



# OWASP API Security Top 10 2019

Τα 10 Κορυφαία Ρίσκα Ασφαλείας API



## Πίνακας Περιεχομένων

TOC Πίνακας Περιεχομένων.....	2
FW Πρόλογος.....	3
I Εισαγωγή.....	4
RN Σημειώσεις .....	5
RISK Ρίσκα Ασφαλείας API.....	6
T10 Τα 10 Κορυφαία Ρίσκα Ασφαλείας API - 2019..	7
API1:2019 Broken Object Level Authorization.....	8
API2:2019 Broken User Authentication.....	10
API3:2019 Excessive Data Exposure.....	12
API4:2019 Lack of Resources & Rate Limiting.....	14
API5:2019 Broken Function Level Authorization..	16
API6:2019 Mass Assignment.....	18
API7:2019 Security Misconfiguration.....	20
API8:2019 Injection.....	22
API9:2019 Improper Assets Management.....	24
API10:2019 Insufficient Logging & Monitoring.....	26
+D Επόμενα Βήματα για Προγραμματιστές.....	28
+DSO Επόμενα Βήματα για DevSecOps.....	29
+DAT Μεθοδολογία και Δεδομένα.....	30
+ACK Ευχαριστίες.....	31

## Σχετικά με το OWASP

Το OWASP (Open Web Application Security Project) είναι μια ανοιχτή κοινότητα που αποσκοπεί στο να βοηθήσει οργανισμούς να παράγουν, να προμηθεύονται και να συντηρούν εφαρμογές και API που θα είναι ασφαλή.

Στο OWASP θα βρείτε δωρεάν διαθέσιμα τα παρακάτω:

- Εργαλεία ασφαλείας εφαρμογών καθώς και πρότυπα ασφαλείας (standards).
- Ολόκληρα βιβλία για testing ασφαλείας εφαρμογών, ανάπτυξη ασφαλούς κώδικα, και ελέγχου (review) ασφαλούς κώδικα.
- Παρουσιάσεις και βίντεο.
- Σκονάκια (cheatsheets) σε πολλά συνήθη θέματα.
- Standard ελέγχους ασφαλείας και βιβλιοθήκες.
- Τοπικά chapters σε όλο τον κόσμο.
- Κορυφαίες έρευνες.
- Συνέδρια σε όλο τον κόσμο.
- Λίστες ταχυδρομείου.

Μάθετε περισσότερα στο: <https://www.owasp.org>.

Όλα τα εργαλεία, έγγραφα, βίντεο, παρουσιάσεις και παραρτήματα (chapters) του OWASP είναι δωρεάν διαθέσιμα και ελεύθερα προσβάσιμα σε όλους όσους ενδιαφέρονται να βελτιώσουν την ασφάλεια των εφαρμογών τους.

Προσεγγίζουμε την ασφάλεια εφαρμογών ως ένα πρόβλημα ανθρώπων, διεργασιών και τεχνολογίας, καθώς οι πιο αποτελεσματικές προσεγγίσεις στο πρόβλημα απαιτούν βελτιώσεις στους παραπάνω τομείς.

Το OWASP είναι ένα νέο είδος οργανισμού. Η ανεξαρτησία μας από τυχόν εμπορικές πιέσεις μας επιτρέπει να παρέχουμε αμερόληπτες, πρακτικές και οικονομικά αποδοτικές πληροφορίες σχετικά με την ασφάλεια εφαρμογών.

Το OWASP δεν είναι συνδεδεμένο με καμία εταιρεία τεχνολογίας. Παρόλα αυτά υποστηρίζουμε την σωστή χρήση της εμπορικής τεχνολογίας ασφαλείας. Το OWASP παράγει υλικό με συλλογικό, διαφανή και ανοιχτό τρόπο.

Το Ίδρυμα OWASP είναι η μη κερδοσκοπική οντότητα που διασφαλίζει την μακροπρόθεσμη επιτυχία αυτού του έργου. Σχεδόν όλοι όσοι σχετίζονται με το OWASP είναι εθελοντές, συμπεριλαμβανομένου του διοικητικού συμβουλίου OWASP, των επικεφαλής των παραρτημάτων (chapters), των επικεφαλής έργων και των μελών των έργων (projects). Υποστηρίζουμε την καινοτόμο έρευνα στον τομέα ασφάλειας με επιχορηγήσεις και υποδομές.

Σας καλούμε να συμμετάσχετε και εσείς στο OWASP!

Ένα θεμελιώδες στοιχείο καινοτομίας στον σημερινό κόσμο που βασίζεται στις εφαρμογές είναι η διεπαφή προγραμματισμού εφαρμογών (API). Από τις τράπεζες, το λιανικό εμπόριο και τις μεταφορές έως το IoT, τα αυτόνομα οχήματα και τις έξυπνες πόλεις, τα API αποτελούν κρίσιμο μέρος των σύγχρονων εφαρμογών για κινητά, SaaS και web. Τα API μπορούν να βρεθούν σε εφαρμογές που απευθύνονται σε πελάτες, σε εφαρμογές που απευθύνονται σε συνεργάτες και σε ενδοεταιρικές εφαρμογές.

Από τη φύση τους, τα APIs αφήνουν εκτεθειμένες ορισμένες πτυχές της επιχειρηματικής λογικής της εφαρμογής (business logic) καθώς και ευαίσθητα δεδομένα όπως οι Προσωπικές Αναγνωριστικές Πληροφορίες (PII) (Σ.τ.Μ. πληροφορίες που επιτρέπουν την αναγνώριση προσώπου). Γι' αυτό τον λόγο τα API γίνονται όλο και περισσότερο στόχος κακόβουλων χρηστών. Χωρίς ασφαλή APIs, η ταχεία καινοτομία θα ήταν αδύνατη.

Παρόλο που ένα γενικευμένο Top 10 με κινδύνους ασφαλείας για web εφαρμογές εξακολουθεί να έχει νόημα, λόγω της ιδιαίτερης φύσης των APIs, απαιτείται μια λίστα ρίσκων ασφαλείας ειδικά για τα API. Η ασφάλεια των APIs εστιάζει σε στρατηγικές και λύσεις για την κατανόηση και την αντιμετώπιση των μοναδικών τρωτών σημείων και ρίσκων ασφαλείας που σχετίζονται με τα APIs.

Εάν είστε εξοικειωμένοι με το [OWASP Top 10 Project](#), τότε θα παρατηρήσετε τις ομοιότητες μεταξύ των δύο καταγραφών: στόχος τους είναι η εύκολη ανάγνωση (readability) και εύκολη υιοθέτηση (adoption). Εάν είστε νέος στη σειρά OWASP Top 10 ίσως είναι καλύτερα να διαβάσετε τις ενότητες [Ρίσκα Ασφαλείας API](#) και [Μεθοδολογία και Δεδομένα](#) πριν μεταβείτε στη λίστα Top 10.

Μπορείτε να συνεισφέρετε στο OWASP API Security Top 10 με τις ερωτήσεις, τα σχόλια και τις ιδέες σας στο ηλεκτρονικό «αποθετήριο» (repository) του έργου στο GitHub:

- <https://github.com/OWASP/API-Security/issues>
- <https://github.com/OWASP/API-Security/blob/master/CONTRIBUTING.md>

Μπορείτε να βρείτε το OWASP API Security Top 10 εδώ:

- [https://www.owasp.org/index.php/OWASP\\_API\\_Security\\_Project](https://www.owasp.org/index.php/OWASP_API_Security_Project)
- <https://github.com/OWASP/API-Security>

Θέλουμε να ευχαριστήσουμε όλους τους συντελεστές που κατέστησαν δυνατό αυτό το έργο με την προσπάθεια και τη συνεισφορά τους. Όλοι τους αναφέρονται στην ενότητα [Ευχαριστίες](#). Σας ευχαριστούμε!



## Καλώς ήρθατε στο OWASP API Security Top 10 - 2019!

Καλώς ήρθατε στην πρώτη έκδοση του OWASP API Security Top 10. Εάν είστε εξοικειωμένοι με τη σειρά OWASP Top 10 τότε θα παρατηρήσετε τις ομοιότητες μεταξύ των δύο καταγραφών: στόχος τους είναι η εύκολη ανάγνωση (readability) και η εύκολη υιοθέτηση (adoption). Σε διαφορετική περίπτωση, εξετάστε το ενδεχόμενο να επισκεφτείτε τη σελίδα [OWASP API Security Project wiki page](#), προτού εμβαθύνετε στους πιο κρίσιμους κινδύνους για την ασφάλεια των APIs.

Τα APIs παίζουν πολύ σημαντικό ρόλο στην αρχιτεκτονική των σύγχρονων εφαρμογών. Εφόσον η δράση ευαισθητοποίησης του κοινού για την ασφάλεια και η καινοτομία έχουν διαφορετικούς ρυθμούς, είναι σημαντικό να εστιάσετε στις συνηθισμένες αδυναμίες ασφαλείας APIs.

Ο πρωταρχικός στόχος του OWASP API Security Top 10 είναι να εκπαιδεύσει όσους εμπλέκονται στην ανάπτυξη και συντήρηση APIs, για παράδειγμα, προγραμματιστές, σχεδιαστές, αρχιτέκτονες, διαχειριστές ή επιχειρήσεις / οργανισμούς.

Στην ενότητα [Μεθοδολογία και Δεδομένα](#), μπορείτε να διαβάσετε περισσότερα για το πώς δημιουργήθηκε αυτή η πρώτη έκδοση. Σε μελλοντικές εκδόσεις, θέλουμε να συνεργαστούμε με τις επιχειρήσεις του τομέα της ασφαλείας εφαρμογών (security industry), με μια δημόσια πρόσκληση για διαμοιρασμό δεδομένων. Προς το παρόν, ενθαρρύνουμε όλους να συνεισφέρουν με ερωτήσεις, σχόλια και ιδέες στο [αποθετήριο \(repository\)](#) [GitHub](#) ή στη [Λίστα αλληλογραφίας](#).

Αυτή είναι η πρώτη έκδοση του OWASP API Security Top 10. Σκοπός μας είναι η έκδοση να ενημερώνεται περιοδικά κάθε τρία ή τέσσερα χρόνια.

Σε μελλοντικές εκδόσεις, σε αντίθεση με την παρούσα έκδοση, θέλουμε να κάνουμε μια δημόσια πρόσκληση για δεδομένα, σε συνεργασία με τις επιχειρήσεις του τομέα της ασφάλειας εφαρμογών (security industry) σε αυτή την προσπάθεια. Στην ενότητα [Μεθοδολογία και Δεδομένα](#) θα βρείτε περισσότερες πληροφορίες σχετικά με τον τρόπο δημιουργίας αυτής της έκδοσης. Για περισσότερες λεπτομέρειες σχετικά με τους κινδύνους ασφαλείας, ανατρέξτε στην ενότητα [Ρίσκα Ασφαλείας API](#).

Είναι σημαντικό να συνειδητοποιήσουμε ότι τα τελευταία χρόνια, η αρχιτεκτονική των εφαρμογών έχει αλλάξει σημαντικά. Επί του παρόντος, τα APIs διαδραματίζουν πολύ σημαντικό ρόλο σε αυτή τη νέα αρχιτεκτονική των μικροϋπηρεσιών (microservices), των εφαρμογών μιας σελίδας (SPA), των εφαρμογών για κινητά, του IoT κ.λπ.

Το OWASP API Security Top 10 ήταν μια απαραίτητη προσπάθεια για την ευαισθητοποίηση του κοινού σχετικά με τα σύγχρονα ζητήματα ασφαλείας των APIs. Αυτό ήταν εφικτό μόνο χάρη στη σημαντική προσπάθεια πολλών εθελοντών οι οποίοι αναφέρονται στην ενότητα [Ευχαριστίες](#). Σας ευχαριστούμε!

Για την ανάλυση κινδύνου χρησιμοποιήθηκε η [Μεθοδολογία Αξιολόγησης Ρίσκου OWASP](#).

Ο παρακάτω πίνακας συνοψίζει την ορολογία που σχετίζεται με τη βαθμολογία κινδύνου.

Παράγοντες Απειλής (Threat Agents)	Εκμεταλλευσιμότητα (Exploitability)	Επικράτηση Αδυναμίας (Weakness Prevalence)	Ανιχνευσιμότητα Αδυναμίας (Weakness Detectability)	Τεχνικός Αντίκτυπος (Technical Impact)	Επιχειρησιακές Επιπτώσεις (Business Impacts)
Εξαρτώνται από το API	Εύκολη: 3	Διαδεδομένη 3	Εύκολη 3	Σοβαρός 3	Εξαρτώνται από την επιχείρηση
	Μεσαία: 2	Κοινή 2	Μεσαία 2	Μεσαίος 2	
	Δύσκολη: 1	Δύσκολη 1	Δύσκολη 1	Μικρός 1	

**Σημείωση:** Η παραπάνω προσέγγιση δεν λαμβάνει υπόψη την πιθανότητα του παράγοντα απειλής (likelihood of threat agent). Επίσης δεν λαμβάνει υπόψη καμία από τις διάφορες τεχνικές λεπτομέρειες που σχετίζονται με τη συγκεκριμένη εφαρμογή σας. Οποιοσδήποτε από αυτούς τους παράγοντες θα μπορούσε να επηρεάσει σημαντικά τη συνολική πιθανότητα ο εισβολέας να βρει και να εκμεταλλευτεί μια συγκεκριμένη ευπάθεια. Αυτή η βαθμολογία δεν λαμβάνει υπόψη τον πραγματικό αντίκτυπο στην επιχείρησή σας. Η επιχείρηση/οργανισμός σας θα πρέπει να αποφασίσει πόσο ρίσκο ασφαλείας από τις εφαρμογές και τα APIs είναι διατεθειμένος να αποδεχτεί, δεδομένης της εταιρικής κουλτούρας/πολιτικών, του κλάδου και του ρυθμιστικού περιβάλλοντος. Ο σκοπός του OWASP API Security Top 10 δεν είναι να κάνει αυτήν την ανάλυση κινδύνου για εσάς.

## Αναφορές (References)

### Αναφορές OWASP

- [OWASP Risk Rating Methodology](#)
- [Article on Threat/Risk Modeling](#)

### Εξωτερικές Αναφορές

- [ISO 31000: Risk Management Std](#)
- [ISO 27001: ISMS](#)
- [NIST Cyber Framework \(US\)](#)
- [ASD Strategic Mitigations \(AU\)](#)
- [NIST CVSS 3.0](#)
- [Microsoft Threat Modeling Tool](#)

API1:2019 - Broken Object Level Authorization	Πολύ συχνά, τα APIs εκθέτουν τελικά σημεία προορισμού (endpoints) που χειρίζονται αναγνωριστικά αντικειμένων (object IDs), δημιουργώντας μία ευρεία επιφάνεια έκθεσης σε πιθανές επιθέσεις επιπέδου ελέγχου πρόσβασης (wide attack surface Level Access Control). Οι έλεγχοι εξουσιοδότησης σε επίπεδο αντικειμένου (Object Level Authorization) θα πρέπει να λαμβάνονται υπόψη σε κάθε λειτουργία που έχει πρόσβαση σε δεδομένα προερχόμενα από χρήστες.
API2:2019 - Broken User Authentication	Οι μηχανισμοί ελέγχου ταυτότητας χρηστών συχνά υλοποιούνται εσφαλμένα, επιτρέποντας στους εισβολείς να διακυβεύουν τα διακριτικά ελέγχου ταυτότητας ή να εκμεταλλεύονται ελαττώματα στην υλοποίηση της εφαρμογής για να προσποούνται προσωρινά ή μόνιμα τις ταυτότητες άλλων χρηστών. Όταν διακυβεύεται η ικανότητα του συστήματος να αναγνωρίζει τον χρήστη / εφαρμογή-πελάτη (client) τότε θέτεται σε κίνδυνο η ασφάλεια των APIs συνολικά.
API3:2019 - Excessive Data Exposure	Στοχεύοντας σε γενικευμένες υλοποιήσεις (generic implementations), οι προγραμματιστές τείνουν να εκθέτουν όλα τα δεδομένα/ιδιότητες των αντικειμένων (object properties) χωρίς να λαμβάνουν υπόψη την ατομική τους ευαισθησία/εμπιστευτικότητα, βασιζόμενοι ότι οι εφαρμογές-πελάτες θα εκτελέσουν το φιλτράρισμα δεδομένων πριν τα εμφανίσουν στον χρήστη.
API4:2019 - Lack of Resources & Rate Limiting	Πολύ συχνά, τα APIs δεν επιβάλλουν περιορισμούς στο μέγεθος ή τον αριθμό των πόρων που μπορεί να ζητήσει ο χρήστης / εφαρμογή-πελάτη (client). Αυτό όχι μόνο μπορεί να επηρεάσει την απόδοση του διακομιστή API, οδηγώντας σε άρνηση υπηρεσίας (DoS), αλλά αφήνει επίσης ανοιχτό το ενδεχόμενο ελαττωμάτων ελέγχου ταυτότητας, όπως η ωμή βία (brute force).
API5:2019 - Broken Function Level Authorization	Πολύπλοκες πολιτικές ελέγχου πρόσβασης με διαφορετικές ιεραρχίες, ομάδες και ρόλους, καθώς και ασαφείς διαχωρισμοί μεταξύ διαχειριστικών/διαβαθμισμένων (administrative) και μη-διαβαθμισμένων λειτουργιών, τείνουν να οδηγούν σε ελαττώματα εξουσιοδότησης. Εκμεταλλευόμενοι αυτά τα ζητήματα, οι εισβολείς αποκτούν πρόσβαση σε πόρους ή/και διαβαθμισμένες λειτουργίες άλλων χρηστών.
API6:2019 - Mass Assignment	Η σύνδεση δεδομένων που προέρχονται από χρήστες ή εφαρμογές-πελάτες (π.χ. JSON) σε μοντέλα δεδομένων, χωρίς το κατάλληλο φιλτράρισμα ιδιοτήτων βάσει μιας λίστας επιτρεπόμενων δεδομένων (whitelist), συνήθως οδηγεί σε μαζική εκχώρηση δεδομένων (Mass Assignment). Οι εισβολείς καταφέρνουν να τροποποιήσουν τις ιδιότητες αντικειμένων που δεν θα έπρεπε να έχουν πρόσβαση να τροποποιήσουν με διάφορους τρόπους, όπως με το να μαντεύουν τις ιδιότητες των αντικειμένων, την εξερεύνηση άλλων τελικών σημείων προορισμού (API endpoints), την μελέτη της τεκμηρίωσης του λογισμικού (documentation) ή την παροχή πρόσθετων ιδιοτήτων αντικειμένων σε ωφέλιμα φορτία αιτημάτων (request payloads).
API7:2019 - Security Misconfiguration	Οι εσφαλμένες ρυθμίσεις ασφαλείας είναι συνήθως αποτέλεσμα μη ασφαλών προεπιλεγμένων ρυθμίσεων, ελλειπών ή αυτοσχέδιων/πρόχειρων (ad-hoc) ρυθμίσεων, χώρου αποθήκευσης cloud με ελλείψεις ή χωρίς περιορισμούς χρήσης, εσφαλμένων ρυθμίσεων κεφαλίδων HTTP (headers), περιττών μεθόδων HTTP, επιτρεπόμενης κοινής χρήσης πόρων Cross-Origin (CORS) και λεπτομερών μηνυμάτων σφάλματος που περιέχουν ευαίσθητες πληροφορίες.
API8:2019 - Injection	Σφάλματα Έγχυσης (Injection Flaws), όπως SQL, NoSQL, Command Injection κ.λπ., συμβαίνουν όταν αποστέλλονται μη αξιόπιστα δεδομένα σε έναν διερμηνέα λογισμικού ως μέρος μιας εντολής ή ερωτήματος (query). Τα κακόβουλα δεδομένα του εισβολέα μπορούν να ξεγελάσουν τον διερμηνέα ώστε να εκτελέσει ακούσιες εντολές ή να αποκτήσει πρόσβαση σε δεδομένα χωρίς την κατάλληλη εξουσιοδότηση.
API9:2019 - Improper Assets Management	Τα APIs τείνουν να εκθέτουν περισσότερα τελικά σημεία προορισμού (endpoints) από τις παραδοσιακές εφαρμογές Ιστού, καθιστώντας τη σωστή και ενημερωμένη τεκμηρίωση (documentation) εξαιρετικά σημαντική. Επίσης σημαντικό ρόλο για τον μετριασμό ελαττωμάτων όπως οι καταργημένες εκδόσεις APIs και τα εκτεθειμένα τελικά σημεία εντοπισμού σφαλμάτων είναι η μεθοδική καταγραφή των servers και των εκδόσεων APIs που έχουν αναπτυχθεί.
API10:2019 - Insufficient Logging & Monitoring	Η ανεπαρκής καταγραφή και παρακολούθηση/εποπτεία, σε συνδυασμό με την έλλειψη ή την αναποτελεσματική ενσωμάτωση και διαλειτουργικότητα με την αντιμετώπιση συμβάντων (incident response), επιτρέπει στους επιτιθέμενους να επεκτείνουν τις επιθέσεις τους, να διατηρήσουν την πρόσβαση τους, να στραφούν σε περισσότερα συστήματα για να παραβιάσουν, να εξαγάγουν ή να καταστρέψουν δεδομένα. Οι περισσότερες μελέτες παραβίασης αποδεικνύουν ότι ο μέσος χρόνος που απαιτείται για τον εντοπισμό μιας παραβίασης ξεπερνάει τις 200 ημέρες και συνήθως εντοπίζεται από εξωτερικούς συνεργάτες και όχι από ενδοεταρχικές διαδικασίες ή παρακολούθηση/εποπτεία των συστημάτων.

<div><div>Threat Agents</div><div>Attack Vectors</div><div>Security Weakness</div><div>Impacts</div></div>					
Εξαρτώνται από το API	Εκμεταλλευσιμότητα : 3	Επικράτηση (Prevalence): 3	Ανιχνευσιμότητα : 2	Τεχνικές Επιπτώσεις : 3	Εξαρτώνται από την Επιχείρηση
Οι εισβολείς μπορούν να εκμεταλλευτούν τελικά σημεία προορισμού API (endpoints) που είναι ευάλωτα σε εσφαλμένη εξουσιοδότηση επιπέδου αντικειμένου (Broken Object Level Authorization), παραποιώντας το αναγνωριστικό ενός αντικειμένου (object ID) που αποστέλλεται εντός ενός αιτήματος (request). Αυτό μπορεί να οδηγήσει σε μη εξουσιοδοτημένη πρόσβαση σε ευαίσθητα δεδομένα. Αυτό το κενό ασφαλείας είναι εξαιρετικά κοινό σε εφαρμογές που βασίζονται σε API, επειδή ο διακομιστής (server) συνήθως δεν έχει γνώση της κατάστασης της εφαρμογής-πελάτη και, αντ' αυτού, βασίζεται σε παραμέτρους όπως τα αναγνωριστικά αντικειμένων, που αποστέλλονται από την εφαρμογή-πελάτη για να αποφασίσει σε ποια αντικείμενα θα έχει πρόσβαση.		Αυτή είναι η πιο κοινή και ιδιαίτερα αποτελεσματική επίθεση σε API. Οι μηχανισμοί εξουσιοδότησης και ελέγχου πρόσβασης στις σύγχρονες εφαρμογές είναι περίπλοκοι και ευρέως διαδεδομένοι. Ακόμα κι αν η εφαρμογή εφαρμόζει μια κατάλληλη υποδομή για ελέγχους εξουσιοδότησης, οι προγραμματιστές μπορεί να ξεχάσουν να χρησιμοποιήσουν αυτούς τους ελέγχους πριν παραχωρήσουν πρόσβαση σε ένα ευαίσθητο/διαβαθμισμένο αντικείμενο. Η ανίχνευση ελέγχου πρόσβασης συνήθως δεν υπόκειται σε αυτοματοποιημένους στατικούς ή δυναμικούς ελέγχους (static or dynamic testing).		Η μη εξουσιοδοτημένη πρόσβαση μπορεί να οδηγήσει σε αποκάλυψη δεδομένων σε μη εξουσιοδοτημένα μέρη, απώλεια δεδομένων ή παραποίηση δεδομένων. Η μη εξουσιοδοτημένη πρόσβαση σε αντικείμενα μπορεί επίσης να οδηγήσει τους επιτιθέμενους να αποκτήσουν πλήρη πρόσβαση του λογαριασμού ενός χρήστη (account takeover).	

## Πότε το API είναι ευάλωτο

Η εξουσιοδότηση επιπέδου αντικειμένου (Object Level Authorization) είναι ένας μηχανισμός ελέγχου πρόσβασης που συνήθως υλοποιείται σε επίπεδο κώδικα για να επιβεβαιώσει ότι ένας χρήστης μπορεί να έχει πρόσβαση μόνο σε αντικείμενα (objects) στα οποία θα έπρεπε να έχει πρόσβαση.

Κάθε τελικό σημείο προορισμού API (endpoint) που λαμβάνει ένα αναγνωριστικό ενός αντικειμένου (object ID) και εκτελεί οποιονδήποτε τύπο ενέργειας στο αντικείμενο, θα πρέπει να εφαρμόζει ελέγχους εξουσιοδότησης σε επίπεδο αντικειμένου. Οι έλεγχοι θα πρέπει να επικυρώνουν ότι ο συνδεδεμένος χρήστης έχει πρόσβαση για να εκτελέσει την απαιτούμενη ενέργεια στο ζητούμενο αντικείμενο.

Οι αποτυχίες σε αυτόν τον μηχανισμό συνήθως οδηγούν σε μη εξουσιοδοτημένη αποκάλυψη πληροφοριών, τροποποίηση ή καταστροφή όλων των δεδομένων.

## Παραδείγματα από Σενάρια Επίθεσης

### Σενάριο Επίθεσης #1

Μια πλατφόρμα ηλεκτρονικού εμπορίου για ηλεκτρονικά καταστήματα παρέχει μια σελίδα με τα διαγράμματα εσόδων για τα καταστήματα που φιλοξενεί. Επιθεωρώντας τα αιτήματα του προγράμματος περιήγησης



(browser requests), ένας εισβολέας μπορεί να αναγνωρίσει τα τελικά σημεία προορισμού API (endpoints) που χρησιμοποιούνται ως πηγή δεδομένων για αυτά τα γραφήματα και το μοτίβο URL τους το οποίο είναι της μορφής: `/shops/{shopName}/revenue_data.json`. Χρησιμοποιώντας ένα άλλο τελικό σημείο προορισμού API (endpoint), ο εισβολέας λαμβάνει τη λίστα με όλα τα ονόματα καταστημάτων που φιλοξενούνται. Ο εισβολέας χρησιμοποιεί ένα απλό script, το οποίο διαχειρίζεται τα ονόματα στη λίστα, αντικαθιστώντας το `{shopName}` στη διεύθυνση URL. Με αυτό τον τρόπο αποκτά πρόσβαση στα δεδομένα πωλήσεων χιλιάδων καταστημάτων ηλεκτρονικού εμπορίου.

## Σενάριο Επίθεσης #2

Κατά την παρακολούθηση της κυκλοφορίας δικτύου μιας φορητής συσκευής, το ακόλουθο αίτημα HTTP PATCH τραβά την προσοχή ενός εισβολέα λόγω της παρουσίας μιας custom κεφαλίδας αιτήματος HTTP `X-User-Id: 54796`. Αντικαθιστώντας την τιμή `X-User-Id` με `54795`, ο εισβολέας λαμβάνει μια επιτυχημένη απάντηση HTTP και μπορεί να τροποποιήσει τα δεδομένα λογαριασμού άλλων χρηστών.





## Τρόπος Πρόληψης

- Δημιουργήστε έναν σωστό μηχανισμό εξουσιοδότησης που βασίζεται στις πολιτικές των χρηστών (user policies) και την ιεραρχία τους.
- Χρησιμοποιήστε έναν μηχανισμό εξουσιοδότησης σε κάθε συνάρτηση (function) που χρησιμοποιεί δεδομένα εισόδου από τη εφαρμογή-πελάτη (client) για να αποκτήσει πρόσβαση σε μια εγγραφή στη βάση δεδομένων. Ο μηχανισμός πρέπει να ελέγχει εάν ο συνδεδεμένος χρήστης έχει πρόσβαση να εκτελέσει την απαιτούμενη ενέργεια στην εγγραφή.
- Προτιμήστε να χρησιμοποιείτε τυχαίες και απρόβλεπτες τιμές ως GUID για τα αναγνωριστικά (object IDs) των εγγραφών.
- Γράψτε αυτοματοποιημένους ελέγχους (tests) για την αξιολόγηση του μηχανισμού εξουσιοδότησης. Μην βγάζετε στην παραγωγή (deploy) ευάλωτες αλλαγές που σπάνε τα tests.

## Αναφορές (References)

### Εξωτερικές Αναφορές

- [CWE-284: Improper Access Control](#)
- [CWE-285: Improper Authorization](#)
- [CWE-639: Authorization Bypass Through User-Controlled Key](#)

<div>Threat Agents</div> 		<div>Attack Vectors</div> 	<div>Security Weakness</div> 	<div>Impacts</div> 	
Εξαρτώνται από το API	Εκμεταλλευσιμότητα: 3	Επικράτηση (Prevalence): 2	Ανιχνευσιμότητα : 2	Τεχνικές Επιπτώσεις : 3	Εξαρτώνται από την Επιχείρηση
Ο έλεγχος ταυτότητας στα APIs είναι ένας πολύπλοκος και μπερδεμένος μηχανισμός. Οι μηχανικοί λογισμικού και ασφαλείας ενδέχεται να έχουν λανθασμένες αντιλήψεις σχετικά με τα όρια του ελέγχου ταυτότητας και πώς να τον εφαρμόσουν σωστά. Επιπλέον, ο μηχανισμός ελέγχου ταυτότητας είναι ένας εύκολος στόχος για τους εισβολείς, καθώς είναι προσβάσιμος σε όλους. Αυτοί οι δύο λόγοι καθιστούν τον μηχανισμό ελέγχου ταυτότητας δυνητικά ευάλωτο σε πολλές επιθέσεις (exploits).		Υπάρχουν δύο υποκατηγορίες του κενού ασφαλείας: 1. Έλλειψη μηχανισμών προστασίας: Τα τελικά σημεία προορισμού APIs που είναι υπεύθυνα για τον έλεγχο ταυτότητας πρέπει να αντιμετωπίζονται διαφορετικά από τα κανονικά τελικά σημεία προορισμού και να εφαρμόζουν επιπλέον επίπεδα προστασίας. 2. Εσφαλμένη εφαρμογή του μηχανισμού: Ο μηχανισμός χρησιμοποιείται / υλοποιείται χωρίς να λαμβάνονται υπόψη τα διανύσματα επίθεσης (attack vectors) ή είναι λάθος η περίπτωση χρήσης του (π.χ. ένας μηχανισμός ελέγχου ταυτότητας που έχει σχεδιαστεί για εφαρμογές-πελάτες (clients) IoT μπορεί να μην είναι η σωστή επιλογή για εφαρμογές Ιστού).		Οι εισβολείς μπορούν να αποκτήσουν τον έλεγχο λογαριασμών άλλων χρηστών στο σύστημα, να διαβάσουν τα προσωπικά τους δεδομένα και να εκτελέσουν ευαίσθητες / διαβαθμισμένες ενέργειες για λογαριασμό τους, όπως συναλλαγές χρημάτων και αποστολή προσωπικών μηνυμάτων.	

## Πότε το API είναι ευάλωτο

Τα τελικά σημεία προορισμού (endpoints) και οι ροές (flows) ελέγχου ταυτότητας είναι στοιχεία που πρέπει να προστατεύονται. Λειτουργίες όπως το "Ξέχασα τον κωδικό πρόσβασης / επαναφορά κωδικού πρόσβασης" θα πρέπει να αντιμετωπίζονται με τον ίδιο τρόπο όπως και οι μηχανισμοί ελέγχου ταυτότητας.

Ένα API είναι ευάλωτο εάν:

- Επιτρέπει [credential stuffing](#) με το οποίο ο εισβολέας έχει την δυνατότητα να αυτοματοποιήσει την προσπάθεια πρόσβασης χρησιμοποιώντας κλεμμένες λίστες με έγκυρα ονόματα χρηστών και κωδικών πρόσβασης.
- Επιτρέπει στους εισβολείς να εκτελούν επίθεση ωμής βίας (brute force attack) στον ίδιο λογαριασμό χρήστη, χωρίς να παρουσιάζουν μηχανισμό captcha ή κλειδώματος λογαριασμού.
- Επιτρέπει αδύναμους κωδικούς πρόσβασης.
- Στέλνει ευαίσθητες λεπτομέρειες ελέγχου ταυτότητας, όπως διακριτικά ταυτότητας (authentication tokens) και κωδικούς πρόσβασης στη διεύθυνση URL (δηλαδή μέσω χρήσης GET HTTP requests).
- Δεν επικυρώνει την αυθεντικότητα των διακριτικών.
- Αποδέχεται ανυπόγραφα/ασθενώς υπογεγραμμένα διακριτικά JWT ("alg":"none")/δεν επικυρώνει την ημερομηνία λήξης τους.
- Διαχειρίζεται κωδικούς πρόσβασης ως απλό κείμενο, χωρίς την χρήση κρυπτογράφησης ή χρησιμοποιεί αδύναμους αλγόριθμους κρυπτογράφησης ή κατακερματισμού (hashing).
- Χρησιμοποιεί αδύναμα κλειδιά κρυπτογράφησης.

## Παραδείγματα από Σενάρια Επίθεσης

### Σενάριο Επίθεσης #1

Το [Credential stuffing](#) (χρησιμοποιώντας [λίστες γνωστών ονομάτων χρήστη/κωδικών πρόσβασης](#)), είναι μια συνηθισμένη επίθεση. Εάν μια εφαρμογή δεν εφαρμόζει προστασία από αυτοματοποιημένες απειλές ή credential stuffing, η εφαρμογή μπορεί να χρησιμοποιηθεί από τους εισβολείς ως ένα μέσο για να προσδιορίσουν εάν τα διαπιστευτήρια είναι έγκυρα. Η τεχνική αυτή μετατρέπει το API σε έναν ελεγκτή (ή μαντείο - oracle) κωδικών πρόσβασης.

## Σενάριο Επίθεσης #2

Ένας εισβολέας ξεκινάει τη διαδικασία ανάκτησης κωδικού πρόσβασης στέλνοντας ένα αίτημα HTTP POST στο `/api/system/verification-codes` και παρέχοντας το όνομα χρήστη στο σώμα του αιτήματος (request). Στη συνέχεια, αποστέλλεται μέσω SMS ένας κωδικός με 6 ψηφία στο τηλέφωνο του θύματος. Επειδή το API δεν διαθέτει προστασία περιορισμού του ρυθμού των αιτημάτων, ο εισβολέας μπορεί να δοκιμάσει όλους τους πιθανούς συνδυασμούς χρησιμοποιώντας μία πολυνηματική εφαρμογή (multi-threaded script) η οποία στέλνει πολλαπλά αιτήματα στο σημείο προορισμού (URL endpoint) `/api/system/verification-codes/{smsToken}` έως ότου ανακαλύψει τον σωστό 6-ψήφιο κωδικό. Ένα τέτοιο σενάριο επίθεσης μπορεί να ολοκληρωθεί μέσα σε λίγα λεπτά.

## Τρόπος Πρόληψης

- Βεβαιωθείτε ότι γνωρίζετε όλες τις πιθανές ροές εκτέλεσης (execution flows) για έλεγχο ταυτότητας στο API (σύνδεσμοι για κινητά/ιστό/deep links που εφαρμόζουν έλεγχο ταυτότητας με ένα κλικ/κ.λπ.)
- Επιβεβαιώστε με τους μηχανικούς σας όλες τις ροές εκτέλεσης.
- Διαβάστε σχετικά με τους μηχανισμούς ελέγχου ταυτότητας. Βεβαιωθείτε ότι καταλαβαίνετε τι και πώς χρησιμοποιούνται. Το OAuth δεν είναι έλεγχος ταυτότητας, ούτε και τα κλειδιά API.
- Μην ανακαλύπτετε ξανά τον τροχό στον έλεγχο ταυτότητας, τη δημιουργία διακριτικών, τους τρόπους αποθήκευσης κωδικών πρόσβασης. Χρησιμοποιήστε τις καθιερωμένες προδιαγραφές (standards).
- Τα τελικά σημεία ανάκτησης διαπιστευτηρίων/λήψης κωδικού πρόσβασης θα πρέπει να αντιμετωπίζονται ως τελικά σημεία σύνδεσης. Εφαρμόστε τα ίδια συστήματα ασφαλείας κατά επιθέσεων όπως την ωμή βία (brute force), τον περιορισμό του ρυθμού (rate limiting) και τις προστασίες κλειδώματος.
- Συμβουλευτείτε και χρησιμοποιήστε το [OWASP Authentication Cheatsheet](#).
- Where possible, implement multi-factor authentication.
- Όπου είναι δυνατόν, εφαρμόστε έλεγχο ταυτότητας πολλαπλών παραγόντων (multi-factor authentication).
- Εφαρμόστε μηχανισμούς κατά της ωμής βίας για τον μετριασμό του credential stuffing, της επίθεσης λεξικού και των επιθέσεων ωμής βίας στα τελικά σημεία ελέγχου ταυτότητας. Αυτός ο μηχανισμός θα πρέπει να είναι πιο αυστηρός από τον κανονικό μηχανισμό περιορισμού ρυθμών στο API σας.
- Εφαρμόστε το μηχανισμό [account lockout](#) / captcha για να αποτρέψετε την ωμή βία εναντίον συγκεκριμένων χρηστών. Εφαρμόστε ελέγχους αδύναμου κωδικού πρόσβασης.
- Τα κλειδιά API δεν πρέπει να χρησιμοποιούνται για έλεγχο ταυτότητας χρήστη, αλλά για [client app/project authentication](#).





## Αναφορές (References)

### Αναφορές OWASP

- [OWASP Key Management Cheat Sheet](#)
- [OWASP Authentication Cheatsheet](#)
- [Credential Stuffing](#)

### Εξωτερικές Αναφορές

- [CWE-798: Use of Hard-coded Credentials](#)

<div>Threat Agents</div> 		<div>Attack Vectors</div> 	<div>Security Weakness</div> 		<div>Impacts</div> 
Εξαρτώνται από το API	Εκμεταλλευσιμότητα: 3	Επικράτηση (Prevalence): 2	Ανιχνευσιμότητα: 2	Τεχνικές Επιπτώσεις: 2	Εξαρτώνται από την Επιχείρηση
Η εκμετάλλευση (exploitation) της υπερβολικής έκθεσης δεδομένων (Excessive Data Exposure) είναι απλή και συνήθως πραγματοποιείται ανιχνεύοντας και αναλύοντας την κίνηση των δεδομένων που επιστρέφονται από το API στον χρήστη. Στόχος της ανάλυσης των δεδομένων είναι η αναζήτηση πιθανής έκθεσης σε ευαίσθητα δεδομένα που δεν πρέπει να επιστραφούν στον χρήστη.		Το κενό ασφαλείας εμφανίζεται όταν APIs βασίζονται στις εφαρμογές-πελάτες (client) για την εκτέλεση του φιλτραρίσματος δεδομένων. Δεδομένου ότι τα APIs χρησιμοποιούνται ως πηγές δεδομένων, συχνά οι προγραμματιστές προσπαθούν να τα υλοποιήσουν με γενικό τρόπο χωρίς να σκεφτούν την ευαισθησία των δεδομένων που εκτίθενται. Τα αυτόματα εργαλεία συνήθως δεν μπορούν να εντοπίσουν αυτόν τον τύπο ευπάθειας, επειδή είναι δύσκολο να γίνει διάκριση μεταξύ των μη-ευαίσθητων δεδομένων που επιστρέφονται από το API και των ευαίσθητων δεδομένων που δεν πρέπει να επιστραφούν. Η δυσκολία αυτή προκύπτει επειδή τα αυτόματα εργαλεία δεν κατανοούν σε βάθος την εφαρμογή.			Η υπερβολική έκθεση δεδομένων οδηγεί συνήθως σε έκθεση ευαίσθητων δεδομένων.

## Πότε το API είναι ευάλωτο

Το API επιστρέφει ευαίσθητα δεδομένα στην εφαρμογή-πελάτη (client) βάσει σχεδίασης. Αυτά τα δεδομένα συνήθως φιλτράρονται από την πλευρά της εφαρμογής-πελάτη πριν παρουσιαστούν στον χρήστη. Ένας εισβολέας μπορεί εύκολα να δει τα ευαίσθητα δεδομένα ανιχνεύοντας την κίνηση.

## Παραδείγματα από Σενάρια Επίθεσης

### Σενάριο Επίθεσης #1

Η ομάδα που ασχολείται με το development για κινητά χρησιμοποιεί το τελικό σημείο προορισμού `/api/articles/{articleId}/comments/{commentId}` στην σελίδα προβολής άρθρων για την εμφάνιση των μεταδεδομένων των σχολίων. Ανιχνεύοντας την κίνηση της εφαρμογής για κινητά, ένας εισβολέας ανακαλύπτει ότι επιστρέφονται και άλλα ευαίσθητα δεδομένα που σχετίζονται με τον συντάκτη του σχολίου. Η υλοποίηση τελικού σημείου προορισμού (endpoint), χρησιμοποιεί μια γενική μέθοδο `toJSON()` για τη σειριοποίηση (serialization) του αντικειμένου στο μοντέλο `User`, το οποίο περιέχει ευαίσθητα δεδομένα (PII).

### Σενάριο Επίθεσης #2

Ένα σύστημα επιτήρησης που βασίζεται στο IOT επιτρέπει στους διαχειριστές να δημιουργούν χρήστες με διαφορετικά δικαιώματα. Ένας διαχειριστής δημιούργησε έναν λογαριασμό χρήστη για έναν νέο φύλακα που θα πρέπει να έχει πρόσβαση μόνο σε συγκεκριμένα κτίρια στον ιστότοπο. Μόλις ο φύλακας χρησιμοποιήσει την εφαρμογή του για κινητά, ενεργοποιείται μια κλήση API στη διεύθυνση: `/api/sites/111/cameras` προκειμένου να ληφθούν δεδομένα σχετικά με τις διαθέσιμες κάμερες και να εμφανίστουν στον πίνακα ελέγχου. Η απάντηση περιέχει μια λίστα με λεπτομέρειες σχετικά με τις κάμερες στην ακόλουθη μορφή: `{"id": "xxx", "live_access_token": "xxxx-bbbbb", "building_id": "yyy"}`. Ενώ το GUI

της εφαρμογής-πελάτη εμφανίζει μόνο κάμερες στις οποίες θα πρέπει να έχει πρόσβαση ο φύλακας, η πραγματική απόκριση API περιέχει μια πλήρη λίστα με όλες τις κάμερες στον ιστότοπο.

### Τρόπος Πρόληψης





- Μην βασίζεστε ποτέ στην πλευρά της εφαρμογής-πελάτη για να φιλτράρετε ευαίσθητα δεδομένα.
- Ελέγξτε τις απαντήσεις από το API για να βεβαιωθείτε ότι περιέχουν μόνο αποδεκτά δεδομένα.
- Οι back-end προγραμματιστές θα πρέπει πάντα να διερωτώνται "ποιος είναι ο καταναλωτής των δεδομένων;", πριν εκθέσουν δημόσια ένα νέο τελικό σημείο προορισμού API.
- Αποφύγετε τη χρήση γενικών μεθόδων όπως `to_json()` και `to_string()`. Αντίθετα, επιλέξτε συγκεκριμένα πεδία που θέλετε πραγματικά να επιστρέψετε.
- Ταξινομήστε όλες τις ευαίσθητες και προσωπικά αναγνωρίσιμες πληροφορίες (PII) τις οποίες αποθηκεύει και συνεργάζεται η εφαρμογή σας, ελέγχοντας όλες τις κλήσεις API που επιστρέφουν τέτοιες πληροφορίες για να δείτε εάν αυτές οι απαντήσεις δημιουργούν πρόβλημα ασφαλείας.
- Εφαρμόστε έναν μηχανισμό επικύρωσης απόκρισης που βασίζεται σε σχήματα (schema-based) ως ένα επιπλέον επίπεδο ασφαλείας. Ως μέρος αυτού του μηχανισμού ορίστε και επιβάλλετε δεδομένα που επιστρέφονται από όλες τις μεθόδους API, συμπεριλαμβανομένων των σφαλμάτων.

### Αναφορές (References)

#### Εξωτερικές Αναφορές

- [CWE-213: Intentional Information Exposure](#)



<div>Threat Agents</div> 		<div>Attack Vectors</div> 	<div>Security Weakness</div> 	<div>Impacts</div> 	
Εξαρτώνται από το API	Εκμεταλλευσιμότητα: 2	Επικράτηση (Prevalence): 3	Ανιχνευσιμότητα: 3	Τεχνικές Επιπτώσεις : 2	Εξαρτώνται από την Επιχείρηση
Η εκμετάλλευση (exploitation) αυτής της αδυναμίας ασφαλείας απαιτεί μόνο την αποστολή απλών αιτημάτων (requests) στο API. Δεν απαιτείται έλεγχος ταυτότητας. Πολλαπλά ταυτόχρονα αιτήματα (requests) μπορούν να αποσταλούν από έναν μόνο τοπικό υπολογιστή ή χρησιμοποιώντας πόρους υπολογιστικού νέφους (cloud computing).		Είναι σύνηθες να βρίσκουμε API που δεν εφαρμόζουν περιορισμό ρυθμού (rate limiting) ή APIs όπου τα όρια περιορισμού δεν έχουν οριστεί σωστά.		Η εκμετάλλευση (exploitation) μπορεί να οδηγήσει σε επιθέσεις άρνησης υπηρεσιών (DoS), καθιστώντας το API μη ανταποκρινόμενο ή ακόμη και μη διαθέσιμο.	

## Πότε το API είναι ευάλωτο

Τα αιτήματα (requests) API καταναλώνουν πόρους (resources) όπως πόρους δικτύου, επεξεργαστή (CPU), μνήμης και αποθήκευσης. Το ποσό των πόρων που απαιτούνται για την ολοκλήρωση ενός αιτήματος εξαρτάται σε μεγάλο βαθμό από τα δεδομένα εισόδου του χρήστη και την επιχειρηματική λογική (business logic) του τελικού σημείου προορισμού (endpoint). Επίσης, λάβετε υπόψη το γεγονός ότι τα αιτήματα από πολλαπλούς πελάτες API ανταγωνίζονται για τους πόρους του συστήματος. Ένα API είναι ευάλωτο εάν τουλάχιστον ένα από τα ακόλουθα όρια περιορισμού λείπει ή τα όρια περιορισμού δεν έχουν οριστεί καταλλήλως (π.χ. πολύ χαμηλά/υψηλά):

- Χρονικά όρια εκτέλεσης (execution timeouts)
- Μέγιστο ποσό δέσμευσης μνήμης
- Αριθμός περιγραφών αρχείου (file descriptors)
- Αριθμός διεργασιών (processes)
- Μέγεθος αιτήματος ωφέλιμου φορτίου (request payload size) (π.χ. μεταφορτώσεις (uploads))
- Αριθμός αιτημάτων ανά πρόγραμμα-πελάτη/πόρο συστήματος
- Αριθμός εγγραφών ανά σελίδα που περιέχονται σε κάθε απάντηση αιτήματος (request response)

## Παραδείγματα από Σενάρια Επίθεσης

### Σενάριο #1

Ένας εισβολέας ανεβάζει μια μεγάλη εικόνα υποβάλλοντας ένα αίτημα POST στο `/api/v1/images`. Όταν ολοκληρωθεί η μεταφόρτωση, το API δημιουργεί πολλές μικρογραφίες (thumbnails) με διαφορετικά μεγέθη. Λόγω του μεγάλου μεγέθους της μεταφορτωμένης εικόνας, η διαθέσιμη μνήμη του συστήματος εξαντλείται κατά τη δημιουργία μικρογραφιών και το API σταματάει να ανταποκρίνεται.

### Σενάριο #2

Έχουμε μια εφαρμογή που περιέχει τη λίστα χρηστών σε μια διεπαφή χρήστη (UI) με όριο 200 χρήστες ανά σελίδα. Η λίστα των χρηστών ανακτάται από τον διακομιστή (server) χρησιμοποιώντας το ακόλουθο ερώτημα: `/api/users?page=1&size=200`. Ένας εισβολέας αλλάζει την παράμετρο μεγέθους σε `200 000`, προκαλώντας προβλήματα απόδοσης (performance) στη βάση δεδομένων. Τα προβλήματα απόδοσης της βάσης

δεδομένων σταματούν το API από το να ανταποκρίνεται και το API δεν είναι πλέον σε θέση να χειριστεί περαιτέρω αιτήματα από τον τρέχων ή άλλους πελάτες (γνωστό και ως DoS).

Το ίδιο σενάριο μπορεί να χρησιμοποιηθεί για την πρόκληση σφαλμάτων υπερχειλίσης ακεραίων (Integer Overflow) ή υπερχειλίσης buffer (Buffer Overflow).

## Τρόπος Πρόληψης

- Το Docker διευκολύνει τον περιορισμό της μνήμης, του επεξεργαστή CPU, τον αριθμό επανεκκινήσεων, των περιγραφών αρχείου (file descriptors), και διεργασιών (processes).
- Εφαρμόστε ένα όριο περιορισμού σχετικά με το πόσο συχνά ένα πρόγραμμα-πελάτης μπορεί να καλεί το API εντός ενός καθορισμένου χρονικού πλαισίου.
- Ειδοποιήστε το πρόγραμμα-πελάτη όταν γίνεται υπέρβαση του ορίου παρέχοντας τον αριθμό ορίου και την ώρα κατά την οποία θα γίνει επαναφορά του ορίου.
- Προσθέστε την κατάλληλη επικύρωση (validation) από την πλευρά του διακομιστή για τις παραμέτρους συμβολοσειράς ερωτήματος (query string) και σώματος αιτήματος (request body), ειδικά αυτή που ελέγχει τον αριθμό των εγγράφων που θα επιστραφούν στην απάντηση.
- Καθορίστε και επιβάλλετε μέγιστο μέγεθος δεδομένων σε όλες τις εισερχόμενες παραμέτρους και ωφέλιμα φορτία (payloads), όπως το μέγιστο μήκος για τις συμβολοσειρές (strings) και τον μέγιστο αριθμό στοιχείων σε πίνακες (elements in arrays).

## Αναφορές (References)

### Αναφορές OWASP

- [Blocking Brute Force Attacks](#)
- [Docker Cheat Sheet - Limit resources \(memory, CPU, file descriptors, processes, restarts\)](#)
- [REST Assessment Cheat Sheet](#)

### Εξωτερικές Αναφορές

- [CWE-307: Improper Restriction of Excessive Authentication Attempts](#)
- [CWE-770: Allocation of Resources Without Limits or Throttling](#)
- “Rate Limiting (Throttling)” - [Security Strategies for Microservices-based Application Systems](#), NIST

Threat Agents		Attack Vectors		Security Weakness		Impacts					
Εξαρτώνται από το API		Εκμεταλλευσιμότητα: 3		Επικράτηση (Prevalence): 2		Ανιχνευσιμότητα: 1		Τεχνικές Επιπτώσεις : 2		Εξαρτώνται από την Επιχείρηση	
Η εκμετάλλευση (exploitation) αυτής της αδυναμίας ασφαλείας απαιτεί από τον εισβολέα (attacker) να στείλει έγκυρες κλήσεις σε τελικό σημείο προορισμού API (endpoint) στο οποίο κανονικά δεν θα πρέπει να έχει πρόσβαση. Αυτά τα τελικά σημεία προορισμού ενδέχεται να εκτίθενται σε ανώνυμους χρήστες ή/και σε κανονικούς, μη προνομιούχους/εξουσιοδοτημένους χρήστες (non-privileged users). Είναι εύκολο να ανακαλύψετε τέτοιου είδους ελαττώματα σε APIs καθώς τα APIs έχουν δομή και ο τρόπος πρόσβασης σε ορισμένες λειτουργίες είναι εύκολα προβλέψιμος (π.χ. αντικατάσταση της μεθόδου HTTP από GET σε PUT ή αλλαγή της συμβολοσειράς "users" στη διεύθυνση URL σε "admins" ).				Οι έλεγχοι εξουσιοδότησης (authorization checks) για μια λειτουργία (function) ή έναν πόρο συστήματος (resource) βασίζονται συνήθως σε ρυθμίσεις (configuration) και μερικές φορές υλοποιούνται σε επίπεδο κώδικα. Η εφαρμογή των κατάλληλων και σωστών ελέγχων εξουσιοδότησης είναι συνήθως μια περίπλοκη εργασία, καθώς οι σύγχρονες εφαρμογές ενδέχεται να περιέχουν πολλούς τύπους ρόλων ή ομάδων και πολύπλοκη ιεραρχία χρηστών (π.χ. υπο-χρήστες, χρήστες με περισσότερους από έναν ρόλους).				Οι αδυναμίες ασφαλείας αυτές επιτρέπουν στους εισβολείς να έχουν πρόσβαση σε μη εξουσιοδοτημένες λειτουργίες. Οι διαχειριστικές λειτουργίες (administrative functions) είναι οι βασικοί στόχοι για αυτό το είδος επίθεσης.			

## Πότε το API είναι ευάλωτο

Ο καλύτερος τρόπος για να βρείτε αδυναμίες ασφαλείας εξουσιοδότησης σε επίπεδο λειτουργιών (Broken Function Level Authorization) είναι να πραγματοποιήσετε μια ενδελεχή ανάλυση (deep analysis) του μηχανισμού εξουσιοδότησης (authorization mechanism). Λάβετε υπόψιν σας την ιεραρχία των χρηστών, τους διαφορετικούς ρόλους ή/και τις ομάδες του συστήματος χρησιμοποιώντας τις ακόλουθες ερωτήσεις:

- Μπορεί ένας κανονικός χρήστης να έχει πρόσβαση στα τελικά σημεία διαχείρισης (administrative endpoints);
- Μπορεί ένας χρήστης να εκτελέσει ευαίσθητες ενέργειες (π.χ. δημιουργία, τροποποίηση ή διαγραφή) στις οποίες δεν θα έπρεπε να έχει πρόσβαση αλλάζοντας απλώς τη μέθοδο HTTP (π.χ. από "GET" σε "DELETE");
- Μπορεί ένας χρήστης από την ομάδα X να αποκτήσει πρόσβαση σε μια λειτουργία που θα πρέπει να εκτίθεται μόνο σε χρήστες από την ομάδα Y, μαντεύοντας απλώς τη διεύθυνση URL του τελικού σημείου προορισμού και τις παραμέτρους (π.χ. `/api/v1/users/export_all`);

Μην υποθέτετε ότι ένα τελικό σημείο API είναι κανονικό ή διαχειριστικό μόνο με βάση τη διαδρομή URL.

Παρόλο που οι προγραμματιστές ενδέχεται να επιλέξουν να εκθέσουν τα περισσότερα από τα διαχειριστικά τελικά σημεία προορισμού σε μια συγκεκριμένη σχετική διαδρομή, όπως `api/admins`, είναι πολύ συνηθισμένο να βρίσκουμε αυτά τα τελικά σημεία διαχείρισης και σε άλλες σχετικές διαδρομές μαζί με κανονικά τελικά σημεία προορισμού, όπως `api/users`.

## Παραδείγματα Σεναρίων Επίθεσης

### Σενάριο Επίθεσης #1

Κατά τη διαδικασία εγγραφής σε μια εφαρμογή που επιτρέπει τη συμμετοχή μόνο σε προσκεκλημένους χρήστες, η εφαρμογή για κινητά εκτελεί μια κλήση API στο `GET /api/invites/{invite_guid}`. Η απάντηση περιέχει ένα

JSON με λεπτομέρειες σχετικές με την πρόσκληση, συμπεριλαμβανομένου του ρόλου του χρήστη και του email του χρήστη.

Ένας εισβολέας αντέγραψε το αίτημα και άλλαξε τη μέθοδο HTTP και το τελικό σημείο σε `POST /api/invites/new`. Αυτό το τελικό σημείο προορισμού θα πρέπει να είναι προσβάσιμο μόνο από διαχειριστές που χρησιμοποιούν την κονσόλα διαχείρισης, το οποίο όμως τελικό σημείο προορισμού δεν εφαρμόζει ελέγχους εξουσιοδότησης σε επίπεδο λειτουργίας.

Ο εισβολέας εκμεταλλεύεται το πρόβλημα και στέλνει στον εαυτό του μια πρόσκληση για να δημιουργήσει έναν λογαριασμό διαχειριστή:

```
POST /api/invites/new
{"email": "hugo@malicious.com", "role": "admin"}
```

## Σενάριο Επίθεσης #2

Ένα API περιέχει ένα τελικό σημείο προορισμού (endpoint) που θα πρέπει να είναι προσβάσιμο μόνο στους διαχειριστές - `GET /api/admin/v1/users/all`. Αυτό το τελικό σημείο προορισμού επιστρέφει τα στοιχεία όλων των χρηστών της εφαρμογής και δεν εφαρμόζει ελέγχους εξουσιοδότησης σε επίπεδο λειτουργίας. Ένας εισβολέας που έμαθε τη δομή του API κάνει μια πιθανή εικασία και καταφέρνει να αποκτήσει πρόσβαση σε αυτό το τελικό σημείο προορισμού, το οποίο εκθέτει ευαίσθητες λεπτομέρειες των χρηστών της εφαρμογής.

## Τρόπος Πρόληψης

Η εφαρμογή σας θα πρέπει να διαθέτει ένα σταθερό και εύκολο στην ανάλυση υποσύστημα εξουσιοδότησης (authorization module) το οποίο θα χρησιμοποιείται από όλες τις επιχειρησιακές λειτουργίες (business functions). Συχνά, μια τέτοια προστασία παρέχεται από ένα ή περισσότερα υποσυστήματα (components) που βρίσκονται εκτός του κώδικα της εφαρμογής.

- Οι μηχανισμοί επιβολής θα πρέπει από προεπιλογή (by default) να απαγορεύουν κάθε πρόσβαση, απαιτώντας ρητές εξουσιοδοτήσεις (explicit grants) σε συγκεκριμένους ρόλους για πρόσβαση σε κάθε λειτουργία.
- Ελέγξτε τα τελικά σημεία προορισμού του API σας σε σχέση με τις αδυναμίες ασφαλείας εξουσιοδότησης σε επίπεδο λειτουργίας (function level), λαμβάνοντας παράλληλα υπόψη την επιχειρησιακή λογική (business logic) της εφαρμογής και της ιεραρχίας των ομάδων χρηστών.
- Βεβαιωθείτε ότι όλα τα διαχειριστικά υποσυστήματα (controllers) βασίζουν την λειτουργία τους σε ένα γενικό διαχειριστικό υποσύστημα (abstract controller) που εφαρμόζει ελέγχους εξουσιοδότησης βάσει της ομάδας και του ρόλου του χρήστη.
- Βεβαιωθείτε ότι οι λειτουργίες διαχείρισης μέσα σε ένα κανονικό διαχειριστικό υποσύστημα (controller) εφαρμόζουν ελέγχους εξουσιοδότησης βάσει της ομάδας και του ρόλου του χρήστη.

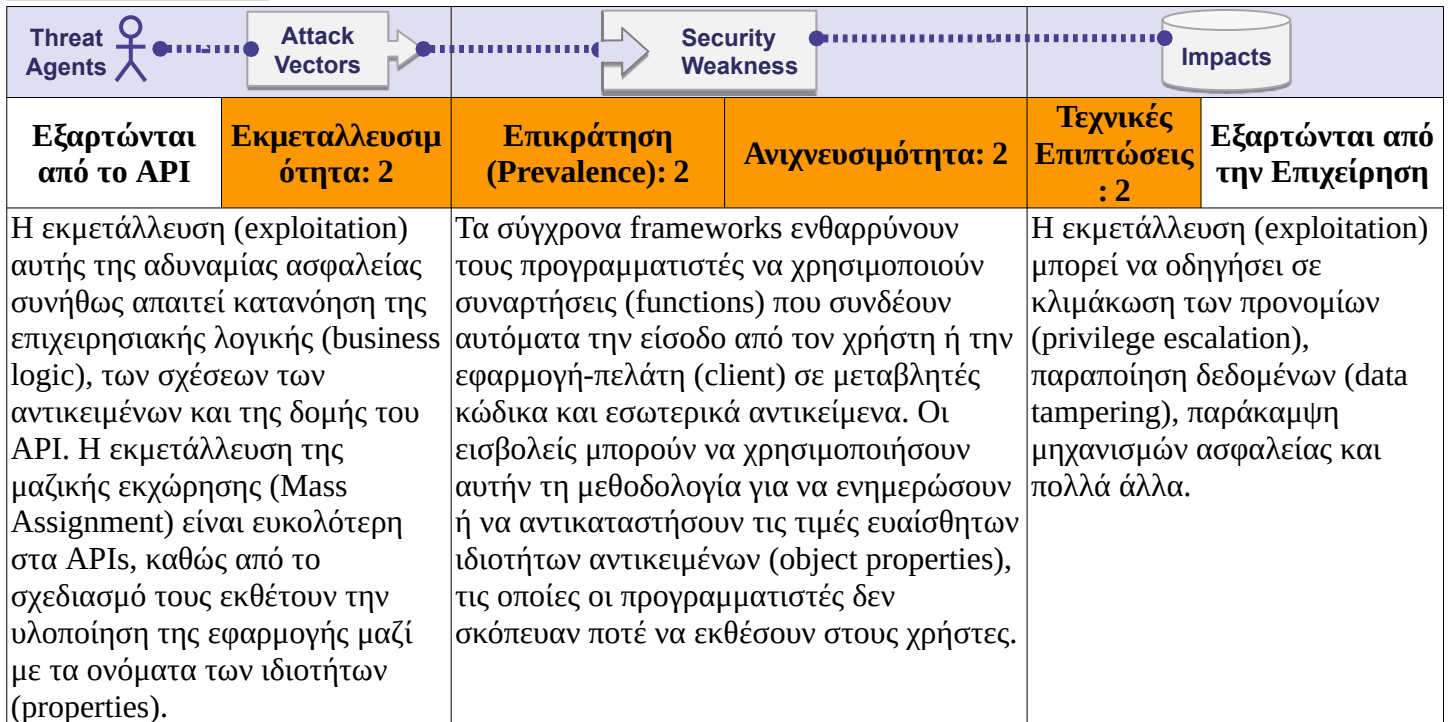
## Αναφορές (References)

### Αναφορές OWASP

- [OWASP Article on Forced Browsing](#)
- [OWASP Top 10 2013-A7-Missing Function Level Access Control](#)
- [OWASP Development Guide: Chapter on Authorization](#)

### External

- [CWE-285: Improper Authorization](#)



## Πότε το API είναι ευάλωτο

Τα αντικείμενα (objects) στις σύγχρονες εφαρμογές μπορεί να περιέχουν πολλές ιδιότητες (properties). Ορισμένες από αυτές τις ιδιότητες θα πρέπει να μπορούν να ενημερώνονται απευθείας από την εφαρμογή-πελάτη (client) (π.χ. `user.first_name` ή `user.address`) και ορισμένες από αυτές δεν θα πρέπει να είναι προσβάσιμες στους χρήστες / εφαρμογές-πελάτες (π.χ. `user.is_vip`).

Ένα τελικό σημείο προορισμού API είναι ευάλωτο εάν μετατρέπει αυτόματα τις παραμέτρους της εφαρμογής-πελάτη σε ιδιότητες εσωτερικού αντικειμένου, χωρίς να λαμβάνεται υπόψη η ευαισθησία και το επίπεδο έκθεσης αυτών των ιδιοτήτων. Αυτό θα μπορούσε να επιτρέψει σε έναν εισβολέα να ενημερώσει τις ιδιότητες αντικειμένων (object properties), στις οποίες δεν θα έπρεπε να έχει πρόσβαση.

Παραδείγματα ευαίσθητων ιδιοτήτων (properties):

- **Ιδιότητες που σχετίζονται με δικαιώματα:** Τα `user.is_admin`, `user.is_vip` θα πρέπει να ορίζονται μόνο από διαχειριστές.
- **Ιδιότητες που εξαρτώνται από κάποια διαδικασία (process):** Το `user.cash` θα πρέπει να ορίζεται εσωτερικά μόνο μετά την επαλήθευση πληρωμής.
- **Εσωτερικές ιδιότητες:** Η ιδιότητα `article.created_time` θα πρέπει να επιτρέπεται να ορίζεται μόνο εσωτερικά από την εφαρμογή.

## Παραδείγματα Σεναρίων Επίθεσης

### Σενάριο Επίθεσης #1

Μια εφαρμογή διαμοιρασμού διαδρομής (ride sharing) παρέχει στον χρήστη την επιλογή να επεξεργαστεί βασικές πληροφορίες για το προφίλ του. Κατά τη διάρκεια αυτής της διαδικασίας, αποστέλλεται μια κλήση API στο `PUT /api/v1/users/me` με το ακόλουθο έγκυρο αντικείμενο JSON:

```
{"user_name": "inons", "age": 24}
```

Το αίτημα `GET /api/v1/users/me` περιλαμβάνει μια πρόσθετη ιδιότητα `credit_balance`:



# API6:2019 Mass Assignment

```
{"user_name": "inons", "age": 24, "credit_balance": 10}
```

Ο εισβολέας επαναλαμβάνει το πρώτο αίτημα με το ακόλουθο payload:

```
{"user_name": "attacker", "age": 60, "credit_balance": 99999}
```

Δεδομένου ότι το τελικό σημείο προορισμού είναι ευάλωτο σε μαζική εκχώρηση (Mass Assignment), ο εισβολέας λαμβάνει πιστώσεις (credits) χωρίς να πληρώσει.

## Σενάριο Επίθεσης #2

Ένα portal διαμοιρασμού βίντεο (video sharing portal) επιτρέπει στους χρήστες να ανεβάζουν περιεχόμενο και να κατεβάζουν περιεχόμενο σε διαφορετικές μορφές (formats). Ένας εισβολέας που εξερευνά το API διαπίστωσε ότι το τελικό σημείο προορισμού `GET /api/v1/videos/{video_id}/meta_data` επιστρέφει ένα αντικείμενο JSON με τις ιδιότητες του βίντεο. Μία από τις ιδιότητες είναι `"mp4_conversion_params": "-v codec h264"`, που υποδεικνύει ότι η εφαρμογή χρησιμοποιεί μια εντολή shell για τη μετατροπή του βίντεο.

Ο εισβολέας διαπίστωσε επίσης ότι το τελικό σημείο προορισμού `POST /api/v1/videos/new` είναι ευάλωτο σε μαζική εκχώρηση (Mass Assignment) και επιτρέπει στην εφαρμογή-πελάτη (client) να ορίσει οποιαδήποτε ιδιότητα του αντικειμένου βίντεο. Ο εισβολέας ορίζει μια κακόβουλη τιμή ως εξής: `"mp4_conversion_params": "-v codec h264 && format C:/"`. Αυτή η τιμή θα προκαλέσει μια έγχυση εντολής shell (shell command injection) μόλις ο εισβολέας κατεβάσει το βίντεο ως MP4.

## Τρόπος Πρόληψης

- Εάν είναι δυνατόν, αποφύγετε τη χρήση συναρτήσεων που μετατρέπουν αυτόματα την είσοδο μιας εφαρμογής-πελάτη σε μεταβλητές κώδικα ή σε εσωτερικά αντικείμενα.
- Δημιουργήστε μια λίστα επιτρεπόμενων (whitelist) ιδιοτήτων που να περιλαμβάνει μόνο τις ιδιότητες που επιτρέπεται να ενημερώνονται από τους χρήστες / εφαρμογές-πελάτες (API clients).
- Χρησιμοποιήστε ενσωματωμένες δυνατότητες του framework σας για να δημιουργήσετε μια λίστα μη-επιτρεπόμενων ιδιοτήτων στις οποίες δεν πρέπει να έχουν πρόσβαση οι εφαρμογές-πελάτες (API clients).
- Εάν είναι εφικτό, ορίστε και επιβάλλετε ρητά σχήματα (schemas) για τα payloads δεδομένων εισόδου.

## Αναφορές (References)

### Εξωτερικές Αναφορές

- [CWE-915: Improperly Controlled Modification of Dynamically-Determined Object Attributes](#)

Threat Agents	Attack Vectors	Security Weakness	Impacts
<b>Εξαρτώνται από το API</b>	<b>Εκμεταλλευσιμότητα: 3</b>	<b>Επικράτηση (Prevalence): 3</b>	<b>Ανιχνευσιμότητα: 3</b>
Οι εισβολείς συχνά επιχειρούν να βρουν μη επιδιορθωμένα ελαττώματα (unpatched flows), συνηθισμένα τελικά σημεία προορισμού ή μη προστατευμένα αρχεία και καταλόγους για να αποκτήσουν μη εξουσιοδοτημένη πρόσβαση ή γνώση του συστήματος.	Λανθασμένες ρυθμίσεις παραμέτρων ασφαλείας (Security Misconfigurations) μπορούν να υπάρξουν σε οποιοδήποτε επίπεδο της στοίβας API (API stack), από το επίπεδο δικτύου έως το επίπεδο εφαρμογής. Διατίθενται αυτοματοποιημένα εργαλεία για τον εντοπισμό και την εκμετάλλευση εσφαλμένων διαμορφώσεων, όπως περιττές υπηρεσίες ή ρυθμίσεις παλαιού τύπου (legacy options).	<b>Τεχνικές Επιπτώσεις : 2</b>	<b>Εξαρτώνται από την Επιχείρηση</b>
		Οι εσφαλμένες ρυθμίσεις ασφαλείας όχι μόνο εκθέτουν ευαίσθητα δεδομένα χρηστών, αλλά εκθέτουν και λεπτομέρειες συστήματος που μπορεί να οδηγήσουν σε πλήρη παραβίαση του διακομιστή (server).	

## Πότε το API είναι ευάλωτο

Το API μπορεί να είναι ευάλωτο όταν:

- Δεν υπάρχει η κατάλληλη θωράκιση ασφαλείας (security hardening) σε όλα τα τμήματα της στοίβας της εφαρμογής, ή υπάρχουν λανθασμένες ρυθμίσεις δικαιωμάτων σε υπηρεσίες Cloud.
- Δεν έχουν εγκατασταθεί οι ενημερωμένες εκδόσεις ασφαλείας, ή τα συστήματα είναι παροχημένα.
- Αχρείαστα χαρακτηριστικά (features) είναι ενεργοποιημένα (για παράδειγμα, το API δέχεται HTTP verbs που δεν χρησιμοποιούνται).
- Δεν υπάρχει Transport Layer Security (TLS).
- Δεν στέλνονται στις εφαρμογές-πελάτες (clients) οι οδηγίες ασφαλείας (security directives)(π.χ., [Επικεφαλίδες Ασφαλείας](#)).
- Δεν υπάρχει πολιτική Cross-Origin Resource Sharing (CORS) ή έχει ρυθμιστεί εσφαλμένα.
- Τα μηνύματα σφαλμάτων περιλαμβάνουν τεχνικές πληροφορίες για τα σφάλματα όπως τα ονόματα και την αλληλουχία των μεθόδων εκτέλεσης (stack trace) ή το API εκθέτει άλλες ευαίσθητες πληροφορίες.

## Παραδείγματα από Σενάρια Επίθεσης

### Σενάριο Επίθεσης #1

Ένας εισβολέας βρίσκει το αρχείο `.bash_history` κάτω από τον κεντρικό φάκελο (root folder) του διακομιστή, το οποίο περιλαμβάνει εντολές που χρησιμοποιούνται από την ομάδα DevOps για να έχουν πρόσβαση στο API:

```
$ curl -X GET 'https://api.server/endpoint/' -H 'authorization: Basic Zm9vOmJhcG=='
```

Ένας εισβολέας θα μπορούσε επίσης να βρει άγνωστα τελικά σημεία προορισμού του API που χρησιμοποιούνται μόνο από την ομάδα DevOps και τα οποία δεν είναι τεκμηριωμένα.

### Σενάριο Επίθεσης #2

Θέτοντας ως στόχο μια συγκεκριμένη υπηρεσία, ένας εισβολέας χρησιμοποιεί μια δημοφιλή μηχανή αναζήτησης για να αναζητήσει υπολογιστές άμεσα προσβάσιμους από το Διαδίκτυο. Ο εισβολέας βρίσκει έναν διακομιστή που τρέχει ένα δημοφιλές σύστημα διαχείρισης βάσεων δεδομένων, το οποίο ακούει στην προεπιλεγμένη θύρα. Ο διακομιστής αυτός χρησιμοποιεί τις προεπιλεγμένες ρυθμίσεις (default configuration), οι οποίες έχουν απενεργοποιημένο τον έλεγχο ταυτότητας, με αποτέλεσμα ο εισβολέας να αποκτήσει πρόσβαση σε εκατομμύρια εγγραφές με προσωπικά δεδομένα (PII), προσωπικές προτιμήσεις και δεδομένα ελέγχου ταυτότητας.

## Σενάριο Επίθεσης #3

Επιθεωρώντας την μεταφορά των δεδομένων μιας εφαρμογής για κινητά, ένας εισβολέας ανακαλύπτει ότι τα δεδομένα HTTP δεν μεταφέρονται ολικά κάτω από ένα ασφαλές πρωτόκολλο (π.χ. TLS). Ο εισβολέας ανακαλύπτει ότι αυτό συμβαίνει ειδικά για τη λήψη εικόνων προφίλ. Καθώς η αλληλεπίδραση του χρήστη με το API είναι δυαδική και παρά το γεγονός ότι η μεταφορά των δυαδικών δεδομένων του API εκτελείται κάτω από ένα ασφαλές πρωτόκολλο, ο εισβολέας ανακαλύπτει ένα μοτίβο όσον αφορά το μέγεθος των απαντήσεων του API. Στην συνέχεια, ο εισβολέας χρησιμοποιεί το μοτίβο για να βγάλει συμπεράσματα για τις προτιμήσεις των χρηστών σε σχέση με το περιεχόμενο που εμφανίζεται (π.χ. εικόνες προφίλ).

## Τρόπος Πρόληψης

Με στόχο την αποτελεσματική πρόληψη, ο κύκλος ζωής των API θα πρέπει να περιλαμβάνει τα παρακάτω:

- Επαναλαμβανόμενη διαδικασία θωράκισης (hardening) που επιτρέπει γρήγορη και εύκολη εγκατάσταση ενός σωστά ασφαλισμένου περιβάλλοντος στο οποίο θα τρέχουν τα APIs.
- Έλεγχο και ενημέρωση των ρυθμίσεων σε ολόκληρη τη στοίβα (stack) των APIs. Ο έλεγχος θα πρέπει να περιλαμβάνει αρχεία που χρησιμοποιούνται κατά την ενορχήστρωση (orchestration), API components και υπηρεσίες cloud (π.χ. δικαιώματα κάδων S3).
- Ένα ασφαλές κανάλι επικοινωνίας για όλες τις αλληλεπιδράσεις του API σε στατικά στοιχεία (π.χ. εικόνες).
- Αυτοματοποιημένη διαδικασία για τη συνεχή αξιολόγηση της αποτελεσματικότητας των ρυθμίσεων σε όλα τα περιβάλλοντα.

Ακόμα:

- Για να αποτρέψετε την πιθανή αποστολή τεχνικών πληροφοριών σε μηνύματα σφαλμάτων (exception trace) και άλλων πολύτιμων πληροφοριών στους εισβολείς, εάν έχετε την δυνατότητα, ορίστε και επιβάλετε συγκεκριμένα σχήματα απαντήσεων API (response payload schemas), συμπεριλαμβανομένων των απαντήσεων σφαλμάτων (error responses).
- Βεβαιωθείτε ότι το API είναι προσβάσιμο μόνο από τα ρήματα/μεθόδους HTTP (HTTP verbs) που έχετε καθορίσει. Όλα τα άλλα ρήματα/μέθοδοι HTTP (HTTP verbs) θα πρέπει να είναι απενεργοποιημένα (π.χ. "HEAD").
- Τα APIs που είναι σχεδιασμένα για να είναι προσβάσιμα από εφαρμογές-πελάτες μέσω προγραμμάτων περιήγησης (π.χ., το front-end μιας εφαρμογής Web) θα πρέπει να εφαρμόζουν μια σωστή πολιτική κοινής χρήσης πόρων μεταξύ προέλευσης (CORS).

## Αναφορές (References)

### Αναφορές OWASP

- [OWASP Secure Headers Project](#)
- [OWASP Testing Guide: Configuration Management](#)
- [OWASP Testing Guide: Testing for Error Codes](#)
- [OWASP Testing Guide: Test Cross Origin Resource Sharing](#)

### Εξωτερικές Αναφορές

- [CWE-2: Environmental Security Flaws](#)
- [CWE-16: Configuration](#)
- [CWE-388: Error Handling](#)
- [Guide to General Server Security](#), NIST
- [Let's Encrypt: a free, automated, and open Certificate Authority](#)

Threat Agents		Attack Vectors		Security Weakness		Impacts	
Εξαρτώνται από το API	Εκμεταλλευσιμότητα: 3	Επικράτηση (Prevalence): 2	Ανιχνευσιμότητα: 3	Τεχνικές Επιπτώσεις : 3	Εξαρτώνται από την Επιχείρηση		
Οι εισβολείς εισάγουν στο API κακόβουλα δεδομένα μέσω οποιασδήποτε διαθέσιμης μεθόδου εισαγωγής δεδομένων αναμένοντας να σταλούν τελικώς σε έναν διερμηνέα λογισμικού (interpreter). Οι μέθοδοι αυτοί που μπορούν να χρησιμοποιηθούν για εισαγωγή κακόβουλων δεδομένων που καταλήγουν σε διερμηνείς λογισμικού ονομάζονται διανύσματα έγχυσης (injection vectors). Παραδείγματα τέτοιων μεθόδων είναι η άμεση εισαγωγή δεδομένων (direct input), οι παράμετροι (parameters), οι ολοκληρωμένες/ενσωματωμένες υπηρεσίες (integrated services) κ.λπ.		Οι ευπάθειες έγχυσης είναι πολύ κοινές και εντοπίζονται συχνά σε ερωτήματα SQL, LDAP ή NoSQL, εντολές λειτουργικού συστήματος, αναλυτές XML και ORM. Αυτές οι ευπάθειες είναι εύκολο να εντοπιστούν κατά τον έλεγχο του πηγαίου κώδικα. Οι επιτιθέμενοι μπορούν να χρησιμοποιήσουν σαρωτές και fuzzers για να εντοπίσουν τέτοιες ευπάθειες.		Η έγχυση (injection) μπορεί να οδηγήσει σε αποκάλυψη πληροφοριών και απώλεια δεδομένων. Μπορεί επίσης να οδηγήσει σε επιθέσεις άρνησης εξυπηρέτησης (DoS) ή πλήρη κατάληψη του κεντρικού υπολογιστή.			

## Πότε το API είναι ευάλωτο

Το API είναι ευάλωτο σε ευπάθεια έγχυσης (injection flaw) όταν:

- Τα δεδομένα που παρέχονται από τους χρήστες ή εφαρμογές-πελάτες δεν επικυρώνονται, δεν φιλτράρονται ή δεν απολυμαίνονται (sanitized) από το API.
- Τα δεδομένα που παρέχονται από τους χρήστες ή εφαρμογές-πελάτες χρησιμοποιούνται απευθείας ή συνδέονται με ερωτήματα SQL/NoSQL/LDAP, εντολές λειτουργικού συστήματος, αναλυτές XML και σχεσιακή αντιστοίχιση αντικειμένων (ORM) / χαρτογράφηση εγγράφων αντικειμένου (ODM).
- Τα δεδομένα που προέρχονται από εξωτερικά συστήματα (π.χ. ολοκληρωμένα/ενσωματωμένα συστήματα) δεν επικυρώνονται (validation), δεν φιλτράρονται (filtering) ή δεν απολυμαίνονται (sanitization) από το API.

## Παραδείγματα από Σενάρια Επίθεσης

### Σενάριο Επίθεσης #1

Το υλικολογισμικό (firmware) μιας συσκευής γονικού ελέγχου παρέχει το τελικό σημείο προορισμού `/api/CONFIG/restore` το οποίο έχει σχεδιαστεί έτσι ώστε η παράμετρος `appId` να αποστέλεται ως παράμετρος πολλαπλών τμημάτων (multipart parameter). Χρησιμοποιώντας έναν απομεταγλωττιστή (decompiler), ένας εισβολέας ανακαλύπτει ότι το `appId` περνάει απευθείας σε μια κλήση συστήματος χωρίς καμία απολύμανση (sanitization):

```
snprintf(cmd, 128, "%srestore_backup.sh /tmp/postfile.bin %s %d",
        "/mnt/shares/usr/bin/scripts/", appId, 66);
system(cmd);
```

Η ακόλουθη εντολή επιτρέπει στον εισβολέα να τερματίσει οποιαδήποτε συσκευή με το ίδιο ευάλωτο υλικολογισμικό:

```
$ curl -k "https://${deviceIP}:4567/api/CONFIG/restore" -F
'appid=$(/etc/pod/power_down.sh)'
```

## Σενάριο Επίθεσης #2

Έχουμε μια εφαρμογή με βασική λειτουργικότητα CRUD για λειτουργίες με κρατήσεις (bookings). Ένας εισβολέας κατάφερε να αναγνωρίσει ότι η εισαγωγή NoSQL μπορεί να είναι δυνατή μέσω της παραμέτρου ερωτήματος `bookingId` στο αίτημα διαγραφής κράτησης. Το αίτημα είναι το εξής: `DELETE /api/bookings?bookingId=678`.

Ο διακομιστής API χρησιμοποιεί την ακόλουθη λειτουργία (function) για να χειριστεί αιτήματα διαγραφής:

```
router.delete('/bookings', async function (req, res, next) {
  try {
    const deletedBooking = await Bookings.findOneAndRemove({_id' : req.query.bookingId});
    res.status(200);
  } catch (err) {
    res.status(400).json({
      error: 'Unexpected error occurred while processing a request'
    });
  }
});
```

Ο εισβολέας υπέκλεψε το αίτημα και άλλαξε την παράμετρο ερωτήματος `bookingId`, όπως φαίνεται παρακάτω. Σε αυτήν την περίπτωση, ο εισβολέας κατάφερε να διαγράψει την κράτηση άλλου χρήστη:

```
DELETE /api/bookings?bookingId[$ne]=678
```

## Τρόπος Πρόληψης

Η πρόληψη της έγχυσης απαιτεί τη διατήρηση των δεδομένων ξεχωριστά από εντολές και ερωτήματα.

- Εκτελέστε επικύρωση (validation) δεδομένων χρησιμοποιώντας μια αξιόπιστη και ενεργά συντηρούμενη βιβλιοθήκη.
- Όλα τα δεδομένα που παρέχονται από τους χρήστες ή εφαρμογές-πελάτες ή άλλα δεδομένα που προέρχονται από ενσωματωμένα συστήματα πρέπει να επικυρώνονται (validation), φιλτράρονται ή/και να απολυμαίνονται (sanitization).
- Οι ειδικοί χαρακτήρες θα πρέπει να διαφεύγονται (escape) χρησιμοποιώντας τη συγκεκριμένη σύνταξη του διεργαστικού λογισμικού που λαμβάνει τα δεδομένα.
- Προτιμήστε ένα ασφαλές API που παρέχει μια παραμετροποιημένη διεπαφή.
- Φροντίστε να περιορίσετε τον αριθμό των επιστρεφόμενων εγγγραφών για να αποτρέψετε τη μαζική παραβίαση δεδομένων σε περίπτωση επίθεσης.
- Επικυρώστε τα εισερχόμενα δεδομένα χρησιμοποιώντας επαρκή φίλτρα για να επιτρέπονται μόνο έγκυρες τιμές για κάθε παράμετρο εισόδου.
- Ορίστε τύπους δεδομένων και αυστηρά μοτίβα για όλες τις παραμέτρους αιτημάτων.

## Αναφορές (References)

### Αναφορές OWASP

- [OWASP Injection Flaws](#)
- [SQL Injection](#)
- [NoSQL Injection Fun with Objects and Arrays](#)
- [Command Injection](#)

### Εξωτερικές Αναφορές

- [CWE-77: Command Injection](#)
- [CWE-89: SQL Injection](#)



# API9:2019 Improper Assets Management

Threat Agents		Attack Vectors		Security Weakness		Impacts	
Εξαρτώνται από το API	Εκμεταλλευσιμότητα: 3	Επικράτηση (Prevalence): 3	Ανιχνευσιμότητα: 2	Τεχνικές Επιπτώσεις : 2	Εξαρτώνται από την Επιχείρηση		
Οι παλιές εκδόσεις ενός API μένουν συνήθως ανενημέρωτες από ενημερώσεις ασφαλείας (unpatched) και έτσι αποτελούν έναν εύκολο τρόπο για την παραβίαση συστημάτων χωρίς να χρειάζεται ο εισβολέας να αντιμετωπίσει μηχανισμούς ασφαλείας τελευταίας τεχνολογίας, οι οποίοι μπορεί να υπάρχουν αλλά να προστατεύουν μόνο τις νέες εκδόσεις ενός API.		Η μη ενημερωμένη τεκμηρίωση (documentation) ενός API καθιστά πιο δύσκολη την εύρεση ή/και τη διόρθωση ευπαθειών. Η έλλειψη μεθοδικής καταγραφής των πληροφοριακών στοιχείων (assets inventory) και η έλλειψη στρατηγικών απόσυρσης (retire strategies) οδηγούν στο να τρέχουν ανενημέρωτα (unpatched) συστήματα, με αποτέλεσμα τη διαρροή ευαίσθητων δεδομένων. Είναι σύνηθες να βρίσκουμε άσκοπα εκτεθειμένους κεντρικούς υπολογιστές API λόγω των σύγχρονων concepts όπως τα microservices, τα οποία καθιστούν τις εφαρμογές ανεξάρτητες και εύκολες στην ανάπτυξη (π.χ. υπολογιστικό νέφος (cloud), k8s).		Οι εισβολείς ενδέχεται να αποκτήσουν πρόσβαση σε ευαίσθητα δεδομένα ή ακόμη και πάρουν τον έλεγχο του διακομιστή μέσω παλιών, μη ενημερωμένων εκδόσεων API που συνδέονται στην ίδια βάση δεδομένων.			

## Πότε το API είναι ευάλωτο

Το API ίσως είναι ευάλωτο όταν:

- Ο σκοπός ενός κεντρικού υπολογιστή API είναι ασαφής και δεν υπάρχουν σαφείς απαντήσεις στις ακόλουθες ερωτήσεις:
  - Σε ποιο περιβάλλον εκτελείται το API (π.χ. παραγωγή (production), σταδιοποίηση (staging), δοκιμή (test), ανάπτυξη (development));
  - Ποιος πρέπει να έχει δικτυακή πρόσβαση στο API (π.χ. δημόσια πρόσβαση, εσωτερική πρόσβαση, πρόσβαση σε συνεργάτες);
  - Ποια έκδοση API εκτελείται;
  - Ποια δεδομένα συλλέγονται και επεξεργάζονται από το API (π.χ. Προσωπικά αναγνωρίσιμα στοιχεία (PII));
  - Ποια είναι η ροή των δεδομένων;
- Δεν υπάρχει τεκμηρίωση (documentation) ή η υπάρχουσα τεκμηρίωση δεν έχει ενημερωθεί.
- Δεν υπάρχει σχέδιο απόσυρσης (retirement plan) για κάθε έκδοση API.
- Δεν υπάρχει αρχείο καταγραφής όλων των hosts (hosts inventory) ή αν υπάρχει δεν είναι ενημερωμένο.
- Το αρχείο καταγραφής ολοκληρωμένων υπηρεσιών (integrated services inventory), είτε της εταιρίας που φτιάχνει το API (first-party), είτε τρίτων μελών (third-party), λείπει ή είναι παλιό.
- Εκτελούνται παλιές ή προηγούμενες εκδόσεις του API χωρίς ενημέρωση.

## Παραδείγματα Σεναρίων Επίθεσης

### Σενάριο Επίθεσης #1

Μετά τον επανασχεδιασμό των εφαρμογών της, μια τοπική υπηρεσία αναζήτησης άφησε μια παλιά έκδοση API (`api.someservice.com/v1`) σε λειτουργία, απροστάτευτη και με πρόσβαση στη βάση δεδομένων των χρηστών. Ένας εισβολέας, ενώ στόχευε μία από τις πιο πρόσφατες εφαρμογές που κυκλοφόρησαν, βρήκε τη διεύθυνση API (`api.someservice.com/v2`). Η αντικατάσταση του «v2» με το «v1» στη διεύθυνση URL έδωσε στον εισβολέα

πρόσβαση στο παλιό, μη προστατευμένο API, εκθέτοντας τα προσωπικά στοιχεία ταυτοποίησης (PII) περισσότερων από 100 εκατομμυρίων χρηστών.

## Σενάριο Επίθεσης #2

Ένα κοινωνικό δίκτυο εφάρμοσε έναν μηχανισμό περιορισμού ρυθμού (rate limiting) που εμποδίζει τους εισβολείς να χρησιμοποιούν επιθέσεις ωμής βίας (brute force attacks) για να μαντέψουν τα διακριτικά επαναφοράς κωδικών πρόσβασης. Αυτός ο μηχανισμός δεν εφαρμόστηκε ως μέρος του ίδιου του κώδικα API, αλλά σε ένα ξεχωριστό στοιχείο (component) μεταξύ του πελάτη και του επίσημου API ([www.socialnetwork.com](http://www.socialnetwork.com)). Ένας ερευνητής βρήκε έναν δεύτερο κεντρικό υπολογιστή ([www.mbasic.beta.socialnetwork.com](http://www.mbasic.beta.socialnetwork.com)) που εκτελεί το ίδιο API, συμπεριλαμβανομένου του μηχανισμού επαναφοράς κωδικού πρόσβασης, αλλά χωρίς μηχανισμό περιορισμού ρυθμού. Ο ερευνητής μπόρεσε να επαναφέρει τον κωδικό πρόσβασης οποιουδήποτε χρήστη χρησιμοποιώντας μια απλή επίθεση ωμής βίας για να μαντέψει το διακριτικό των 6 ψηφίων.

## Τρόπος Πρόληψης


- Καταγράψτε όλους τους υπολογιστές (hosts) που φιλοξενούν API. Καταγράψτε τα περιβάλλοντα που τρέχουν τα API (production, staging, test, development). Επίσης καταγράψτε ποιος θα πρέπει να έχει πρόσβαση σε αυτά (ανοιχτά σε όλους, εσωτερική πρόσβαση, πρόσβαση σε συνεργάτες) καθώς και την έκδοσή τους.
- Καταγράψτε τις ολοκληρωμένες υπηρεσίες (integrated services) δηλαδή τις εξωτερικές υπηρεσίες που χρησιμοποιούν το API. Τεκμηριώστε σημαντικές πτυχές τους όπως ο ρόλος τους στο σύστημα, ποια δεδομένα ανταλλάσσονται (ροή δεδομένων) και η ευαισθησία τους.
- Τεκμηριώστε όλες τις πτυχές του API σας, όπως τον έλεγχο ταυτότητας, τα σφάλματα, τις ανακατευθύνσεις, τον περιορισμό ρυθμού, την πολιτική και τα τελικά σημεία κοινής χρήσης πόρων μεταξύ προέλευσης (CORS), συμπεριλαμβανομένων των παραμέτρων, των αιτημάτων και των απαντήσεών τους.
- Δημιουργήστε τεκμηρίωση αυτόματα υιοθετώντας ανοιχτά πρότυπα (open standards). Συμπεριλάβετε την αυτόματη δημιουργία τεκμηρίωσης στο σύστημα CI/CD σας.
- Παραχωρήστε πρόσβαση στην τεκμηρίωση API σε όσους είναι εξουσιοδοτημένοι να χρησιμοποιούν το API.
- Χρησιμοποιήστε εξωτερικά μέτρα προστασίας, όπως τείχη προστασίας ασφαλείας API για όλες τις εκτεθειμένες εκδόσεις των API σας και όχι μόνο για την τρέχουσα έκδοση παραγωγής.
- Αποφύγετε τη χρήση δεδομένων παραγωγής σε μη παραγωγικές διανομές (deployments) του API. Εάν αυτό είναι αναπόφευκτο, αυτά τα τελικά σημεία προορισμού θα πρέπει να τυγχάνουν της ίδιας μεταχείρισης ασφαλείας με αυτά της παραγωγής.
- Όταν νεότερες εκδόσεις των API περιλαμβάνουν βελτιώσεις ασφαλείας, πραγματοποιήστε ανάλυση κινδύνου για να αποφασίσετε για τις ενέργειες μετριασμού που απαιτούνται για τις παλαιότερες εκδόσεις: για παράδειγμα, εάν είναι δυνατή η υποστήριξη των βελτιώσεων χωρίς να διαταραχθεί η συμβατότητα API ή εάν πρέπει να αφαιρέσετε τις παλαιότερες εκδόσεις γρήγορα και να αναγκάσετε όλους τους χρήστες / εφαρμογές-πελάτες να μετακινηθούν στην πιο πρόσφατη έκδοση.

## Αναφορές (References)

### Εξωτερικές Αναφορές

- [CWE-1059: Incomplete Documentation](#)
- [OpenAPI Initiative](#)

# API10:2019 Insufficient Logging & Monitoring

					
Εξαρτώνται από το API	Εκμεταλλευσιμότητα: 2	Επικράτηση (Prevalence): 3	Ανιχνευσιμότητα: 1	Τεχνικές Επιπτώσεις : 2	Εξαρτώνται από την Επιχείρηση
Οι επιτιθέμενοι εκμεταλλεύονται την έλλειψη καταγραφής συμβάντων (logging) και παρακολούθησης (monitoring) για να επιτεθούν σε συστήματα χωρίς να γίνουν αντιληπτοί.		Χωρίς καταγραφή και παρακολούθηση ή με ανεπαρκή καταγραφή και παρακολούθηση, είναι σχεδόν αδύνατο να παρακολουθήσετε ύποπτες δραστηριότητες και να αντιδράσετε έγκαιρα.		Χωρίς ορατότητα σε συνεχείς κακόβουλες δραστηριότητες, οι εισβολείς έχουν άφθονο χρόνο για να υπονομεύσουν πλήρως τα συστήματά σας.	

## Πότε το API είναι ευάλωτο

Το API είναι ευάλωτο όταν:

- Δεν παράγει κανένα αρχείο καταγραφής συμβάντων, το επίπεδο καταγραφής δεν έχει ρυθμιστεί σωστά ή τα μηνύματα καταγραφής δεν περιλαμβάνουν αρκετές λεπτομέρειες.
- Η ακεραιότητα των μηνυμάτων καταγραφής δεν είναι εγγυημένη (π.χ. [Log Injection](#)).
- Τα αρχεία καταγραφής δεν παρακολουθούνται συνεχώς.
- Η υποδομή API δεν παρακολουθείται συνεχώς.

## Παραδείγματα από Σενάρια Επίθεσης

### Σενάριο Επίθεσης #1

Τα κλειδιά πρόσβασης ενός διαχειριστικού API διέρρευσαν σε ένα δημόσια προσβάσιμο χώρο αποθήκευσης κώδικα (αποθετήριο) (public repository). Ο κάτοχος του αποθετηρίου ειδοποιήθηκε μέσω email σχετικά με την πιθανή διαρροή, αλλά χρειάστηκαν περισσότερες από 48 ώρες για να αντιμετωπιστεί το συμβάν και η έκθεση των κλειδιών πρόσβασης μπορεί να επέτρεψε την πρόσβαση σε ευαίσθητα δεδομένα. Λόγω ανεπαρκούς καταγραφής, η εταιρεία δεν είναι σε θέση να αξιολογήσει σε ποια δεδομένα είχαν πρόσβαση κακόβουλοι παράγοντες.

### Σενάριο Επίθεσης #2

Μια πλατφόρμα κοινής χρήσης βίντεο χτυπήθηκε από μια «μεγάλης κλίμακας» επίθεση τροφοδότησης διαπιστευτηρίων (credential stuffing). Παρά το γεγονός ότι αποτυχημένες προσπάθειες σύνδεσης καταγράφηκαν στα αρχεία καταγραφής συμβάντων, δεν στάλθηκαν ειδοποιήσεις ασφαλείας κατά τη διάρκεια του χρόνου της επίθεσης. Έπειτα από παράπονα χρηστών, τα αρχεία καταγραφής συμβάντων του API αναλύθηκαν και η επίθεση εντοπίστηκε. Η εταιρεία αναγκάστηκε να αναφέρει το περιστατικό στις ρυθμιστικές αρχές και να κάνει μια δημόσια ανακοίνωση ζητώντας από τους χρήστες να επαναφέρουν τους κωδικούς πρόσβασής τους.

## Τρόπος Πρόληψης

- Καταγράψτε όλες τις αποτυχημένες προσπάθειες ελέγχου ταυτότητας, την άρνηση πρόσβασης και τα σφάλματα επικύρωσης εισαγωγής (input validation errors).
- Τα αρχεία καταγραφής πρέπει να συντάσσονται χρησιμοποιώντας μια μορφή κατάλληλη για αξιοποίηση από μια εφαρμογή διαχείρισης αρχείων καταγραφής συμβάντων (log management solution) και θα πρέπει να περιλαμβάνουν αρκετές λεπτομέρειες για τον εντοπισμό του κακόβουλου παράγοντα.
- Τα αρχεία καταγραφής θα πρέπει να αντιμετωπίζονται ως ευαίσθητα δεδομένα και η ακεραιότητά τους θα πρέπει να είναι εγγυημένη κατά την αποθήκευση και τη μεταφορά.
- Χρησιμοποιήστε ένα σύστημα παρακολούθησης (monitoring system) για συνεχή παρακολούθηση της υποδομής, του δικτύου και της λειτουργίας των APIs.
- Χρησιμοποιήστε ένα σύστημα διαχείρισης πληροφοριών ασφαλείας και συμβάντων (SIEM) για να συγκεντρώσετε και να διαχειριστείτε αρχεία καταγραφής από όλα τα στοιχεία των APIs και των κεντρικών υπολογιστών.
- Δημιουργήστε εξατομικευμένους πίνακες παρακολούθησης ειδοποιήσεων (dashboards). Οι πίνακες αυτοί θα διευκολύνουν τον γρηγορότερο εντοπισμό και την διαχείριση ύποπτων δραστηριοτήτων.

## Αναφορές (References)

### Αναφορές OWASP

- [OWASP Logging Cheat Sheet](#)
- [OWASP Proactive Controls: Implement Logging and Intrusion Detection](#)
- [OWASP Application Security Verification Standard: V7: Error Handling and Logging Verification Requirements](#)

### Εξωτερικές Αναφορές

- [CWE-223: Omission of Security-relevant Information](#)
- [CWE-778: Insufficient Logging](#)

Η εργασία δημιουργίας και συντήρησης ασφαλούς λογισμικού καθώς και η επιδιόρθωση αυτού είναι δύσκολη υπόθεση. Το ίδιο ισχύει και για τα APIs.

Πιστεύουμε ότι η εκπαίδευση και η ευαισθητοποίηση είναι βασικοί παράγοντες για τη δημιουργία ασφαλούς λογισμικού. Τα υπόλοιπα που απαιτούνται για την επίτευξη του στόχου, εξαρτώνται από τη **δημιουργία και τη χρήση επαναλαμβανόμενων διαδικασιών και τυπικών ελέγχων ασφαλείας**.

Το OWASP διαθέτει πολλούς δωρεάν και ανοιχτούς πόρους για την αντιμετώπιση της ασφάλειας από την αρχή του έργου. Επισκεφτείτε τη [σελίδα OWASP Projects](#) για την ολοκληρωμένη λίστα των διαθέσιμων έργων.

<b>Εκπαίδευση</b>	Μπορείτε να ξεκινήσετε να διαβάζετε το <a href="#">υλικό του OWASP Education Project</a> ανάλογα με το επάγγελμα και το ενδιαφέρον σας. Για πρακτική μάθηση, προσθέσαμε το <b>crAPI - Completely Ridiculous API</b> στον <a href="#">οδικό μας χάρτη (roadmap)</a> . Εν τω μεταξύ, μπορείτε να εξασκηθείτε στο WebAppSec χρησιμοποιώντας το <a href="#">OWASP DevSlop Pixi Module</a> , μια ευάλωτη υπηρεσία WebApp και API που έχει σκοπό να διδάξει στους χρήστες πώς να δοκιμάζουν σύγχρονες εφαρμογές ιστού και API για ζητήματα ασφαλείας και πώς να γράφουν πιο ασφαλή APIs στο μέλλον. Μπορείτε επίσης να παρακολουθήσετε εκπαιδευτικές συνεδρίες του <a href="#">Συνεδρίου OWASP AppSec</a> ή να <a href="#">εγγραφείτε στο τοπικό σας τμήμα</a> .
<b>Απαιτήσεις Ασφαλείας</b>	Η ασφάλεια πρέπει να είναι μέρος κάθε έργου (project) από την αρχή. Όταν βρίσκεστε στο στάδιο της εξαγωγής απαιτήσεων (requirements elicitation), είναι σημαντικό να ορίσετε τι σημαίνει "ασφαλές" για το έργο σας. Το OWASP συνιστά να χρησιμοποιείτε το <a href="#">Πρότυπο Επαλήθευσης Ασφάλειας Εφαρμογών OWASP (ASVS)</a> ως οδηγό για τον καθορισμό των απαιτήσεων ασφαλείας. Εάν αναθέτετε το έργο σας σε εξωτερικούς συνεργάτες (outsourcing), εξετάστε το <a href="#">Παράρτημα Σύμβασης Ασφαλούς Λογισμικού OWASP</a> , το οποίο θα πρέπει να προσαρμοστεί σύμφωνα με την τοπική νομοθεσία και τους κανονισμούς της.
<b>Αρχιτεκτονική Ασφαλείας</b>	Η ασφάλεια θα πρέπει να λαμβάνεται υπόψη σε όλα τα στάδια του έργου. Τα <a href="#">Σκονάκια Πρόληψης OWASP (OWASP Prevention Cheat Sheets)</a> είναι ένα καλό σημείο εκκίνησης για καθοδήγηση σχετικά με τον τρόπο σχεδιασμού ασφαλείας κατά τη φάση σχεδιασμού / αρχιτεκτονικής. Μεταξύ πολλών άλλων, θα βρείτε το σκονάκι <a href="#">REST Security Cheat Sheet</a> και το σκονάκι <a href="#">REST Assessment Cheat Sheet</a> .
<b>Τυπικοί Έλεγχοι Ασφαλείας</b>	Η υιοθέτηση Τυποποιημένων Μηχανισμών Ελέγχων Ασφαλείας (Standard Security Controls) μειώνει τον κίνδυνο εισαγωγής αδυναμιών ασφαλείας κατά την διάρκεια υλοποίησης της λογικής του λογισμικού σας. Παρά το γεγονός ότι πολλά σύγχρονα frameworks διαθέτουν πλέον ενσωματωμένους τυπικούς αποτελεσματικούς ελέγχους, τα <a href="#">OWASP Proactive Controls</a> σας παρέχουν μια καλή επισκόπηση των στοιχείων ελέγχου ασφαλείας που πρέπει να συμπεριλάβετε στο έργο σας. Το OWASP παρέχει επίσης ορισμένες βιβλιοθήκες και εργαλεία που μπορεί να σας φανούν πολύτιμα, όπως στοιχεία ελέγχου επικύρωσης (validation controls).
<b>Κύκλος Ζωής Ασφαλούς Ανάπτυξης Λογισμικού</b>	Μπορείτε να χρησιμοποιήσετε το <a href="#">OWASP Software Assurance Maturity Model (SAMM)</a> για να βελτιώσετε τη διαδικασία κατά τη δημιουργία των APIs. Πολλά ακόμα έργα OWASP είναι διαθέσιμα για να σας βοηθήσουν σε όλες τις διαφορετικές φάσεις ανάπτυξης API, π.χ., το <a href="#">Έργο Αναθεώρησης Κώδικα OWASP (OWASP Code Review Project)</a> .



Λόγω της σημασίας τους στις σύγχρονες αρχιτεκτονικές εφαρμογών, η δημιουργία ασφαλών API είναι ζωτικής σημασίας. Η ασφάλεια δεν μπορεί να παραμεληθεί και θα πρέπει να αποτελεί μέρος ολόκληρου του κύκλου ζωής της ανάπτυξης των APIs. Η σάρωση (scanning) και οι ετήσιες δοκιμές διείσδυσης (penetration tests) δεν είναι πλέον αρκετές.

Οι DevSecOps θα πρέπει να συμμετάσχουν στην προσπάθεια ανάπτυξης, διευκολύνοντας τις συνεχείς δοκιμές ασφαλείας σε ολόκληρο τον κύκλο ζωής ανάπτυξης λογισμικού. Στόχος τους είναι να ενισχύσουν τον αγωγό ανάπτυξης (development pipeline) με αυτοματισμό ασφαλείας (security automation), χωρίς να επηρεάζουν την ταχύτητα ανάπτυξης.

Για περισσότερες πληροφορίες, μπορείτε να ελέγχετε συχνά το [DevSecOps Manifesto](#) για να μείνετε ενημερωμένοι.

<b>Κατανοήστε το Μοντέλο Απειλής (Threat Model)</b>	Οι προτεραιότητες δοκιμών (testing priorities) προέρχονται από ένα μοντέλο απειλής (threat model). Εάν δεν έχετε μοντέλο απειλής, εξετάστε το ενδεχόμενο να χρησιμοποιήσετε το <a href="#">Πρότυπο Επαλήθευσης Ασφάλειας Εφαρμογών OWASP (OWASP Application Security Verification Standard) (ASVS)</a> και τον <a href="#">Οδηγό Δοκιμής OWASP (OWASP Testing Guide)</a> . Η συμμετοχή της ομάδας ανάπτυξης μπορεί να βοηθήσει τους προγραμματιστές να γίνουν περισσότερο ενήμεροι για την ασφάλεια.
<b>Κατανοήστε το SDLC</b>	Γίνετε μέλος της ομάδας ανάπτυξης για να κατανοήσετε καλύτερα τον Κύκλο Ζωής Ανάπτυξης Λογισμικού. Η συνεισφορά σας στις συνεχείς δοκιμές ασφαλείας θα πρέπει να είναι συμβατή με τα άτομα, τις διαδικασίες και τα εργαλεία που έχουν θεσπιστεί. Να επισημανθεί ότι πρέπει να συμφωνούν όλοι με τη διαδικασία, ώστε να μην υπάρχουν περιττές τριβές ή αντιστάσεις.
<b>Στρατηγικές Testing</b>	Καθώς η εργασία σας δεν πρέπει να επηρεάζει την ταχύτητα ανάπτυξης του API, θα πρέπει να επιλέξετε με σύνεση την καλύτερη (απλή, ταχύτερη, πιο ακριβή) τεχνική για να επαληθεύσετε τις απαιτήσεις ασφαλείας. Το <a href="#">Γνωσιακό Πλαίσιο Ασφαλείας OWASP (OWASP Security Knowledge Framework)</a> και το <a href="#">Πρότυπο Επαλήθευσης Ασφάλειας Εφαρμογών OWASP (OWASP Application Security Verification Standard)</a> μπορούν να αποτελέσουν εξαιρετικές πηγές λειτουργικών και μη λειτουργικών απαιτήσεων ασφαλείας. Υπάρχουν άλλες εξαιρετικές πηγές για <a href="#">Έργα</a> and <a href="#">Εργαλεία</a> παρόμοιες με αυτήν που προσφέρει η <a href="#">Κοινότητα DevSecOps</a> .
<b>Επίτευξη Κάλυψης (Coverage) και ακρίβειας (Accuracy)</b>	Είστε η γέφυρα μεταξύ προγραμματιστών και ομάδων λειτουργιών (operation teams). Για να πετύχετε κάλυψη (coverage), δεν πρέπει να εστιάσετε μόνο στη λειτουργικότητα, αλλά και στην ενορχήστρωση. Εργαστείτε από την αρχή κοντά στις ομάδες ανάπτυξης και λειτουργίας, ώστε να βελτιστοποιήσετε τον χρόνο και την προσπάθειά σας. Θα πρέπει να στοχεύετε σε μια κατάσταση όπου η βασική ασφάλεια επαληθεύεται συνεχώς.
<b>Επικοινωνήστε με σαφήνεια τα ευρήματα</b>	Συνεισφέρετε αξία με λιγότερη ή καθόλου τριβή. Παραδώστε τα ευρήματα σας έγκαιρα, μέσα στα εργαλεία που χρησιμοποιούν οι ομάδες ανάπτυξης (όχι αρχεία PDF). Γίνετε μέλος της ομάδας ανάπτυξης για να αντιμετωπίσετε τα ευρήματα. Εκμεταλλευτείτε την ευκαιρία για να τους εκπαιδεύσετε, περιγράφοντας ξεκάθαρα την αδυναμία και πώς μπορεί να γίνει κατάχρηση της, συμπεριλαμβάνοντας ένα σενάριο επίθεσης για να γίνετε κατανοητοί.

## Σύνοψη

Δεδομένου ότι η βιομηχανία AppSec δεν έχει εστιάσει στην πιο πρόσφατη αρχιτεκτονική εφαρμογών, στην οποία τα API διαδραματίζουν σημαντικό ρόλο, η σύνταξη μιας λίστας με τους δέκα πιο κρίσιμους κινδύνους ασφαλείας API, με βάση μια δημόσια πρόσκληση για δεδομένα, θα ήταν δύσκολο έργο. Παρά το γεγονός ότι δεν υπάρχει δημόσια κλήση δεδομένων, η προκύπτουσα λίστα Top 10 εξακολουθεί να βασίζεται σε δημόσια διαθέσιμα δεδομένα, συνεισφορές ειδικών σε θέματα ασφαλείας και σε ανοιχτή συζήτηση με την κοινότητα ασφαλείας.

## Μεθοδολογία και Δεδομένα

Στην πρώτη φάση συλλέχθηκαν, εξετάστηκαν και κατηγοριοποιήθηκαν δημόσια δεδομένα σχετικά με συμβάντα ασφαλείας API από μια ομάδα ειδικών σε θέματα ασφαλείας. Τα δεδομένα αυτά συλλέχθηκαν από πλατφόρμες επιβράβευσης σφαλμάτων (bug bounty platforms) και βάσεις δεδομένων ευπάθειας, τα οποία καταχωρήθηκαν στο χρονικό διάστημα ενός έτους και χρησιμοποιήθηκαν για στατιστικούς σκοπούς.

Στην επόμενη φάση οι επαγγελματίες ασφαλείας με εμπειρία σε δοκιμές διείσδυσης (penetration testing) κλήθηκαν να συντάξουν τη δική τους λίστα Top 10.

Η [OWASP Risk Rating Methodology](#) χρησιμοποιήθηκε για την εκτέλεση της Ανάλυσης Κινδύνου. Οι βαθμολογίες συζητήθηκαν και αναθεωρήθηκαν μεταξύ των επαγγελματιών ασφαλείας. Για περισσότερες πληροφορίες σχετικά με αυτά τα θέματα ανατρέξτε στην ενότητα [Κίνδυνοι Ασφαλείας API](#).

Το πρώτο προσχέδιο του OWASP API Security Top 10 2019 προέκυψε από συναίνεση των στατιστικών αποτελεσμάτων της πρώτης φάσης και από τις λίστες που προετοίμασαν οι επαγγελματίες ασφαλείας. Το προσχέδιο υποβλήθηκε στη συνέχεια για εκτίμηση και αξιολόγηση από άλλη ομάδα επαγγελματιών ασφαλείας, με σχετική εμπειρία στους τομείς ασφαλείας API.

Το OWASP API Security Top 10 2019 παρουσιάστηκε για πρώτη φορά στην εκδήλωση OWASP Global AppSec Tel Aviv (Μάιος 2019). Από τότε είναι διαθέσιμο στο GitHub για δημόσια συζήτηση και συνεισφορές.

Η λίστα των συντελεστών είναι διαθέσιμη στην ενότητα [Ευχαριστίες](#).

## Ευχαριστίες προς τους Συντελεστές

Θα θέλαμε να ευχαριστήσουμε τους ακόλουθους συντελεστές που συνεισέφεραν δημόσια στο GitHub ή με άλλα μέσα:

- 007divyachawla
- Abid Khan
- Adam Fisher
- anotherik
- bkimminich
- caseysoftware
- Chris Westphal
- dsopas
- DSotnikov
- emilva
- ErezYalon
- flascelles
- Guillaume Benats
- IgorSasovets
- Inonshk
- JonnySchnittger
- jmanico
- jmdx
- Keith Casey
- kozmic
- LauraRosePorter
- Matthieu Estrade
- nathanawmk
- PauloASilva
- pentagramz
- philippederyck
- pleothaud
- r00ter
- Raj kumar
- Sagar Popat
- Stephen Gates
- thomaskonrad
- xycloops123
- Athanasios Emmanouilidis
- Apostolos Giannakidis