οι παραπάνω στόχοι επιτεύχθηκαν, εκτός από την εύρεση του κόστους και του αριθμού χρηστών που μπορούν να εξυπηρετηθούν από το κάθε σύστημα, καθώς και τα δύο διαφέρουν ανά περίπτωση και οι παράμετροι που συμβάλλουν στον καθορισμό τους είναι πολλοί. Πέραν αυτών, αναπτύχθηκε ένα custom σύστημα διαλειτουργικής αυθεντικοποίησης, ώστε να ταιριάζει περισσότερο στις ανάγκες της πτυχιακής αυτής, σε σύγκριση με το ήδη υπάρχον της Microsoft, το οποίο είναι καθορισμένο και δεν μπορεί να παραμετροποιηθεί εύκολα. Επιπλέον, χρησιμοποιήθηκε το fingerprint του browser ως αναγνωριστικό του χρήστη και ο αποκλεισμός

της IP έγινε σε επίπεδο δικτύου. Τα παραπάνω, δεν ήταν στόχοι από την αρχή, αλλά προέκυψαν κατά την ανάπτυξη του τεχνικού κομματιού.

1.3 Διάρθρωση της πτυχιακής

Στο πρώτο και εισαγωγικό κεφάλαιο αυτής της εργασίας, παρουσιάζεται το αντικείμενο μελέτης, ο σκοπός της, η σημασία των συστημάτων ταυτοποίησης στη σύγχρονη κυβερνοασφάλεια και η συνολική δομή της εργασίας.

Το κεφάλαιο που ακολουθεί, αποτελεί το θεωρητικό υπόβαθρο της εργασίας, όπου και πραγματοποιείται εκτενής ανασκόπηση των πρωτοκόλλων OAuth2.0, OpenID Connect, Kerberos και SAML όπου είναι τα πλέον διαδεδομένα διαλειτουργικά συστήματα αυθεντικοποίησης. Επίσης, αναλύεται η λειτουργία του AD, η ασφάλεια των APIs καθώς και οι αρχές του MFA.

Στο τρίτο κεφάλαιο πραγματοποιείται συγκριτική αξιολόγηση μεταξύ υφιστάμενων λύσεων αυθεντικοποίησης, όπως είναι το ADFS που προσφέρει η Microsoft, το λογισμικό ανοιχτού κώδικα Keycloak και το προσαρμοσμένο (custom) API που αναπτύχθηκε. Συγκεκριμένα, η σύγκριση γίνεται βάσει κριτηρίων, όπως η υποστήριξη SSO, η δυνατότητα χρήσης MFA, οι απαιτήσεις υποδομής καθώς και η επεκτασιμότητα τους.

Στο τέταρτο κεφάλαιο γίνεται αναφορά στο τεχνολογικό περιβάλλον και τις υποδομές που χρησιμοποιήθηκαν σε αυτό. Αναλυτικότερα, αναλύονται οι βασικές λειτουργίες και η δομή του AD, οι απαιτήσεις συστήματος σε επίπεδο υλικού και λογισμικού, καθώς και τεχνολογίες που επιλέχθηκαν για την ανάπτυξη του API.

Στο πέμπτο κεφάλαιο πραγματοποιείται εκτενής περιγραφή της τεχνικής υλοποίησης της εφαρμογής. Παρουσιάζονται αναλυτικά η αρχιτεκτονική του συστήματος, οι ροές αυθεντικοποίησης, συμπεριλαμβανομένου και του μηχανισμού TOTP, καθώς και τα Endpoints που αναπτύχθηκαν. Επιπλέον, δίνεται ιδιαίτερη έμφαση στους μηχανισμούς ασφαλείας που ενσωματώθηκαν, όπως η καταγραφή του αποτυπώματος- fingerprint του browser του χρήστη, ο αποκλεισμός της πραγματικής IP του χρήστη σε επίπεδο firewall και η προστασία του από επιθέσεις brute force. Στο τέλος του κεφαλαίου παρατίθενται σενάρια ελέγχου και μετρήσεις απόδοσης του συστήματος.

Στο έκτο και τελευταίο κεφάλαιο παρουσιάζονται συνοπτικά τα συμπεράσματα της εργασίας σχετικά με τη σημαντικότητα των διαλειτουργικών συστημάτων αυθεντικοποίησης και προτείνονται μελλοντικές επεκτάσεις και βελτιώσεις του κώδικα.

Στο τέλος της εργασίας παρατίθεται το Παράρτημα Α, το οποίο περιέχει τους συνδέσμους για τα αποθετήρια κώδικα (repositories), καθώς και μερικά από τα σημαντικότερα αποσπάσματα κώδικα της υλοποίησης.

Κεφάλαιο 2ο: Διαλειτουργικά συστήματα αυθεντικοποίησης

2.1 Διεπαφή Προγραμματισμού Εφαρμογής

Η ΔΠΕ ή αλλιώς API είναι ένα σύνολο κανόνων, πρωτοκόλλων και εργαλείων που δίνει τη δυνατότητα σε τρίτους να αλληλεπιδράσουν με τα δεδομένα και τις λειτουργίες μιας εφαρμογής ή υπηρεσίας, χωρίς να γνωρίζουν τον τρόπο λειτουργίας και να έχουν πρόσβαση σε όλη τη βάση δεδομένων της. [5] [6] Πιο σύνηθες είναι να υπάρχουν δύο εφαρμογές, όπου η μια έχει τις πληροφορίες και η άλλη τις ζητάει. Για την καλύτερη "επικοινωνία" μεταξύ τους, οι πληροφορίες μορφοποιούνται και στέλνονται σε JSON ή XML, διότι είναι εύκολες στη χρήση και τη διαχείριση. [6]

Με την ανάπτυξη ενός API υπάρχει μεγάλη πιθανότητα να δημιουργηθούν κενά ασφαλείας στο σύστημα, διότι δίνει πρόσβαση σε δεδομένα και λειτουργίες μιας εφαρμογής ή εταιρίας σε μια άλλη. Ένα μεγάλο πρόβλημα των διεπαφών είναι πως τα κενά αυτά δεν μπορούν να εντοπιστούν με τα έως τώρα υπάρχοντα εργαλεία. Αναλυτικότερα, περίπου το 90% των εταιρειών που χρησιμοποιούν παραγωγικά APIs, δηλαδή API για μεταφορά πληροφοριών σε τρίτες εταιρείες, έχουν αναφέρει ότι έχουν αντιμετωπίσει ή αντιμετωπίζουν θέματα ασφαλείας. [5] [6]

Παρατηρώντας το θέμα στατιστικά, το 41% των εταιρειών που δέχθηκαν επίθεση, δεν έδωσε παραπάνω πληροφορίες για το είδους της, ενώ από αυτές που έδωσαν πληροφορίες το 41% δέχθηκαν επιθέσεις χρησιμοποιώντας τα κενά ασφαλείας που υπήρχαν κατά την αυθεντικοποίηση των χρηστών ή των συστημάτων που προσπαθούν να έχουν πρόσβαση στο API τους. Το 31% υπέστη επιθέσεις που οδήγησαν στην έκθεση προσωπικών ή ευαίσθητων δεδομένων, ενώ το 17% υπέστη ολική παραβίαση ασφαλείας (full-blow security breach).[5] [6]

Παραδείγματα εταιρειών που παρά το μέγεθος τους έχουν δεχτεί τέτοιου τύπου επιθέσεις και το γεγονός γνωστοποιήθηκε, είναι η Google, το Facebook, η T-Mobile, το Twitter[5] και το LinkedIn[7]. Αποτέλεσμα αυτού είναι να επηρεάζεται η φήμη τους, να κλονίζεται η εμπιστοσύνη των χρηστών τους και κατ' επέκταση να έχουν απώλεια χρημάτων.

Το OWASP (Open Web Application Security Project) είναι ένας μη κερδοσκοπικός οργανισμός που έχει στόχο την βελτιοποίηση της ασφάλειας των εφαρμογών. Για να το πετύχει αυτό, έχει δημιουργήσει μια λίστα με τις 10 πιο συχνά εμφανιζόμενες ευπάθειες των APIs, την οποία ανανεώνει κάθε 3 χρόνια, ώστε να είναι πάντα ενημερωμένη.[8] Σύμφωνα με την λίστα

του 2023, οι 10 πιο συχνά εμφανιζόμενες ευπάθειες των APIs ήταν αυτές που παρουσιάζονται στον παρακάτω πίνακα.

Πίνακας 1: Λίστα OWASP 2023

Code	**Name of vulnerability**
API1:2023	Broken Object Level Authorization
API2:2023	Broken Authentication
API3:2023	Broken Object Property Level Authorization
API4:2023	Unrestricted Resource Consumption
API5:2023	Broken Function Level Authorization
API6:2023	Unrestricted Access to Sensitive Business Flows
API7:2023	Server Side Request Forgery
API8:2023	Security Misconfiguration
API9:2023	Improper Inventory Management
API10:2023	Unsafe Consumption of APIs

2.1.1 Τρόποι προστασίας

Για την ενίσχυση της ασφάλειας των APIs υπάρχουν κάποιες πρακτικές και εργαλεία που μπορούν να χρησιμοποιηθούν, όπως το JWT και το OAuth 2.0, τα οποία βοηθούν στην αυθεντικοποίηση των χρηστών και των συστημάτων που προσπαθούν να έχουν πρόσβαση στο API [9]. Συγκεκριμένα το JWT, το οποίο και επιλέχθηκε ανάμεσα σε άλλα παρόμοια για την διεκπεραίωση του προγραμματιστικού κομματιού της πτυχιακής αυτής, δημιουργεί tokens που είναι δύσκολο να προβλεφθούν. [6] Ο λόγος που επιλέχθηκε έναντι άλλων, παρουσιάζεται στη συνέχεια.

Μερικά σημεία στα οποία πρέπει να δοθεί ιδιαίτερη προσοχή όταν δημιουργείται ένα API, ώστε να μειώνονται όσο γίνεται τα κενά ασφαλείας του, παρουσιάζονται παρακάτω.

Αρχικά, το API πρέπει να επικοινωνεί μόνο με μια συγκεκριμένη λίστα εφαρμογών και συστημάτων που είναι εξουσιοδοτημένα, το οποίο σημαίνει πως οποιοδήποτε αίτημα από εφαρμογή εκτός της λίστας απευθείας απορρίπτεται. Αυτό γίνεται στα πλαίσια του ελέγχου των δικαιωμάτων.[6] Επιπλέον, για κάθε λειτουργία του, το API πρέπει να απαντάει από ξεχωριστό endpoint, δηλαδή να υπάρχει περιορισμός στα δεδομένα που επιστρέφονται. Σημαντικό είναι, πως ο έλεγχος των δεδομένων αυτών, θα πρέπει να πραγματοποιείται από το API και όχι από την εφαρμογή και πως τα δεδομένα αυτά πρέπει να στέλνονται κρυπτογραφημένα. [6] [10] Σε περίπτωση σφάλματος από την πλευρά του API, ο έλεγχος πρέπει να γίνεται πάλι από το API, προτού απαντήσει στον χρήστη, ώστε να μην εκτίθενται πληροφορίες για την λειτουργία και τις ευπάθειες του. [10]

Στα πλαίσια του ελέγχου που αφορά τους χρήστες και τα συστήματα που χρησιμοποιούν API, σημαντικό είναι να ελέγχονται και οι κινήσεις τους (πχ. αιτήματα, προσπάθειες σύνδεσης κλπ). Αυτό μπορεί να γίνει με τη δημιουργία ειδικών ανιχνευτών, οι οποίοι θα εντοπίζουν επιθέσεις Κατανεμημένης Άρνησης Παροχής Υπηρεσιών (Distributed Denial of Services - DDoS) και Οικονομικής Άρνησης Παροχής Υπηρεσιών (Economic Denial of Services - EDoS). [5] Οι κινήσεις αυτές τι περισσότερες φορές καταγράφονται αυτόματα για τη δημιουργία αρχείου (logs), το οποίο μπορεί να χρησιμοποιηθεί για την ανίχνευση μιας απειλής. Αμέσως μετά την ανίχνευση, ακολουθεί ενημέρωση των διαχειριστών του API σε πραγματικό χρόνο, ώστε να αντιμετωπιστεί όσο πιο άμεσα γίνεται. [5]

Επιπλέον, στην περίπτωση αυθεντικοποίησης χρηστών του AD, πολύ σημαντικό είναι, οι admin χρήστες να έχουν περιορισμό ως προς τις ενέργειες που μπορούν να πραγματοποιήσουν μέσω του API. Για παράδειγμα, διεργασίες όπως η δημιουργία ή η άρση αποκλεισμού ενός χρήστη, να πραγματοποιούνται μόνο από το UI του Windows Server, [10] ώστε σε περίπτωση πρόσβασης ενός επιτιθέμενου στο προφίλ ενός χρήστη admin, να μην μπορεί να παραμετροποιηθεί πολύ ο διακομιστής. Τέλος, πρέπει να δημιουργηθεί Documentation για το API, να αποσταλεί στις εταιρείες ή τους διαχειριστές των εφαρμογών που το χρησιμοποιούν και να ενημερώνεται κάθε φορά που κάτι λειτουργικό τροποποιείται, ώστε να είναι πάντα ενημερωμένο.[6]

2.1.2 Αυθεντικοποίηση

Για να γίνει σύνδεση σε μια εφαρμογή, τα στοιχεία σύνδεσης κωδικοποιούνται, στέλνονται στο API, αυτό επαληθεύει την εγκυρότητα τους και δίνει ένα μοναδικό token. Για κάθε αίτημα το API αναγνωρίζοντας αυτό το token και τα δικαιώματα που έχει ο

συγκεκριμένος λογαριασμός, ελέγχει το αίτημα και στη συνέχεια επιστρέφει τις πληροφορίες που ζητήθηκαν ή εμφανίζει την ένδειξη σφάλματος, εάν κάτι πάει λάθος. [6]

Το token περιέχει τις απαραίτητες πληροφορίες (ρόλος, ID και δικαιώματα χρήστη) και είναι αποθηκευμένο και από την πλευρά του χρήστη και από την πλευρά του διακομιστή, ώστε ο διακομιστής να μην χρειάζεται να διατηρεί ανοιχτό session για κάθε χρήστη. Για την επικοινωνία του χρήστη με τον διακομιστή, αρκεί να σταλεί από τον χρήστη ένα αίτημα μαζί με το token του. [9]

Για να αποφευχθούν κενά ασφαλείας σε όλη αυτή τη διαδικασία, πρέπει να δοθεί προσοχή σε ορισμένα σημεία. [6] Μερικά από αυτά είναι, τα endpoints που σχετίζονται με την αυθεντικοποίηση των χρηστών, όπως το login και η αλλαγή κωδικού πρόσβασης, καθώς είναι τα πιο ευπαθή σε επιθέσεις τύπου brute force και αποκλεισμού του χρήστη. [6] Για περαιτέρω ενίσχυση της ασφάλειας συστήνεται να χρησιμοποιούνται multi-factor authentication (MFA).[10] Οι χρήστες βέβαια από μόνοι τους ή ύστερα από προτροπή από τους IT πρέπει να χρησιμοποιούν ισχυρούς κωδικούς, τους οποίους συχνά θα ενημερώνουν. [6] [10] Επιπλέον, χρήσιμη μπορεί να φανεί η οριοθέτηση του αριθμού των αιτημάτων που μπορούν να πραγματοποιηθούν προς το API από έναν χρήστη ή ένα σύστημα, μέσα σε ένα χρονικό διάστημα. Όριο πρέπει να υπάρχει και στον όγκο των δεδομένων που μπορούν να επιστραφούν από το API για κάθε αίτημα. [6]

2.2 Διαλειτουργικά συστήματα αυθεντικοποίησης

Στην ενότητα αυτή θα παρουσιαστούν μερικές βασικές πληροφορίες σχετικές με μερικά από τα πιο χρησιμοποιούμενα διαλειτουργικά συστήματα αυθεντικοποίησης, το OAuth2.0, το OpenID Connect, το SAML και το Kerberos.

2.2.1 OAuth2.0

Το OAuth2.0 είναι ένα πρωτόκολλο εξουσιοδότησης, όχι αυθεντικοποίησης χρηστών, το οποίο λειτουργεί ως SSO και χρησιμοποιείται ευρέως στο διαδίκτυο από εταιρείες όπως η Google, το Facebook, η Microsoft και το LinkedIn.[1] [2] Δημιουργεί και διαχειρίζεται ένα προσωρινό token, το οποίο ονομάζεται access_token και είναι συνήθως JSON Web Token (JWT). Αυτό επιτρέπει σε τρίτες εφαρμογές να έχουν πρόσβαση σε πόρους του χρήστη, χωρίς να χρειάζεται να γνωρίζουν τα απαραίτητα διαπιστευτήρια. [5] [2] [11]

Οι βασικοί ρόλοι των χρηστών στο OAuth2.0 είναι ο Resource Owner (RO), ο Client ή αλλιώς Relying Party (RP), ο Authorization server ή αλλιώς Identity Provider (IdP) και σε

μερικές περιπτώσεις ο Resource Server (RS). Αναλυτικότερα, ο RO μπορεί να είναι ένας χρήστης ή μια ανεξάρτητη εφαρμογή που κατέχει έναν προστατευόμενο πόρο ή έχει δικαίωμα πρόσβασης σε αυτόν. Ο RP είναι μια εφαρμογή, η οποία στέλνει το αίτημα πρόσβασης στους προστατευόμενους πόρους, εκ μέρους του RO. Ο IdP είναι ένας διακομιστής που αυθεντικοποιεί τον RO, εκδίδει τα access_token και το προωθεί στον RP.[12] [11] [13] Τέλος, ο RS είναι ένας διακομιστής που φιλοξενεί τους πόρους του χρήστη και τους παρέχει εφόσον ολοκληρωθεί η αυθεντικοποίηση και υπάρχει έγκυρο access_token.[12] [13] Ο RS χρησιμοποιείται όταν τα δεδομένα που χρειάζεται ο RO βρίσκονται σε άλλο διακομιστή.

Το OAuth2.0 έχει 4 βασικούς τύπους ροών για την εξουσιοδότηση χρηστών. Ο πρώτος και πιο ευρέως χρησιμοποιούμενος είναι ο Authorization Code. Στον συγκεκριμένο τύπο, ο χρήστης - browser δεν έχει επαφή με το access_token και το access_token αποστέλλεται μόνο στον RP από τον IdP, αυξάνοντας έτσι την ασφάλεια του συστήματος και κάνοντας τον, τον πιο ασφαλή τύπος ροής. Ο Resource Owner Password Credentials είναι ένας ακόμα τύπος, ο οποίος χρησιμοποιείται κυρίως σε εφαρμογές που εμπιστεύονται τον RP. Ο RO προωθεί τα διαπιστευτήρια του στον RP, ο οποίος στην συνέχεια τα προωθεί στον IdP και με την σειρά του επιστρέφει το access_token στον RP. Ο τύπος Client Credentials, σε αντίθεση με τις υπόλοιπες ροές, δεν απαιτεί την συμμετοχή του χρήστη, καθώς ο RP στέλνει τα διαπιστευτήρια του στον IdP και λαμβάνει το access_token. Χρησιμοποιείται κυρίως για επικοινωνίες από API προς API. Τέλος, ο Implicit χρησιμοποιείται κυρίως σε mobile και web εφαρμογές. Μετά την επιτυχημένη εξουσιοδότηση του χρήστη, ο IdP επιστρέφει το access_token στον RP μέσω της διεύθυνσης URL του browser. Στις μέρες μας δεν προτείνεται, διότι θεωρείται απαρχαιωμένο. [11] [14] [13]

Το OAuth2.0, όπως και όλα τα υπάρχοντα συστήματα διαλειτουργικής αυθεντικοποίησης, έχουν πλεονεκτήματα και μειονεκτήματα. Τα μειονεκτήματα σε τέτοια συστήματα αποτελούν τις ευπάθειες τους. Μια σημαντική ευπάθεια του προκύπτει όταν μετά την ολοκληρωμένη εξουσιοδότηση, ο χρήστης ανακατευθύνεται σε μη ελεγμένα URL, τα οποία εμπεριέχεται στα αντίστοιχα URI. Προσοχή πρέπει να δίνεται λοιπόν στο αν τα URL που χρησιμοποιούνται είναι αξιόπιστα και όχι κακόβουλα. Επιπλέον, ευπάθεια δημιουργείται όταν ο κωδικός εξουσιοδότησης που στέλνεται από τον IdP κατά την ροή Authorization Code, δεν είναι ασφαλής ή επιτρέπεται η επαναχρησιμοποίηση του. Ο κωδικός αυτός στέλνεται στον RP, μέσω της αναδρομολόγησης, και εκείνος τον επιστρέφει στον IdP, ώστε να λάβει το access_token. Κατά την ίδια ροή, κενό ασφαλείας μπορεί να προκύψει εάν δεν ρυθμιστεί σωστά το πρόσθετο Proof Key for Code Exchange (PKCE), το οποίο προσθέτει ένα επίπεδο

ασφαλείας κατά την διαδικασία ανταλλαγής του κωδικό εξουσιοδότησης και του access_token, μεταξύ RP και IdP. Τέλος, ευπάθεια αποτελεί και η γενικότερα λανθασμένη ανανέωση των access_token. [15]

2.2.2 OpenID Connect

Το OpenID Connect (OIDC), είναι ένα ακόμα σύστημα διαλειτουργικής αυθεντικοποίησης, το οποίο δημιουργήθηκε από την OpenID Foundation. Είναι ένα πρωτόκολλο που βασίζεται στο OAuth2.0, και όχι στο OpenID, παρά το όνομα του, και προσθέτει επιπλέον λειτουργίες στην αυθεντικοποίηση χρηστών και το authorization τους.[16] [13] [3] Το σύστημα αυτό, επιτρέπει στους χρήστες να συνδέονται σε εφαρμογές χρησιμοποιώντας τα διαπιστευτήρια τους από έναν κεντρικό διακομιστή ταυτοποίησης (IdP), όμοια με το OAuth2.0.[16] [13] Χρησιμοποιείται από εταιρείες όπως η Google, η PayPal, η Saleforce, η Microsoft, η Deutsche Telekom και άλλες.[16] Ανάλογα τον τρόπο δημιουργίας του, μπορεί να χρησιμοποιεί ταυτόχρονα το access_token του OAuth2.0 για να έχει πρόσβαση σε APIs (πχ UserInfo endpoint) και το ID token για την αυθεντικοποίηση του χρήστη. [16] [13] [3] Το access_token περιέχει αναφορές για τα δικαιώματα πρόσβασης και όχι απαραίτητα πληροφορίες για τον χρήστη [3], ενώ το ID token περιέχει πληροφορίες για τον χρήστη και για τα session σε ένα υπογεγραμμένο ή και κρυπτογραφημένο JWT. [16] [13] [3]

Οι ρόλοι του OIDC είναι ίδιοι με του OAuth2.0[16] [13], ωστόσο διαφέρουν στην λειτουργία και στις αρμοδιότητες κάθε ρόλου. Πριν την οποιαδήποτε επικοινωνία ανάμεσα στον RP και τον IdP, πρέπει πρώτα να πραγματοποιηθεί Discovery και Dynamic Client Registration, ώστε να πιστοποιηθεί και από τις δύο πλευρές η αυθεντικότητα της άλλης. Μια ακόμη ομοιότητα μεταξύ τους είναι πως και τα δύο επιτρέπουν στον χρήστη να εξουσιοδοτήσει τον RP, ώστε να έχει πρόσβαση στα δεδομένα του από τον IdP.[16]

Το OIDC, έχει τρεις βασικούς τύπους ροών, που αλλιώς ονομάζονται modes. Το Authorization Code και το Implicit, το οποίο είναι ίδιο με τα αντίστοιχα του OAuth2.0 και το Hybrid Mode, το οποίο είναι συνδυασμός των δύο παραπάνω. Στο τελευταίο, ο IdP στέλνει στον browser το token και κατά την αναδρομολόγηση του χρήστη, ο IdP στέλνει στον RP το access_token και το id_token.[16] [13]

Οι ευπάθειες του OIDC είναι αρκετές, για να μπορέσουν να παρουσιαστούν όμως με περισσότερη ευκολία, έχουν κατηγοριοποιηθεί σε αυτές που σχετίζονται με αναγνώριση και πιστοποίηση του IdP, με state, nonce και session integrity, με κακή διαχείριση tokens και

στοιχείων σύνδεσης χρηστών, με επιθέσεις τύπου Injection - manipulation και τέλος με επιθέσεις δικτύου και υποδομής.

Αναλυτικότερα, όσον αφορά την πρώτη κατηγορία ευπαθειών, υπάρχει η IdP Mix-Up Attack, κατά την οποία ο επιτιθέμενος εκμεταλλεύεται την έλλειψη σωστού ελέγχου ταυτότητας του IdP από τον RP. Στόχος του είναι να παραπλανήσει τον RP, ώστε η αυθεντικοποίηση να πραγματοποιηθεί από κακόβουλο IdP. [16] Μια παρόμοια επίθεση είναι η ID Spoofing, στην οποία ο επιτιθέμενος δημιουργεί έναν IdP ο οποίος ισχυρίζεται ότι είναι ο αυθεντικός IdP.[17] Τέλος, μια άλλη πιθανή επίθεση είναι η Naïve RP Session Integrity Attack, στην οποία ο RP προωθεί τον χρήστη στη σελίδα του IdP για την αυθεντικοποίηση, χωρίς να έχει αποθηκεύσει ποιόν IdP έχει επιλέξει. Στη συνέχεια, περιμένει την πληροφορία αυτή μέσω του URI, δημιουργώντας κενό ασφαλείας, το οποίο μπορεί να εκμεταλλευτεί ο επιτιθέμενος για να παραπλανήσει τον RP, με απώτερο σκοπό να τον πείσει πως ο χρήστης έχει αυθεντικοποιηθεί από τον IdP που έχει επιλέξει, ενώ στην πραγματικότητα έχει αυθεντικοποιηθεί από κακόβουλο IdP.[16]

Οι ευπάθειες σχετικά με state, nonce και session integrity, μπορεί να είναι Session Handling, σε περιπτώσεις όπου δεν ανανεώνεται το sessionID του χρήστη μετά από κάθε επιτυχημένη σύνδεση. Δίνεται λοιπόν η ευκαιρία στον επιτιθέμενο να επαναχρησιμοποιήσει παλιά sessionIDs. Μπορεί ακόμα να είναι Attacks on the State Parameter, όταν ο διακομιστής αυθεντικοποίησης δεν καθαρίζει τα states όταν λήγουν ή απενεργοποιούνται οι ενεργές συνεδρίες, επιτρέποντας έτσι σε έναν επιτιθέμενο να επαναχρησιμοποιήσει ένα state. Το state χρησιμοποιείται ως δικλείδα ασφαλείας και είναι μια συμβολοσειρά από τυχαίους χαρακτήρες που δημιουργείται στον RP και αποστέλλεται στον IdP μαζί με το αίτημα αυθεντικοποίησης. Τέλος, σε αυτή τη κατηγορία ευπαθειών ανήκουν τα Attacks and Third-Party Login Initiation, στην οποία δεν υπάρχει CSRF προστασία ή επιβεβαίωση χρήστη. Αυτό έχει σαν αποτέλεσμα ένας επιτιθέμενος να μπορεί να δημιουργήσει έναν σύνδεσμο με τα δικά του στοιχεία σύνδεσης για login flow, τον οποίο στέλνει στο θύμα και αφού αυτό το χρησιμοποιήσει, οι αλλαγές που κάνει είναι ορατές στον επιτιθέμενο. Αυτό είναι εφικτό διότι στο OIDC υπάρχει endpoit που επιτρέπει σε τρίτες εφαρμογές τη δημιουργία login flow χωρίς κάποια πιστοποίηση. [16]

Όσον αφορά τις ευπάθειες που σχετίζονται με την κακή διαχείριση token και στοιχείων σύνδεσης χρηστών, υπάρχει η Code/Token/State Leakage, η 307 Redirect Attack, η Wrong Recipient και τέλος η Replay. Στην πρώτη, μετά την ολοκλήρωση της αυθεντικοποίησης ο IdP

ανακατευθύνει τον χρήστη στην σελίδα του RP, μέσω του URI που περιέχει και παραμέτρους όπως authorization code και state.

Στην περίπτωση που ο χρήστης επιλέξει συνδέσμους προς τρίτα domain ή στην σελίδα που βρίσκεται υπάρχουν εξωτερικοί πόροι (πχ scripts και εικόνες), τότε το πλήρες URL μπορεί να αποσταλεί σε τρίτους μέσω του HTTP Referer header εκθέτοντας τα tokens. [16] Στην δεύτερη, τα στοιχεία που έχει εισάγει ο χρήστης εκτίθενται σε τρίτους (πχ στον RP), λόγω της αναδρομολόγησης του με HTTP 307. Στην Wrong Recipient, ένας επιτιθέμενος μπορεί να χρησιμοποιήσει tokens που έχουν δημιουργηθεί σε άλλους RP και να αποκτήσει πρόσβαση σε ευαίσθητες πληροφορίες του χρήστη. Αυτό μπορεί να γίνει λόγω του ότι ο RP δεν ελέγχει σωστά το client_id που λαμβάνει από τον IdP. Το client_id είναι μια ξεχωριστή μεταβλητή για κάθε RP. Τέλος, στην ευπάθεια Replay, η οποία είναι παρόμοια με την Session Handling, δεν πραγματοποιείται σωστός έλεγχος στη λήξη του access_token. [17]

Στις ευπάθειες τύπου Injection - manipulation, αν ο RP δεν ελέγχει τα δεδομένα που λαμβάνει από τον IdP, και ο IdP είναι μολυσμένος, τότε ο επιτιθέμενος μπορεί να εκμεταλλευτεί την κατάσταση και να εισάγει κακόβουλο εκτελέσιμο κώδικα στον RP. Αυτός ο τύπος επίθεσης ονομάζεται XSS (Cross-Site Scripting) & SQL Injection.[16] Σε άλλη περίπτωση, όταν παρακάμπτεται ή απενεργοποιείται ο έλεγχος κρυπτογράφησης της υπογραφής του token, το περιεχόμενο του μπορεί να παραμετροποιηθεί χωρίς να θεωρηθεί άκυρο, δημιουργώντας μεγάλο κενό ασφάλειας. Επιθέσεις τέτοιου τύπου ονομάζονται Signature Bypass. [17]

Στην τελευταία κατηγορία επιθέσεων του OIDC, δηλαδή στην κατηγορία που αφορά το δίκτυο και την υποδομή, περιέχονται οι επιθέσεις Server-Side Request Forgery (SSRF), Transport Layer Security και Third-Party Resources. Στο πρώτο είδος επίθεσης, ένας μολυσμένος RP ή IdP στέλνει αιτήματα σε άλλους διακομιστές ή εξυπηρετητές (hosts) εντός του δικτύου. Αποτέλεσμα αυτού είναι, είτε να καλεί υπηρεσίες ή να στέλνει πολλά αιτήματα προς έναν διακομιστή προκαλώντας Denial of Service attacks. Κενά ασφαλείας δημιουργούνται και με την μη χρήση HTTPS, Strict Transport Security και Public Key Pinning, σε επίπεδο μεταφοράς (Transport Layer Security). Τέλος, όταν οι σελίδες των RP και IdP περιέχουν πόρους τρίτων (πχ scrpits, εικόνες, css κλπ), εκτίθενται ευαίσθητες πληροφορίες, οι οποίες μπορεί να κλαπούν.[16]

2.2.3 Security Assertion Markup Language (SAML)

Το SAML είναι ένα XML-based πρωτόκολλο, το οποίο βασίζεται στην ανταλλαγή μηνυμάτων μεταξύ του IdP και του SP για την αυθεντικοποίηση χρηστών[18] [3]. Είναι ανοιχτού κώδικα, επεκτάσιμο και δεν απαιτεί την χρήση cookies για την επικοινωνία μεταξύ χρήστη και server, σε αντίθεση με την επικοινωνία μεταξύ RP και IdP, η οποία παρότι μπορεί να γίνει και χωρίς cookies, συνήθως τα χρησιμοποιεί. Η ιδιότητα αυτή του πρωτοκόλλου βασίζεται στις ανακατευθύνσεις του browser και στο HTTP.[18]

Έχει όμοιους ρόλους με τα παραπάνω πρωτόκολλα, διαφέρει όμως στην ροή. Η διαδικασία που ακολουθεί ξεκινάει με την αυθεντικοποίηση του χρήστη από τον IdP, τον έλεγχο του εάν ο browser είναι αυτός από τον οποίο έχει πραγματοποιηθεί η αυθεντικοποίηση, και εάν επαληθευτεί, ο χρήστης δρομολογείται στον RP. Η παραπάνω διαδικασία ελέγχου ονομάζεται και user tracking. Στην συνέχεια ο RP ζητάει από τον IdP το SAML Response/Assertion, το ελέγχει και εάν είναι σωστό, τότε επιτρέπει στον χρήστη την πρόσβαση στους πόρους του.[18] [3] Το SAML assertion (ισχυρισμός) είναι μια ψηφιακή ταυτότητα, η οποία μπορεί να είναι κρυπτογραφημένη, ανάλογα την πολιτική ασφαλείας, και περιέχει πληροφορίες για τον χρήστη, την διάρκεια ζωής του assertion και άλλες πληροφορίες για την πολιτική ασφαλείας. Προωθείται από τον IdP στον RP. [3]

Το SAML όπως και τα υπόλοιπα συστήματα που παρουσιάστηκαν ήδη, έχει ορισμένες γνωστές ευπάθειες που το κάνουν ευάλωτο σε συγκεκριμένα είδη επιθέσεων. Σε επίπεδο SSL/TLS Binding, η χρήση μη ασφαλών αλγορίθμων ή εκδόσεων μπορεί να δώσει την δυνατότητα σε έναν πιθανό επιτιθέμενο, να υποκλέψει ή να τροποποιήσει SAML μηνύματα. Για να αποφευχθεί αυτό το κενό ασφαλείας χρειάζεται ιδιαίτερη προσοχή κατά την υλοποίηση του https, ssl/ tls binding. [18]

Ο τρόπος λειτουργίας του SAML, το κάνει ευάλωτο σε μια σειρά επιθέσεων. Ένας τύπος επίθεσης που μπορεί να δεχθεί είναι το Connection Hijacking σε συνδυασμό με το Replay Attack, στα οποία ο επιτιθέμενος δεν γνωρίζει, αλλά μπορεί να χρησιμοποιήσει τα δεδομένα αυθεντικοποίησης ενός χρήστη για να στείλει περεταίρω αιτήματα. Την πρόσβαση αυτή αποκτά παρακολουθώντας την επικοινωνία μεταξύ του IdP και του RP, και λόγω λάθος ταυτοποίησης δεδομένων που στέλνει και λαμβάνει ο IdP από και προς τον RP.[18] Επιπλέον, πιθανό είναι να δεχθεί επίθεση Man-in-the-Middle Attack, στην οποία ο επιτιθέμενος τοποθετηθεί τον εαυτό του ανάμεσα στον χρήστη και στον IdP, με χρήση DNS Spoofing. Με αυτόν τον τρόπο, μπορεί να υποδυθεί τον IdP και έτσι να υποκλέψει ή να τροποιήσει τα

δεδομένα που ανταλλάσσονται. Στην HTTP Referrer Attack, ο επιτιθέμενος έχει την δυνατότητα να εκμεταλλευτεί συγκεκριμένα artifacts για μη εξουσιοδοτημένη πρόσβαση. Αυτό συμβαίνει σε περίπτωση αποτυχίας της ανακατεύθυνσης του χρήστη από τον IdP στον RP. Ο IdP επιστρέφει στον χρήστη ένα HTTP Referrer header, το οποίο περιέχει το SAML artifact και το οποίο μπορεί να αποκαλύψει σε τρίτους τις τιμές αυτές που δεν έχουν χρησιμοποιηθεί ακόμα. [18] [3]

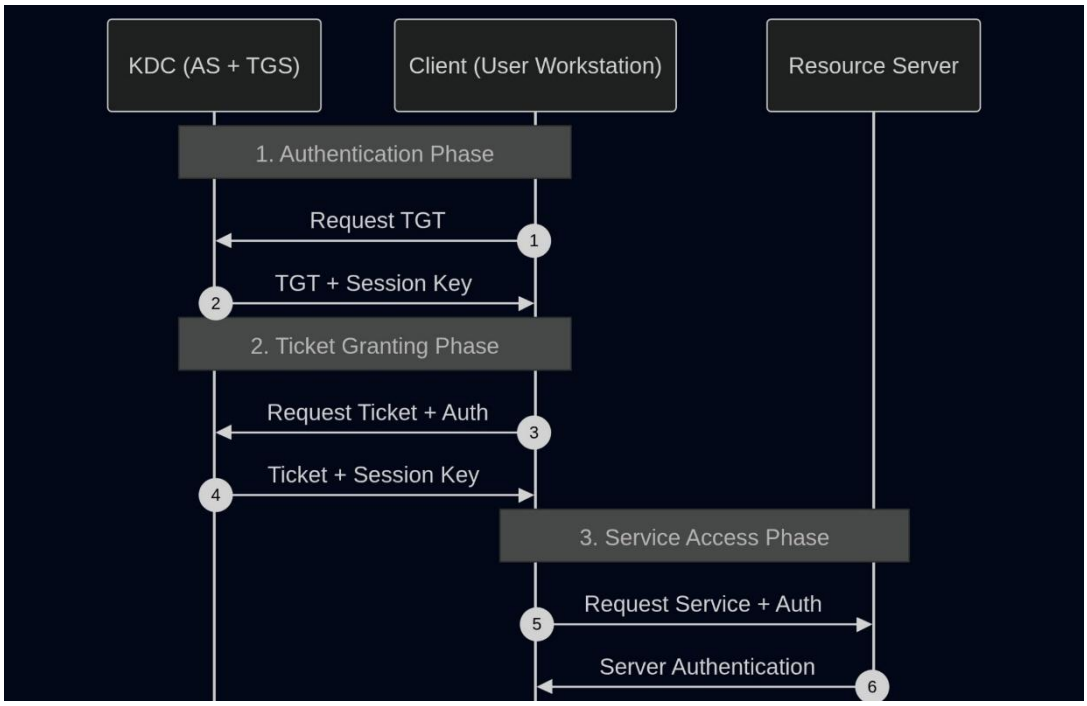
Μια ακόμα επίθεση που μπορεί να δεχθεί είναι η Denial-of-Service (DoS) Attack. Αυτή συμβαίνει όταν ο IdP δεν κάνει σωστό έλεγχο και διαχείριση των αιτημάτων που λαμβάνει, με αποτέλεσμα ο επιτιθέμενος, εκμεταλλευόμενος το redirect από τον RP στον IdP, να μπορεί να στέλνει πολλά αιτήματα στον IdP, ώστε να τον καταστήσει μη λειτουργικό. Τέλος, πιθανή είναι η επίθεση Cross-Site Scripting (XSS) Attack, η οποία σχετίζεται με εισαγωγή κακόβουλου κώδικα, είτε μέσω του SAML Response, είτε μέσω του πεδίου RelayState. Ο επιτιθέμενος το επιτυγχάνει αυτό όταν ο έλεγχος των δεδομένων που λαμβάνει ο RP από τον IdP δεν γίνεται σωστά. Οι μεταβλητές SAML Response και το πεδίο RelayState προωθούνται από τον IdP προς τον RP μέσω του URL. [3]

2.2.4 Kerberos

Το διαλειτουργικό σύστημα Kerberos είναι ένα πρωτόκολλο αυθεντικοποίησης όπου αναπτύχθηκε από το Ινστιτούτο Τεχνολογίας της Μασαχουσέτης (Massachusetts Institute of Technology - MIT). Έχει σχεδιαστεί για την αυθεντικοποίηση εντός κλειστών δικτύων, σε τοπικά δηλαδή domains, και βασίζεται στη λογική των tickets, για την επικοινωνία μεταξύ χρηστών και διακομιστών που ανήκουν στο ίδιο δίκτυο.[19]

Τα βασικά στοιχεία που αποτελούν τον Kerberos, είναι το Key Distribution Center (KDC), και ο Service Server(SS). Το KDC απαρτίζεται από 2 υπηρεσίες, τον Authentication Server (AS), ο οποίος επαληθεύει τα στοιχεία σύνδεσης που έχει δώσει ο χρήστης και εάν είναι σωστά εκδίδει το Ticket Granting Ticket (TGT), και τον Ticket Granting Server (TGS), ο οποίος λαμβάνει το TGT και το χρησιμοποιεί για να εκδώσει το Service Ticket (ST), το οποίο δίνει πρόσβαση σε συγκεκριμένες υπηρεσίες. Στην συνέχεια, ο Service Server(SS), λαμβάνει το ST από τον χρήστη και του επιστρέφει τα δεδομένα που έχει δικαίωμα να δει. [19] [20]

Η παρακάτω απεικόνιση έχει δημιουργηθεί για τις ανάγκες αυτής της πτυχιακής εργασίας και δείχνει τους διακομιστές που συμμετέχουν στην διαδικασία αυθεντικοποίησης χρηστών στο πρωτοκόλλο Kerberos.



Εικόνα 1: Αυθεντικοποίηση Kerberos

Στη συνέχεια, παρουσιάζονται ορισμένες καταγεγραμμένες κατηγορίες επιθέσεων, έναντι του πρωτοκόλλου αυτού. Στην επίθεση Golden Ticket Attack, ο επιτιθέμενος αποκτά την κρυπτογράφηση του λογαριασμού krbtgt από τον Domain Controller και μέσω αυτού δημιουργεί πλαστά TGTs, ώστε να έχει πρόσβαση σε οποιαδήποτε υπηρεσία του δικτύου. Σε γενικές γραμμές είναι δύσκολο να ανιχνευθεί, ωστόσο η ισχυρή κρυπτογράφηση του TGT, ο έλεγχος της διάρκειας ζωής των TGTs, αλλά και ο έλεγχος των αρχείων καταγραφής (logs) μπορούν να βοηθήσουν σημαντικά στον εντοπισμό μιας τέτοιου είδους επίθεσης. Στην Silver Ticket Attack, ο επιτιθέμενος αποκτά πρόσβαση στο κλειδί μιας υπηρεσίας και δημιουργεί ένα δικό του ticket για την υπηρεσία αυτή. Η ανίχνευση της επίθεσης είναι δύσκολη, ωστόσο η διατήρηση αρχείων καταγραφής (logs) και η παρακολούθηση των κινήσεων των χρηστών μπορεί να βοηθήσει στον εντοπισμό μιας τέτοιου είδους επίθεσης. [19]

Στην Skeleton Ticket Attack - Skeleton Key Attack, χρησιμοποιείται ένα κακόβουλο λογισμικό το οποίο δημιουργεί έναν καθολικό κωδικό πρόσβασης στην μνήμη Local Security Authority Subsystem Service - LSASS, το οποίο επιτρέπει την αυθεντικοποίηση οποιουδήποτε χρήστη στο δίκτυο, χωρίς να γνωρίζει τον ενεργό κωδικό πρόσβασης του. Ο συνδυαστικός έλεγχος των αρχείων καταγραφής (logs), η παρακολούθηση των ρυθμίσεων κρυπτογράφησης του Kerberos και η ενεργοποίηση της επιλογής monitoring, του αρχείου Lsass.exe, μπορούν να

βοηθήσουν στον εντοπισμό αυτού του τύπου επιθέσεων. Τέλος, στην AdminSDHolder Attack, ο επιτιθέμενος χρησιμοποιεί το αντικείμενο AdminSDHolder, του Active Directory, το οποίο δημιουργεί καλούπια για τα δικαιώματα των χρηστών και των ομάδων που έχουν δικαιώματα διαχειριστή στο δίκτυο. Με αυτόν τον τρόπο προσθέσει αυθαίρετα χρήστες με δικαιώματα διαχειριστή. Για τον εντοπισμό σχετικών επιθέσεων, χρήσιμη είναι η συχνή παρακολούθηση του AdminSDHolder ACL για πιθανές αλλαγές στις λίστες των χρηστών και των ομάδων που έχουν δικαιώματα διαχειριστή. [19]

2.3 Σύγκριση πρωτοκόλλων

Στην προηγούμενη ενότητα, με τίτλο διαλειτουργικά συστήματα αυθεντικοποίησης (2.2), έγινε αναλυτική περιγραφή των πρωτοκόλλων OAuth2.0, OpenID Connect, SAML και Kerberos, ενώ σε αυτήν γίνεται σύγκριση τους με βάση ορισμένα κριτήρια. Για την ευκολότερη ανάδειξη ομοιοτήτων και διαφορών, πλεονεκτημάτων και μειονεκτημάτων, αλλά και ορισμένων από τα CVEs των τελευταίων πέντε χρόνων για κάθε πρωτόκολλο, επιλέχθηκε η κατασκευή ενός πίνακα, ο οποίος φαίνεται παρακάτω. Οι πληροφορίες που περιέχει αντλήθηκαν από σχετική βιβλιογραφία και συνοψίστηκαν χάριν ευκολίας, ενώ τα δεδομένα για τα CVEs, αντλήθηκαν από το API που προσφέρει η opencve.io και αφορούν αποκλειστικά τις καταγεγραμμένες ευπάθειες των πρωτοκόλλων, και όχι των τεχνολογιών που χρησιμοποιούνται γύρω από αυτά, έως και τον Νοέμβριο του 2025.

Πίνακας 2 Σύγκριση πρωτοκόλλων

Πρωτόκολλο	**Τύπος**	**Tokens / Format**	**Τυπικές Χρήσεις**	**Πλεονεκτήματα**	**Μειονεκτήματα / Ευπάθειες**	**Καταγεγραμμένες ευπάθειες τα τελευταία 5 χρόνια**
OAuth 2.0	Εξουσιοδότηση (όχι αυθεντικοποίηση από μόνο του) [1] [2] [11]	Access Tokens (JWT ή opaque) [5] [2] [11]	Web & Mobile apps, API-to-API [1] [2]	Ευρεία χρήση [1] [2], υποστήριξη πολλών flows (Authorization Code + PKCE, Client Credentials) [11] [14] [13], κατάλληλο για API access delegation	Δεν επαληθεύει ποιος είναι ο χρήστης χωρίς OIDC [16], ροές όπως Implicit & ROPC παρωχημένες [14], ευπάθειες σε redirect URI, code reuse, token leakage αν δεν γίνει σωστό hardening [15]	Σύνολο 365, με την OAuth2 Flow Parameters 36.7% και το 12.6% του συνολικού ορίζονται Critical

OpenID Connect (OIDC)	Αυθεντικοποίηση + Εξουσιοδότηση (βασισμένο σε OAuth 2.0) [16] [13] [3]	ID Token (JWT), Access Token (JWT ή opaque) [16] [13]	Social logins, federated identity σε web & mobile apps [16]	ID Token υπογεγραμμένο ή και κρυπτογραφημένο, υποστήριξη Discovery & Dynamic Client Registration, ευρεία υποστήριξη από μεγάλα IdPs [16]	Απαιτεί αυστηρή υλοποίηση state/nonce/PKCE [16], ευπάθειες σε mix-up attacks, token leakage, CSRF, XSS [16] [17]	Σύνολο 200, με XSS 12.5% και το 8.0% του συνολικού ορίζονται Critical
SAML	Αυθεντικοποίηση (Web SSO) [18] [3]	SAML Assertions (XML) [18] [3]	Enterprise SSO, B2B federation [18] [3]	Ανεξάρτητο από cookies, ισχυρή υποστήριξη σε enterprise εφαρμογές [18], επεκτάσιμο, mature standard [3]	Μεγάλου μεγέθους μηνύματα (XML), πιο πολύπλοκο στην υλοποίηση [18], ευπάθειες σε XML Signature Wrapping, replay, HTTP Referrer leaks [18] [3]	Σύνολο 10712, Execute Code Command Injection 23.9% και το 4.4% του συνολικού ορίζονται Critical
Kerberos	Αυθεντικοποίηση (κυρίως σε κλειστά δίκτυα / AD) [19] [20]	Tickets (TGT, Service Ticket) [19]	Windows Active Directory, intranet, enterprise apps [19]	Πολύ αποδοτικό σε τοπικά δίκτυα, δεν στέλνει password σε plaintext, single sign-on στο domain [19], ισχυρή κρυπτογράφηση	Χρειάζεται συγχρονισμό ώρας, περιορισμένο σε εσωτερικά δίκτυα, επιθέσεις Golden/Silver Ticket, Skeleton Key, απαιτεί σωστή διαχείριση service accounts και κλειδιών [19]	Σύνολο 4442, Memory Overflow 33.5% και το 5.0% του συνολικού ορίζονται Critical

Όπως φαίνεται και από τον παραπάνω πίνακα το OAuth 2.0 και το OIDC χρησιμοποιούν ελαφριά tokens όπως είναι το JWT, κάνοντας τα να κυριαρχούν στο σύγχρονο διαδίκτυο. Ωστόσο, απαιτούν προσοχή κατά την υλοποίηση για την αποφυγή ευπαθειών όπως XSS και token leakage. Σε αντίθεση, το πρωτόκολλο SAML, φαίνεται να είναι η πιο συχνή επιλογή για επιχειρησιακά περιβάλλοντα (Enterprise SSO/B2B), καθώς δίνει έμφαση στον ισχυρή υποστήριξη χωρίς να κάνει χρήση από cookies. Εμφανίζει όμως σημαντικά περισσότερες καταγεγραμμένες ευπάθειες, λόγω της πολυπλοκότητας του XML format, σε σχέση με τα υπόλοιπα πρωτόκολλα. Τέλος, το Kerberos διαφοροποιείται από τα υπόλοιπα πρωτόκολλα καθώς είναι σχεδιασμένο για κλειστά, εσωτερικά δίκτυα Windows AD με υψηλή αποδοτικότητα μέσω Tickets. Σε μερικές βέβαια περιπτώσεις, η σχεδίαση αυτή καθώς και η

εξάρτηση από τον συγχρονισμό ώρας μεταξύ διακομιστών και χρήστη, το καθιστούν ακατάλληλο για ευρεία χρήση στο δημόσιο διαδίκτυο.

Κεφάλαιο 3ο: Συγκριτική αξιολόγηση συστημάτων αυθεντικοποίησης

Μία από τις πλέον διαδεδομένες λύσεις της Microsoft για διαλειτουργική αυθεντικοποίηση είναι το ADFS, το οποίο υποστηρίζει Ενιαία Σύνδεση (SSO – Single Sign-On) και Identity Federation σε πολλαπλές εφαρμογές και domains [21]. Μια άλλη ιδιαίτερα δημοφιλής λύση, είναι η χρήση του συστήματος ανοιχτού κώδικα Keycloak [22], το οποίο υποστηρίζει πρωτόκολλα όπως το OpenID Connect και SAML 2.0, ενώ προσφέρει REST API για την πλήρη διαχείριση χρηστών και ρόλων.

Μια σύντομη παρουσίαση των βασικών χαρακτηριστικών του ADFS, Keycloak και Custom API γίνεται με τη βοήθεια του Πίνακα 3. Περισσότερες πληροφορίες για τα εργαλεία που χρησιμοποιήθηκαν και την τεχνική υλοποίηση, παρουσιάζονται στα κεφάλαια 5 και 6 αντίστοιχα.

Σημαντικό είναι να αναφερθεί πως τα συγκεκριμένα συστήματα επιλέχθηκαν ως μέτρα σύγκρισης για το custom API που αναπτύχθηκε, καθώς αποτελούν μερικά από τα πιο πολυχρησιμοποιούμενα συστήματα αυθεντικοποίησης. Το ADFS αποτελεί την πρόταση της Microsoft, στην οποία βασίζεται και ο διακομιστή της παρούσας πτυχιακής εργασίας, προσφέροντας απόλυτη συμβατότητα με το AD. Ωστόσο το κόστος του είναι μια μεταβλητή που σε μερικές περιπτώσεις το καθιστούν επιλογή χαμηλότερη στην λίστα προτίμησης. Από την άλλη, το Keycloak επιλέχθηκε ως η κορυφαία εναλλακτική ανοιχτού κώδικα, προσφέροντας υποστήριξη πολλαπλών πρωτοκόλλων αυθεντικοποίησης και δωρεάν χρήση. Τέλος, το custom API είναι μια ακόμη επιλογή και στη συγκεκριμένη περίπτωση αναπτύχθηκε με στόχο το συνδυασμό των θετικών και των δύο προαναφερθέντων συστημάτων αυθεντικοποίησης, καθιστώντας το ως μια ενδιάμεση λύση.

Πίνακας 3: Σύγκριση συστημάτων αυθεντικοποίησης

Κριτήριο	**ADFS**	**Keycloak**	**Custom API**
SSO	Πλήρης υποστήριξη	Πλήρης υποστήριξη	Πλήρης υποστήριξη
Ομοσπονδιακή Αυθεντικοποίηση	Ναι (AD trust + claims rules)	Ναι (IDP/SP SAML, OIDC)	Όχι

2FA	Μερική (μέσω policies ή MFA)	Ναι (SMS, TOTP, WebAuth)	Ναι (TOTP)
Διαχείριση συστήματος	Σύνθετη	GUI/CLI, πιο εύκολη	Custom εργαλεία, logs
Απαιτήσεις Υποδομής	Υψηλές (ADFS, certs, IIS)	Java-based, container-ready	Ελαφρύ – ISS + .NET + AD
Ενσωμάτωση με APIs	Πολύ καλή (claims-based auth)	Native για REST APIs	JWT με ρυθμιζόμενα roles
Επεκτασιμότητα	Μέτρια	Υψηλή	Υψηλή (μελλοντικά OIDC-ready)

Κεφάλαιο 4ο: Τεχνολογικό περιβάλλον πειραματικής υλοποίησης

Σε αυτή την ενότητα παρουσιάζονται τα βασικά πρωτόκολλα λειτουργίας και οι υφιστάμενες τεχνολογίες που χρησιμοποιήθηκαν για το τεχνικό κομμάτι της πτυχιακής. Αναλυτικότερα, θα παρουσιαστούν πληροφορίες για τον διακομιστή και το API που δημιουργήθηκε.

4.1 Πρωτόκολλα διακομιστών windows

Ως Active Directory (AD) ορίζεται ένα σύστημα καταλόγου της Microsoft, που σαν κύριο στόχο έχει να βοηθήσει εταιρίες, οργανισμούς ή και όποιον άλλο το χρησιμοποιεί, να οργανώσει, να διαχειριστεί και να ελέγχει αποτελεσματικά τους πόρους του δικτύου του. Αναλυτικότερα, στηρίζεται στην ύπαρξη ενός διαχειριστή (Administrator), ο οποίος έχει πρόσβαση στη κεντρική βάση δεδομένων, δηλαδή σε όλα τα αρχεία και τις πληροφορίες του δικτύου (χρήστες, υπολογιστές, ομάδες κλπ). Αυτός λοιπόν, έχει την δυνατότητα δημιουργίας και διαχείρισης χρηστών και ομάδων, οριοθέτησης της προσβασιμότητας του κάθε χρήστη, ανάλογα με τις ανάγκες της θέσης του, παρακολούθησης και καταγραφής όλων των δραστηριοτήτων των χρηστών εντός του δικτύου, χρησιμοποιώντας εργαλεία παρακολούθησης (monitoring) του δικτύου και των πόρων του. Στην πράξη όλα τα παραπάνω, αλλά και η εφαρμογή πολιτικών ασφαλείας και ελέγχου πρόσβασης, μερικές φορές με χρήση του πρωτοκόλλου Kerberos, όπως αναφέρθηκε παραπάνω [23], δεν γίνονται από ένα άτομο, τον διαχειριστή, αλλά από μια ομάδα Διαχειριστών Συστημάτων Πληροφορικής, τους αποκαλούμενους IT (Information Technology). [24]

Μερικοί από τους βασικούς όρους των AD είναι ο Τομέας (Domain), το Δέντρο (Tree), το Δάσος (Forest) και οι Οργανωτικές μονάδες (Organizational Units - OUs). Ο ευκολότερος τρόπος να παρουσιαστούν και να γίνει ξεκάθαρη η σχέση μεταξύ τους είναι μέσω ενός παραδείγματος. Έστω μια εταιρία με όνομα “ silken touch ” και domain silken-touch.gr. Όλοι οι εργαζόμενοι, οι υπολογιστές, τυχόν εκτυπωτές και άλλα μηχανήματα είναι εγγεγραμμένα στο παραπάνω domain, το οποίο σημαίνει πως όλοι αυτοί υπακούουν σε ίδιους κανόνες και πολιτικές. Ανάλογα με το τμήμα της εταιρείας όμως, είναι άλλες οι εργασίες που πρέπει να γίνουν και κατ’ επέκταση άλλες οι ανάγκες πρόσβασης σε αρχεία και πληροφορίες. Για να γίνει αυτός ο διαμοιρασμός δικαιωμάτων, ενώ υπάρχει ένα κεντρικό root domain (silken-touch.gr), κάθε τμήμα έχει το δικό του child domain (sales.silken-touch.gr, hr.silken-touch.gr, developers.silken-touch.gr κα.) το οποίο έχει άλλα δικαιώματα πρόσβασης και αποτελεί μια

οργανωτική ομάδα (OU). Το root domain και τα child domains αποτελούν ένα δέντρο. Η εταιρεία αυτή μπορεί να δραστηριοποιείται και εκτός Ελλάδας, επομένως να έχει και ένα δεύτερο domain (silken-touch.com) με ίδια όμως δομή, άρα με παρόμοιο σκελετό tree. Τα δύο αυτά trees αποτελούν ένα forest, μεταξύ των οποίων υπάρχει εμπιστοσύνη και έτσι επιτρέπεται η ανταλλαγή πόρων και πληροφοριών μεταξύ τους. [23] [24]

Το συγκεκριμένο σύστημα καταλόγου (AD) έχει ευπάθειες όπως και όλες οι άλλες τεχνολογίες, μιας και όλα φτιάχνονται από τον άνθρωπο. Μερικές από αυτές προκύπτουν από κενά ασφαλείας, ενώ μερικές άλλες από λάθος ή μη επαρκείς ρυθμίσεις που σαν αποτέλεσμα έχουν να εκθέτουν το δίκτυο. Για παράδειγμα, πρόβλημα μπορεί να δημιουργηθεί όταν μεμονωμένοι χρήστες ή και ολόκληρες ομάδες έχουν παραπάνω δικαιώματα από αυτά που χρειάζονται. Αυτό μπορεί να συμβεί, όταν πέραν της ομάδας των IT, που είναι Domain Admins, οριστούν και άλλα άτομα ως Domain Admins (πχ. κάποιος από της ομάδα των Developers ή και ολόκληρη η ομάδα). Οποιοδήποτε άτομο έχει αυτή την ιδιότητα, έχει απόλυτη εξουσία στον διακομιστή και μπορεί να παραμετροποιήσει πολλές ρυθμίσεις σε αυτόν, με αποτέλεσμα να πρέπει η ιδιότητα αυτή να δίνεται με πολλή προσοχή και μόνο στα άτομα όπου πραγματικά χρειάζεται. Κενό ασφαλείας μπορεί να προκύψει και από τη χρήση μη ισχυρών κωδικών. Αυτό μπορεί να σημαίνει χρήση μικρών σε αριθμών χαρακτήρων κωδικών πρόσβασης, μη ενημέρωση τους ή χρήση του ίδιου κωδικού για το προφίλ του Administrator σε πολλούς ή και σε όλους τους υπολογιστές. Τέλος, ευπάθεια αποτελεί η συνειδητή ή μη, χρήση παλαιών εκδόσεων λογισμικού του διακομιστή ή των εφαρμογών. [24]

Οι πιθανοί τύποι επιθέσεων που μπορεί να πραγματοποιηθούν, είναι brute force για την τυχαία εύρεση κωδικών πρόσβασης μέσω πολλαπλών δοκιμών συνηθισμένων κωδικών, phishing για την απόσπαση κωδικών πρόσβασης μέσω εξαπάτησης των χρηστών του δικτύου, malware δηλαδή χρήση κακόβουλου λογισμικού με σκοπό την παραβίαση του δικτύου και των πόρων του και social engineering για την εξαπάτηση χρηστών, με απώτερο σκοπό την απόσπαση ευαίσθητων πληροφοριών σχετικά με το δίκτυο και τους πόρους του.

Για την αποφυγή επιθέσεων σαν τις παραπάνω, χρήσιμο είναι η ομάδα των IT, να δημιουργήσει και να εφαρμόσει πολιτικές ασφαλείας που αφορούν τα δικαιώματα χρηστών και την προσβασιμότητα σε πόρους του δικτύου, αλλά και να εκπαιδεύσει τους χρήστες και γενικότερα όλους τους εργαζόμενους της εταιρείας σχετικά με τα παραπάνω. Στα πλαίσια αυτής της ενημέρωσης, οι IT μπορούν να δώσουν οδηγίες που πρέπει να ακολουθηθούν σε διάφορα θεωρητικά σενάρια, όπως πχ. σε περίπτωση που οι εργαζόμενοι λάβουν e-mail από

κάποιον εκτός εταιρείας ή από οποιοδήποτε μη εταιρικό e-mail, το οποίο ζητάει εμπιστευτικές πληροφορίες, να μην απαντήσουν. Επιπλέον, καλό είναι η ομάδα των IT να χρησιμοποιεί εργαλεία παρακολούθησης και καταγραφής των κινήσεων των χρηστών του δικτύου, με χρήση αρχείων καταγραφών (logs) και εργαλεία ανάλυσης αυτών, ώστε ακόμα και αν υπάρξει κάποια διαρροή στοιχείων, αυτή να γίνει εύκολα και γρήγορα αντιληπτή ή ακόμα και αν συμβεί κάποιο μη σκόπιμο λάθος, να βρεθεί άμεσα ένας τρόπος διόρθωσης του. Τέλος, αν και φαντάζει πολύ προφανές, είναι πολύ σημαντικό να γίνονται συχνές ενημερώσεις και αναβαθμίσεις του λειτουργικού συστήματος και των εφαρμογών που υπάρχουν στο δίκτυο και να χρησιμοποιούνται εργαλεία προστασίας του δικτύου από επιθέσεις και κακόβουλα λογισμικά, όπως τα firewalls και τα antivirus.

Ένας εντελώς διαφορετικός τρόπος ενίσχυσης της ασφάλειας μιας εφαρμογής, είναι η χρήση του IIS, το οποίο είναι μία ευρέως χρησιμοποιούμενη πλατφόρμα για την φιλοξενία web APIs και ιστοσελίδων. Αυτό προσφέρει μία ευρεία γκάμα από ενσωματωμένα εργαλεία, χρήσιμα για την επίτευξη του στόχου αυτού. Περιλαμβάνει μηχανισμούς αυθεντικοποίησης, κάνει έλεγχο πρόσβασης, κρυπτογράφηση σύνδεσης μέσω SSL και υποστηρίζει ψηφιακές πιστοποιήσεις. Ωστόσο, για την διατήρηση ενός ασφαλούς περιβάλλοντος και την αποφυγή πιθανών κενών ασφαλείας, απαραίτητη είναι η σωστή ρύθμιση των προαναφερθέντων εργαλείων, καθώς και η συνεχής παρακολούθηση και η ενημέρωση του. [25]

4.2 Απαιτήσεις συστήματος

Ο server που χρησιμοποιήθηκε φιλοξενείται σε VM (virtual machine) και έχει τα εξής χαρακτηριστικά, τα οποία είναι αναγκαία για τη σωστή λειτουργία του. Αρχικά, σαν λειτουργικό σύστημα έχει Windows Server 2022 Standard Evaluation και οι υπηρεσίες που διαθέτει είναι, το Active Directory Domain Services (AD DS) για τη διαχείριση χρηστών και ομάδων, το IIS για τη φιλοξενία του API, το SQLite για τη διαχείριση της βάσης δεδομένων και τέλος τον DNS Server για την επίλυση ονομάτων στο δίκτυο. Με κάθε εκκίνηση του server, γίνεται αυτόματη εκκίνηση των IIS, DNS και AD DS. Οι εφαρμογές που χρησιμοποιεί είναι η .NET Hosting Bundle 8.0.15 για την εκτέλεση του API που είναι αναπτυγμένο σε ASP.NET Core 8 και η DB Browser for SQLite για την διαχείριση της βάσης του API.

Τα τεχνικά χαρακτηριστικά του διακομιστή είναι, επεξεργαστής τεσσάρων πυρήνων, μνήμη RAM 8 GB , αποθηκευτικός χώρος 60 GB SSD, από τα οποία χρησιμοποιούνται περίπου τα 22 GB. Σε δοκιμή αποστολής 200 αιτημάτων με καθυστέρηση 1200ms ανά αίτημα, το API χρησιμοποίησε μόλις το 2.08% της διαθέσιμης RAM, γεγονός που υποδηλώνει χαμηλή

κατανάλωση πόρων. Η σύνδεση στο δίκτυο πραγματοποιείται μέσω Ethernet, ενώ στις ρυθμίσεις του VM, επιλέχθηκε επιλεχθεί η λειτουργία “Bridget Adapter”, Αυτό σημαίνει πως ο διακομιστής λαμβάνει IP διεύθυνση από το ίδιο δίκτυο του host και είναι προσβάσιμος από άλλες συσκευές στο δίκτυο σαν ξεχωριστή συσκευή LAN.

4.3 Τεχνολογίες που χρησιμοποιήθηκαν για το API

Στο τεχνολογικό κομμάτι της πτυχιακής, επιλέχθηκε η χρήση ενός αλγορίθμου δημιουργίας token, ώστε ο χρήστης να μη χρειάζεται να προωθεί κάθε φορά τα στοιχεία σύνδεσης του στο API, εκθέτοντας τα στο δίκτυο. Για την εν λόγω διαδικασία, υπάρχουν πολλοί αλγόριθμοι, ο καθένας από τους οποίους έχει τα προτερήματα του ανάλογα τα σημεία στα οποία οι προγραμματιστές θέλουν να δώσουν έμφαση (πχ. στην ταχύτητα, το μέγεθος κλειδιού, τη χρήση υπάρχοντος γνωστού αλγορίθμου ή τη χρήση μοναδικού αλγορίθμου κα). Μετά από εξέταση των διαθέσιμων επιλογών, επιλέχθηκε ο αλγόριθμος Json Web Tokens (JWT), λόγω της ευρείας χρήση του στον κλάδο των εφαρμογών ιστοσελίδων και της δυνατότητας επιλογής αλγορίθμου υπογραφής που προσφέρει. Μερικοί από τους αλγόριθμους είναι ο HMAC, ο RSA, ο ECDSA, οι οποίοι είναι ευρέως γνωστοί για την αξιοπιστία τους.

Για να γίνει πιο έντονα κατανοητό γιατί επιλέχθηκε έναντι άλλων, παρακάτω γίνεται η σύγκριση του με το PASETO, το οποίο είναι νεότερο πρότυπο και φαίνεται να υπερτερεί στον τομέα της ασφάλειας. Ένα ακόμα θετικό του είναι ο μεγαλύτερος μέσος όρος χαρακτήρων σε κάθε token. Συγκεκριμένα, ο μέσος όρος για τον αλγόριθμο αυτό είναι 320, σε αντίθεση με του JWT που είναι 186. Τέλος, ο κώδικας κρυπτογράφησης του δεν είναι γνωστός στο κοινό, το οποίο αποτελεί και προτέρημα και μειονέκτημα. Παρά τα χαρακτηριστικά του αυτά, υστερεί στην απόδοση ταχύτητας έναντι του JWT, το οποίο χρειάζεται λιγότερο χρόνο για την δημιουργία και την αποστολή του token. [9]

Από τους αλγορίθμους υπογραφής που προσφέρονται από το JWT, επιλέχθηκε ο HMAC-SHA256 (HmacSha256Signature). Σε σύγκριση με τους υπόλοιπους, δεν απαιτεί την χρήση δημόσιων και ιδιωτικών κλειδιών, μιας και χρησιμοποιεί ένα και μόνο κλειδί για την υπογραφή και την επικύρωση του μηνύματος από και προς τις δύο πλευρές (αποστολέα και παραλήπτη). Ενισχύει την ασφάλεια, όχι όμως στον βαθμό που την ενισχύει ο RSA, και είναι απλός στην χρήση, γεγονός που τον καθιστά ιδανικό για μικρές και μεσαίες εφαρμογές, όπως αυτή που δημιουργήθηκε στα πλαίσια αυτής της πτυχιακής. [26]

Η μελέτη περίπτωσης της εργασίας αυτής, αφορά την ανάπτυξη ενός προσαρμοσμένου συστήματος αυθεντικοποίησης το οποίο είναι βασισμένο στο AD, και προσφέρεται από τη Microsoft μέσω του Windows Server.

Κεφάλαιο 5ο: Εφαρμογή που αναπτύχθηκε

5.1 Χαρακτηριστικά της Υλοποίησης

Σε αυτή την ενότητα θα γίνει αναλυτική περιγραφή της τελικής μορφής της υλοποίησης. Εννοείται πως η μορφή αυτή προέκυψε μετά από πολλές αλλαγές και διορθώσεις, οι οποίες δε χρειάζεται να αναφερθούν. Όλο το τεχνικό κομμάτι της πτυχιακής υπάρχει ανεβασμένο στο github, ο σύνδεσμο για το οποίο υπάρχει στο παράρτημα.

5.1.1 Επισκόπηση αρχιτεκτονικής

Κατά τη δημιουργία του API και προσπαθώντας να δοθεί έμφαση στην ασφάλεια, προσοχή δόθηκε σε μερικά σημεία, τα οποία αποτελούν τα κύρια υποσυστήματα της πτυχιακής. Αναλυτικότερα, αυτά είναι οι Controllers και τα Services του API, οι δύο ιστοσελίδες που στόχο έχουν να δείξουν την λειτουργικότητα του SSO, και η οργάνωση της βάσης και των αρχείων (καταγραφών), το καθένα από τα οποία θα παρουσιαστούν παρακάτω.

Οι Controllers που δημιουργήθηκαν είναι, ο TotpController και ο UserController. Ο πρώτος, εξυπηρετεί αιτήματα που σχετίζονται με την αυθεντικοποίηση του χρήστη, καθώς και την δημιουργία και τον έλεγχο του TOTP κωδικού. Ο δεύτερος, εξυπηρετεί αιτήματα, που σχετίζονται με τις πληροφορίες του χρήστη στο Active Directory. Ένα επιπλέον υποσύστημα στο οποίο δόθηκε έμφαση, είναι οι υπηρεσίες, κοινώς γνωστές στην ASP.NET, ως Services. Οι υπηρεσίες αυτές υπάρχουν στο backend και παρουσιάζονται παρακάτω, μαζί με μια σύντομη περιγραφή της κάθε μιας.

a. Η Active Directory Service, ασχολείται με ότι αφορά τα στοιχεία του χρήστη στο AD. Ελέγχει εάν υπάρχει ο χρήστης και επιστρέφει πληροφορίες που υπάρχουν στον διακομιστή για αυτόν (πχ. Name, First Name, Initials, Last Name, Description, Office, Telephone Number, Email Account).

b. Η Token Service, είναι υπεύθυνη για τη δημιουργία, τον έλεγχο, την αποθήκευση και την ανανέωση του χρόνου ζωής των token των χρηστών.

c. Η Login Security Service, ακολουθώντας την πολιτική ασφαλείας που έχει επιλεχθεί, καταγράφει τις επιτυχημένες και τις αποτυχημένες προσπάθειες σύνδεσης αλλά και τις υπόλοιπες ενέργειες των χρηστών. Επιπλέον, αποκλείει IP εκτελώντας script μέσω PowerShell και διατηρεί αρχείο των αποκλεισμένων (σε περιπτώσεις όπως, η υπέρβαση του επιτρεπτού αριθμού προσπαθειών σύνδεσης, σε προσπάθεια δημιουργίας totp σε χρήστη που έχει ήδη κα.). Ο αποκλεισμός μιας IP ονομάζεται και ban.

d. Η Session Cleanup Service, εκτελείται αυτόματα κάθε πέντε λεπτά ή έπειτα από όσο χρόνο έχει οριστεί, και ελέγχει εάν ισχύουν ακόμα τα bans και τα ενεργά tokens των χρηστών, τα οποία έχουν κάποια χρονική ισχύ (στο συγκεκριμένο παράδειγμα ορίστηκε σε 30 λεπτά). Μετά το πέρας του χρόνου αυτού, λήγουν και έτσι η ίδια υπηρεσία μπορεί να τα διαγράψει για να μην συσσωρεύονται.

e. Η Api Client Trust Service, είναι υπεύθυνη για μια σειρά εργασιών που σχετίζονται με το fingerprint ενός client. Αναλυτικότερα, αποκλείει fingerprint σε περίπτωση που ενώ κάποιος χρήστης είναι συνδεδεμένος, έρθει αίτημα για σύνδεση στον ίδιο χρήστη ή για οποιαδήποτε άλλη λειτουργία, από διαφορετικό fingerprint. Τέλος, προχωρά σε έλεγχο των fingerprints που έρχονται, ώστε να επιβεβαιώσει πως δεν είναι αποκλεισμένα.

Σημαντικό είναι να αναφερθεί πως υπάρχουν και άλλα Services (πχ. SecretEncryption και SessionRepository), όμως παρουσιάστηκαν συνοπτικά τα πιο σημαντικά για την πτυχιακή αυτή.

Οι 2 ιστοσελίδες αναπτύχθηκαν σε Node.js/Express. Η πρώτη ιστοσελίδα είναι αυτή με την οποία θέλει να αλληλεπιδράσει ο χρήστης. Για να μπορέσει όμως να το κάνει αυτό, πρέπει πρώτα να συνδεθεί με SSO, το οποίο σημαίνει πως μόλις επιλέξει αυτόν τον τρόπο σύνδεσης, αυτόματα θα οδηγηθεί στην δεύτερη σε σειρά ιστοσελίδα στην οποία θα γίνει και η αυθεντικοποίηση του. Αφού ολοκληρωθεί η διαδικασία αυτή, ο χρήστης ανακατευθύνεται στην πρώτη σελίδα και πλέον μπορεί να δει εμπιστευτικά δεδομένα, η επίδειξη των οποίων απαιτεί αυθεντικοποίηση. Η δεύτερη ιστοσελίδα που αναπτύχθηκε και στην οποία έγινε αναφορά, είναι το SSO frontend, το οποίο επικοινωνεί με το API μέσω του αρχείου server.js. Το εν λόγω αρχείο περιλαμβάνει δύο proxy endpoints, τα οποία λαμβάνουν πληροφορίες που δίνει ο χρήστης κατά τη προσπάθεια σύνδεσης και τις προωθούν στο API, ώστε αυτό να κάνει περαιτέρω έλεγχο. Το πρώτο είναι το /auth/authenticate, το οποίο λαμβάνει τα στοιχεία σύνδεσης του χρήστη, ενώ το δεύτερο είναι το /auth/validate, το οποίο λαμβάνει τον εξαψήφιο κωδικό TOTP. Και τα δύο endpoints ελέγχουν το τμήμα του URI που καθορίζει τον σύνδεσμο στον οποίο θα δρομολογηθεί ο χρήστης μετά την επιτυχημένη σύνδεση. Ο έλεγχος αυτός είναι αναγκαίος ώστε να εξακριβωθεί αν ο προορισμός είναι επιτρεπτός. Οποιαδήποτε επικοινωνία του SSO με το API πραγματοποιείται μέσω HTTPS για να μην εκτεθούν πληροφορίες στο δίκτυο, όπως το fingerprint του client/browser, το client type, η IP και τα cookies.

Ένα άλλο κύριο υποσύστημα της πτυχιακής αυτής αφορά την αποθήκευση και καταγραφή ενεργειών των χρηστών. Τα αρχεία από τις καταγραφές του API αποθηκεύονται

τοπικά στον διακομιστή, στον φάκελο C:\apilogs. Στον φάκελο αυτό υπάρχει το αρχείο της βάσης δεδομένων (sessions.db), όπου διατηρεί τους πίνακες για τα ενεργά sessions που υπάρχουν, τις απόπειρες σύνδεσης και τις αποκλεισμένες IP. Επιπλέον, υπάρχει αρχείο στο οποίο διατηρούνται κρυπτογραφημένα όλα τα κλειδιά από τα ενεργά TOTPs των χρηστών (totp_secrets.json), και αρχείο που καταγράφει όλες τις κινήσεις των χρηστών εντός του API (login, αποτυχημένες προσπάθειες σύνδεσης, προσπάθειες πρόσβασης σε endpoints) με χρονική σήμανση (user_activity.log). Τέλος, υπάρχει το εκτελέσιμο script, το οποίο είναι υπεύθυνο για τον αποκλεισμό των IP (block_firewall.ps1), τα οποία αποθηκεύονται απευθείας σε έναν εσωτερικό φάκελο (ps_transcripts) για την καλύτερη οργάνωση του όλου συστήματος.

5.1.2 Ροή υψηλού επιπέδου

Για την ευκολότερη κατανόηση του τρόπου με τον οποίο γίνεται σύνδεση ενός χρήστη στον server, γίνεται επιγραμματική περιγραφή των βημάτων που ακολουθούνται.

1. Ο χρήστης ο οποίος βρίσκεται στην πρώτη ιστοσελίδα επιλέγει να κάνει LOG IN with SSO και ανακατευθύνεται στην δεύτερη ιστοσελίδα, η οποία καταγράφει το fingerprint της συσκευής/client με χρήση του FingerprintJS.
2. Ο χρήστης βάζει τα στοιχεία σύνδεσης του (username και password) και το frontend τα προωθεί μαζί με το fingerprint και την IP στο API.
3. Το API αφού επιβεβαιώσει πως δεν υπάρχει ενεργός αποκλεισμός της IP και του fingerprint ή κάποιο ενεργό session για τον χρήστη, τον ανακατευθύνει στην πρώτη σελίδα μαζί με το access_token.
4. Σε περίπτωση όμως που ο χρήστης έχει ενεργοποιήσει στο προφίλ του το MFA (Multi Factor Authentication) το API ενημερώνει το frontend ώστε να προχωρήσει την αυθεντικοποίηση με χρήση του TOTP.
5. Ο χρήστης προωθεί τον εξαψήφιο κωδικό, το frontend τον στέλνει στο API, το οποίο προχωράει στον έλεγχο του και επιστρέφει στον χρήστη το access_token σε μορφή cookie.
6. Ο χρήστης ανακατευθύνεται στο redirect URI, όταν αυτό υπάρχει στο URL και ανήκει στα επιτρεπόμενα domains. Στην περίπτωση όπου στο URL δεν υπάρχει redirect URI, τότε το API επιστρέφει ανάλογο μήνυμα σφάλματος, όπως και όταν το redirect URI δεν βρίσκεται στη λίστα των επιτρεπτών.

7. Όλες οι επιτυχείς και οι μη επιτυχείς προσπάθειες σύνδεσης και οι αποκλεισμοί καταγράφονται στο αρχείο user_activity.log, ώστε αν κάποια στιγμή χρειαστεί να μπορεί να γίνει έλεγχος μέσω τερματικού ή εφαρμογών επεξεργασίας κειμένου.
8. Τα Background services καθαρίζουν ληγμένα sessions, bans και client trust records για να μην συσσωρεύονται.

5.1.3 Μηχανισμοί προστασίας του API

Πολύ σημαντικό είναι το API να είναι επαρκώς προστατευμένο. Για την προστασία του έγινε μια προσπάθεια χρήσης ήδη υπάρχοντων αλλά και εναλλακτικών μηχανισμών. Στην πρώτη κατηγορία, ανήκει το TOTP setup, το rate limiting & logging, η πραγματοποίηση όλων των κλήσεων στη βάση από παραμετροποιημένα queries και ο περιορισμός της δικαιοδοσίας του χρήστη όσον αφορά την αποθήκευση των input του. Στην δεύτερη κατηγορία, ανήκουν το fingerprint, ως επιπλέον τρόπος επαλήθευσης της μοναδικότητας του χρήστη και ο αποκλεισμός IP, μέσω του Windows Firewall. Μερικά από αυτά θα αναλυθούν εκτενέστερα παρακάτω.

Το API αποθηκεύει το fingerprint του browser/client του χρήστη, το οποίο δημιουργείται από το FingerprintJS στο frontend, και τα στοιχεία της συνεδρίας-session. Εάν υπάρξει αίτημα με διαφορετικό fingerprint για τον ίδιο χρήστη, τότε ενεργοποιείται ο μηχανισμός αποκλεισμού της IP και του fingerprint από όπου στάλθηκε το νέο αίτημα, και το περιστατικό καταγράφεται στην βάση δεδομένων. Ωστόσο, η εν λόγω πολιτική συνήθως δεν προτείνεται, λόγω του ότι κάνει λιγότερο ευχάριστη την εμπειρία για τον χρήστη. Παρά το γεγονός αυτό, η παραπάνω πολιτική χρησιμοποιήθηκε στο πλαίσιο της πτυχιακής αυτής εργασίας, για να την ανάδειξη της χρήσης του fingerprint, ως δεύτερο αναγνωριστικό ταυτότητας του χρήστη. Θα μπορούσε βέβαια να χρησιμοποιηθεί με πολλές διαφορετικές παραλλαγές, οι οποίες θα παρουσιαστούν στον τελευταίο κεφάλαιο της εργασίας.

Το endpoint setup του TOTP είναι προσβάσιμο στο δίκτυο, το οποίο σημαίνει πως ανά πάσα στιγμή μπορεί να λάβει αίτημα από οποιονδήποτε έχει τα στοιχεία πρόσβασης του χρήστη, με σκοπό την αντικατάσταση ή την δημιουργία νέου TOTP secret. Για την αποφυγή αυτού, παραμετροποιήθηκε το endpoint setup του TOTP, έτσι ώστε να δημιουργεί ένα μόνο TOTP secret ανά χρήστη, δηλαδή να επιτρέπει την ύπαρξη ενός μόνο ενεργού TOTP secret. Σε περίπτωση που πραγματοποιηθει αίτημα προς το API για δημιουργία εκ νέου secret, τότε η εν

λόγω ενέργεια θεωρείται ύποπτη και το API μπλοκάρει την IP, από την οποία έγινε το αίτημα, και καταγράφει το γεγονός στην βάση δεδομένων.

Όσον αφορά τους δύο παραπάνω μηχανισμούς, ο αποκλεισμός δε γίνεται μόνο σε επίπεδο εφαρμογής (API) αλλά και σε επίπεδο δικτύου, με προσθήκη κανόνα στο Firewall του διακομιστή.

Ένας ακόμα μηχανισμός προστασίας που χρησιμοποιήθηκε, είναι ο καθορισμός του επιτρεπτού ορίου αιτημάτων που μπορεί να εξυπηρετήσει το API, από κάθε IP (rate limiting), σε συγκεκριμένο χρονικό διάστημα. Εάν ξεπεραστεί το όριο αυτό, τότε το API επιστρέφει σφάλμα και οποιοδήποτε νεότερο αίτημα από αυτή την IP δεν εξυπηρετείται μέχρι το τέλος του χρονικού αυτού διαστήματος, προστατεύοντας το API από επιθέσεις brute-force και κατάχρηση των πόρων του διακομιστή. Το χρονικό διάστημα αυτό μπορεί να οριστεί από τον προγραμματιστή κατά τη δημιουργία του endpoint ή σε οποιαδήποτε αναβάθμιση του.

5.2 Προαπαιτούμενα για τη λειτουργία του API

Για να λειτουργήσει το API χρειάζεται συγκεκριμένο λογισμικό, ορισμένες εφαρμογές και καθορισμένες ρυθμίσεις στον διακομιστή. Αναλυτικότερα, αναγκαίο είναι να υπάρχει εγκατεστημένο λογισμικό Windows Server 2022+ σε διακομιστή μέλος του domain windowsserver, ώστε να συνδέεται το API απευθείας με το Active Directory μέσω PrincipalContext*. Όσον αφορά τις εφαρμογές, θα πρέπει να είναι εγκατεστημένο το .NET SDK/Runtime (LTS έκδοση) για την εκτέλεση του API και το Powershell, ώστε το API να μπορεί να δημιουργεί τους κανόνες στο Firewall για αποκλεισμό IP που έχουν ύποπτη δραστηριότητα. Τέλος, χρειάζεται ο IT να έχει δώσει στον χρήστη που εκπροσωπεί το API, δικαίωμα ανάγνωσης των χρηστών του AD και δικαιώματα ανάγνωσης και αλλαγής των πληροφοριών που βρίσκονται στο φάκελο που διατηρεί τα logs του API (C:\apilogs).

*Η σύνδεση του API με το AD γίνεται μέσω της κλάσης PrincipalContext, της βιβλιοθήκης System.DirectoryServices.AccountManagement. Η χρήση αυτής της κλάσης, παρέχει στο API τη δυνατότητα ασφαλούς και κεντρικής διαχείρισης των χρηστών (αναζήτηση, επαλήθευση κωδικών, ενημέρωση ιδιοτήτων), και επιλέχθηκε επειδή προσφέρει επικύρωση διαπιστευτηρίων απευθείας από το AD, χωρίς το API να χρειάζεται να αποθηκεύει ή να έχει πρόσβαση στους κωδικούς των χρηστών.

5.2.1 Ρύθμιση αρχείων παραμέτρων

Ένα από τα πιο βασικά αρχεία για την λειτουργία των ASP.NET Core εφαρμογών/API είναι το appsettings.json. Το αρχείο αυτό μπορεί να παραμετροποιηθεί, ώστε να υπάρξουν αλλαγές στο API, χωρίς να γίνουν αλλαγές στον πηγαίο κώδικα. Ακολουθεί απόσπασμα από το αρχείο, καθώς και οι βασικές ρυθμίσεις που χρησιμοποιούνται στην υλοποίηση.

```
{
 "JwtSettings": {
  "SecretKey": "<BASE64-κλειδί-256bit>"
 },
 "TotpSettings": {
  "SecretsFileFolder": "C:\\apilogs"
 },
 "LoggingPaths": {
  "LogFolder": "C:\\apilogs"
 },
 "IpRateLimiting": {
  "EnableEndpointRateLimiting": true,
  "GeneralRules": [
   { "Endpoint": "*:/api/*", "Period": "1m", "Limit": 60 }
  ]
 },
 "AllowedHosts": "example.local;sso.example.local;adapi.example.edu",
}
```

Συνοπτικά, σε αυτό το κομμάτι κώδικα υπάρχει η δήλωση του μυστικού κλειδιού SecretKey, το οποίο χρησιμοποιείται για την ψηφιακή υπογραφή και την επικύρωση των JWT tokens. Το κλειδί έχει μέγεθος 256 bit και για την κρυπτογράφηση του έχει επιλεχθεί ο αλγόριθμός Base64. Επιπλέον, δηλώνεται ο φάκελος, C:\apilogs, στον οποίο αποθηκεύονται τα μυστικά κλειδιά που σχετίζονται με τον TOTP μηχανισμό, σε κρυπτογραφημένη μορφή, για

λόγους ασφαλείας. Στον ίδιο φάκελο διατηρούνται τα αρχεία καταγραφής (logs) του API, ώστε να υπάρχει μια κεντρική συλλογή αυτών για τον ευκολότερο έλεγχο περιστατικών ασφαλείας και monitoring των κινήσεων, από και προς το API.

Τέλος, ενεργοποιείται ο μηχανισμός RateLimiting για όλα τα endpoints του API (*:/api/*) ώστε να μειωθεί ο κίνδυνος επιθέσεων τύπου Brute Force ή Denial of Service και μέσω της μεταβλητής AllowedHosts καθορίζονται οι hosts που επιτρέπεται να εξυπηρετηθούν από το API.

5.2.2 Πολιτική Cookies

Αφού το API αυθεντικοποιήσει τον χρήστη, του στέλνει επιβεβαιωτικό μήνυμα και δημιουργεί το access token, το οποίο πλέον θα χρησιμοποιείται για την αυθεντικοποίηση του χρήστη, αντί των στοιχείων σύνδεσης του, ώστε αυτά να μην κυκλοφορούν συχνά στο δίκτυο. Το access token δημιουργείται από την Token Service και αποστέλλεται σε cookie με χρήση της μεθόδου Response.Cookies.Append(), η οποία προσφέρεται από την ASP.NET Core. Η διαδικασία αυτή φαίνεται πιο αναλυτικά στο παρακάτω απόσπασμα κώδικα.

```
Response.Cookies.Append("access_token", token, new CookieOptions
{
   HttpOnly = true,
   Secure = true,
   SameSite = SameSiteMode.Lax,
   Expires = DateTimeOffset.Now.AddMinutes(30),
   Path = "/",
   Domain = "example.local"
});
```

Επιγραμματικά τα κύρια χαρακτηριστικά της πολιτικής που επιλέχθηκε για την αποστολή του access token, είναι τα εξής:

1. Το cookie δεν είναι προσβάσιμο από JavaScript, γεγονός που περιορίζει τον κίνδυνο από επιθέσεις τύπου XSS. Αυτό επιτυγχάνεται μέσω της επιλογής HttpOnly = true.

2. Το cookie αποστέλλεται αποκλειστικά μέσω HTTPS, προστατεύοντας έτσι από πιθανή υποκλοπή (πχ. μέσω packet sniffing), ορίζοντας την μεταβλητή Secure = true.
3. Επιλέχθηκε η κοινή χρήση του cookie μεταξύ όλων των υποτομέων (subdomain) του example.local, διευκολύνοντας έτσι το σενάριο του SSO (Domain = “example.local”). Επίσης, ορίστηκε το SameSite να είναι Lax, το οποίο σημαίνει πως το cookie μπορεί να αποσταλεί μόνο σε αιτήματα που προέρχονται από την ίδια ιστοσελίδα ή subdomain. Έτσι περιορίζεται ο κίνδυνος επιθέσεων τύπου Cross-Site Request Forgery (CSRF). Σύμφωνα με τη βιβλιογραφία (OWASP CSRF Prevention Cheat Sheet) η ενσωμάτωση μηχανισμών anti-CSRF, διασφαλίζει πλήρως την ακεραιότητα των αιτημάτων. [27]
4. Το cookie έχει περιορισμένο χρόνο ζωής, ο οποίος είναι 30 λεπτά και ανανεώνεται με κάθε επιτυχές αίτημα του χρήστη. Το συγκεκριμένο χρονικό διάστημα επιλέχθηκε, για να είναι φιλικό προς τον χρήστη και ώστε να περιορίσει το χρονικό διάστημα διατήρησης των token στον διακομιστή, μειώνοντας το ρίσκο υποκλοπής τους και την μετέπειτα χρήση τους από κάποιο τρίτο.

5.2.3 Build & Run

Το API μπορεί να εκτελεστεί με δυο τρόπους στον διακομιστή που φιλοξενείται. Ο πρώτος απαιτεί αρχικά εγκατάσταση και λειτουργία της εφαρμογής NET command-line interface (CLI), στον διακομιστή όπου επιθυμούμε να εκτελεστεί το API. Στην συνέχεια, γίνεται χρήση εντολών, όπως η «dotnet restore» με την οποία πραγματοποιείται ανάκτηση απαραίτητων για το API πακέτων NuGet, και η εντολή «dotnet build -c Release», η οποία δημιουργεί το εκτελέσιμο σε Release Mode (λειτουργία παραγωγής). Για την εκκίνηση του API χρειάζεται να εκτελεστεί η εντολή «dotnet ActiveDirectoryAPI.dll», η οποία εκκινεί τον HTTP server της εφαρμογής. Το API είναι διαθέσιμο από το URL και το port που έχουν οριστεί από τον κώδικα (configured URL/port).

Ο δεύτερος τρόπος είναι η υλοποίηση του API να γίνει publish, είτε μέσα από το περιβάλλον του Visual Studio, είτε με εκτέλεση της εντολής «dotnet publish -c Release -o C:\path\to\api». Στην συνέχεια, ο IT χρειάζεται να διαμορφώσει τον IIS Host στον διακομιστή και να μεταφέρει τα αρχεία του φακέλου publish στον αντίστοιχο φάκελο του IIS. Η φιλοξενία μέσω IIS θεωρείται πιο μόνιμη και σταθερή στο παραγωγικό περιβάλλον, καθώς παρέχει μεταξύ άλλων αυτόματη εκκίνηση της εφαρμογής και ενσωμάτωση HTTPS/SSL πιστοποιητικών για ασφαλή επικοινωνία.

5.2.4 Δικαιώματα στο Active Directory

Για να μπορέσει το API να επικοινωνήσει με το AD, χρειάζεται μια σειρά ρυθμίσεων του διακομιστή. Απαραίτητο είναι να υπάρχει ένας ειδικός λογαριασμός υπηρεσίας (service account), ο οποίος να είναι μέλος του domain και να έχει μια σειρά δικαιωμάτων. Στον λογαριασμό αυτό, πρέπει να έχουν παραχωρηθεί δικαιώματα ανάγνωσης σε χρήστες και ομάδες του AD (username etc), με σκοπό να επαληθεύει τα στοιχεία σύνδεσης που δέχεται από τους χρήστες κατά το πρώτο βήμα αυθεντικοποίησης, και άρα το API να μην χρειάζεται να τα αποθηκεύει. Η κατοχή των δικαιωμάτων αυτών, σε συνδυασμό με την έλλειψη των δικαιωμάτων ενημέρωσης και αλλαγής πληροφοριών στο AD, συνεισφέρουν στον περιορισμό του κινδύνου, μιας και ο χρήστης δεν μπορεί να παραμετροποιήσει πληροφορίες του διακομιστή. Επιπλέον, χρειάζεται να έχει δικαιώματα ανάγνωσης και επεξεργασίας του φακέλου apilogs, καθώς και των αρχείων εντός αυτού, για να διατηρεί αρχεία καταγραφών, τα οποία μπορούν να φανούν χρήσιμα σε ποικίλες περιπτώσεις. Σημαντικό είναι ακόμα, ο λογαριασμός υπηρεσίας να είναι μέλος της τοπικής ομάδας Administrators, ώστε να έχει τη δυνατότητα προσθήκης κανόνων στο Firewall του διακομιστή.

5.3 Λειτουργική περιγραφή & Endpoints

Στο πλαίσιο της παρούσας πτυχιακής χρησιμοποιήθηκαν συγκεκριμένες ροές αυθεντικοποίησης και διαχείρισης των πληροφοριών των χρηστών του AD. Παρακάτω παρουσιάζονται τα endpoints που δημιουργήθηκαν μαζί με μια σύντομη περιγραφή του τρόπου λειτουργίας τους και του τι προσφέρουν.

5.3.1 Ροές αυθεντικοποίησης (2 βήματα)

Η διαδικασία σύνδεσης του χρήστη πραγματοποιείται είτε σε ένα στάδιο, είτε σε δύο στάδια, εάν έχει ενεργοποιηθεί η επιλογή της χρήσης TOTP. Και στις δύο ροές το πρώτο βήμα είναι να στείλει ο client του χρήστη τα στοιχεία σύνδεσης του (σε json μορφή: { "username": "<ADusername>", "password": "<ADpassword>" }), στο endpoint «POST /api/totp/authenticate» .Το API επικυρώνει, μέσω του AD, τα στοιχεία πρόσβασης που έχει λάβει, ενώ ταυτόχρονα ενεργοποιεί μηχανισμούς ασφαλείας, όπως ο έλεγχος αποκλεισμού fingerprint/IP και η καταγραφή αποτυχημένων προσπαθειών. Στην περίπτωση που ο χρήστης έχει απενεργοποιημένη την λειτουργία MFA, τότε το API επιστρέφει σε μορφή cookie το access_token, μαζί με τα metadata της συνεδρίας (πχ sessionId, χρόνος λήξης του token).

Στην περίπτωση που ο χρήστης έχει ενεργοποιημένη την λειτουργία MFA, τότε το API ενημερώνει τον client πως για την ολοκλήρωση της διαδικασίας αυθεντικοποίησης χρειάζεται

να αποστείλει στο API το username μαζί με τον 6 ψήφιο κωδικό TOTP που έχει παραχθεί από την εφαρμογή Aunthenticator που αυτός έχει επιλέξει ({ "username": "<samAccountName>", "code": "123456" }). Η διαδικασία αυτή γίνεται με τη χρήση του endpoint «POST /api/totp/validate». Το API με την σειρά του επαληθεύει τα στοιχεία που έλαβε και επιστρέφει σε μορφή cookie το access_token, μαζί με τα metadata της συνεδρίας (πχ sessionId, χρόνος λήξης του token).

Όταν ο χρήστης θελήσει να αποσυνδεθεί, καλείται το endpoint «POST /api/totp/signout», το οποίο διαγράφει το session από την βάση και ακυρώνει το cookie του χρήστη. Για τη διαδικασία αυτή απαιτείται Authorization.

5.3.2 Ενεργοποίηση TOTP

Για να ενεργοποιήσει ένας χρήστης τον μηχανισμό TOTP στον λογαριασμό του, πρέπει να στείλει τα στοιχεία σύνδεσης του AD στο endpoint «POST /api/totp/setup». Το API τα ελέγχει και αν είναι σωστά, δημιουργεί μυστικό κλειδί TOTP και του το προωθεί. Σε περίπτωση που ο χρήστης θέλει να ελέγξει αν η παραπάνω διαδικασία έγινε σωστά, μπορεί είτε να συνδεθεί απευθείας μέσω στου client, είτε να στείλει τον κωδικό στο endpoint «POST /api/totp/verify».

5.3.3 Προφίλ χρήστη

Για να μπορέσει ένας χρήστης να δει τις πληροφορίες που υπάρχουν στο προφίλ του στο AD, αρκεί να στείλει αίτημα στο endpoint «GET /api/user/me?fields=firstName,lastName,emailAccount,...», μαζί με το token του. Για λόγους ασφαλείας επιλέχθηκε οι πληροφορίες αυτές να μην μπορούν να αλλάξουν από τον χρήστη εντός του API, αλλά μόνο μέσω των προσφερόμενων επιλογών της Microsoft.

5.3.4 Πίνακας endpoints

Πίνακας 4: Endpoints που αναπτύχθηκαν

Μέθοδος	Διαδρομή	Auth	Περιγραφή
POST	/api/totp/setup	Όχι	Δημιουργία μυστικού TOTP (με AD login)
POST	/api/totp/verify	Όχι	Δοκιμαστική επαλήθευση TOTP

POST	/api/totp/authenticate	Όχι*	Βήμα 1 – έλεγχος AD, απόφαση για TOTP / άμεσο login
POST	/api/totp/validate	Όχι**	Βήμα 2 – TOTP, JWT, session/cookie
POST	/api/totp/signout	Όχι (headers + TOTP)	Logout, ακύρωση συνεδρίας/cookie
GET	/api/user/me	Ναι	Προβολή επιλεγμένων πεδίων χρήστη

*Header που χρησιμοποιείται: X-Client-Fingerprint

**Headers που χρησιμοποιούνται: X-Client-Fingerprint, X-Client-Type.

Cookie: access_token (HttpOnly, Secure, SameSite=Lax, Domain=example.local, Path=/).

5.4 Μηχανισμοί Ασφαλείας

Με σκοπό την ελαχιστοποίηση του ρίσκου από επιθέσεις XSS/CSRF, brute-force, κατάχρηση πόρων και μη εξουσιοδοτημένης πρόσβασης, στο τεχνικό κομμάτι της πτυχιακής αυτής, χρησιμοποιήθηκε συνδυασμός μέτρων ασφαλείας σε επίπεδο εφαρμογής, δικτύου και αποθήκευσης. Παρακάτω θα περιγραφούν αναλυτικά και τα τρία επίπεδα.

5.4.1 Επίπεδο εφαρμογής (Application)

Σε επίπεδο εφαρμογής, η ασφάλεια ενισχύεται με διάφορους τρόπους. Ένας από αυτούς είναι η έκδοση και η αποστολή του access token αποκλειστικά ως HttpOnly & Secure cookie, με χρήση της επιλογής SameSite = Lax. Το μεγάλο πλεονέκτημα αυτής της μεθόδου είναι ότι τα token δεν είναι προσβάσιμα από JavaScript, ελαχιστοποιώντας έτσι τις επιπτώσεις από επιθέσεις τύπου XSS και CSRF. Το cookie και η συνεδρία που είναι καταγεγραμμένη στον διακομιστή, ευθυγραμμίζονται χρονικά για την ύπαρξη συνάφειας στην υπηρεσία. Όπως αναφέρθηκε και προηγουμένως, η επιλογή SameSite=Lax καλύπτει βασικές περιπτώσεις, ωστόσο προτείνεται σε μελλοντική αναβάθμιση του API να προστεθούν anti-CSRF token μηχανισμοί.

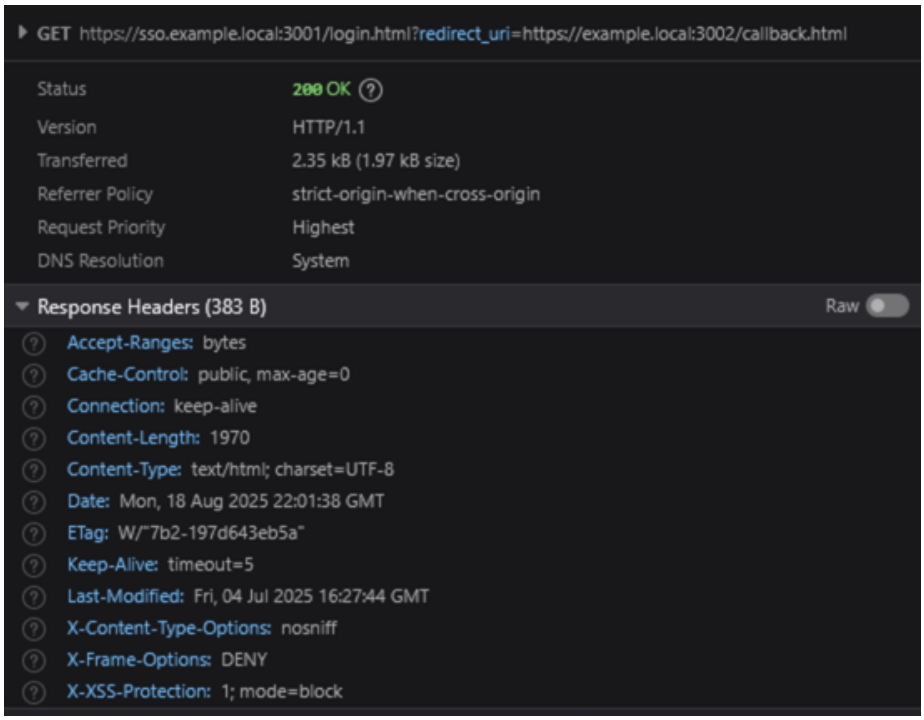
Ένας άλλος τρόπος ενίσχυσης της ασφάλειας του API είναι η εισαγωγή δεύτερης βαθμίδας αυθεντικοποίησης. Αυτό επιτυγχάνεται με χρήση της βιβλιοθήκης Otp.NET, η οποία είναι υπεύθυνη για την δημιουργία και τον έλεγχο των TOTP. Ο χρήστης με εκτέλεση εντολών στο τερματικό ή χρησιμοποιώντας κάποια εφαρμογή όπως η Postman, μπορεί να δημιουργήσει

μυστικό κλειδί και να το επαληθεύσει. Ως πολιτική ασφαλείας έχει οριστεί η αντιμετώπιση οποιασδήποτε προσπάθειας επανάληψης δημιουργίας κλειδιού, ως ύποπτη και η ενεργοποίηση μηχανισμού αποκλεισμού της IP που έστειλε το αίτημα. Για την επαλήθευση του TOTP έχει οριστεί μικρό παράθυρο ανοχής ± 1 περίοδο, για αποφυγή false negative προσπαθειών. Συγκεκριμένα, η μια περίοδος αντιστοιχεί σε 30 δευτερόλεπτα.

Για την ελαχιστοποίηση των κενών ασφαλείας έναντι επιθέσεων τύπου brute-force και DoS attack και για την προστασία των πόρων του διακομιστή και της εφαρμογής, έχει ενεργοποιηθεί περιορισμός στο πλήθος των αιτημάτων που μπορούν να εξυπηρετηθούν ανά IP (60 req/min για endpoint). Σε περίπτωση υπέρβασης των συγκεκριμένων ορίων (60/λεπτό ή 10/10 δευτερόλεπτα), η εφαρμογή απορρίπτει επιπλέον αιτήματα από αυτή την IP, για το υπόλοιπο του χρονικού διαστήματος και καταγράφει το γεγονός.

Ένας ακόμα τρόπος ενίσχυσης της ασφάλειας του API, είναι ο προσωρινός αποκλεισμός μιας συγκεκριμένης IP και του αντίστοιχου fingerprint, όταν εντοπίζονται συνεχόμενες αποτυχημένες προσπάθειες σύνδεσης σε μικρό χρονικό διάστημα. Το ίδιο συμβαίνει, και όταν ενώ ο χρήστης είναι συνδεδεμένος, εντοπίζεται αίτημα ανάγνωσης πόρων από IP ή fingerprint πέραν της δικής του.

Τέλος, για την μείωση των γνωστών επιθέσεων σε επίπεδο εφαρμογής, έχουν εφαρμοστεί τρία βασικά HTTP headers. Στο στιγμιότυπο οθόνης που παρατίθεται παρακάτω, φαίνεται το αίτημα https://sso.example.local:3001/login.html και η απάντηση σε αυτό, η οποία είναι 200 OK. Τα HTTP headers που χρησιμοποιήθηκαν, είναι με τη σειρά, το X-Content-Type-Options, το οποίο παίρνει την τιμή nosniff και αποτρέπει το MIME-sniffing από τον browser, το X-Frame-Options, το οποίο παίρνει την τιμή DENY και αποτρέπει την ενσωμάτωση της σελίδας σε iframe και τέλος, το X-XSS-Protection, το οποίο παίρνει τις τιμές 1 και mode=block και ενεργοποιεί φίλτρα XSS σε παλαιότερους browsers. Αυτά τα τρία headers χρησιμοποιούνται αμφίδρομα στην επικοινωνία του SSO proxy και του API.



Εικόνα 2: Βασικά HTTP headers

5.4.2 Επίπεδο δικτύου & πλαισίου φιλοξενίας (Network/Hosting)

Η ασφάλεια της υλοποίησης σε επίπεδο δικτύου προσεγγίστηκε με τους παρακάτω τρόπους. Αρχικά, όλες οι ροές ανάμεσα σε SSO, API και frontend πραγματοποιούνται αποκλειστικά μέσω HTTPS καναλιών. Η ρύθμιση αυτή επιλέχθηκε για την επιλογή Secure=true στο response cookie. Η λογική πίσω από αυτή την επιλογή έχει ήδη εξηγηθεί στο υποκεφάλειο, 6.2.3. Επιπλέον, έχει οριστεί στους Forwarded headers η τιμή X-Forwarded-For/proxy, με σκοπό να εξασφαλιστεί πως η IP που λαμβάνει το API κατά τη διάρκεια κάποιου αιτήματος, είναι η πραγματική IP του χρήστη. Με αυτόν τον τρόπο, εκτελούνται σωστά οι περιορισμοί και διατηρούνται αξιόπιστα αρχεία καταγραφών. Όσον αφορά τους περιορισμούς, αυτοί γίνονται σε επίπεδο Windows Firewall μέσω PowerShell, άρα σε επίπεδο δικτύου.

5.4.3 Επίπεδο δεδομένων & αποθήκευσης (Data/Storage)

Για την ενίσχυση της ασφάλειας σε επίπεδο αποθήκευσης δεδομένων, χρησιμοποιήθηκαν 3 πίνακες στη βάση δεδομένων, οι οποίοι κρατούν τις απαραίτητες πληροφορίες για τις συνεδρίες και τα συμβάντα ασφαλείας. Στον πρώτο πίνακα που έχει τίτλο “sessions”, διατηρούνται οι καταγραφές που σχετίζονται με τις ενεργές συνεδρίες που έχουν πραγματοποιηθεί μέσω του SSO. Οι πληροφορίες που μπορούν να κρατηθούν είναι ποικίλες, επιλέχθηκαν όμως χάριν ευκολίας μερικές πολύ βασικές, όπως το token (JWT), το username,

το sessionId (GUID), η IP, το clientType, το fingerprint, καθώς και το loginTime, το οποίο καθορίζει και το πότε θα λήξει το session αυτό. Οι μεταβλητές αυτές αποτελούν τις στήλες του πίνακα.

token	username	expiration	sessionId	ip	clientType	fingerprint	loginTime
Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ…	ManiaJr	2025-09-22 20:27:07	57af520b-1155-44f8-8591-48d0ba87873e	192	web	a7f8	2025-09-22 19:57:06

Εικόνα 3: Στιγμιότυπο Πίνακα sessions

Στον δεύτερο πίνακα, ο οποίος έχει τίτλο “login_attempts”, καταγράφονται οι αποτυχημένες προσπάθειες σύνδεσης ανά χρήστη, μαζί με μερικές πληροφορίες για τον χρήστη αυτόν. Αναλυτικότερα, αποθηκεύονται το username, η IP, το fingerprint και το timestamp. Όταν ο χρήστης φτάσει το όριο των επιτρεπόμενων αποτυχημένων προσπαθειών (3), ενεργοποιείται η διαδικασία αποκλεισμού του.

id	username	ip	fingerprint	timestamp
Filter	Filter	Filter	Filter	Filter
1	ManiaJr2	192	a7f8	2025-09-22 19:48:12
2	ManiaJr2	192	a7f8	2025-09-22 19:48:34
3	ManiaJr2	192	a7f8	2025-09-22 19:48:35

Εικόνα 4: Στιγμιότυπο Πίνακα login_attempts

Ο τελευταίος πίνακας έχει τίτλο “banned_ips” και καταγράφει πληροφορίες όπως, ποιες IP και ποια fingerprint είναι αποκλεισμένα, γιατί αποκλείστηκαν, πότε ξεκίνησε και μέχρι πότε ισχύει ο περιορισμός τους. Οι πληροφορίες είναι πολύ χρήσιμες στον διαχειριστή του API, για το καλύτερο monitoring και συντήρηση της εφαρμογής.

id	ip	fingerprint	reason	timestamp	expires_at
Filter	Filter	Filter	Filter	Filter	Filter
64	192	a7f	invalid_credentials	2025-09-22 19:48:35	2025-09-22 20:18:35

Εικόνα 5: Στιγμιότυπο Πίνακα banned_ips

Όλες οι καταγραφές που κάνει το API αποθηκεύονται είτε σε αρχεία, είτε στην βάση δεδομένων, τα οποία/και περιέχονται στο φάκελο C:\apilogs. Αναλυτικότερα, στον φάκελο υπάρχει ένα αρχείο μέσα στο οποίο διατηρούνται κρυπτογραφημένα τα κλειδιά TOTP που δημιουργούνται με χρήση του DPAPI/LocalMachine. Επιπλέον, υπάρχει το user_activity.log, το οποίο καταγράφει όλες τις κινήσεις που γίνονται προς το API, και το block_firewall.ps1 το οποίο είναι το εκτελέσιμο αρχείο για τον αποκλεισμό. Τέλος, υπάρχει ο φάκελος ps_trasncripts, μέσα στον οποίο αποθηκεύονται τα αποτελέσματα του block_firewall.ps1.

5.5 Πειραματική λειτουργία

Σε αυτήν την υποενότητα, παρουσιάζονται αναλυτικά ένα υποθετικό σενάριο χρήσης του SSO που δημιουργήθηκε, ένα υποθετικό σενάριο ρύθμισης του TOTP και τα αποτελέσματα των μετρήσεων των χρόνων απόκρισης κάθε endpoint.

5.5.1 Σενάριο σύνδεσης χρήστη

Για την παρουσίαση των σεναρίων χρήσης/ λειτουργίας του API αναγκαία ήταν η χρήση ενός δοκιμαστικού περιβάλλοντος. Στο δοκιμαστικό περιβάλλον που επιλέχθηκε περιλαμβάνονται Το δοκιμαστικό περιβάλλον περιλαμβάνονται τα εξής: χρειάζεται κατάλληλος Browser. Αναλυτικότερα, συμβατά αποδείχθηκαν η τελευταία έκδοση του Firefox και του Chrome, σε αντίθεση με το Brave, το οποίο επιστρέφει διαφορετικό fingerprint σε κάθε σελίδα για την προστασία του απορρήτου του χρήστη, δημιουργώντας έτσι πρόβλημα στην δοκιμαστική διαδικασία. Όσον αφορά το frontend, χρησιμοποιήθηκε το domain example.local:3002, το οποίο είναι η αρχική σελίδα από την οποία ο χρήστης ζητάει να συνδεθεί και το SSO proxy sso.example.local:3001. Το API απαντάει από το domain https://adapi.example.edu, ωστόσο επειδή δεν έχει αγοραστεί και δεν έχει καταχωρηθεί ώστε να είναι προσβάσιμο από οποιονδήποτε εκτός local έχει παραμετροποιηθεί το hosts αρχείο του υπολογιστή στον οποίο γίνεται η δοκιμή. Η διάρκεια ζωής κάθε συνεδρίας εξαρτάται σε μεγάλο βαθμό από τον χρήστη. Όσο εκείνος στέλνει νέα αιτήματα, η διάρκεια της συνεδρίας επεκτείνεται πέραν των 30 λεπτών που είναι ο ελάχιστος χρόνος ζωής της.

Παρακάτω παρουσιάζονται πιθανές υποπεριπτώσεις προσπάθειας σύνδεσης ενός χρήστη.

1. Σωστό AD password + ενεργοποιημένη η δευτεροβάθμια επαλήθευση μέσω TOTP:
   Αρχικά, ο SSO proxy στέλνει στο API, αίτημα «POST /api/totp/authenticate» μαζί με τα στοιχεία σύνδεσης του χρήστη, την IP, το fingerprint και τα headers που αναφέρθηκαν στο υποκεφάλαιο 6.4.1. Αυτό αφού τα επαληθεύσει, επιστρέφει 200 με προαπαιτούμενο Totp=true. Στην συνέχεια, ο χρήστης εισάγει τον 6-ψήφιο κωδικό και ο proxy τον προωθεί στο API με το αίτημα «POST /api/totp/validate». Το API με την σειρά του, δημιουργεί το access_token και το επιστρέφει στον χρήστη. Συνέπεια αυτού είναι, ο χρήστης να αναδρομολογηθεί στην ιστοσελίδα που αναγράφεται στο URI. Ταυτόχρονα, το API δημιουργεί εγγραφή στον πίνακα sessions και καταγράφει το γεγονός στο user_activity.log.
2. Σωστό AD password + απενεργοποιημένη η δευτεροβάθμια επαλήθευση μέσω TOTP:

Όπως και στο πρώτο σενάριο ο SSO proxy προωθεί στο API το αίτημα «POST /api/totp/authenticate». Το API επιστρέφει 200 και εκδίδει άμεσα cookie access_token. Η αναδρομολόγηση και οι εγγραφές εκτελούνται κανονικά.

3. Λανθασμένα στοιχεία σύνδεσης:

   Σε περίπτωση που τα στοιχεία πρόσβασης καταχωρηθούν λάθος, η απάντηση του API στο αίτημα «POST /api/totp/authenticate», είναι 401 Unauthorized. Σε συνέχεια αυτού, το API εγγράφει την αποτυχημένη προσπάθεια στον πίνακα login_attempts, μαζί με την συγκεκριμένη IP και το fingerprint. Εάν φτάσει το όριο των αποτυχημένων προσπαθειών, τότε η συγκεκριμένη IP και το fingerprint εγγράφονται στον πίνακα των αποκλεισμένων, ο οποίος έχει τίτλο “banned_ips”, μαζί με τον λόγο αποκλεισμού τους, ο οποίος στην συγκεκριμένη περίπτωση είναι το «invalide_credentials». Ο αποκλεισμός αυτός ισχύει για 30 λεπτά και στην συνέχεια αίρεται αυτόματα. Ταυτόχρονα, ενεργοποιείται κανόνας στο Windows Firewall.

4. Λανθασμένο TOTP:

   Στο σενάριο αυτό, παρότι το πρώτο στάδιο αυθεντικοποίησης ολοκληρώνεται με επιτυχία, στο δεύτερο εισάγεται λάθος 6-ψήφιος κωδικός. Αυτό έχει σαν αποτέλεσμα, το API να επιστρέφει 401, και όπως και στο τρίτο σενάριο, προστίθεται η κατάλληλη εγγραφή στον πίνακα login_attempts. Οι επαναλαμβανόμενες αποτυχημένες προσπάθειες οδηγούν σε αποκλεισμό και προσθήκη firewall κανόνα.

5. Παράλληλη σύνδεση από διαφορετική IP/fingerprint:

   Σε περίπτωση που ένα από τα δύο πρώτα σενάρια έχει ολοκληρωθεί με επιτυχία και το API λάβει αίτημα σύνδεσης από συσκευή διαφορετική από του χρήστη (διαφορετική IP ή και fingerprint), τότε το API απορρίπτει το νέο αίτημα και προχωράει σε αποκλεισμό τους.

6. Logout:

   Ο χρήστης μέσω του frontend στέλνει αίτημα «POST /api/totp/signout» μαζί με τις μοναδικές του πληροφορίες. Το API ελέγχει αν ταυτίζονται με αυτές που έχει αποθηκευμένες στη βάση και αν αυτό ισχύει, επιστρέφει 200 και προχωράει σε διαγραφή της συνεδρίας από τον πίνακα sessions. Ταυτόχρονα, ακυρώνει το ενεργό cookie του χρήστη και σε επόμενο αίτημα «GET /api/user/me» επιστρέφει 401.

7. Ληγμένη συνεδρία:

Στην περίπτωση που ο χρήστης παραμείνει ανενεργός για 30 λεπτά, δηλαδή δεν στείλει νέο αίτημα GET/POST προς το API, τότε αυτό προχωράει σε αυτόματο logout, παραλείποντας αυτή τη φορά την επαλήθευση των μοναδικών πληροφοριών.

5.5.2 Σενάριο για την δημιουργία TOTP

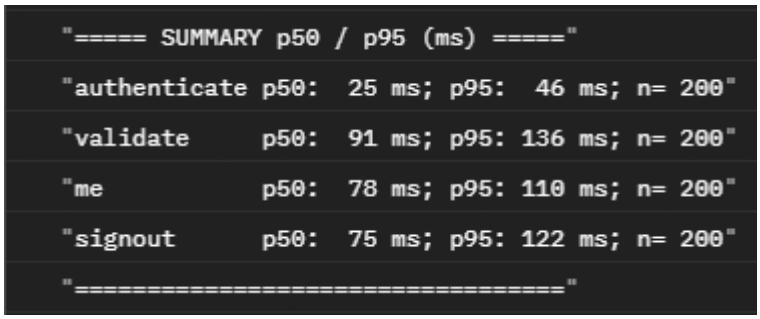
Σε αυτό το σενάριο, επιλέχθηκε οι ενέργειες για την δημιουργία και τον έλεγχο του TOTP, να πραγματοποιούνται μέσω της εφαρμογής Postman ή τερματικού. Η επιλογή αυτή βασίστηκε στο ότι δεν έχει δημιουργηθεί UI (user interface). Παρακάτω παρουσιάζονται μερικά υποσενάρια της δημιουργίας TOTP.

1. Ενεργοποίηση TOTP (setup & verify – επιτυχία):
   Ο χρήστης στέλνει το αίτημα «POST /api/totp/setup», με σωστά AD credentials, και λαμβάνει από το API απάντηση 200, μαζί με τα στοιχεία για την ενεργοποίηση του TOTP. Το μυστικό κλειδί, όπως αναφέρθηκε και προηγουμένως, αποθηκεύεται με κρυπτογραφημένη μορφή, με χρήση DPAPI/LocalMachine στο αρχείο totp_secrets.json και προστίθεται εγγραφή στο user_activity.log. Στην συνέχεια, για να επιβεβαιωθεί πως η διαδικασία έχει ολοκληρωθεί με επιτυχία και από τις 2 μεριές, ο χρήστης προωθεί το αίτημα «POST /api/totp/verify» με σωστό 6-ψήφιο από την εφαρμογή Authenticator και λαμβάνει απάντηση 200.
2. Ενεργοποίηση TOTP – αποτυχία επιβεβαίωσης:
   Όμοια με το προηγούμενο σενάριο, το πρώτο βήμα ολοκληρώνεται με επιτυχία. Στο δεύτερο όμως, κατά την επαλήθευση του κωδικού TOTP, το API επιστρέφει 401 Unauthorized, λόγω λανθασμένου κωδικού. Στις 3 αποτυχημένες προσπάθειες το API προχωράει σε αποκλεισμό της συγκεκριμένης IP και προσθήκη κανόνα στο firewall .
3. Επαναλαμβανόμενο setup (re-setup) για τον ίδιο χρήστη:
   Σε περίπτωση που πραγματοποιηθεί αίτημα «POST /api/totp/setup» για χρήστη ο οποίος έχει ήδη ενεργό μυστικό κλειδί, το API ενεργοποιεί κανόνα «ύποπτου re-setup». Αναλυτικότερα, το API, επιστρέφει σφάλμα και εφαρμόζει προσωρινό ban για 30 λεπτά. Επιπλέον, προχωρά σε εγγραφή της IP στον κατάλληλο πίνακα και προσθήκη κανόνα στο Firewall του διακομιστή.
4. Ανοχή χρονικού σφάλματος (clock skew):
   Για την αποφυγή false negative προσπαθειών 2FA, στα endpoint (/api/totp/verify και /api/totp/validate), πραγματοποιείται αποδοχή του εξαψήφιου κωδικού εντός

παραθύρου ±1 περιόδου (30s). Σε οποιαδήποτε προσπάθεια εκτός παραθύρου επιστρέφει 401 και την καταγράφει.

5.5.3 Αποτελέσματα ελέγχων- μετρήσεις

Οι μετρήσεις εκτελέστηκαν με χρήση του προγράμματος Postman και για κάθε ένα endpoint που έχει αναπτυχθεί (authenticate, validate, me, signout) εκτελέστηκαν 200 επαναλήψεις. Λόγω της ύπαρξης rate limiting ανά IP, εφαρμόστηκε delay μεταξύ των αιτημάτων, 1200ms ανά αίτημα. Οι χρόνοι απόκρισης συλλέχθηκαν σε μεταβλητές περιβάλλοντος, με την μορφή rt_*(rt_auth. Etc) και στο τέλος υπολογίστηκαν οι τιμές p50 και p95 με τον nearest-rank ορισμό. Το p50 είναι η μεσαία τιμή των δειγμάτων, τα οποία είναι ταξινομημένα σε αύξουσα σειρά, ενώ το p95 είναι η τιμή που αντιστοιχεί στο 95% του συνόλου των ταξινομημένων δειγμάτων (πχ. το 95ο στα 100 ή το 190ο στα 200). Καθώς οι μετρήσεις εκτελέστηκαν από μηχάνημα, διαφορετικό του VM στο οποίο υπάρχει το API, οι τιμές περιλαμβάνουν το RTT host με VM, καθώς έτσι προσομοιάζει την εμπειρία ενός χρήστη.



Εικόνα 6: Αποτελέσματα απόδοσης του API

Όπως φαίνεται στο παραπάνω στιγμιότυπο οθόνης, την χαμηλότερη καθυστέρηση εμφάνισε το endpoint authenticate, με p50 ≈ 25ms, ενώ τη μεγαλύτερη καθυστέρηση εμφάνισε το validate, πιθανόν λόγω των επιπλέον ελέγχων που εκτελεί. Η συνολική συμπεριφορά του API, όσον αφορά τους χρόνους απόκρισης, είναι σταθερή σε μεγάλο βαθμό, καθώς οι τιμές p95 παραμένουν κάτω από το διπλάσιο του p50 σε όλα τα endpoints.

Κεφάλαιο 6ο: Συμπεράσματα και μελλοντικές επεκτάσεις

6.1 Συμπεράσματα

Η πτυχιακή αυτή είναι μια προσπάθεια ανάδειξης της σημαντικότητας της χρήσης συστημάτων διαλειτουργικής αυθεντικοποίησης εντός εταιρειών, οργανισμών κ.α., που σκοπό είχε την μείωσης της πολυπλοκότητας των συστημάτων αυθεντικοποίησης για τους διαχειριστές, αυξάνοντας την απόδοση τους, και την πιο ευχάριστη εμπειρία για τους χρήστες. Τα πρωτόκολλα που μπορούν να χρησιμοποιηθούν για την επίτευξη αυτού του σκοπού είναι πολλά, για αυτό έγινε σοβαρή και ενδελεχής έρευνα, για τα πλεονεκτήματα και τα μειονεκτήματα μερικών ευρέως γνωστών. Επιπλέον, λήφθηκε υπόψιν η ανάγκη για διεξαγωγή ελέγχου των αναγκών της εταιρείας, ώστε το πρωτόκολλο που θα επιλεγεί να ταιριάζει όντως τις ανάγκες της.

Παρουσιάστηκε επιπλέον η δυνατότητα δημιουργίας ενός custom πρωτοκόλλου, το οποίο μπορεί να δημιουργηθεί και να χρησιμοποιηθεί στη περίπτωση που στο τέλος της παραπάνω έρευνας, κανένα από τα «έτοιμα» πρωτόκολλα δεν ικανοποιεί τις απαιτήσεις της εταιρεία και δε ταιριάζει στις ανάγκες της. Το νέο αυτό προσαρμοσμένο πρωτόκολλο προσφέρει πολύ μεγάλη ελευθερία στον τρόπο υλοποίησης και τις τεχνολογίες που θα χρησιμοποιήσει, ενώ ταυτόχρονα επιδιώκεται να είναι συμβατό με τις ήδη υπάρχουσες τεχνολογίες της εταιρείας. Έτσι, μπορεί να χρησιμοποιηθεί χωρίς να απαιτούνται μερικές ή και ολικές αλλαγές στις υπάρχουσες υποδομές της εταιρείας.

Τα παραπάνω αποτελούν στόχους της παρούσας πτυχιακής που στέφθηκαν με επιτυχία. Για την ικανοποιητική τεκμηρίωση τους, επιλέχθηκε να παρουσιαστούν αναλυτικά ορισμένοι σημαντικοί παράγοντες που πρέπει να ληφθούν υπόψη πριν την κάθε επιλογή. Όμοια με τα παραπάνω, μελετήθηκε και παρουσιάστηκε η επιλογή που προσφέρει η Microsoft, η οποία εμπεριέχει AD και ορισμένα σημαντικά εργαλεία και τεχνολογίες για την διαχείριση χρηστών, τα οποία μπορούν να φανούν πολύ χρήσιμα στους διαχειριστές. Τέλος, δημιουργήθηκε ένα API, το οποίο βέβαια θα μπορούσε μελλοντικά να τροποποιηθεί ή να επεκταθεί για να καλύπτει διαφορετικές ανάγκες από αυτές που επιλέχθηκαν να παρουσιαστούν. Στην αμέσως επόμενη ενότητα γίνεται εκτενής παρουσίαση πιθανών τροποποιήσεων και σε επίπεδο κώδικα και σε επίπεδο εφαρμογής.

Σημαντικό όμως είναι πως με το πέρας αυτής της εργασίας, νιώθω πως έχω κατανοήσει σε μεγάλο βαθμό τις έννοιες που είναι σχετικές με το θέμα της και πως αν μελλοντικά χρειαστεί να παρουσιάσω κάποιο μέρος της, θα είμαι σε θέση να το κάνω. Επίσης, θα μπορούσα να

συμβουλέψω κάποιον για να πάρει την απόφαση επιλογής πρωτοκόλλου ή διαμόρφωσης του δικού του προσωποποιημένου API και πως θα μπορούσα εγώ ο ίδιος να ασχοληθώ επιπλέον με την υλοποίηση του. Αναλυτικότερα, τα επιτεύγματα της πτυχιακής εργασίας παρουσιάζονται και στην ενότητα 1.2.

6.2 Πιθανές αλλαγές στον κώδικα με μελλοντικές επεκτάσεις

Στα πλαίσια αυτής της πτυχιακής, παραμετροποιήθηκε ένας windows server, δημιουργήθηκε ένα API, σχεδιάστηκαν 2 ιστοσελίδες (frontend) και μελετήθηκαν οι συνδέσεις μεταξύ τους. Οι αποφάσεις που λαμβάνονταν κατά τη διάρκεια της συνολικής δημιουργίας του τεχνικού κομματιού είχαν γνώμονα την ασφάλεια. Εννοείται πως πολλά πράγματα θα μπορούσαν να έχουν γίνει διαφορετικά, οδηγώντας σε ένα τελείως διαφορετικό αποτέλεσμα. Η παραπάνω υλοποίηση μπορεί να χρησιμοποιηθεί ως θεμέλιο-βάση για τη δημιουργία μιας πιο εξελιγμένης και πλήρους μορφής του. Στην συνέχεια θα παρουσιαστούν συνοπτικά μερικές πιθανές αλλαγές.

6.2.1 Πιθανές μελλοντικές τροποιήσεις της εφαρμογής

Ξεκινώντας από τα βασικά, οι ιστοσελίδες και ο server θα μπορούσαν να έχουν ονομαστεί διαφορετικά, ο χρόνος ανανέωσης του TOTP αλλά και γενικά οποιαδήποτε εντολή σχετίζεται με χρονικό περιθώριο μπορεί να αλλάξει, ώστε να προσαρμοστεί στην κάθε περίπτωση, όμοια και ο αριθμός επιτρεπτών προσπαθειών. Μια άλλη πιο σημαντική αλλαγή που θα μπορούσε να γίνει, αφορά το πλήθος των επιτρεπτών ταυτόχρονων συνδέσεων ανά χρήστη. Σε αντίθεση με την ισχύουσα πολιτική, θα μπορούσε το όριο να γίνει μεγαλύτερο της μονάδας ή να δίνεται η επιλογή στον χρήστη να ορίσει επιτρεπτή μια δεύτερη συσκευή. Αυτό μπορεί να γίνει μέσω email ή sms ή κάποιας εφαρμογής που θα μπορεί να έχει πρόσβαση μόνο ο ιδιοκτήτης του λογαριασμού. (εναλλακτική του 5.1.3)

Για το καλύτερο monitoring και την προστασία του διακομιστή και της εφαρμογής, θα μπορούσε να αλλάξει η πολιτική ανίχνευσης κάποιας απειλής. Αντί να γίνεται αποκλεισμός στον χρήστη που προσπαθεί να συνδεθεί δεύτερος σε σειρά, θα μπορούσε να γίνεται αποκλεισμός και των δύο, μέχρι να εξακριβωθεί ποιός είναι ο εξουσιοδοτημένος, μιας και μπορεί ο επιτιθέμενος να είναι αυτός που συνδέθηκε πρώτος. Εναλλακτικά, θα μπορούσε να αποκλείεται ο AD χρήστης, ώστε κανένας να μην μπορεί να συνδεθεί μέχρι να επιλυθεί το ζήτημα. (εναλλακτική του 5.4.1)

Μια άλλη αλλαγή που θα μπορούσε να γίνει, αφορά τον τρόπο διατήρησης των αρχείων καταγραφών των αιτημάτων που πραγματοποιούνται προς το API. Παρότι εδώ επιλέχθηκε να καταγράφονται όλα μαζί σε ένα αρχείο txt με τίτλο "user_activity.log", θα μπορούσαν να διαχωρίζονται οι ύποπτες κινήσεις των τελευταίων ημερών, να συμπιέζονται σε ένα έγγραφο το οποίο θα αποθηκεύεται και ως αρχείο txt να μένουν οι πληροφορίες μόνο της μέρας που διανύουμε. Με αυτό τον τρόπο μπορεί να μειωθεί σημαντικά ο όγκος των πληροφοριών που θα κρατούνται. Ένας άλλος εναλλακτικός τρόπος είναι η αποθήκευση τους σε βάση δεδομένων. (εναλλακτική του 5.4.3)

Δεν υπάρχει τρόπος να λάβει το API ένα σταθερό fingerprint από το brave, ή όποιο άλλο privacy-focused browser, ωστόσο για να μπορεί να είναι λειτουργικό το API με οιοιδήποτε browser μπορεί να γίνει αλλαγή/ προσθήκη του πηγαίου κώδικα, ώστε να δημιουργείται ένα browser ID, το οποίο να αντικαταστήσει ή να δουλεύει παράλληλα με το fingerprint ως αναγνωριστικό του χρήστη.(εναλλακτική του 5.1.1)

6.2.2 Πιθανές μελλοντικές προσθήκες στην εφαρμογή

Μια εύκολη αλλά ταυτόχρονα σημαντική αλλαγή που μπορεί να γίνει από όποιον θελήσει να χρησιμοποιήσει την εφαρμογή αυτή, είναι η παραμετροποίηση του πηγαίου κώδικα του API, ώστε ανάλογα το client Type, να κάνει τον απαιτούμενο έλεγχο, ο οποίος είναι διαφορετικός για κάθε είδος. Αυτή τη στιγμή το API είναι λειτουργικό μόνο από αιτήματα ιστοσελίδων, θα μπορούσε όμως να εξυπηρετεί εφαρμογές κινητών και υπολογιστών. Για να γίνει αυτό δεν χρειάζονται μεγάλες αλλαγές μιας και η βάση έχει σχηματιστεί έτσι ώστε να εξυπηρετήσει και άλλους τύπους client. Ήδη στον πίνακα με τίτλο "sessions", ο οποίος διατηρεί τις καταγραφές που σχετίζονται με τις ενεργές συνεδρίες, υπάρχει στήλη που αποθηκεύει το clientType. (εναλλακτική του 5.4.3)

Μια ακόμη πιθανή αλλαγή είναι η ενσωμάτωση στο σύστημα, μοντέλων μηχανικής μάθησης (machine learning) για την αυτόματη ανίχνευση ύποπτης δραστηριότητας και την εφαρμογή πολιτικών ασφαλείας σε πραγματικό χρόνο. Σε ένα τέτοιο μοντέλο, δεδομένα όπως η διεύθυνση IP του χρήστη, το αποτύπωμα του προγράμματος περιήγησης (browser fingerprint), η συχνότητα και ο τρόπος χρήσης των tokens, καθώς και τα μοτίβα πρόσβασης στον χρόνο και στον χώρο, θα αξιοποιούνται ώστε να εκτιμάται ο βαθμός κινδύνου κάθε αιτήματος. Ανάλογα το αποτέλεσμα, θα εφαρμόζονται δυναμικά μέτρα όπως προσωρινοί αποκλεισμοί (bans), επιπρόσθετη ταυτοποίηση (step-up MFA) ή ανάκληση ενεργών

συνεδριών. Τέλος, το μοντέλο θα εκπαιδεύεται συνεχώς από ιστορικά δεδομένα και αναλύσεις αποκλίσεων από τη φυσιολογική συμπεριφορά.

Σε μια άλλη μελλοντική αναβάθμιση του συστήματος, θα μπορούσε να δημιουργηθεί μια ξεχωριστή εξωτερική βάση δεδομένων που θα περιέχει όλες τις απαραίτητες πληροφορίες για τους χρήστες και τα αρχεία καταγραφών του API, ένας διακομιστής υπεύθυνος για την φιλοξενία και λειτουργία του API, και ένας διακομιστής για το AD. Σε αντίθεση με τη πρόταση αυτή, αυτήν την στιγμή το API φιλοξενείται στον ίδιο διακομιστή στον οποίον υπάρχουν και οι χρήστες του AD και η βάση δεδομένων. Με αυτή την τροποποίηση, απομονώνονται κρίσιμες υπηρεσίες, μειώνεται η επιφάνεια επιθέσεων και προσφέρεται η δυνατότητα κλιμάκωσης του API ανεξάρτητα από τον διακομιστή ταυτοποίησης ή τη βάση δεδομένων. Παράλληλα, διευκολύνεται η εφαρμογή πολιτικών ασφαλείας σε επίπεδο δικτύου, με αυστηρούς κανόνες πρόσβασης μεταξύ των υπηρεσιών και χρήση ασφαλών πρωτοκόλλων επικοινωνίας, όπως LDAPS και TLS.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] T. Huang και F. Guo, ‘Research on Single Sign-on Technology for Educational Administration Information Service Platform’, στο *2021 3rd International Conference on Computer Communication and the Internet (ICCCI)*, Nagoya, Japan: IEEE, Ιουνίου 2021, σελ. 69–72. doi: 10.1109/ICCCI51764.2021.9486813.

[2] L. Wu, H. J. Cai, και H. Li, ‘SGX-UAM: A Secure Unified Access Management Scheme With One Time Passwords via Intel SGX’, *IEEE Access*, τ. 9, σελ. 38029–38042, 2021, doi: 10.1109/ACCESS.2021.3063770.

[3] N. Naik και P. Jenkins, ‘Securing digital identities in the cloud by selecting an apposite Federated Identity Management from SAML, OAuth and OpenID Connect’, στο *2017 11th International Conference on Research Challenges in Information Science (RCIS)*, Brighton, United Kingdom: IEEE, Μαΐου 2017, σελ. 163–174. doi: 10.1109/RCIS.2017.7956534.

[4] A. Puesche, D. Bothe, M. Niemeyer, S. Sachweh, N. Pohlmann, και I. Kunold, ‘Concept of Smart Building Cyber-physical Systems Including Tamper Resistant Endpoints’, στο *2018 International IEEE Conference and Workshop in Óbuda on Electrical and Power Engineering (CANDO-EPE)*, Budapest: IEEE, Νοεμβρίου 2018, σελ. 000127–000132. doi: 10.1109/CANDO-EPE.2018.8601130.

[5] S. Prinakaa, B. V, S. S, S. Srinivasan, και S. V, ‘A Real-Time Approach to Detecting API Abuses Based on Behavioral Patterns’, στο *2024 8th International Conference on Cryptography, Security and Privacy (CSP)*, Osaka, Japan: IEEE, Απριλίου 2024, σελ. 24–28. doi: 10.1109/CSP62567.2024.00012.

[6] A. M. Abdurrahman και E. Husni, ‘A Secure Digital Image Marketplace: Microservices and OWASP API Security Using Spring Boot’, στο *2024 International Conference on ICT for Smart Society (ICISS)*, Bandung, Indonesia: IEEE, Σεπτεμβρίου 2024, σελ. 1–8. doi: 10.1109/ICISS62896.2024.10750956.

[7] B. Gibson, S. Townes, D. Lewis, και S. Bhunia, ‘Vulnerability in Massive API Scraping: 2021 LinkedIn Data Breach’, στο *2021 International Conference on Computational Science and Computational Intelligence (CSCI)*, Las Vegas, NV, USA: IEEE, Δεκεμβρίου 2021, σελ. 777–782. doi: 10.1109/CSCI54926.2021.00191.

[8] Y. Ishida, M. Hanada, A. Waseda, και M. W. Kim, ‘Automated Vulnerability Assessment Approach for Web API that Considers Requests and Responses’, στο *2024 26th International Conference on Advanced Communications Technology (ICACT)*, Pyeong

Chang, Korea, Republic of: IEEE, Φεβρουαρίου 2024, σελ. 1521–1533. doi: 10.23919/ICACT60172.2024.10471939.

[9] A. F. Nugraha, H. Kabetta, I. K. S. Buana, και R. B. Hadiprakoso, 'Performance and Security Comparison of Json Web Tokens (JWT) and Platform Agnostic Security Tokens (PASETO) on RESTful APIs', στο *2023 IEEE International Conference on Cryptography, Informatics, and Cybersecurity (ICoCICs)*, Bogor, Indonesia: IEEE, Αυγούστου 2023, σελ. 15–22. doi: 10.1109/ICoCICs58778.2023.10277377.

[10] A. Kumar και A. Gandhi, 'A Study Using OWASP On Secure Open Banking Architecture', στο *2023 3rd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*, Greater Noida, India: IEEE, Μαΐου 2023, σελ. 2062–2067. doi: 10.1109/ICACITE57410.2023.10182656.

[11] A. Beshiri, A. Mishev, και I. Chorbev, 'Security Issues in the RESTful API (Service) using OAuth 2.0 for Authentication and Authorization', 2021.

[12] J. Chen, M. Hoppen, D. Boken, J. Reitz, M. Schluse, και J. Rosmann, 'Identity, Authentication and Authorization in Forestry 4.0 Using OAuth 2.0', στο *2022 3rd International Informatics and Software Engineering Conference (IISEC)*, Ankara, Turkey: IEEE, Δεκεμβρίου 2022, σελ. 1–6. doi: 10.1109/IISEC56263.2022.9998287.

[13] W. Li και C. J. Mitchell, 'User Access Privacy in OAuth 2.0 and OpenID Connect', στο *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, Genoa, Italy: IEEE, Σεπτεμβρίου 2020, σελ. 664–6732. doi: 10.1109/EuroSPW51379.2020.00095.

[14] D. Fett, R. Küsters, και G. Schmitz, 'A Comprehensive Formal Security Analysis of OAuth 2.0', στο *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, Vienna Austria: ACM, Οκτωβρίου 2016, σελ. 1204–1215. doi: 10.1145/2976749.2978385.

[15] P. Philippaerts, J. Vanhoof, T. Van Cutsem, και W. Joosen, 'Is Your OAuth Middleware Vulnerable? Evaluating Open-Source Identity Providers' Security', στο *GLOBECOM 2024 - 2024 IEEE Global Communications Conference*, Cape Town, South Africa: IEEE, Δεκεμβρίου 2024, σελ. 3607–3612. doi: 10.1109/GLOBECOM52923.2024.10901500.

[16] D. Fett, R. Kusters, και G. Schmitz, 'The Web SSO Standard OpenID Connect: In-depth Formal Security Analysis and Security Guidelines', στο *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, Santa Barbara, CA: IEEE, Αυγούστου 2017, σελ. 189–202. doi: 10.1109/CSF.2017.20.

[17] C. Mainka, V. Mladenov, J. Schwenk, και T. Wich, ‘SoK: Single Sign-On Security — An Evaluation of OpenID Connect’, στο *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*, Paris: IEEE, Απριλίου 2017, σελ. 251–266. doi: 10.1109/EuroSP.2017.32.

[18] T. Gross, ‘Security analysis of the SAML single sign-on browser/artifact profile’, στο *19th Annual Computer Security Applications Conference, 2003. Proceedings.*, Las Vegas, Nevada, USA: IEEE, 2003, σελ. 298–307. doi: 10.1109/CSAC.2003.1254334.

[19] Z. Senturk και E. Irmak, ‘Persistence Techniques in Microsoft Active Directory: Detection and Mitigation Strategies’, στο *2024 12th International Symposium on Digital Forensics and Security (ISDFS)*, San Antonio, TX, USA: IEEE, Απριλίου 2024, σελ. 01–06. doi: 10.1109/ISDFS60797.2024.10527234.

[20] W. Matsuda, M. Fujimoto, και T. Mitsunaga, ‘Detecting APT Attacks Against Active Directory Using Machine Leaning’, στο *2018 IEEE Conference on Application, Information and Network Security (AINS)*, Langkawi, Malaysia: IEEE, Νοεμβρίου 2018, σελ. 60–65. doi: 10.1109/AINS.2018.8631486.

[21] P. Pengsart, A. R. X. Belo, J. X. Vaz, J. B. S. Marques, και E. Junior, ‘ADFS Authentication for Healthcare System’, 2017.

[22] A. B. Kretarta και H. Kabetta, ‘Secure User Management Gateway for Microservices Architecture APIs Using Keycloak on XYZ’, στο *2022 5th International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, Yogyakarta, Indonesia: IEEE, Δεκεμβρίου 2022, σελ. 7–13. doi: 10.1109/ISRITI56927.2022.10052901.

[23] J. R. Dora και L. Hluchy, ‘Attacks on Active Directory - Kerberos Delegation: *Exploitation of Active Directory using Kerberos Constrained Delegation’, στο *2024 IEEE 6th International Symposium on Logistics and Industrial Informatics (LINDI)*, Karaganda, Kazakhstan: IEEE, Οκτωβρίου 2024, σελ. 103–108. doi: 10.1109/LINDI63813.2024.10820404.

[24] A. Binduf, H. O. Alamoudi, H. Balahmar, S. Alshamrani, H. Al-Omar, και N. Nagy, ‘Active Directory and Related Aspects of Security’, στο *2018 21st Saudi Computer Society National Computer Conference (NCC)*, Riyadh: IEEE, Απριλίου 2018, σελ. 4474–4479. doi: 10.1109/NCG.2018.8593188.

[25] P. Zhang, Y. Zhang, Z. Wu, και W. Du, ‘IIS Security Mechanisms’, στο *2010 Third International Symposium on Intelligent Information Technology and Security Informatics*, Jian, China: IEEE, Απριλίου 2010, σελ. 619–622. doi: 10.1109/IITSI.2010.179.

[26] D. Ravilla και C. S. R. Putta, ‘Implementation of HMAC-SHA256 algorithm for hybrid routing protocols in MANETs’, στο *2015 International Conference on Electronic Design, Computer Networks & Automated Verification (EDCAV)*, Shillong, India: IEEE, Ιανουαρίου 2015, σελ. 154–159. doi: 10.1109/EDCAV.2015.7060558.

[27] ‘OWASP CSRF Prevention Cheat Sheet’, Ιστοσελίδα. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο:https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html

ΠΑΡΑΡΤΗΜΑ Α : Αποθετήριο και αποσπάσματα κώδικας του API

Α.1 Αποθετήρια κώδικα

API: https://github.com/ManiaJr/Thesis

Πρώτη ιστοσελίδα: https://github.com/ManiaJr/example.local

Δεύτερη ιστοσελίδα (SSO): https://github.com/ManiaJr/sso.example.local

Α.2 Αποσπάσματα κώδικα

Παρακάτω γίνεται αναφορά σε μερικά από τα βασικά τμήματα κώδικα της υλοποίησης του API, μαζί με σύντομη επεξήγηση τους. Πλήρης κώδικας της υλοποίησης θα ακολουθήσει στο Παράρτημα/Repository.

Α.2.1 Ανάγνωση JWT από HttpOnly cookie

```
builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)
    .AddJwtBearer(opts =>
    {
        opts.Events = new JwtBearerEvents
        {
            OnMessageReceived = context =>
            {
                var token = context.Request.Cookies["access_token"];
                if (!string.IsNullOrEmpty(token))
                {
                    context.Token = token;
                }
                return Task.CompletedTask;
            }
        };

        // ... validation params ...
    });
```

Εικόνα 7: Ανάγνωση JWT από HttpOnly cookie

Αυτό το κομμάτι κώδικα, επιτρέπει [Authorize] χωρίς να εκθέτει το token σε JavaScript.

Α.2.2 Κρυπτογράφηση μυστικών κλειδιών TOTP (DPAPI/LocalMachine)

```
1 reference
public static byte[] Encrypt(byte[] data) =>
    ProtectedData.Protect(data, null, DataProtectionScope.LocalMachine);

1 reference
public static byte[] Decrypt(byte[] encrypted) =>
    ProtectedData.Unprotect(encrypted, null, DataProtectionScope.LocalMachine);
```

Εικόνα 8: Κρυπτογράφηση μυστικών κλειδιών TOTP (DPAPI/LocalMachine)

Ο παραπάνω κώδικας κάνει την κρυπτογράφηση και την αποκρυπτογράφηση των μυστικών κλειδιών TOTP. Σημαντικό είναι πως οι διαδικασίες αυτές πραγματοποιούνται μόνο στο συγκεκριμένο μηχάνημα.

Α.2.3 Έλεγχος TOTP με χρονικό παράθυρο

```
var totp = new Totp(secret);
bool isValid = totp.VerifyTotp(request.Code, out _, new VerificationWindow(1, 1));
```

Εικόνα 9: Έλεγχος TOTP με χρονικό παράθυρο

Στο παραπάνω κομμάτι κώδικα, ορίζεται το παράθυρο αποδοχής TOTP μιας περιόδου, το οποίο ισούται με 30 δευτερόλεπτα. Για την αποφυγή false negatives αιτημάτων, δίνεται το περιθώριο +- (±) μιας περιόδου.

Α.2.4 Καταγραφή αποτυχημένων logins

```
7 references
public void BanIp(string ip, TimeSpan duration, string reason , string fingerprint)
{
    _sessionRepo.DeleteSessionsByIpOrFingerprint(ip, fingerprint);
    _sessionRepo.LogBanIp(ip, fingerprint, reason, duration);

    var durText = duration == TimeSpan.MaxValue ? "permanent" : duration.ToString();
    LogUserActivity(ip, "-", "Ban", "Blocked", $"Reason: {reason}; FP: {fingerprint}; Duration: {durText}");

    try
    {
        RunPowerShellBlock(ip);
    }
    catch (Exception ex)
    {
        Console.WriteLine($"[ERROR] Failed to ban firewall: {ex.Message}");
    }

}
```

Εικόνα 10: Καταγραφή αποτυχημένων logins

Το κομμάτι αυτό κώδικα χρησιμοποιείται για την άμεση προστασία της εφαρμογής και του διακομιστή από επιθέσεις brute-force.

A.2.5 Έκδοση cookie (με & χωρίς TOTP) — TotpController

```
Response.Cookies.Append("access_token", token, new CookieOptions
{
    HttpOnly = true,
    Secure = true,
    SameSite = SameSiteMode.Lax,
    Expires = DateTimeOffset.Now.AddMinutes(30),
    Path = "/",
    Domain = "example.local"
});
```

Εικόνα 11: Έκδοση cookie (με & χωρίς TOTP) — TotpController

Τα παραπάνω flags (HttpOnly/Secure/SameSite/Domain/Path), δηλώνονται ρητά κατά την έκδοση του token, με ή χωρίς χρήση TOTP.

A.2.6 Logout – ακύρωση cookie & session

```
[HttpPost("signout")]
[Authorize]
0 references
public new IActionResult SignOut()
{

    var user = _userContext.GetCurrentUserFromToken(Request);

    var token = Request.Cookies["access_token"];

    if (user == null)
        return Unauthorized();

    var fingerprint = Request.Headers["X-Client-Fingerprint"].ToString();
    var clientType = Request.Headers["X-Client-Type"].ToString();
    var ip = HttpContext.Connection.RemoteIpAddress?.ToString();

    if (string.IsNullOrWhiteSpace(fingerprint) || string.IsNullOrWhiteSpace(clientType) || string.IsNullOrWhiteSpace(ip))
        return BadRequest("Missing client info");



    _clientTrust.RemoveClientTrustRecord(user.Name.ToLowerInvariant());
    if (token != null)
        _tokenService.RemoveToken(token);

    //this does not work
    Response.Cookies.Delete("access_token", new CookieOptions
    {
        Path = "/",
    });

    return Ok(new { message = "Signed out" });
}
```

Εικόνα 12: Logout – ακύρωση cookie & session

Με την παραπάνω μέθοδο, διαγράφεται το token του χρήστη και στον διακομιστή αλλά και στον browser του.

Α.2.7 Forwarded headers – πραγματική IP πίσω από proxy

```
app.UseForwardedHeaders(new ForwardedHeadersOptions
{
    ForwardedHeaders = ForwardedHeaders.XForwardedFor | ForwardedHeaders.XForwardedProto
});
```

Εικόνα 13: Forwarded headers – πραγματική IP πίσω από proxy

Το παραπάνω κομμάτι κώδικα, δείχνει πως χρησιμοποιείται η πραγματική IP του χρήστη για την καταγραφή logs, rate-limit και ban.

Α.2.8 Rate limiting – παράδειγμα ρύθμισης

```
"IpRateLimiting": {
  "EnableEndpointRateLimiting": true,
  "StackBlockedRequests": false,
  "RealIpHeader": "X-Real-IP",
  "ClientIdHeader": "X-ClientId",
  "HttpStatusCode": 429,
  "GeneralRules": [
    {
      "Endpoint": "*",
      "Period": "1m",
      "Limit": 60
    },
    {
      "Endpoint": "*",
      "Period": "10s",
      "Limit": 10
    }
  ]
}
```

Εικόνα 14: Rate limiting – παράδειγμα ρύθμισης

Με αυτό το κομμάτι κώδικα, γίνεται περιορισμός των αιτημάτων για την μείωση καταχρήσεων και brute-force επιθέσεων.

A.2.9 block_firewall.ps1 script

```
1 reference
private void EnsureBlockScript()
{
    lock (_psLock)
    {
        if (File.Exists(ScriptPath) && !string.IsNullOrWhiteSpace(File.ReadAllText(ScriptPath)))
            return;

        var transcriptEsc = TranscriptDir.Replace("\\", "\\\\");
        var script = @"
param([Parameter(Mandatory=$true)][string]$IP)
$ErrorActionPreference = 'Stop'

$nullRef = $null
if (-not [System.Net.IPAddress]::TryParse($IP, [ref]$nullRef)) { Write-Error ""Invalid IP: $IP""; exit 1 }

$ruleName = ""SSO-Ban-$IP""
$group    = ""SSO-Bans""

$logDir = '" + transcriptEsc + @"'
New-Item -ItemType Directory -Force -Path $logDir | Out-Null
$tsPath = Join-Path $logDir (""block_{0}_{1}.log"" -f $IP, (Get-Date -Format ""yyyyMMdd_HHmmss""))

Start-Transcript -Path $tsPath -Append
try {
  if (-not (Get-NetFirewallRule -DisplayName $ruleName -ErrorAction SilentlyContinue)) {
    New-NetFirewallRule -DisplayName $ruleName -Group $group -Direction Inbound -RemoteAddress $IP -Action Block -Profile Any | Out-Null
  }
  Write-Output ""Blocked $IP with rule '$ruleName'""
  exit 0
}
catch { Write-Error $_; exit 1 }
finally { Stop-Transcript | Out-Null }
";
        File.WriteAllText(ScriptPath, script);
    }
}
```

Εικόνα 15: block_firewall.ps1 script

Το παραπάνω κομμάτι κώδικα δείχνει τον αποκλεισμό μιας IP σε επίπεδο δικτύου του διακομιστή.

A.2.10Security headers (API & proxy)

// API (Program.cs)

```
app.Use(async (context, next) =>
{
    context.Response.Headers.Add("X-Content-Type-Options", "nosniff");
    context.Response.Headers.Add("X-Frame-Options", "DENY");
    context.Response.Headers.Add("X-XSS-Protection", "1; mode=block");
    await next();
});
```

Εικόνα 16: Security headers (API & proxy) API (Program.cs)

// SSO proxy/frontend (server.js)

```
app.use((req, res, next) => {
  res.setHeader('X-Content-Type-Options', 'nosniff');
  res.setHeader('X-Frame-Options', 'DENY');
  res.setHeader('X-XSS-Protection', '1; mode=block');
```

Εικόνα 17: Security headers (API & proxy) SSO proxy/frontend (server.js)

Τα παραπάνω κομμάτια κώδικα, χρησιμοποιούνται για την προστασία της εφαρμογής-API από XSS επιθέσεις. Το πρώτο, είναι μέρος του API, ενώ το δεύτερο, είναι μέρος των αρχείων server.js του SSO proxy και του frontend.

A.2.11 TOTP setup – ενδεικτική απόκριση (redacted/περιορισμένο)

```
var uri = $"otpauth://totp/ActiveDirectoryAPI:{normalizedUsername}?secret={base32}&issuer=ActiveDirectoryAPI";

return Ok(new
{
    message = "Copy this secret to KeePassXC",
    secret = base32,
    provisioningUri = uri
});
```

Εικόνα 18: TOTP setup – ενδεικτική απόκριση (redacted/περιορισμένο)

Το κομμάτι αυτό κώδικα, δείχνει πως το API επιστρέφει το μυστικό κλειδί σε base32 και τυπικό otpauth URI για εισαγωγή στην εφαρμογή που θα επιλέξει ο χρήστης. Για λόγους ασφαλείας το μυστικό κλειδί παραμένει κρυμμένο.

A.2.12 Frontend SSO server (Node.js proxy) — προώθηση Set-Cookie

```
const setCookie = response.headers.raw()['set-cookie'];
  if (setCookie) {
    res.setHeader('Set-Cookie', setCookie);
    res.setHeader('Access-Control-Allow-Credentials', 'true');
  }
```

Εικόνα 19: Frontend SSO server (Node.js proxy) — προώθηση Set-Cookie

Με αυτό το κομμάτι κώδικα, ο proxy(server.js) απομονώνει το API και προωθεί ασφαλώς τα cookies.

A.2.13 React login component — branching με requiresTotp

```
if (data.requiresTotp) {
setShowTotp(true);
} else {

if (redirectUri) {
window.location.href = redirectUri;
} else {
window.location.href = '/';
}
```

Εικόνα 20: React login component — branching με requiresTotp

Αυτό το κομμάτι κώδικα δείχνει πως ο SSO proxy, πραγματοποιεί δυναμική εναλλαγή μεταξύ βήματος 1 και 2, ανάλογα την απόκριση που θα λάβει απο το API.

A.2.14 AllowedHosts – ρητή λίστα

```
"AllowedHosts": "example.local;sso.example.local;adapi.example.edu",
```

Εικόνα 21: AllowedHosts – ρητή λίστα

Στην παραπάνω γραμμή κώδικα, φαίνεται η αποδοχή επιλεγμένων Host headers για την αποφυγή λαθών ή κατάχρησης από μη εξουσιοδοτημένους hosts.