ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πρόγραμμα Μεταπτυχιακών Σπουδών
Κυβερνοασφάλεια και Επιστήμη Δεδομένων

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Αυτοματοποίηση Ελέγχου Διείσδυσης μέσω Αρχιτεκτονικής Πολλαπλών AI Agents** **Penetration Testing Automation through Multi-Agent AI Architecture**
Ονοματεπώνυμο Φοιτητή	**Δημήτρης Μπαλτζής**
Πατρώνυμο	**Σωκράτης**
Αριθμός Μητρώου	**ΜΠΚΕΔ24026**
Επιβλέπων	**ΚΑΡΑΝΤΖΙΑΣ ΑΘΑΝΑΣΙΟΣ, Διδάσκων ΠΜΣ**

Ημερομηνία Παράδοσης / Ιούλιος 2026

Τριμελής Εξεταστική Επιτροπή

Δρ. Αθανάσιος Καραντζιάς Διδάσκων ΠΜΣ	Δρ. Σπυρίδων Παπαστεργίου Διδάσκων ΠΜΣ	Παναγιώτης Κοτζανικολάου Καθηγητής

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή, Δρ. Θάνο Καραντζιά, για την καθοδήγηση και την υποστήριξή του καθ' όλη τη διάρκεια εκπόνησης της παρούσας διατριβής. Επίσης, ευχαριστώ τα μέλη της τριμελούς εξεταστικής επιτροπής για τον χρόνο και τις παρατηρήσεις τους.

Η παρούσα διατριβή εκπονήθηκε με την υποστήριξη υποτροφίας από το Κοινωφελές Ίδρυμα Ιωάννη Σ. Λάτση, το οποίο ευχαριστώ θερμά για την οικονομική στήριξη κατά τη διάρκεια των μεταπτυχιακών μου σπουδών.

Περίληψη

Στο πλαίσιο της παρούσας διπλωματικής εργασίας προτείνεται, σχεδιάζεται και υλοποιείται ένα αυτοματοποιημένο σύστημα ελέγχου διείσδυσης (penetration testing) βασισμένο σε αρχιτεκτονική πολλαπλών AI agents. Το σύστημα αξιοποιεί το Model Context Protocol (MCP) και το μοντέλο DeepSeek για τον συντονισμό δύο εξειδικευμένων agents: τον Agent Ανάλυσης, ο οποίος εκτελεί παθητική αναγνώριση και παράγει δομημένο πλάνο επίθεσης, και τον Pentester Agent, ο οποίος αναλαμβάνει την εκτέλεση και επιβεβαίωση των ευπαθειών. Η φάση αναγνώρισης υλοποιείται μέσω της αλυσίδας Wappalyzer, FastCVE και ExploitDB, ενώ η εκτέλεση βασίζεται κυρίως σε templates του Nuclei με εναλλακτικό μονοπάτι μέσω ExploitDB. Το σύστημα αξιολογήθηκε σε δύο ελεγχόμενα περιβάλλοντα Docker, επιβεβαιώνοντας ευπάθειες τύπου path traversal (CVE-2021-41773) και απομακρυσμένης εκτέλεσης κώδικα μέσω reverse shell (CVE-2017-5638). Τα αποτελέσματα έδειξαν ότι το σύστημα μπορεί να αυτοματοποιήσει αξιόπιστα το πλήρες pipeline ενός penetration test με ελάχιστη ανθρώπινη παρέμβαση.

Abstract

This thesis proposes, designs, and implements an automated penetration testing system based on a multi-agent AI architecture, which is subsequently evaluated in a controlled environment. The system leverages the Model Context Protocol (MCP) and the DeepSeek language model to coordinate two specialized agents: an Analyst Agent, responsible for passive reconnaissance and generating a structured attack plan, and a Pentester Agent, which handles exploit execution and vulnerability confirmation. The reconnaissance phase is implemented through a pipeline consisting of Wappalyzer, FastCVE, and ExploitDB, while execution relies primarily on Nuclei templates with a fallback path via ExploitDB. The system was evaluated in two controlled Docker environments, successfully confirming a path traversal vulnerability (CVE-2021-41773) and a remote code execution vulnerability via reverse shell (CVE-2017-5638). Results demonstrate that the system can reliably automate the full penetration testing pipeline with minimal human intervention, validating the effectiveness of the multi-agent role separation approach.

Πίνακας Περιεχομένων

1 Εισαγωγή

Το παρόν κεφάλαιο αποτελεί την εισαγωγή στη διπλωματική εργασία και έχει ως στόχο να παρουσιάσει το πλαίσιο μέσα στο οποίο αναπτύχθηκε η έρευνα, το πρόβλημα που καλείται να αντιμετωπίσει, καθώς και τους σκοπούς και τα όρια της. Αρχικά παρουσιάζεται το κίνητρο και το πρόβλημα που οδήγησε στην ανάπτυξη του συστήματος, ακολουθεί η περιγραφή του σκοπού και της συνεισφοράς της εργασίας, τα ερευνητικά ερωτήματα που διέπουν την έρευνα, οι στόχοι που τέθηκαν, τα όρια του συστήματος και τέλος η δομή της εργασίας.

1.1 Κίνητρο και Πρόβλημα

Το σύγχρονο ψηφιακό περιβάλλον χαρακτηρίζεται από μια διαρκώς επιταχυνόμενη αύξηση των κυβερνοαπειλών, τόσο σε όγκο όσο και σε πολυπλοκότητα. Οι κυβερνοεπιθέσεις δεν αποτελούν πλέον σπάνιο φαινόμενο, είναι καθημερινή πραγματικότητα για οργανισμούς κάθε μεγέθους, με συνέπειες που κυμαίνονται από οικονομικές ζημίες και απώλεια εμπιστευτικών δεδομένων έως πλήρη διακοπή λειτουργίας κρίσιμων υποδομών. Το κόστος, τόσο σε χρήμα όσο και σε εμπιστοσύνη, καθιστά την αντιμετώπιση της ασφάλειας επιτακτική ανάγκη και όχι προαιρετική επιλογή.

Ωστόσο, ένα από τα πιο κρίσιμα προβλήματα στον τομέα δεν είναι μόνο η ανακάλυψη ευπαθειών, αλλά η πρακτική επικύρωση τους. Οι οργανισμοί επενδύουν σε εργαλεία σάρωσης που παράγουν εκτεταμένες λίστες αδυναμιών, χωρίς όμως να αποδεικνύουν αν αυτές είναι πραγματικά εκμεταλλεύσιμες στο συγκεκριμένο περιβάλλον. Δημιουργείται έτσι ένα χάσμα μεταξύ θεωρητικής ανάλυσης κινδύνου και πρακτικής απόδειξης, ένα χάσμα που μόνο ο έμπειρος penetration tester μπορεί να γεφυρώσει. Εκεί ακριβώς εντοπίζεται το δεύτερο κρίσιμο πρόβλημα: η έλλειψη εξειδικευμένων επαγγελματιών. Οι pentesters αποτελούν μία από τις πιο δυσεύρετες ειδικότητες στον χώρο, με αποτέλεσμα ένας πλήρης έλεγχος διείσδυσης να είναι χρονοβόρος και δαπανηρός και συχνά απρόσιτος για μικρότερους οργανισμούς. Η ζήτηση ξεπερνά κατά πολύ την προσφορά, και αυτή η ανισορροπία δεν αναμένεται να αντιστραφεί σύντομα.

Στην αγορά υπάρχουν βεβαίως εργαλεία που επιχειρούν να καλύψουν αυτό το κενό, όπως το Metasploit Framework και το Burp Suite. Παρόλα αυτά, και τα δύο προϋποθέτουν βαθιά τεχνική γνώση όπου δεν αποτελούν ολοκληρωμένα pipelines, αλλά εργαλεία που απαιτούν ο χρήστης να γνωρίζει ήδη τι ψάχνει, πού να κοιτάξει και πώς να ερμηνεύσει τα αποτελέσματα. Η λογική της επίθεσης, η οποία περιλαμβάνει την απόφαση για το ποια ευπάθεια να δοκιμαστεί, με ποια σειρά και με ποιον συγκεκριμένο τρόπο, παραμένει αποκλειστικά ανθρώπινη ευθύνη.

Η εμφάνιση των Μεγάλων Γλωσσικών Μοντέλων (Large Language Models, LLMs) μετασχηματίζει αυτό το δεδομένο, επιτρέποντας την ουσιαστική αυτοματοποίηση της σύνθετης αυτής στρατηγικής. Τα LLMs διαθέτουν την ικανότητα να επεξεργάζονται πολύπλοκη και ετερογενή πληροφορία, να συνθέτουν αποφάσεις και να κατευθύνουν εξειδικευμένα εργαλεία σε δομημένες ακολουθίες ενεργειών.

Σε συνδυασμό με σύγχρονα frameworks ορχήστρωσης, όπως το **Model Context Protocol** (MCP), καθίσταται πλέον εφικτή η κατασκευή συστημάτων που δεν περιορίζονται στην απλή σάρωση, αλλά αναλύουν το περιβάλλον και δρουν αυτόνομα με στόχο την έμπρακτη απόδειξη εκμεταλλευσιμότητας (exploitability). Η παρούσα εργασία αναπτύσσεται ακριβώς σε αυτό το τεχνολογικό παράθυρο ευκαιρίας, γεφυρώνοντας το χάσμα μεταξύ της στατικής ανίχνευσης και της ενεργητικής αξιολόγησης ασφαλείας.

1.2 Σκοπός και συμβολή

Η ραγδαία εξέλιξη του τοπίου των κυβερνοαπειλών καθιστά την απλή καταγραφή ευπαθειών ανεπαρκή, αν αυτή δεν συνοδεύεται από έμπρακτη επικύρωση του κινδύνου. Η παρούσα

διπλωματική εργασία παρουσιάζει την ανάπτυξη μιας καινοτόμου εφαρμογής αυτοματοποιημένου ελέγχου διείσδυσης (Penetration Testing), η οποία μεταβαίνει από τη θεωρητική ανάλυση στην πρακτική απόδειξη (Proof of Concept).

Η αρχιτεκτονική του συστήματος βασίζεται στο Model Context Protocol (MCP), αξιοποιώντας την ευφυΐα του μοντέλου DeepSeek για τον συντονισμό δύο εξειδικευμένων AI Agents. Η διαδικασία ξεκινά με τη φάση της αναγνώρισης (reconnaissance), όπου μέσω εργαλείων όπως τα Wappalyzer, Fastcve και ExploitDB, συλλέγονται δεδομένα για το τεχνολογικό υπόβαθρο και τις πιθανές αδυναμίες του στόχου.

Η καινοτομία της εφαρμογής έγκειται στον διαχωρισμό των ρόλων:

- Ο Analyst Agent επεξεργάζεται τον όγκο της πληροφορίας για να συνθέσει ένα δυναμικό στρατηγικό πλάνο επίθεσης (Attack Plan).
- Ο Pentester Agent αναλαμβάνει την επιχειρησιακή δράση, χρησιμοποιώντας εργαλεία όπως το Nuclei και άμεσα HTTP requests για την εκτέλεση και επιβεβαίωση των exploits.

Η αξία του εργαλείου δεν περιορίζεται στην απλή παράθεση πηγών και ευπαθειών, η ουσιαστική συνεισφορά του είναι η ενεργός αξιοποίηση των ευρημάτων. Με τον τρόπο αυτό, το σύστημα δεν πληροφορεί απλώς τον χρήστη για το "τι" είναι ευάλωτο, αλλά αποδεικνύει στην πράξη το "πώς" μπορεί να παραβιαστεί, προσφέροντας μια ρεαλιστική και αυτοματοποιημένη προσέγγιση στην επιθετική ασφάλεια (Offensive Security).

1.3 Ερευνητικά Ερωτήματα

Η παρούσα εργασία διέπεται από τρία κεντρικά ερευνητικά ερωτήματα, τα οποία προέκυψαν φυσικά από τις προκλήσεις που ανέκυψαν κατά τη διάρκεια της υλοποίησης.

Το πρώτο αφορά στον βαθμό στον οποίο ένα multi-agent σύστημα μπορεί να αυτοματοποιήσει αξιόπιστα το πλήρες pipeline ενός penetration test από τη φάση αναγνώρισης έως την εκτέλεση exploit, διασφαλίζοντας ότι κάθε εργαλείο χρησιμοποιείται επαρκώς και με τη σωστή σειρά, ακόμα και όταν ενδιάμεσα βήματα αποτυγχάνουν ή αποδίδουν ατελή αποτελέσματα.

Το δεύτερο εξετάζει κατά πόσο ο διαχωρισμός ρόλων μεταξύ Analyst Agent και Pentester Agent, και ο τρόπος ορισμού των αντίστοιχων prompts και ορίων αρμοδιότητας, αποτελεί αποτελεσματική στρατηγική για τη διαχείριση της πολυπλοκότητας και την εξασφάλιση συνεκτικής ροής εκτέλεσης.

Το τρίτο αφορά στα πρακτικά όρια του συστήματος που απορρέουν από την εξάρτησή του από εξωτερικά εργαλεία, όπως η ατελής κάλυψη τεχνολογιών από το Wappalyzer ή η ανομοιογένεια των αρχείων του ExploitDB, και στον τρόπο με τον οποίο το σύστημα διαχειρίζεται αυτές τις αβεβαιότητες, για παράδειγμα μέσω της προτεραιοποίησης Nuclei templates έναντι exploit αρχείων αδιευκρίνιστης αξιοπιστίας.

1.4 Στόχοι Εργασίας

Με βάση τα παραπάνω ερευνητικά ερωτήματα, οι στόχοι της εργασίας ορίζονται ως εξής. Πρώτος στόχος είναι ο σχεδιασμός και η υλοποίηση του reconnaissance pipeline μέσω της ενσωμάτωσης των εργαλείων Wappalyzer, FastCVE και ExploitDB σε μια ενιαία και συνεκτική αλυσίδα που παράγει δομημένο προφίλ ευπαθειών. Δεύτερος στόχος είναι ο σχεδιασμός και η υλοποίηση της αρχιτεκτονικής multi-agent συστήματος, αξιοποιώντας το Model Context Protocol και το μοντέλο DeepSeek ως υποκείμενη νοημοσύνη. Τρίτος στόχος είναι η ανάπτυξη του Agent Ανάλυσης, ο οποίος επεξεργάζεται το προφίλ ευπαθειών και συνθέτει ένα δυναμικό Attack Plan με ιεράρχηση προτεραιοτήτων βάσει της διαθεσιμότητας exploit. Τέταρτος στόχος είναι η ανάπτυξη του Pentester Agent για την αυτόνομη εκτέλεση και επιβεβαίωση ευπαθειών μέσω Nuclei templates και HTTP

requests, με εναλλακτικό μονοπάτι εκτέλεσης μέσω ExploitDB. Πέμπτος και τελευταίος στόχος είναι η αξιολόγηση του συνολικού συστήματος σε ελεγχόμενο περιβάλλον, με σκοπό την ανάδειξη των δυνατοτήτων και των περιορισμών του.

1.5 Όρια Συστήματος

Είναι σημαντικό να οριστεί με σαφήνεια το πεδίο εφαρμογής του συστήματος, ώστε να αποφευχθούν παρερμηνείες ως προς τις δυνατότητές του. Το σύστημα εστιάζει αποκλειστικά στο web application penetration testing και καλύπτει τις φάσεις της αναγνώρισης, της ανάλυσης και της εκτέλεσης exploit. Όλες οι δοκιμές πραγματοποιούνται σε ελεγχόμενα και εξουσιοδοτημένα περιβάλλοντα, και το σύστημα δεν προορίζεται, ούτε έχει σχεδιαστεί, για χρήση σε μη εξουσιοδοτημένους στόχους.

Εκτός πεδίου εφαρμογής βρίσκονται οι φάσεις post-exploitation, όπως η εγκατάσταση backdoors, η πλευρική κίνηση εντός δικτύου και η εξαγωγή δεδομένων. Επίσης, το σύστημα δεν καλύπτει network-level ή infrastructure pentesting, ούτε τεχνικές παράκαμψης συστημάτων ανίχνευσης όπως WAF evasion ή IDS bypass. Τέλος, η αυτόματη παραγωγή τελικής έκθεσης (report) δεν αποτελεί μέρος της παρούσας υλοποίησης και αφήνεται ως πεδίο μελλοντικής επέκτασης.

1.6 Δομή Εργασίας

Η εργασία οργανώνεται σε εννέα κεφάλαια, ακολουθώντας τη λογική του pipeline που υλοποιήθηκε. Το παρόν κεφάλαιο έθεσε το πλαίσιο, τους στόχους και τα ερευνητικά ερωτήματα της έρευνας. Το Κεφάλαιο 2 παρουσιάζει το θεωρητικό υπόβαθρο, καλύπτοντας έννοιες από τον χώρο του penetration testing, των ευπαθειών λογισμικού, των LLM agents και του Model Context Protocol. Το Κεφάλαιο 3 αναλύει τη φάση αναγνώρισης και τα εργαλεία που τη συνθέτουν. Το Κεφάλαιο 4 περιγράφει τον Agent Ανάλυσης και τη διαδικασία παραγωγής του Attack Plan. Το Κεφάλαιο 5 αφιερώνεται στον Pentester Agent και στην εκτέλεση των ευπαθειών. Το Κεφάλαιο 6 παρουσιάζει την ολοκλήρωση του συστήματος ως σύνολο, με έμφαση στη ροή δεδομένων και τις σχεδιαστικές επιλογές. Το Κεφάλαιο 7 περιλαμβάνει την αξιολόγηση του συστήματος μέσω δύο σεναρίων εκτέλεσης σε ελεγχόμενο περιβάλλον. Το Κεφάλαιο 8 αποτιμά τα αποτελέσματα, αναλύει τους περιορισμούς και παρουσιάζει τις προοπτικές μελλοντικής επέκτασης. Το Κεφάλαιο 9 παρουσιάζει τα συμπεράσματα της εργασίας.

2 Θεωρητικό Υπόβαθρο

2.1 Έλεγχος Διείσδυσης

Ο έλεγχος διείσδυσης (Penetration Testing ή εν συντομία Pentest) είναι μια μεθοδολογική διαδικασία κατά την οποία εξουσιοδοτημένοι επαγγελματίες προσομοιώνουν πραγματικές κυβερνοεπιθέσεις εναντίον συστημάτων, δικτύων ή εφαρμογών, με σκοπό τον εντοπισμό και την αξιολόγηση των αδυναμιών τους πριν αυτές εκμεταλλευτούν από κακόβουλους τρίτους.Η βασική διαφορά από άλλες μεθόδους ασφαλείας είναι ότι το penetration testing δεν περιορίζεται στην καταγραφή ευπαθειών, επιχειρεί ενεργά να τις αξιοποιήσει, προσφέροντας έτσι μια ρεαλιστική εικόνα του πραγματικού κινδύνου.

Στην πράξη, ένας πλήρης έλεγχος διείσδυσης ακολουθεί μια δομημένη ακολουθία φάσεων. Η πρώτη είναι η φάση αναγνώρισης (Reconnaissance), κατά την οποία συλλέγονται πληροφορίες για τον στόχο, τεχνολογίες που χρησιμοποιεί, εκδόσεις λογισμικού, πιθανά σημεία εισόδου. Ακολουθεί η φάση σάρωσης και ανάλυσης (Scanning & Analysis), όπου τα συλλεχθέντα δεδομένα αντιστοιχίζονται με γνωστές ευπάθειες και αξιολογούνται ως προς την εκμεταλλευσιμότητά τους. Στη φάση

εκμετάλλευσης (Exploitation) επιχειρείται η πρακτική αξιοποίηση των ευπαθειών που εντοπίστηκαν, με στόχο την απόδειξη ότι ο κίνδυνος είναι πραγματικός και όχι θεωρητικός. Τέλος, η φάση αναφοράς (Reporting) συγκεντρώνει τα ευρήματα σε δομημένη έκθεση με προτάσεις αντιμετώπισης.

Είναι σημαντικό να διακριθούν δύο βασικές κατηγορίες ελέγχου. Στο manual penetration testing, ένας έμπειρος επαγγελματίας διενεργεί ολόκληρη τη διαδικασία με βάση την κρίση και την εμπειρία του, προσαρμόζοντας δυναμικά την προσέγγισή του σε κάθε στόχο. Στο automated penetration testing, εργαλεία και συστήματα αναλαμβάνουν μέρος ή το σύνολο της διαδικασίας, με κέρδος ταχύτητας και κλιμάκωσης αλλά συνήθως με μικρότερη ευελιξία. Η παρούσα εργασία κινείται ακριβώς στο μεταίχμιο αυτών των δύο κατηγοριών, επιχειρώντας να προσεγγίσει την ευελιξία του manual testing μέσω της αυτοματοποίησης.

2.2 Ευπάθειες Λογισμικού και Μηχανισμοί Εκμετάλλευσης τους

Για να κατανοηθεί η λειτουργία του συστήματος που αναπτύχθηκε, είναι απαραίτητο να παρουσιαστεί ο τρόπος με τον οποίο οργανώνεται και κυκλοφορεί η γνώση γύρω από τις ευπάθειες λογισμικού, αυτό που αποκαλείται ecosystem των exploits.

Κάθε γνωστή ευπάθεια λογισμικού λαμβάνει έναν μοναδικό αναγνωριστικό κωδικό, γνωστό ως CVE (Common Vulnerabilities and Exposures). Το σύστημα αυτό διατηρείται από τον οργανισμό MITRE και αποτελεί το παγκόσμιο πρότυπο αναφοράς για την καταγραφή και επικοινωνία ευπαθειών. Κάθε CVE συνοδεύεται από περιγραφή της αδυναμίας, το λογισμικό που επηρεάζει και συχνά από μια βαθμολογία σοβαρότητας βάσει του συστήματος CVSS (Common Vulnerability Scoring System), η οποία κυμαίνεται από 0 έως 10. Η βαθμολογία αυτή επιτρέπει την ιεράρχηση των ευπαθειών και την εστίαση σε αυτές που παρουσιάζουν τον υψηλότερο κίνδυνο.

Η ύπαρξη ενός CVE ωστόσο δεν σημαίνει αυτόματα ότι υπάρχει και έτοιμος κώδικας εκμετάλλευσής του. Εκεί έρχεται ο ρόλος του ExploitDB, μιας δημόσιας βάσης δεδομένων που συγκεντρώνει έτοιμα exploit scripts και Proof of Concept κώδικες, αντιστοιχισμένα με τα αντίστοιχα CVEs. Το ExploitDB επιτρέπει στον pentester να βρει γρήγορα αν για μια συγκεκριμένη ευπάθεια υπάρχει ήδη διαθέσιμο εργαλείο εκμετάλλευσης. Ωστόσο, η ποιότητα και η αξιοπιστία των αρχείων αυτών δεν είναι ομοιογενής, κάποια είναι πλήρως λειτουργικά, άλλα απαιτούν προσαρμογή, και άλλα ενδέχεται να μην λειτουργούν καθόλου στο συγκεκριμένο περιβάλλον. Αυτή η ανομοιογένεια αποτελεί μία από τις βασικές προκλήσεις που αντιμετωπίζει το σύστημα της παρούσας εργασίας, όπως αναλύεται στα επόμενα κεφάλαια.

Συμπληρωματικά, εργαλεία όπως το Nuclei παρέχουν μια διαφορετική προσέγγιση: αντί για raw exploit κώδικα, χρησιμοποιούν δομημένα templates που ορίζουν με ακρίβεια πώς να ελεγχθεί μια συγκεκριμένη ευπάθεια. Τα templates αυτά είναι πιο αξιόπιστα και ευκολότερα στη χρήση, καθώς έχουν σχεδιαστεί ειδικά για αυτοματοποιημένη εκτέλεση. Για αυτόν ακριβώς τον λόγο, στο σύστημα που αναπτύχθηκε, ο Pentester Agent προτεραιοποιεί τα Nuclei templates έναντι των αρχείων του ExploitDB, καταφεύγοντας στα τελευταία μόνο όταν δεν υπάρχει διαθέσιμο template.

2.3 Μεγάλα Γλωσσικά Μοντέλα και AI Agents

Τα Μεγάλα Γλωσσικά Μοντέλα (Large Language Models — LLMs) αποτελούν μια κατηγορία τεχνητής νοημοσύνης που εκπαιδεύεται σε τεράστιες ποσότητες κειμένου, αποκτώντας την ικανότητα να κατανοεί και να παράγει φυσική γλώσσα με αξιοσημείωτη συνέπεια και πολυπλοκότητα. Η ικανότητα αυτή δεν περιορίζεται στην απλή παραγωγή κειμένου, τα LLMs μπορούν να αναλύουν δομημένη πληροφορία, να συνθέτουν συμπεράσματα, να σχεδιάζουν ακολουθίες ενεργειών και να προσαρμόζουν τη συμπεριφορά τους βάσει του πλαισίου που τους δίνεται. Αυτές οι ιδιότητες τα καθιστούν ιδιαίτερα κατάλληλα για εργασίες που απαιτούν κρίση και όχι απλή εκτέλεση σταθερών κανόνων.

Ένα κρίσιμο βήμα στην εξέλιξη των LLMs ήταν η ανάπτυξη της ικανότητας χρήσης εξωτερικών εργαλείων (tool use). Αντί να περιορίζονται στην παραγωγή κειμένου, τα μοντέλα αυτά μπορούν να καλούν εξωτερικές συναρτήσεις και υπηρεσίες, όπως αναζήτηση στο διαδίκτυο, εκτέλεση κώδικα ή κλήση APIs, και να ενσωματώνουν τα αποτελέσματα στη ροή της σκέψης τους. Με αυτόν τον τρόπο το LLM δεν είναι απλώς ένα μοντέλο που απαντά, αλλά ένα σύστημα που ενεργεί.

Η ικανότητα αυτή τεκμηριώθηκε ερευνητικά από τους Schick et al. [13], οι οποίοι έδειξαν ότι τα γλωσσικά μοντέλα μπορούν να μάθουν αυτόνομα πότε και πώς να καλούν εξωτερικά εργαλεία μέσα στη ροή επεξεργασίας τους. Ένα κρίσιμο ζήτημα ωστόσο δεν είναι μόνο αν το μοντέλο μπορεί να καλέσει ένα εργαλείο, αλλά πώς οργανώνει τη σκέψη του γύρω από τη χρήση του. Οι Yao et al. [12] πρότειναν το ReAct, ένα πλαίσιο στο οποίο το μοντέλο εναλλάσσει βήματα συλλογισμού (reasoning) και δράσης (acting) αντί να τα εκτελεί ανεξάρτητα. Στην πράξη αυτό σημαίνει ότι ο agent δεν σχεδιάζει εκ των προτέρων μια πλήρη ακολουθία ενεργειών για να την εκτελέσει τυφλά. Αντίθετα, μετά από κάθε ενέργεια εξετάζει το αποτέλεσμα, αξιολογεί αν η πληροφορία που έλαβε είναι επαρκής, και αποφασίζει δυναμικά το επόμενο βήμα. Αυτός ο κύκλος παρατήρησης, συλλογισμού και δράσης επαναλαμβάνεται μέχρι να επιτευχθεί ο στόχος ή να εξαντληθούν οι διαθέσιμες επιλογές.

Η αρχιτεκτονική αυτή είναι ιδιαίτερα κατάλληλη για εργασίες penetration testing, όπου κάθε βήμα εξαρτάται από τα ευρήματα του προηγούμενου και η στρατηγική πρέπει να προσαρμόζεται συνεχώς στα δεδομένα που προκύπτουν. Στο σύστημα που αναπτύχθηκε στο πλαίσιο της παρούσας εργασίας, τόσο ο Analyst Agent όσο και ο Pentester Agent ακολουθούν αυτό το μοτίβο: ο Pentester Agent, για παράδειγμα, δεν εκτελεί όλα τα CVEs του Attack Plan με προκαθορισμένο τρόπο, αλλά αξιολογεί το αποτέλεσμα κάθε δοκιμής και αποφασίζει αν θα συνεχίσει με Nuclei template ή θα μεταβεί στο εναλλακτικό μονοπάτι μέσω ExploitDB.

Όταν ένα LLM συνδυάζεται με εργαλεία, στόχους και την ικανότητα να λαμβάνει αποφάσεις για το πώς θα τα χρησιμοποιήσει, τότε ορίζεται ως ένας AI Agent. Ο Agent δεν εκτελεί απλώς εντολές, παρατηρεί το περιβάλλον, αποφασίζει ποια ενέργεια είναι καταλληλότερη, εκτελεί και αξιολογεί το αποτέλεσμα, συνεχίζοντας μέχρι να επιτύχει τον στόχο του. Αυτή η λογική είναι ιδιαίτερα σημαντική στο πλαίσιο της παρούσας εργασίας, καθώς τόσο ο Analyst Agent όσο και ο Pentester Agent λειτουργούν ακριβώς με αυτή τη λογική: παρατηρούν, αποφασίζουν, εκτελούν και αξιολογούν σε κάθε βήμα του pipeline.

2.4 Το Πρωτόκολλο MCP και η Ενορχήστρωση Εργαλείων

Το Model Context Protocol (MCP) είναι ένα ανοιχτό πρωτόκολλο επικοινωνίας με σκοπό την τυποποίηση του τρόπου με τον οποίο τα LLMs αλληλεπιδρούν με εξωτερικά εργαλεία, υπηρεσίες και πηγές δεδομένων. Πριν από την εμφάνισή του, κάθε ενσωμάτωση ενός μοντέλου με εξωτερικά συστήματα απαιτούσε custom υλοποίηση. Το MCP έρχεται να θεσπίσει έναν κοινό τρόπο επικοινωνίας, ανεξάρτητα από το μοντέλο ή το εργαλείο που χρησιμοποιείται.

Η αρχιτεκτονική του MCP βασίζεται σε μια σχέση client-server. Ο MCP server εκθέτει ένα σύνολο εργαλείων (tools) που το μοντέλο μπορεί να καλέσει, ενώ ο MCP client, ο οποίος ουσιαστικά αποτελεί το ίδιο το μοντέλο ή το framework διαχείρισης του, αποφασίζει πότε και πώς θα χρησιμοποιήσει αυτά τα εργαλεία. Κάθε εργαλείο περιγράφεται με σαφήνεια ως προς την είσοδο που δέχεται και την έξοδο που επιστρέφει, επιτρέποντας στο μοντέλο να κατανοεί τις δυνατότητές του και να τις αξιοποιεί κατάλληλα.

Αυτό που καθιστά το MCP ιδιαίτερα σημαντικό για την παρούσα εργασία είναι η δυνατότητα ενορχήστρωσης πολλαπλών εργαλείων σε μια συνεκτική ροή. Αντί να απαιτείται από τον προγραμματιστή ο χειροκίνητος ορισμός κάθε βήματος της αλληλεπίδρασης, το μοντέλο μέσω του MCP δύναται να αποφασίζει δυναμικά ποιο εργαλείο θα καλέσει, με ποιες παραμέτρους και σε ποια σειρά, ανάλογα με το πλαίσιο και τον στόχο που έχει τεθεί. Στο σύστημα που αναπτύχθηκε, το MCP

αποτελεί τον κεντρικό κόμβο που συνδέει τους δύο agents με τα εργαλεία αναγνώρισης και εκμετάλλευσης, επιτρέποντας τη δομημένη και ελεγχόμενη εκτέλεση του pipeline.

2.5 Συστήματα Πολλαπλών Agents

Ένα σύστημα πολλαπλών agents (Multi-Agent System) είναι μια αρχιτεκτονική στην οποία περισσότεροι από ένας AI agents συνεργάζονται για την επίτευξη ενός κοινού στόχου, με κάθε έναν να αναλαμβάνει ένα συγκεκριμένο και καλά ορισμένο ρόλο. Αντί ένας μοναδικός agent να επιφορτίζεται με το σύνολο μιας σύνθετης εργασίας, η λογική κατανέμεται σε εξειδικευμένες οντότητες που επικοινωνούν και συντονίζονται μεταξύ τους.

Η βασική τεκμηρίωση αυτής της προσέγγισης είναι η διαχείριση της πολυπλοκότητας. Όταν μια εργασία απαιτεί διαφορετικές δεξιότητες, διαφορετικά εργαλεία και διαφορετικό τρόπο σκέψης σε κάθε φάση, η ανάθεσή της σε έναν μόνο agent οδηγεί συχνά σε συγχύσεις, λάθη προτεραιοτήτων και αναξιόπιστη εκτέλεση. Ο διαχωρισμός ρόλων επιτρέπει σε κάθε agent να εστιάζει αποκλειστικά στο δικό του πεδίο αρμοδιότητας, με αποτέλεσμα πιο προβλέψιμη και ελεγχόμενη συμπεριφορά.

Στο πλαίσιο της παρούσας εργασίας, η επιλογή δύο εξειδικευμένων agents έναντι ενός ενιαίου δεν είναι τυχαία. Ο Analyst Agent και ο Pentester Agent αντιπροσωπεύουν δύο θεμελιωδώς διαφορετικές λειτουργίες: η ανάλυση και η σύνθεση στρατηγικής από τη μία, και η επιχειρησιακή εκτέλεση από την άλλη. Ο διαχωρισμός αυτός αντικατοπτρίζει τη δομή που ακολουθεί και ένας έμπειρος pentester στην πράξη, όπου η σκέψη και η δράση είναι διακριτά αλλά αλληλένδετα στάδια μιας συνεκτικής διαδικασίας.

2.6 Υφιστάμενα Εργαλεία Αυτοματοποιημένου Penetration Testing

Η αυτοματοποίηση του ελέγχου διείσδυσης μέσω τεχνητής νοημοσύνης αποτελεί ένα ταχέως εξελισσόμενο πεδίο, με αρκετά εργαλεία να έχουν εμφανιστεί τα τελευταία χρόνια. Η κατανόηση των υφιστάμενων προσεγγίσεων, των δυνατοτήτων και των περιορισμών τους, είναι απαραίτητη για την τοποθέτηση της παρούσας εργασίας στο ερευνητικό τοπίο. Στην παρούσα ενότητα παρουσιάζονται δύο από τα πιο αντιπροσωπευτικά εργαλεία, το PentestGPT και το HexStrike AI, και ακολουθεί συγκριτική ανάλυση με το σύστημα που αναπτύχθηκε στο πλαίσιο αυτής της εργασίας.

2.6.1 PentestGPT

Το PentestGPT είναι ένα εργαλείο αυτοματοποιημένου ελέγχου διείσδυσης που βασίζεται σε Μεγάλα Γλωσσικά Μοντέλα, δημοσιευμένο στο συνέδριο USENIX Security 2024 [15]. Αρχικά κυκλοφόρησε το 2023 ως ανοιχτού κώδικα έργο και αποτελεί ένα από τα πρώτα εργαλεία που επιχείρησαν να αξιοποιήσουν τις δυνατότητες των LLMs στο πεδίο του penetration testing.

Η αρχιτεκτονική του PentestGPT βασίζεται σε τρία βασικά συστατικά, ένα module δημιουργίας εντολών, ένα module καθοδήγησης δοκιμών και ένα module ανάλυσης αποτελεσμάτων, τα οποία λειτουργούν μέσω ενός ενοποιημένου terminal handler. Ως υποκείμενη νοημοσύνη χρησιμοποιεί το ChatGPT και υποστηρίζει πολλαπλές κατηγορίες δοκιμών, συμπεριλαμβανομένων Web, Crypto, Reversing, Forensics, PWN και Privilege Escalation. Το εργαλείο διαθέτει επίσης session persistence, επιτρέποντας την αποθήκευση και συνέχιση δοκιμών μεταξύ συνεδριών.

Ωστόσο, το PentestGPT λειτουργεί ουσιαστικά ως σύμβουλος παρά ως αυτόνομος agent εκτέλεσης. Ο χρήστης είναι υπεύθυνος για την εκτέλεση κάθε εργαλείου, την αντιγραφή των αποτελεσμάτων στη διεπαφή και την υποβολή ερωτήσεων σχετικά με την ερμηνεία τους. Δεν διαθέτει δικό του runtime περιβάλλον, δεν εκτελεί αυτόνομα εντολές και δεν χρησιμοποιεί τυποποιημένο πρωτόκολλο επικοινωνίας με εξωτερικά εργαλεία. Επιπλέον, οι εντολές που παράγει δεν είναι πάντα

βελτιστοποιημένες για τη συγκεκριμένη περίπτωση χρήσης, ενώ η αξιοπιστία των αποτελεσμάτων του εξαρτάται άμεσα από τις δυνατότητες και τους περιορισμούς του υποκείμενου μοντέλου ChatGPT. Η βασική αξία του εργαλείου εντοπίζεται στην εκπαιδευτική του διάσταση και στην καθοδήγηση λιγότερο έμπειρων χρηστών στη μεθοδολογία του penetration testing.

2.6.2 HexStrike AI

Το HexStrike AI είναι ένα framework επιθετικής ασφάλειας ανοιχτού κώδικα, που αναπτύχθηκε από τον ερευνητή ασφαλείας Muhammad Osama και διατέθηκε στο GitHub τον Ιούλιο του 2025 [16]. Σε αντίθεση με το PentestGPT, το HexStrike AI σχεδιάστηκε εξαρχής για αυτόνομη εκτέλεση, αξιοποιώντας το Model Context Protocol (MCP) ως πρωτόκολλο επικοινωνίας μεταξύ LLM agents και εργαλείων ασφαλείας.

Η αρχιτεκτονική του HexStrike AI, στην τρέχουσα έκδοσή του (v6.0), βασίζεται σε μια πολυ-agent αρχιτεκτονική με ένα κεντρικό Intelligent Decision Engine. Αυτό το συστατικό αναλύει αυτόνομα τους στόχους, επιλέγει τα κατάλληλα εργαλεία και βελτιστοποιεί τις παραμέτρους τους. Διαθέτει πάνω από 12 εξειδικευμένους AI agents (όπως BugBountyWorkflowManager, CVEIntelligenceManager κ.ά.) και ενσωματώνει περισσότερα από 150 επαγγελματικά εργαλεία ασφαλείας, μεταξύ των οποίων τα Nmap, Metasploit, Burp Suite και John the Ripper. Το εργαλείο είναι LLM-agnostic, υποστηρίζοντας πολλαπλά μοντέλα όπως Claude, GPT και Copilot, και είναι πλέον διαθέσιμο ως πακέτο στο Kali Linux.

Η βασική καινοτομία του HexStrike AI έγκειται στην ικανότητά του να μεταφράζει γενικές εντολές φυσικής γλώσσας σε ακριβείς, αλληλουχημένες τεχνικές ενέργειες. Ο χρήστης μπορεί να δώσει μια γενική οδηγία και το σύστημα αναπτύσσει αυτόματα την κατάλληλη αλυσίδα εργαλείων. Ωστόσο, αυτή ακριβώς η ισχύ αποτέλεσε και το κυριότερο πρόβλημά του, καθώς σύντομα μετά την κυκλοφορία του, κακόβουλοι χρήστες άρχισαν να το χρησιμοποιούν σε πραγματικές επιθέσεις. Η Check Point κατέγραψε περιπτώσεις όπου το εργαλείο αξιοποιήθηκε για exploitation zero-day ευπαθειών του Citrix NetScaler λίγες μόνο ώρες μετά τη δημοσίευσή τους, αφήνοντας ελάχιστα περιθώρια αντίδρασης στις ομάδες ασφαλείας των οργανισμών. Το γεγονός ότι ένα τέτοιο εργαλείο μειώνει ουσιαστικά την τεχνική εμπειρία που απαιτείται για την εκτέλεση σύνθετων επιθέσεων, θέτει εύλογα ερωτήματα σχετικά με τα όρια και τις ευθύνες που συνοδεύουν την ανάπτυξη εργαλείων επιθετικής ασφάλειας.

2.6.3 Συγκριτική Ανάλυση και Τοποθέτηση του Παρόντος Συστήματος

Η σύγκριση των τριών συστημάτων αναδεικνύει θεμελιώδεις διαφορές τόσο στην αρχιτεκτονική φιλοσοφία όσο και στον βαθμό αυτονομίας, την ασφάλεια εκτέλεσης και την ερευνητική τεκμηρίωση.

Αρχιτεκτονική και διαχωρισμός ρόλων. Το PentestGPT ακολουθεί μονολιθική αρχιτεκτονική χωρίς εσωτερικό διαχωρισμό ρόλων σε επίπεδο agent. Το HexStrike AI υιοθετεί πολυ-agent αρχιτεκτονική με 12+ agents, αλλά χωρίς σαφή διαχωρισμό μεταξύ φάσης ανάλυσης και φάσης εκτέλεσης. Το σύστημα της παρούσας εργασίας υιοθετεί μια ελεγχόμενη dual-agent αρχιτεκτονική, όπου ο διαχωρισμός μεταξύ Analyst Agent και Pentester Agent αντικατοπτρίζει ρητά τη θεμελιώδη διάκριση μεταξύ στρατηγικής σκέψης και επιχειρησιακής δράσης. Αυτή η επιλογή δεν είναι μόνο σχεδιαστική αλλά και ερευνητική, καθώς επιτρέπει τη μεμονωμένη αξιολόγηση κάθε φάσης.

Βαθμός αυτονομίας. Τα τρία εργαλεία τοποθετούνται σε ένα φάσμα που εκτείνεται από τον παθητικό σύμβουλο έως την πλήρη αυτονομία. Το PentestGPT βρίσκεται στο ένα άκρο: δεν εκτελεί τίποτα αυτόνομα, εξαρτώμενο πλήρως από τον χρήστη για κάθε ενέργεια. Το HexStrike AI βρίσκεται στο αντίθετο άκρο, με σχεδόν πλήρη αυτονομία στην εκτέλεση. Το σύστημα της παρούσας εργασίας τοποθετείται σκόπιμα στο μέσο αυτού του φάσματος, με ελεγχόμενη αυτονομία: εκτελεί αυτόνομα τις τεχνικές ενέργειες, αλλά η τελική απόφαση για το ποιες ευπάθειες θα δοκιμαστούν παραμένει στον

χρήστη. Η προσέγγιση αυτή εξασφαλίζει ότι ο ανθρώπινος έλεγχος διατηρείται στα κρίσιμα σημεία, χωρίς να θυσιάζεται η αυτοματοποίηση στα τεχνικά βήματα.

Σχεδιασμός επίθεσης. Ένα σημαντικό σημείο διαφοροποίησης αφορά τη φάση σχεδιασμού της επίθεσης. Το PentestGPT παρέχει γενικές συμβουλές βήμα-βήμα χωρίς δομημένη ιεράρχηση. Το HexStrike AI ενσωματώνει τη λογική σχεδιασμού στο Intelligent Decision Engine, χωρίς όμως να παράγει ξεχωριστό και ορατό πλάνο επίθεσης πριν από την εκτέλεση. Το σύστημα της παρούσας εργασίας εισάγει ρητή και ελεγχόμενη φάση σχεδιασμού μέσω του Attack Plan, το οποίο ιεραρχεί τις ευπάθειες βάσει της διαθεσιμότητας exploit, διαχωρίζει τους actionable στόχους από τους θεωρητικούς κινδύνους, και παρέχει αιτιολογημένη σύσταση προτεραιότητας. Η δημιουργία αυτού του ενδιάμεσου αποτελέσματος ενισχύει τη διαφάνεια και την αξιολογησιμότητα του συστήματος.

Ασφαλιστικές δικλείδες. Η αυτοματοποίηση της επιθετικής ασφάλειας εγείρει εγγενείς κινδύνους κατάχρησης, όπως καταδεικνύεται από την περίπτωση του HexStrike AI. Το PentestGPT αποφεύγει αυτόν τον κίνδυνο εκ κατασκευής, καθώς δεν εκτελεί τίποτα αυτόνομα. Το HexStrike AI, παρότι διαθέτει retry logic και recovery handling, αποδείχθηκε στην πράξη ευάλωτο σε κακόβουλη χρήση. Το σύστημα της παρούσας εργασίας ενσωματώνει ρητό σύστημα Guardrails στον Pentester Agent, το οποίο αποτρέπει επαναλαμβανόμενες ενέργειες, θέτει ανώτατα όρια εκτέλεσης ανά CVE, και προστατεύει από ατέρμονες βρόχους, ένα γνωστό πρόβλημα στη συμπεριφορά των LLM agents.

Εύρος εφαρμογής. Ως προς το εύρος, το PentestGPT καλύπτει πολλαπλές κατηγορίες δοκιμών αλλά μόνο σε συμβουλευτικό επίπεδο. Το HexStrike AI καλύπτει ολόκληρο το φάσμα της επιθετικής ασφάλειας, συμπεριλαμβανομένων network και infrastructure testing, με πλήρη αυτονομία εκτέλεσης. Το σύστημα της παρούσας εργασίας εστιάζει αποκλειστικά στο web application penetration testing μέσω HTTP, μια συνειδητή σχεδιαστική επιλογή που διατηρεί το σύστημα εστιασμένο και αξιολογήσιμο, αφήνοντας παράλληλα χώρο για μελλοντική επέκταση.

Ο Πίνακας 2.1 συνοψίζει τα βασικά χαρακτηριστικά των τριών συστημάτων.

Χαρακτηριστικό	PentestGPT	HexStrike AI	Παρόν Σύστημα
Αρχιτεκτονική	Μονολιθική (3 components)	Multi-agent (12+ agents)	Dual-agent (Analyst + Pentester)
Πρωτόκολλο	Κανένα (copy-paste)	MCP	MCP
LLM Backend	ChatGPT	Claude, GPT, Copilot κ.ά.	DeepSeek
Αυτόνομη εκτέλεση	Όχι	Πλήρης	Ελεγχόμενη (human-in-the-loop)
Attack Plan	Όχι (γενικές συμβουλές)	Εσωτερικό (μη ορατό)	Ρητό, δομημένο, αξιολογήσιμο
Φάση αναγνώρισης	Χειροκίνητη	Αυτόματη (150+ εργαλεία)	Αυτόματη (Wappalyzer, FastCVE, ExploitDB)
Εκτέλεση exploits	Μόνο συμβουλευτική	Πλήρης (πολλά πρωτόκολλα)	HTTP-based (Nuclei + ExploitDB fallback)
Guardrails	Δεν απαιτούνται	Retry logic, recovery	Ρητό σύστημα ελέγ-

			χου ανά CVE
Πεδίο εφαρμογής	Web + CTF (συμβουλευτικά)	Full-spectrum offensive	Web application pentesting
Ακαδημαϊκή δημοσίευση	USENIX Security 2024	Καμία	Παρούσα εργασία

Πίνακας 2.1: Συγκριτικός πίνακας PentestGPT, HexStrike AI και παρόντος συστήματος

Συνοψίζοντας, το σύστημα της παρούσας εργασίας τοποθετείται σε ένα σημείο που δεν καλύπτεται επαρκώς από τα υφιστάμενα εργαλεία: προσφέρει πραγματική αυτόνομη εκτέλεση, σε αντίθεση με το PentestGPT, αλλά με ελεγχόμενο και διαφανή τρόπο, σε αντίθεση με το HexStrike AI. Η κεντρική συνεισφορά δεν είναι μόνο τεχνική αλλά και αρχιτεκτονική: ο ρητός διαχωρισμός ανάλυσης και εκτέλεσης, η ορατή φάση σχεδιασμού μέσω του Attack Plan, και η ενσωμάτωση ασφαλιστικών δικλείδων αποδεικνύουν ότι η αυτοματοποίηση του penetration testing μπορεί να επιτευχθεί χωρίς να θυσιαστεί ο ανθρώπινος έλεγχος και η συστηματική τεκμηρίωση.

3 Φάση Αναγνώρισης

Η φάση αναγνώρισης (Reconnaissance) αποτελεί το θεμέλιο ολόκληρου του pipeline. Χωρίς αξιόπιστη και πλήρη πληροφορία για τον στόχο, κανένα από τα επόμενα στάδια δεν μπορεί να λειτουργήσει αποτελεσματικά. Ο σκοπός της φάσης αυτής είναι η συλλογή δεδομένων για το τεχνολογικό υπόβαθρο του στόχου, ο εντοπισμός πιθανών ευπαθειών και η δημιουργία ενός δομημένου προφίλ που θα αποτελέσει την είσοδο για τον Analyst Agent. Η φάση αυτή υλοποιείται μέσω τριών εργαλείων που λειτουργούν συμπληρωματικά και ακολουθούν μια συγκεκριμένη σειρά εκτέλεσης, τα οποία έχουν αναπτυχθεί και εκτελούνται σε περιβάλλον Docker, εξασφαλίζοντας αναπαραγωγιμότητα και απομόνωση από το υπόλοιπο σύστημα.

3.1 Wappalyzer

Το Wappalyzer είναι ένα εργαλείο αναγνώρισης τεχνολογιών (technology fingerprinting) που αναλύει έναν ιστότοπο και εντοπίζει το τεχνολογικό του υπόβαθρο. Εξετάζει στοιχεία όπως τις επικεφαλίδες HTTP, τον κώδικα HTML, τα αρχεία JavaScript και τα cookies, αντιστοιχίζοντάς τα με γνωστές υπογραφές τεχνολογιών. Το αποτέλεσμα είναι μια λίστα με τις τεχνολογίες που χρησιμοποιεί ο στόχος, frameworks, CMS, web servers, βιβλιοθήκες, συστήματα ανάλυσης και άλλα.

Κρίσιμο στοιχείο της εξόδου του Wappalyzer είναι ότι κάθε εντοπισμένη τεχνολογία συνοδεύεται από έναν αναγνωριστικό κωδικό CPE (Common Platform Enumeration). Το CPE είναι ένα τυποποιημένο σχήμα ονοματολογίας που περιγράφει με ακρίβεια ένα προϊόν λογισμικού, συμπεριλαμβανομένης της έκδοσής του, και αποτελεί το κοινό "γλωσσάρι" που επιτρέπει στα εργαλεία ασφαλείας να επικοινωνούν μεταξύ τους. Στο pipeline που αναπτύχθηκε, ο κωδικός CPE αποτελεί τον συνδετικό κρίκο μεταξύ του Wappalyzer και του FastCVE. Η έξοδος του ενός γίνεται η είσοδος του άλλου.

Ωστόσο, όπως αναφέρθηκε στα ερευνητικά ερωτήματα, η κάλυψη του Wappalyzer δεν είναι πάντα πλήρης. Κάποιες τεχνολογίες ενδέχεται να μην ανιχνευτούν, είτε λόγω απουσίας τους από τη βάση υπογραφών είτε λόγω τεχνικών αποκρύψεων από τον ιστότοπο. Αυτός ο περιορισμός λαμβάνεται υπόψη στον σχεδιασμό του συστήματος και αναλύεται περαιτέρω στο κεφάλαιο της αξιολόγησης.

3.2 FastCVE

Με βάση τους κωδικούς CPE που παρήγαγε το Wappalyzer, το FastCVE αναλαμβάνει την αναζήτηση γνωστών ευπαθειών για κάθε τεχνολογία που εντοπίστηκε. Για κάθε CPE που λαμβάνει, εκτελεί αίτημα στη βάση δεδομένων ευπαθειών και επιστρέφει τα αντίστοιχα CVEs, εμπλουτισμένα με πληροφορίες για τη σοβαρότητά τους βάσει του συστήματος CVSS και το είδος της αδυναμίας που περιγράφουν.

Η σημασία του FastCVE στο pipeline έγκειται στο ότι μετατρέπει τη γενική πληροφορία για τις τεχνολογίες του στόχου σε συγκεκριμένες και δομημένες πληροφορίες για πιθανές αδυναμίες. Αποτελεί ουσιαστικά τη γέφυρα μεταξύ της αναγνώρισης τεχνολογιών και της αναζήτησης εκμεταλλεύσιμων ευπαθειών, και η έξοδός του τροφοδοτεί άμεσα το επόμενο βήμα του pipeline.

3.3 ExploitDB

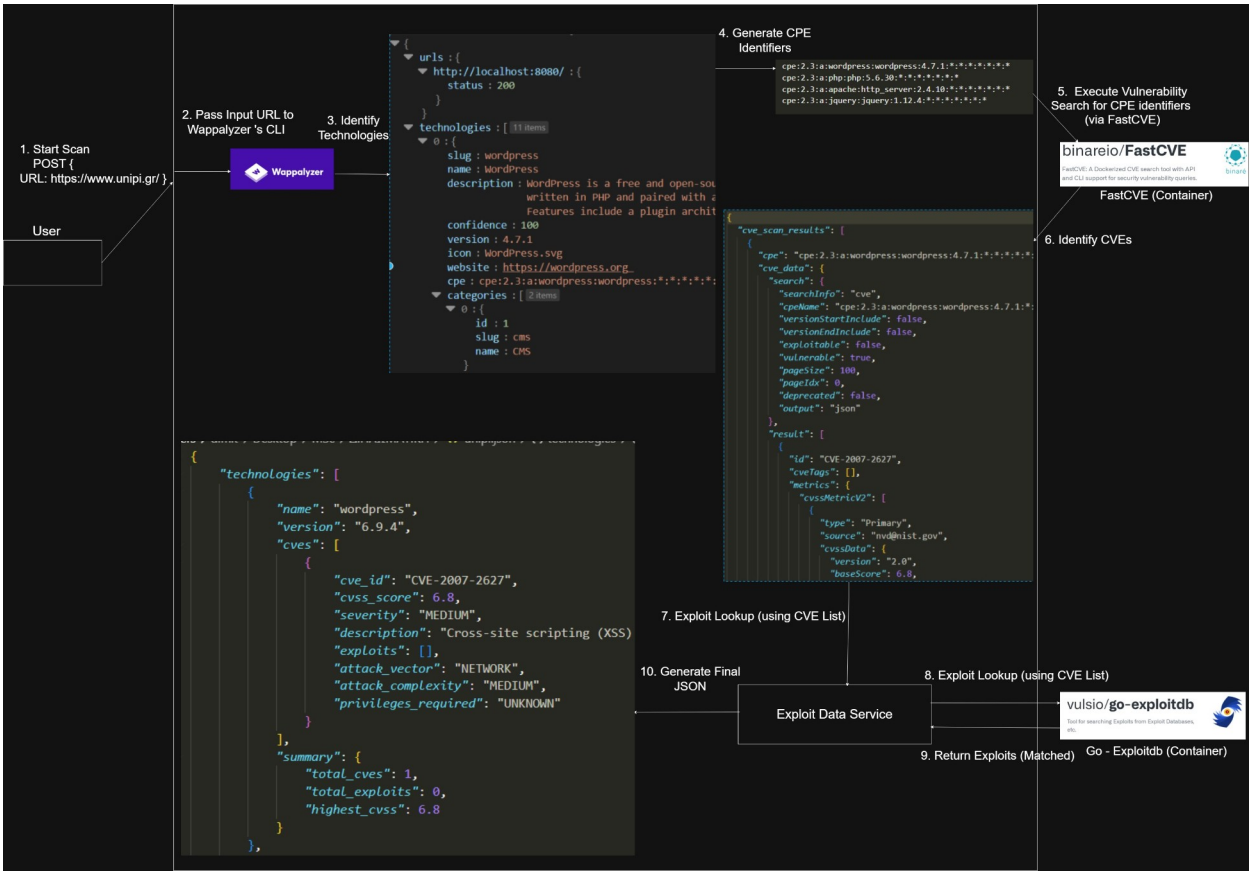
Για κάθε CVE που εντόπισε το FastCVE εκτελείται ξεχωριστό αίτημα στο ExploitDB με σκοπό να διαπιστωθεί αν υπάρχει διαθέσιμος κώδικας εκμετάλλευσης. Τα αιτήματα αυτά εκτελούνται ταυτόχρονα, εξασφαλίζοντας αποδοτική διαχείριση του χρόνου ανεξάρτητα από τον όγκο των ευπαθειών που εντοπίστηκαν. Το αποτέλεσμα της διαδικασίας εμπλουτίζει το προφίλ ευπαθειών με πρακτική πληροφορία για το τι είναι άμεσα εκμεταλλεύσιμο, ολοκληρώνοντας έτσι την αλυσίδα από την τεχνολογία στο CVE και από το CVE στο exploit.

Όπως αναλύθηκε στο Κεφάλαιο 2, η ανομοιογένεια των αρχείων του ExploitDB αποτελεί έναν πραγματικό περιορισμό, καθώς η αξιοπιστία και η λειτουργικότητά τους δεν είναι πάντα εγγυημένες. Για αυτόν το λόγο τα αποτελέσματά του δεν χρησιμοποιούνται άκριτα. Ο Pentester Agent, όπως θα αναλυθεί στο Κεφάλαιο 5, τα αντιμετωπίζει ως δευτερεύουσα πηγή, προτιμώντας όπου είναι δυνατόν τα δομημένα templates του Nuclei, τα οποία προσφέρουν μεγαλύτερη αξιοπιστία και ευκολία εκτέλεσης.

3.4 Έξοδος Φάσης Αναγνώρισης

Το αποτέλεσμα της φάσης αναγνώρισης είναι ένα δομημένο αρχείο JSON που οργανώνει την πληροφορία σε τρία επίπεδα. Στο πρώτο επίπεδο βρίσκονται οι εντοπισμένες τεχνολογίες με την έκδοσή τους. Στο δεύτερο επίπεδο, για κάθε τεχνολογία, παρατίθενται τα αντίστοιχα CVEs με πληροφορίες για τη σοβαρότητα, το διάνυσμα επίθεσης και την πολυπλοκότητα εκμετάλλευσης. Στο τρίτο επίπεδο, για όσα CVEs διαθέτουν αντίστοιχο exploit, περιλαμβάνονται στοιχεία όπως ο τύπος, η πλατφόρμα και ο σύνδεσμος του κώδικα εκμετάλλευσης. Αυτή η δομή επιτρέπει στον Analyst Agent να επεξεργαστεί την πληροφορία με συστηματικό τρόπο και να ιεραρχήσει τις ευπάθειες βάσει της σοβαρότητας και της διαθεσιμότητας exploit.

Η φάση αναγνώρισης, όπως αποτυπώνεται στην Εικόνα 3.1, υλοποιείται μέσω μιας σαφούς αλυσίδας δέκα βημάτων. Ο χρήστης εκκινεί τη διαδικασία παρέχοντας το URL στόχου, το οποίο διαβιβάζεται στο Wappalyzer για ανάλυση. Το Wappalyzer εντοπίζει τις τεχνολογίες του στόχου και παράγει τους αντίστοιχους CPE κωδικούς, οι οποίοι διαβιβάζονται στο FastCVE. Το FastCVE εκτελεί αναζήτηση ευπαθειών για κάθε CPE και επιστρέφει τη λίστα των αντίστοιχων CVEs. Η λίστα αυτή διαβιβάζεται στο Exploit Data Service, το οποίο επικοινωνεί με το go-exploitdb container για να εντοπίσει διαθέσιμα exploits για κάθε CVE. Τα αποτελέσματα συγκεντρώνονται και παράγεται το τελικό δομημένο JSON προφίλ ευπαθειών που αποτελεί την είσοδο του Analyst Agent.



Εικόνα 3.1 Ροή φάσης αναγνώρισης: από το URL στόχου στο δομημένο προφίλ ευπαθειών

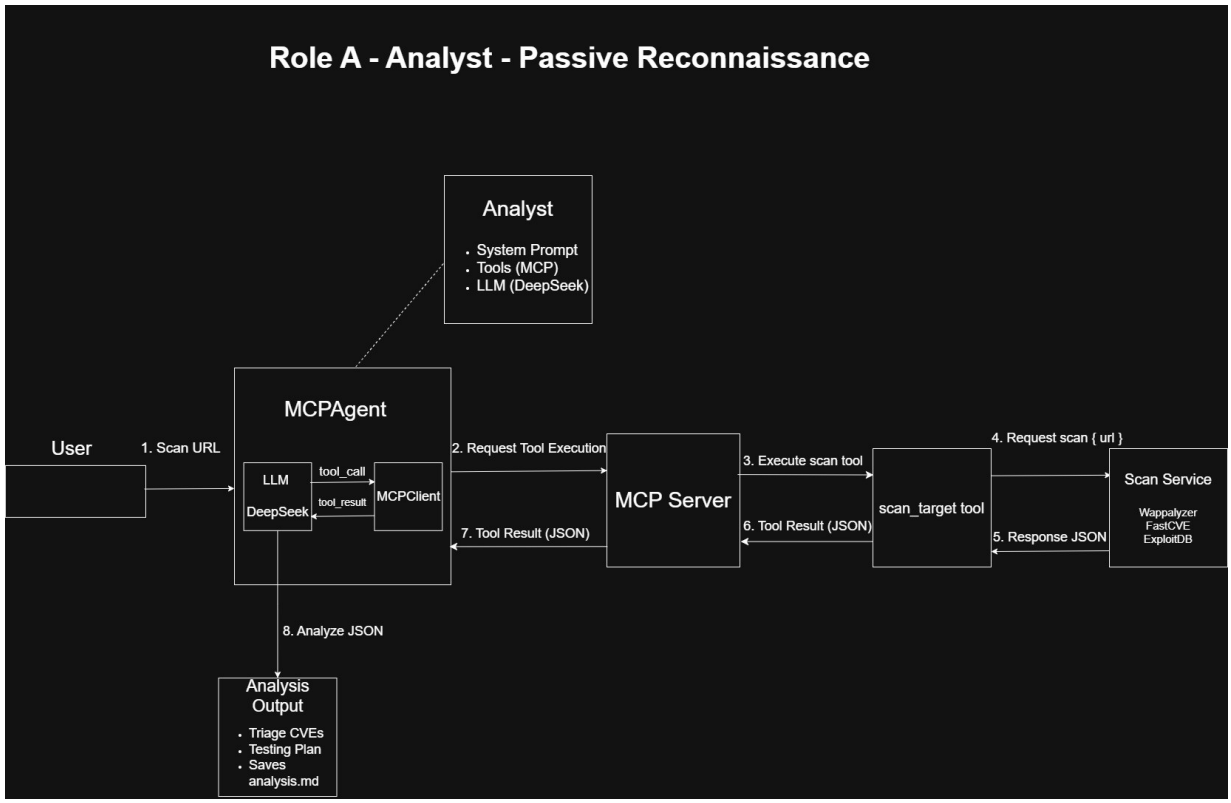
4 Agent Ανάλυσης και Στρατηγικού Σχεδιασμού

Ο Agent Ανάλυσης (Analyst Agent) αποτελεί το πρώτο από τα δύο εξειδικευμένα συστατικά του multi-agent pipeline και αναλαμβάνει τον ρόλο της ανάλυσης και της στρατηγικής σκέψης. Λαμβάνει ως είσοδο το δομημένο προφίλ ευπαθειών που παρήγαγε η φάση αναγνώρισης και έχει ως αποστολή να το επεξεργαστεί και να παράγει ένα δομημένο Attack Plan που θα καθοδηγήσει τον Pentester Agent στο επόμενο στάδιο. Ο διαχωρισμός αυτός αντανακλά μια θεμελιώδη αρχή της επιθετικής ασφάλειας: η ποιότητα της επίθεσης εξαρτάται άμεσα από την ποιότητα της ανάλυσης που έχει προηγηθεί.

4.1 Αρχιτεκτονική και Ενσωμάτωση με το MCP

Ο Analyst Agent υλοποιείται μέσω του Model Context Protocol, αξιοποιώντας το μοντέλο DeepSeek ως υποκείμενη νοημοσύνη. Λειτουργεί σε αποκλειστικά παθητική λειτουργία, το μοναδικό εργαλείο που έχει στη διάθεσή του είναι η σάρωση του στόχου, ενώ του απαγορεύεται ρητά η εκτέλεση exploit κώδικα ή η αποστολή επιθετικών αιτημάτων. Αυτός ο σχεδιαστικός περιορισμός δεν είναι τυχαίος, διασφαλίζει τον σαφή διαχωρισμό ρόλων μεταξύ των δύο agents και αποτρέπει την ανάμειξη φάσεων που θα μπορούσε να οδηγήσει σε αναξιόπιστα αποτελέσματα.

Στο παρακάτω διάγραμμα (Εικόνα 4.1) αποτυπώνεται αναλυτικά η ροή εκτέλεσης του Analyst Agent. Η διαδικασία ξεκινά με τον χρήστη που παρέχει το URL στόχου (βήμα 1). Ο MCPAgent, ο οποίος αποτελείται από το μοντέλο DeepSeek και τον MCPClient, λαμβάνει την εντολή και αιτείται την εκτέλεση του κατάλληλου εργαλείου (βήμα 2). Ο MCP Server δέχεται το αίτημα και εκκινεί το scan_target tool (βήμα 3), το οποίο αποστέλλει αίτημα σάρωσης στο Scan Service (βήμα 4). Το Scan Service, που αποτελείται από την αλυσίδα Wappalyzer, FastCVE και ExploitDB, εκτελεί την αναγνώριση και επιστρέφει το αποτέλεσμα σε μορφή JSON (βήμα 5). Το JSON διαβιβάζεται πίσω μέσω του scan_target tool στον MCP Server (βήμα 6), ο οποίος το προωθεί στον MCPAgent (βήμα 7). Το μοντέλο DeepSeek αναλύει το JSON (βήμα 8) και παράγει το Analysis Output, το οποίο περιλαμβάνει την κατηγοριοποίηση των CVEs, το Testing Plan με τους actionable στόχους και την αποθήκευση των αποτελεσμάτων σε αρχείο analysis.md.



Εικόνα 4.1 Αρχιτεκτονική και ροή εκτέλεσης Analyst Agent

4.2 Σχεδιασμός του Prompt

Ένα από τα κρίσιμα ζητήματα στην ανάπτυξη του Analyst Agent είναι ο σχεδιασμός του prompt που καθορίζει τη συμπεριφορά του. Το prompt δεν είναι απλώς μια οδηγία, είναι ο μηχανισμός μέσω του οποίου ορίζεται ο ρόλος του agent, τα εργαλεία που έχει στη διάθεσή του, η λογική επεξεργασίας των αποτελεσμάτων και η μορφή της εξόδου του. Η πρόκληση έγκειται στο ότι ένα LLM δεν ακολουθεί εγγυημένα τις οδηγίες που του δίνονται, η διατύπωση, η σειρά και η ακρίβεια των οδηγιών επηρεάζουν άμεσα τη συμπεριφορά του, και μικρές αποκλίσεις μπορούν να οδηγήσουν σε μη αναμενόμενη ροή εκτέλεσης.

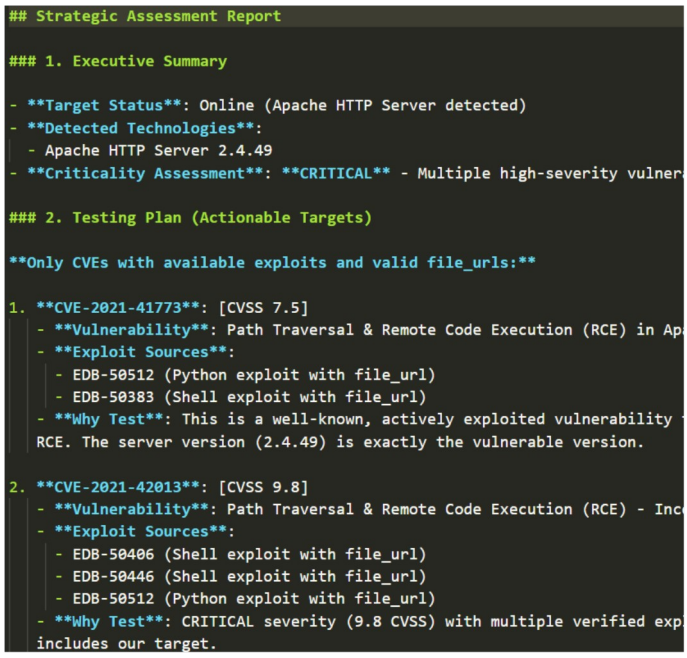
Στην προσέγγιση που επιλέχθηκε ιδιαίτερη έμφαση δόθηκε στην αποφυγή μη παραγωγικών βρόχων εκτέλεσης, ένα γνωστό πρόβλημα των LLM Agents, που τείνουν να επαναλαμβάνουν αποτυχημένες ενέργειες με την ελπίδα διαφορετικού αποτελέσματος. Με ρητές οδηγίες τερματισμού όταν ο στόχος είναι μη προσβάσιμος ή δεν ανιχνευτεί καμία τεχνολογία, το σύστημα αποφεύγει αυτή την παθολογία διατηρώντας παράλληλα αξιόπιστη και προβλέψιμη συμπεριφορά.

4.3 Παραγωγή του Attack Plan

Το τελικό αποτέλεσμα του Analyst Agent είναι το Attack Plan, ένα δομημένο έγγραφο που διαχωρίζει και ιεραρχεί τις ευπάθειες που εντοπίστηκαν. Η λογική ιεράρχησης είναι σκόπιμα συντηρητική: μια ευπάθεια εντάσσεται στο ενεργό πλάνο δοκιμών μόνο εφόσον συνοδεύεται από διαθέσιμο κώδικα εκμετάλλευσης. Ευπάθειες με υψηλό CVSS score αλλά χωρίς διαθέσιμο exploit κατατάσσονται σε ξεχωριστή κατηγορία παρακολούθησης και δεν οδηγούνται στον Pentester Agent. Η προσέγγιση αυτή εξασφαλίζει ότι το Attack Plan περιέχει μόνο στόχους που μπορούν να δοκιμαστούν (actionable targets) και όχι θεωρητικές ευπάθειες που δεν μπορούν να επαληθευτούν στην πράξη.

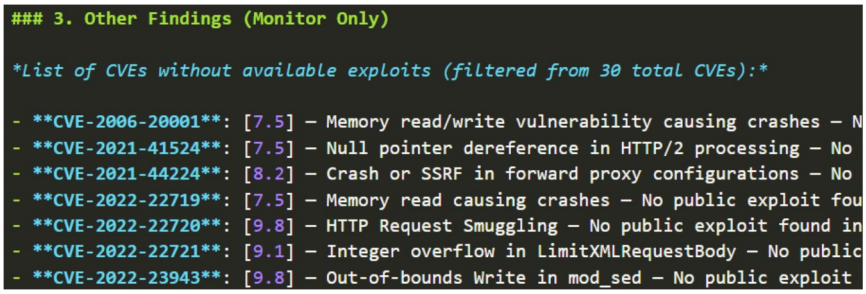
Το Attack Plan παράγεται δυναμικά βάσει των ευρημάτων κάθε εκτέλεσης και προσαρμόζεται στο συγκεκριμένο προφίλ του στόχου. Περιλαμβάνει επίσης σύσταση προτεραιότητας, δηλαδή ποια ευπάθεια συνίσταται να δοκιμαστεί πρώτη και για ποιον λόγο, παρέχοντας στον χρήστη και στον Pentester Agent ένα σαφές σημείο εκκίνησης.

Στο παρακάτω παράδειγμα (Εικόνες 4.3.1 — 4.3.3) αποτυπώνεται το Attack Plan, το οποίο οργανώνεται σε τέσσερις ενότητες. Η πρώτη είναι το Executive Summary και το Testing Plan (Εικόνα 4.3.1), όπου καταγράφεται η κατάσταση του στόχου, οι εντοπισμένες τεχνολογίες και η συνολική εκτίμηση κινδύνου, ακολουθούμενη από τα CVEs που διαθέτουν έγκυρο exploit, με την περιγραφή της ευπάθειας, τις πηγές exploit και την αιτιολόγηση της επιλογής για καθένα από αυτά. Η δεύτερη ενότητα (Εικόνα 4.3.2) καταγράφει τα CVEs που εντοπίστηκαν αλλά δεν διαθέτουν exploit, τα οποία παρακολουθούνται χωρίς να οδηγούνται στον Pentester Agent. Τέλος, η τρίτη ενότητα (Εικόνα 4.3.3) παρέχει τη σύσταση προτεραιότητας με σαφή αιτιολόγηση, κατευθύνοντας τον χρήστη στο πιο κρίσιμο σημείο εκκίνησης.



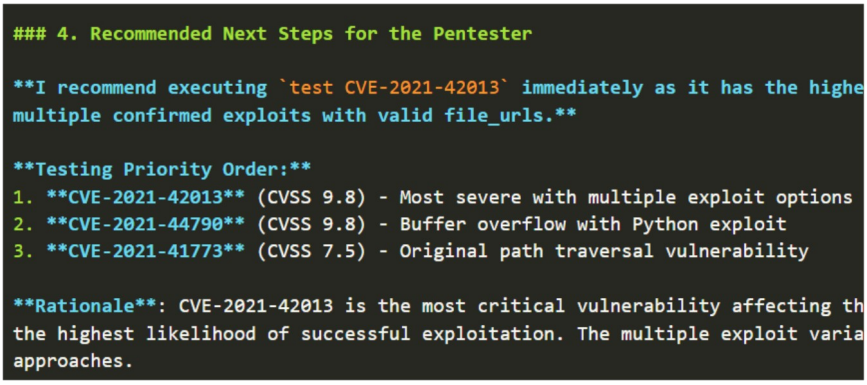
```
## Strategic Assessment Report

### 1. Executive Summary

- **Target Status**: Online (Apache HTTP Server detected)
- **Detected Technologies**:
  - Apache HTTP Server 2.4.49
- **Criticality Assessment**: **CRITICAL** - Multiple high-severity vulner

### 2. Testing Plan (Actionable Targets)

**Only CVEs with available exploits and valid file_urls:**

1. **CVE-2021-41773**: [CVSS 7.5]
   - **Vulnerability**: Path Traversal & Remote Code Execution (RCE) in Ap
   - **Exploit Sources**:
     - EDB-50512 (Python exploit with file_url)
     - EDB-50383 (Shell exploit with file_url)
   - **Why Test**: This is a well-known, actively exploited vulnerability
   RCE. The server version (2.4.49) is exactly the vulnerable version.

2. **CVE-2021-42013**: [CVSS 9.8]
   - **Vulnerability**: Path Traversal & Remote Code Execution (RCE) - Inc
   - **Exploit Sources**:
     - EDB-50406 (Shell exploit with file_url)
     - EDB-50446 (Shell exploit with file_url)
     - EDB-50512 (Python exploit with file_url)
   - **Why Test**: CRITICAL severity (9.8 CVSS) with multiple verified exp
   includes our target.
```

Εικόνα 4.3.1 Σύνοψη και Σχέδιο Δοκιμών Επίθεσης



```
### 3. Other Findings (Monitor Only)

*List of CVEs without available exploits (filtered from 30 total CVEs):*

- **CVE-2006-20001**: [7.5] — Memory read/write vulnerability causing crashes — N
- **CVE-2021-41524**: [7.5] — Null pointer dereference in HTTP/2 processing — No
- **CVE-2021-44224**: [8.2] — Crash or SSRF in forward proxy configurations — No
- **CVE-2022-22719**: [7.5] — Memory read causing crashes — No public exploit fou
- **CVE-2022-22720**: [9.8] — HTTP Request Smuggling — No public exploit found in
- **CVE-2022-22721**: [9.1] — Integer overflow in LimitXMLRequestBody — No public
- **CVE-2022-23943**: [9.8] — Out-of-bounds Write in mod_sed — No public exploit
```

Εικόνα 4.3.2 Ευπάθειες υπό παρακολούθηση χωρίς διαθέσιμο exploit



```
### 4. Recommended Next Steps for the Pentester

**I recommend executing `test CVE-2021-42013` immediately as it has the highe
multiple confirmed exploits with valid file_urls.**

**Testing Priority Order:**
1. **CVE-2021-42013** (CVSS 9.8) - Most severe with multiple exploit options
2. **CVE-2021-44790** (CVSS 9.8) - Buffer overflow with Python exploit
3. **CVE-2021-41773** (CVSS 7.5) - Original path traversal vulnerability

**Rationale**: CVE-2021-42013 is the most critical vulnerability affecting th
the highest likelihood of successful exploitation. The multiple exploit varia
approaches.
```

Εικόνα 4.3.3 Σύσταση προτεραιότητας και επόμενα βήματα για τον Pentester Agent

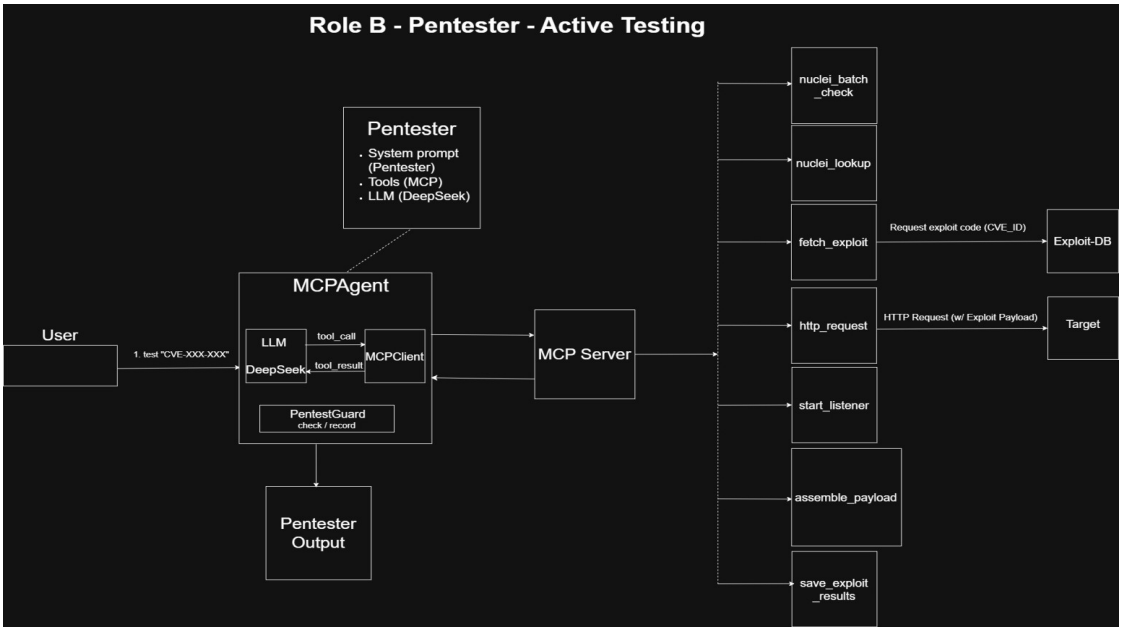
5 Agent Εκτέλεσης και Επιβεβαίωσης Επιθέσεων

Ο Pentester Agent αποτελεί το δεύτερο και τελευταίο συστατικό του multi-agent pipeline και αναλαμβάνει τον ρόλο της επιχειρησιακής δράσης. Ο χρήστης, αξιοποιώντας το Attack Plan που παρήγαγε ο Analyst Agent, επιλέγει ποιες ευπάθειες θα δοκιμαστούν. Στη βάση αυτής της επιλογής, ο Pentester Agent αναλαμβάνει την εκτέλεση, δοκιμάζει τις ευπάθειες, εκτελεί τα κατάλληλα εργαλεία και επιβεβαιώνει αν αυτές είναι πραγματικά εκμεταλλεύσιμες στο συγκεκριμένο περιβάλλον. Η ποιότητα του Attack Plan επηρεάζει άμεσα την αποτελεσματικότητα της εκτέλεσης όπου ένα καλά δομημένο και ιεραρχημένο πλάνο επιτρέπει στον agent να δρα με ακρίβεια και στοχευμένα.

5.1 Αρχιτεκτονική και Ενσωμάτωση με το MCP

Ο Pentester Agent υλοποιείται μέσω του Model Context Protocol αξιοποιώντας το μοντέλο DeepSeek, όπως και ο Analyst Agent. Το σύνολο εργαλείων που έχει στη διάθεσή του είναι θεμελιωδώς διαφορετικό, αντί για εργαλεία συλλογής πληροφορίας, διαθέτει εργαλεία εκτέλεσης επιθέσεων και επαλήθευσης ευπαθειών. Μια εξίσου σημαντική συνιστώσα της αρχιτεκτονικής είναι το σύστημα Guardrails, το οποίο παρεμβάλλεται σε κάθε κλήση εργαλείου και διασφαλίζει ότι ο agent ακολουθεί τη σωστή ροή εκτέλεσης. Το σύστημα αυτό αναλύεται διεξοδικά στην ενότητα 5.5.

Όπως αποτυπώνεται στην Εικόνα 5.1, ο χρήστης εκκινεί τη διαδικασία επιλέγοντας ένα CVE προς δοκιμή. Ο MCPAgent, αποτελούμενος από το μοντέλο DeepSeek και τον MCPClient, λαμβάνει την εντολή και αρχίζει να καλεί διαδοχικά τα εργαλεία που έχει στη διάθεσή του μέσω του MCP Server. Ανάλογα με τη φύση του CVE και τα αποτελέσματα κάθε βήματος, ο agent ακολουθεί είτε το πρωτεύον μονοπάτι μέσω Nuclei templates είτε το εναλλακτικό μέσω ExploitDB — καλώντας το fetch_exploit για ανάκτηση κώδικα και στη συνέχεια το assemble_payload για κατασκευή του τελικού payload. Το http_request αναλαμβάνει την αποστολή του payload στον στόχο και επιστρέφει την απάντηση για αξιολόγηση. Μετά την ολοκλήρωση όλων των CVEs, το save_exploit_results αποθηκεύει τα αποτελέσματα, ενώ το Pentester Output παράγεται απευθείας από το μοντέλο και παρουσιάζεται στον χρήστη.



Εικόνα 5.1 Αρχιτεκτονική και ροή εκτέλεσης Agent Εκτέλεσης και Επιβεβαίωσης Επιθέσεων

5.2 Nuclei

Το Nuclei αποτελεί το πρωτεύον εργαλείο εκτέλεσης του Pentester Agent. Πριν ξεκινήσει η επεξεργασία οποιουδήποτε CVE, ο agent εκτελεί μια μαζική αναζήτηση μέσω του nuclei_batch_check για να διαπιστώσει ποια από τα CVEs του Attack Plan διαθέτουν έτοιμο Nuclei template. Με βάση αυτή την πληροφορία, τα CVEs κατατάσσονται σε δύο ομάδες: αυτά για τα οποία υπάρχει template και επεξεργάζονται πρώτα, και αυτά που δεν διαθέτουν template και οδηγούνται στο εναλλακτικό μονοπάτι εκτέλεσης.

Για κάθε CVE της πρώτης ομάδας, ο agent ανακτά το αντίστοιχο template μέσω του nuclei_lookup, το οποίο περιέχει δομημένα payloads έτοιμα προς εκτέλεση, καθώς και matchers, κριτήρια που ορίζουν αν η απάντηση του στόχου επιβεβαιώνει την ευπάθεια. Τα payloads αποστέλλονται στον στόχο και η αξιολόγηση του αποτελέσματος γίνεται αυτόματα. Εάν κανένα payload δεν επιτύχει, ο agent μεταβαίνει στο εναλλακτικό μονοπάτι.

5.3 Εκτέλεση HTTP Αιτημάτων

Το εργαλείο http_request αποτελεί τον κεντρικό μηχανισμό εκτέλεσης όπου όλα τα payloads, ανεξάρτητα από την πηγή τους, αποστέλλονται μέσω αυτού. Δέχεται ως παραμέτρους τη μέθοδο, το URL, τις επικεφαλίδες, το σώμα του αιτήματος και τα matchers, και επιστρέφει την πλήρη απάντηση του στόχου. Κρίσιμο χαρακτηριστικό του είναι ότι αξιολογεί αυτόματα τα matchers που του παρέχονται και επιστρέφει σαφή ένδειξη αν η ευπάθεια επιβεβαιώθηκε. Επιπλέον, καταγράφει κάθε αίτημα που εκτελείται, παρέχοντας πλήρες ιστορικό για τους σκοπούς της αξιολόγησης. Ο Pentester Agent περιορίζεται σε επιθέσεις που υλοποιούνται μέσω HTTP, καλύπτοντας ένα ευρύ φάσμα κατηγοριών όπως SQL injection, XSS, SSRF, LFI, path traversal και RCE μέσω web.

5.4 Εναλλακτικό Μονοπάτι και Αξιοποίηση Exploits

Για τα CVEs που δεν διαθέτουν Nuclei template, ο agent ακολουθεί ένα εναλλακτικό μονοπάτι βασισμένο στα αρχεία του ExploitDB μέσω του εργαλείου fetch_exploit. Ανακτά τον κώδικα εκμετάλλευσης, αναλύει τη δομή του και προσαρμόζει το payload ώστε να στοχεύει το συγκεκριμένο περιβάλλον. Όταν ο κώδικας απαιτεί την έγχυση εντολής σε συγκεκριμένο σημείο του payload, το εργαλείο assemble_payload αναλαμβάνει την κατασκευή του τελικού αιτήματος, αντικαθιστώντας τον κατάλληλο placeholder με την εντολή-στόχο. Η διαδικασία αυτή είναι πιο απαιτητική και λιγότερο προβλέψιμη σε σχέση με τα Nuclei templates, γεγονός που αντικατοπτρίζει την ανομοιογένεια των αρχείων του ExploitDB που αναλύθηκε στο Κεφάλαιο 2. Ειδικές κατηγορίες ευπαθειών, όπως οι ευπάθειες απομακρυσμένης εκτέλεσης κώδικα (Remote Code Execution — RCE), αντιμετωπίζονται με εξειδικευμένη ροή εκτέλεσης που διαφέρει ως προς τον τρόπο κατασκευής του payload και επιβεβαίωσης του αποτελέσματος.

5.5 Σύστημα Guardrails

Ένα από τα πιο κρίσιμα στοιχεία του Pentester Agent είναι το ενσωματωμένο σύστημα Guardrails, το οποίο λειτουργεί ως στρώμα ελέγχου σε κάθε κλήση εργαλείου. Ο ρόλος του είναι διπλός: αφενός διασφαλίζει ότι ο agent ακολουθεί τη σωστή ροή εκτέλεσης, αφετέρου αποτρέπει περιττές ή επαναλαμβανόμενες ενέργειες που θα μπορούσαν να οδηγήσουν σε αναξιόπιστα αποτελέσματα.

Συγκεκριμένα, το σύστημα αποτρέπει την εκτέλεση περαιτέρω ενεργειών για CVEs που έχουν ήδη επιβεβαιωθεί ή απορριφθεί, θέτει όριο στον αριθμό των HTTP requests ανά CVE και εξασφαλίζει ότι η αποθήκευση των αποτελεσμάτων γίνεται μία μόνο φορά στο τέλος της εκτέλεσης. Επιπλέον, ορίζει

ένα συνολικό ανώτατο όριο βημάτων εκτέλεσης, προστατεύοντας το σύστημα από ατέρμονες βρόχους, ένα γνωστό πρόβλημα στη συμπεριφορά των LLM agents.

5.6 Αποθήκευση Αποτελεσμάτων

Μετά την ολοκλήρωση της επεξεργασίας όλων των CVEs, ο agent αποθηκεύει τα αποτελέσματα σε δομημένη μορφή μέσω του εργαλείου save_exploit_results. Κάθε CVE καταγράφεται με το αποτέλεσμά του, επιβεβαιωμένο, πιθανό, μη ευάλωτο ή παραλειφθέν, μαζί με τις λεπτομέρειες των αιτημάτων που εκτελέστηκαν και συνοπτικά στατιστικά για το σύνολο της εκτέλεσης. Τα αποτελέσματα αυτά αποτελούν τη βάση για την αξιολόγηση του συστήματος που παρουσιάζεται στο Κεφάλαιο 7.

6 Ολοκλήρωση Συστήματος

Τα προηγούμενα κεφάλαια παρουσίασαν τα επιμέρους συστατικά του συστήματος που αναπτύχθηκε στο πλαίσιο της παρούσας εργασίας, όπου περιλάμβαναν τη φάση αναγνώρισης, τον Analyst Agent και τον Pentester Agent, αναλύοντας τον ρόλο και τη λειτουργία του καθενός ξεχωριστά. Το παρόν κεφάλαιο εξετάζει το σύστημα ως σύνολο, εστιάζοντας στον τρόπο με τον οποίο τα επιμέρους συστατικά συνδέονται, επικοινωνούν και συνεργάζονται για την επίτευξη του κοινού στόχου.

6.1 Ροή Δεδομένων και Επικοινωνία μεταξύ Συνιστωσών

Η ροή δεδομένων στο σύστημα ακολουθεί μια γραμμική αλλά δομημένη πορεία, όπου η έξοδος κάθε σταδίου αποτελεί την είσοδο του επόμενου. Η διαδικασία ξεκινά με την παροχή του URL στόχου από τον χρήστη. Η πληροφορία αυτή τροφοδοτεί τη φάση αναγνώρισης, η οποία παράγει το δομημένο προφίλ ευπαθειών. Το προφίλ αυτό παραδίδεται στον Analyst Agent, ο οποίος το επεξεργάζεται και παράγει το Attack Plan. Ο χρήστης στη συνέχεια επιλέγει ποιες ευπάθειες θα δοκιμαστούν και ο Pentester Agent αναλαμβάνει την εκτέλεση, αποθηκεύοντας τα αποτελέσματα στο τέλος της διαδικασίας.

Κρίσιμο χαρακτηριστικό αυτής της αρχιτεκτονικής είναι ότι κάθε στάδιο είναι αυστηρά απομονωμένο από τα υπόλοιπα. Ο Agent Ανάλυσης δεν γνωρίζει και δεν επηρεάζει τον τρόπο εκτέλεσης του Pentester Agent, και αντίστροφα. Η επικοινωνία τους περιορίζεται αποκλειστικά στην ανταλλαγή τυποποιημένων δεδομένων του Attack Plan, αποκλείοντας κάθε άμεση αλληλεπίδραση. Αυτή η προσέγγιση διασφαλίζει τη μέγιστη προβλεψιμότητα, την ανεξαρτησία των συνιστωσών και την ευκολία στη συντήρηση ή αναβάθμιση του συστήματος.

6.2 Ο Ρόλος του Χρήστη στο Pipeline

Παρότι το σύστημα είναι σε μεγάλο βαθμό αυτοματοποιημένο, ο χρήστης διαδραματίζει έναν συγκεκριμένο και σκόπιμο ρόλο στη ροή εκτέλεσης. Μετά την παραγωγή του Attack Plan, ο χρήστης αποφασίζει ποιες ευπάθειες θα δοκιμαστούν, δεν είναι παθητικός παρατηρητής αλλά ενεργός συμμέτοχος που ασκεί έλεγχο στο τι εκτελείται. Αυτή η επιλογή δεν είναι τυχαία στον σχεδιασμό, εξασφαλίζει ότι η τελική απόφαση για την εκτέλεση επιθετικών ενεργειών παραμένει στα χέρια του ανθρώπου και όχι του συστήματος.

6.3 Σχεδιαστικές Επιλογές και Trade-offs

Κατά τον σχεδιασμό του συστήματος έγιναν ορισμένες επιλογές που αξίζει να αναλυθούν ως προς τα πλεονεκτήματα και τους συμβιβασμούς που συνεπάγονται.

Η πρώτη αφορά τη φύση της αναγνώρισης. Το σύστημα βασίζεται αποκλειστικά σε παθητική αναγνώριση, το Wappalyzer αναλύει το στόχο χωρίς να εκτελεί επιθετικές σαρώσεις ή να αλληλεπιδρά με αυτόν με τρόπο που θα μπορούσε να προκαλέσει ανίχνευση. Αυτή η επιλογή διασφαλίζει ότι η φάση αναγνώρισης παραμένει αθόρυβη, με αντάλλαγμα όμως πιθανή ατέλεια στην κάλυψη των τεχνολογιών που ανιχνεύονται, όπως αναλύθηκε στο Κεφάλαιο 3.
Η δεύτερη επιλογή αφορά τον περιορισμό των εργαλείων που έχει στη διάθεσή του ο Analyst Agent. Σκόπιμα του αποδίδεται μόνο το εργαλείο σάρωσης, χωρίς πρόσβαση σε εργαλεία εκτέλεσης. Αυτό διασφαλίζει τον σαφή διαχωρισμό ρόλων αλλά σημαίνει ότι το Attack Plan βασίζεται αποκλειστικά στη στατική ανάλυση και όχι σε δυναμική επαλήθευση.

Η τρίτη επιλογή αφορά τον διαχωρισμό σε δύο agents. Η προσέγγιση αυτή προσφέρει σαφήνεια ρόλων και ευκολότερη διαχείριση πολυπλοκότητας, με αντάλλαγμα την ανάγκη για προσεκτικό σχεδιασμό των αντίστοιχων prompts και σαφή ορισμό της διεπαφής μεταξύ τους.

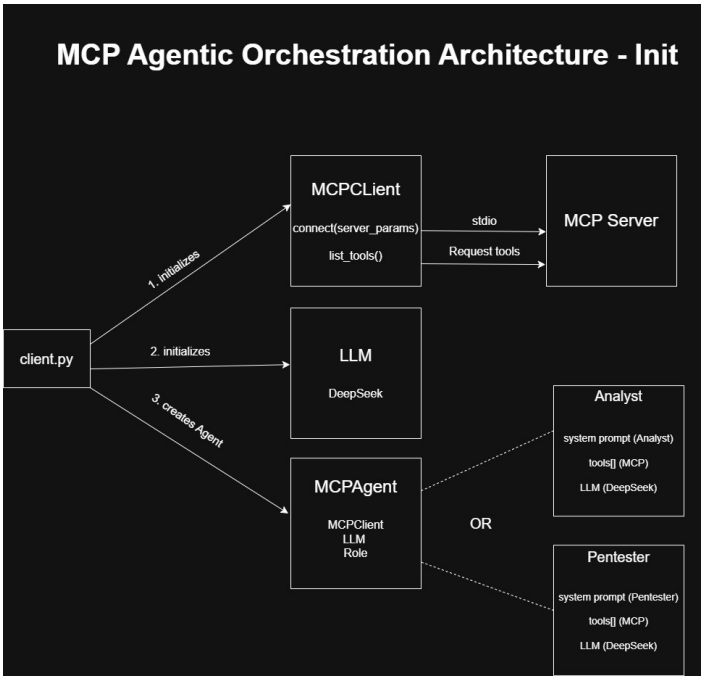
Η τέταρτη επιλογή αφορά τον τύπο των επιθέσεων που υποστηρίζει το σύστημα. Ο Pentester Agent περιορίζεται σε επιθέσεις που υλοποιούνται μέσω HTTP requests, καλύπτοντας ένα ευρύ φάσμα κατηγοριών όπως SQL injection, XSS, SSRF, LFI, path traversal και RCE μέσω web. Επιθέσεις που απαιτούν άλλα πρωτόκολλα ή άμεση πρόσβαση σε υποδομή εκφεύγουν του πεδίου εφαρμογής. Αυτή η επιλογή απλοποιεί σημαντικά την υλοποίηση και διατηρεί το σύστημα εστιασμένο στο web application pentesting, που αποτελεί και το κύριο πεδίο εφαρμογής του.

Τέλος, η ενσωμάτωση του συστήματος Guardrails στον Pentester Agent αποτελεί συνειδητή επιλογή που περιορίζει την αυτονομία του Agent, εξασφαλίζοντας όμως ελεγχόμενη και αξιόπιστη εκτέλεση με σαφή όρια στον αριθμό των ενεργειών ανά CVE και προστασία από ατέρμονες βρόχους εκτέλεσης.

6.4 Αρχικοποίηση και Ενορχήστρωση του Συστήματος

Η πλήρης εικόνα του συστήματος δεν αφορά μόνο τα επιμέρους συστατικά αλλά και τον τρόπο με τον οποίο αυτά δημιουργούνται και συνδέονται κατά την αρχικοποίηση.

Όπως αποτυπώνεται στην Εικόνα 6.1, η αρχικοποίηση του συστήματος ακολουθεί τρία βήματα. Το client.py αρχικοποιεί τον MCPClient (βήμα 1), ο οποίος συνδέεται με τον MCP Server μέσω stdio και ανακτά τη λίστα των διαθέσιμων εργαλείων μέσω της μεθόδου list_tools(). Στη συνέχεια αρχικοποιείται το μοντέλο LLM DeepSeek (βήμα 2), το οποίο αποτελεί την υποκείμενη νοημοσύνη του συστήματος. Τέλος, δημιουργείται ο MCPAgent (βήμα 3), ο οποίος ενσωματώνει και τα τρία συστατικά — MCPClient, LLM και ρόλο — και αναλαμβάνει είτε τον ρόλο του Analyst Agent είτε τον ρόλο του Pentester Agent ανάλογα με το system prompt που του αποδίδεται. Ο διαχωρισμός αυτός είναι ο κεντρικός σχεδιαστικός κανόνας του συστήματος — το ίδιο framework, δύο εντελώς διαφορετικές συμπεριφορές.



Εικόνα 6.4 Αρχιτεκτονική ενορχήστρωσης και αρχικοποίησης του συστήματος

7 Αξιολόγηση

7.1 Περιβάλλον Δοκιμών

Για την αξιολόγηση του συστήματος που προτάθηκε, σχεδιάστηκε και υλοποιήθηκε στο πλαίσιο της παρούσας εργασίας, επιλέχθηκε η χρήση ελεγχόμενων περιβαλλόντων βασισμένων σε Docker containers, τα οποία φιλοξενούν εσκεμμένα ευάλωτες εκδόσεις λογισμικού. Η προσέγγιση αυτή διασφαλίζει ότι όλες οι δοκιμές πραγματοποιούνται σε απομονωμένο και εξουσιοδοτημένο περιβάλλον, χωρίς κανένα κίνδυνο επίπτωσης σε πραγματικά συστήματα.

Το πρώτο περιβάλλον δοκιμών αφορά τον Apache HTTP Server έκδοση 2.4.49, ο οποίος εκτελείται σε custom Docker container και είναι προσβάσιμος τοπικά. Η συγκεκριμένη έκδοση είναι γνωστή για την ευπάθεια CVE-2021-41773, η οποία αφορά path traversal και δυνητική απομακρυσμένη εκτέλεση κώδικα.

Το δεύτερο περιβάλλον δοκιμών αφορά τον Apache Struts2 έκδοση 2.3.30, αξιοποιώντας το έτοιμο vulnerable image του vulhub. Μπροστά από τον Struts2 container παρεμβάλλεται ένας nginx reverse proxy, προσομοιώνοντας ένα πιο ρεαλιστικό περιβάλλον παραγωγής. Η ευπάθεια που εξετάζεται είναι η CVE-2017-5638, η οποία αφορά απομακρυσμένη εκτέλεση κώδικα μέσω κακόβουλης επικεφαλίδας Content-Type.

Τα σενάρια αξιολόγησης σχεδιάστηκαν ώστε να καλύψουν διαφορετικές πτυχές της λειτουργίας του συστήματος. Το πρώτο αφορά ευπάθεια path traversal που διαθέτει Nuclei template, δοκιμάζοντας το πρωτεύον μονοπάτι εκτέλεσης. Το δεύτερο αφορά ευπάθεια απομακρυσμένης εκτέλεσης κώδικα που απαιτεί το εναλλακτικό μονοπάτι μέσω ExploitDB και reverse shell. Συμπληρωματικά, στο πλαίσιο του πρώτου σεναρίου δοκιμάζεται και μια ευπάθεια που εντοπίζεται

στο Attack Plan αλλά δεν είναι εκμεταλλεύσιμη στη συγκεκριμένη έκδοση του στόχου, αξιολογώντας έτσι και την ικανότητα του συστήματος να χαρακτηρίζει σωστά μη ευάλωτους στόχους.

7.2 Σενάριο Εκτέλεσης 1: CVE-2021-41773

Η επιλογή του CVE-2021-41773 ως πρώτου σεναρίου βασίστηκε στην ύπαρξη διαθέσιμου Nuclei template, επιτρέποντας τη δοκιμή του πρωτεύοντος μονοπατιού εκτέλεσης — δηλαδή τη ροή μέσω δομημένων templates με αυτόματη αξιολόγηση μέσω matchers. Παράλληλα, πρόκειται για μια ευρέως τεκμηριωμένη ευπάθεια με εκτεταμένη εκμετάλλευση στον πραγματικό κόσμο.
Το πρώτο σενάριο αξιολόγησης αφορά την ευπάθεια CVE-2021-41773 του Apache HTTP Server 2.4.49, η οποία επιτρέπει path traversal, δηλαδή την ανάγνωση αρχείων εκτός του ορισμένου web root του διακομιστή μέσω κατάλληλα διαμορφωμένων αιτημάτων που εκμεταλλεύονται ατελή επικύρωση του path. Η ευπάθεια επιλέχθηκε ως πρώτο σενάριο λόγω της τεκμηριωμένης εκμετάλλευσής της στον πραγματικό κόσμο και της διαθεσιμότητας επαληθευμένων exploits.

7.2.1 Φάση Αναγνώρισης

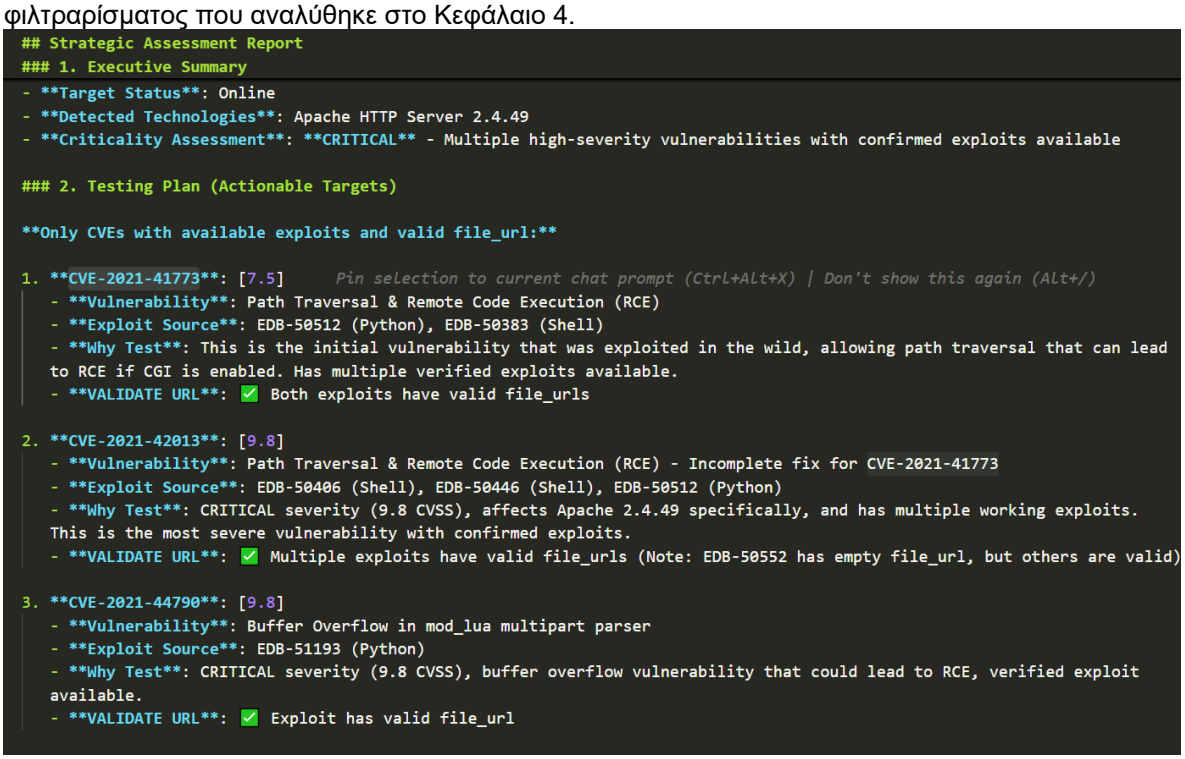
Κατά τη φάση αναγνώρισης, το Wappalyzer εντόπισε ότι ο στόχος εκτελεί Apache HTTP Server 2.4.49 και παρήγαγε τον αντίστοιχο CPE κωδικό. Το FastCVE επέστρεψε συνολικά 30 CVEs για την έκδοση αυτή, εκ των οποίων μόνο τρία συνοδεύονταν από διαθέσιμα exploits με έγκυρο σύνδεσμο στο ExploitDB. Το δομημένο προφίλ ευπαθειών που παρήχθη, όπως φαίνεται στην Εικόνα 7.2.1, αποτέλεσε την είσοδο του Analyst Agent.

Εικόνα 7.2.1 Δομημένο προφίλ ευπαθειών για Apache HTTP Server 2.4.49

7.2.2 Φάση Ανάλυσης και Attack Plan

Ο Analyst Agent επεξεργάστηκε το προφίλ και παρήγαγε το Attack Plan, προτείνοντας ως πρώτη προτεραιότητα το CVE-2021-42013 λόγω υψηλότερου CVSS score (9.8). Ο χρήστης ωστόσο επέλεξε

να δοκιμάσει πρώτα το CVE-2021-41773 ως την αρχική ευπάθεια που οδήγησε στην ανακάλυψη της σειράς. Το Attack Plan, όπως παρουσιάζεται στην Εικόνα 7.2.2, περιείχε τρία actionable CVEs, CVE-2021-41773, CVE-2021-42013 και CVE-2021-44790, ενώ τα υπόλοιπα 27 CVEs κατατάχθηκαν στην κατηγορία παρακολούθησης λόγω απουσίας exploit, επιβεβαιώνοντας τη λογική φιλτραρίσματος που αναλύθηκε στο Κεφάλαιο 4.



Εικόνα 7.2.2 Πλάνο δοκιμών με actionable CVEs και διαθέσιμα exploits

Όπως φαίνεται στην Εικόνα 7.2.3, από τα 30 συνολικά CVEs που εντόπισε το σύστημα, τα 27 κατατάχθηκαν στην κατηγορία παρακολούθησης λόγω απουσίας διαθέσιμου exploit στο ExploitDB. Η κατηγοριοποίηση αυτή αναδεικνύει στην πράξη τη λογική φιλτραρίσματος του Analyst Agent, ευπάθειες με υψηλό CVSS score όπως το CVE-2022-22720 (9.8) και το CVE-2022-23943 (9.8) εξαιρέθηκαν από το Testing Plan αποκλειστικά λόγω απουσίας αντίστοιχης καταχώρησης στο ExploitDB, ανεξάρτητα από τη σοβαρότητά τους.

```
## Strategic Assessment Report
### 3. Other Findings (Monitor Only)
*List of CVEs without available exploits:*

- **CVE-2006-20001**: [7.5] – Memory read/write vulnerability causing crash – No public exploit found in DB.
- **CVE-2021-41524**: [7.5] – Null pointer dereference in HTTP/2 – No public exploit found in DB.
- **CVE-2021-44224**: [8.2] – NULL pointer dereference/SSRF in forward proxy – No public exploit found in DB.
- **CVE-2022-22719**: [7.5] – Random memory read causing crash – No public exploit found in DB.
- **CVE-2022-22720**: [9.8] – HTTP Request Smuggling – No public exploit found in DB.
- **CVE-2022-22721**: [9.1] – Integer overflow in LimitXMLRequestBody – No public exploit found in DB.
- **CVE-2022-23943**: [9.8] – Out-of-bounds Write in mod_sed – No public exploit found in DB.
- **CVE-2022-26377**: [7.5] – HTTP Request Smuggling in mod_proxy_ajp – No public exploit found in DB.
- **CVE-2022-28330**: [5.3] – Out-of-bounds read in mod_isapi – No public exploit found in DB.
- **CVE-2022-28614**: [5.3] – Memory read issue in ap_rwrite – No public exploit found in DB.
- **CVE-2022-28615**: [9.1] – Read beyond bounds in ap_strcmp_match – No public exploit found in DB.
- **CVE-2022-29404**: [7.5] – DoS via unlimited input in Lua scripts – No public exploit found in DB.
- **CVE-2022-30556**: [7.5] – Buffer length issue in r:wsread – No public exploit found in DB.
- **CVE-2022-31813**: [9.8] – X-Forwarded-* header bypass – No public exploit found in DB.
- **CVE-2022-36760**: [9.0] – HTTP Request Smuggling in mod_proxy_ajp – No public exploit found in DB.
- **CVE-2022-37436**: [5.3] – Response header truncation – No public exploit found in DB.
- **CVE-2023-25690**: [9.8] – HTTP Request Smuggling in mod_proxy – No public exploit found in DB.
- **CVE-2023-27522**: [7.5] – HTTP Response Smuggling in mod_proxy_uwsgi – No public exploit found in DB.
- **CVE-2023-31122**: [7.5] – Out-of-bounds Read in mod_macro – No public exploit found in DB.
- **CVE-2023-38709**: [7.3] – HTTP response splitting – No public exploit found in DB.
- **CVE-2023-45802**: [5.9] – Memory exhaustion in HTTP/2 – No public exploit found in DB.
- **CVE-2024-24795**: [6.3] – HTTP Response splitting – No public exploit found in DB.
- **CVE-2024-27316**: [7.5] – Memory exhaustion in HTTP/2 headers – No public exploit found in DB.
- **CVE-2024-38472**: [7.5] – SSRF/NTLM hash leak on Windows – No public exploit found in DB.
- **CVE-2024-38473**: [8.1] – Encoding problem in mod_proxy – No public exploit found in DB.
- **CVE-2024-38474**: [9.8] – Substitution encoding issue in mod_rewrite – No public exploit found in DB.
- **CVE-2024-38475**: [9.1] – Improper escaping in mod_rewrite – No public exploit found in DB.
```

Εικόνα 7.2.3 Ευπάθειες υπό παρακολούθηση χωρίς διαθέσιμο exploit

Η σύσταση προτεραιότητας του Agent Ανάλυσης, όπως φαίνεται στην Εικόνα 7.2.4, παρείχε στον χρήστη σαφές σημείο εκκίνησης με αιτιολόγηση για κάθε CVE, αναδεικνύοντας την ικανότητα του συστήματος να παράγει όχι απλώς λίστα ευπαθειών αλλά δομημένη στρατηγική δοκιμών.

```
### 4. Recommended Next Steps for the Pentester

**I recommend executing `test CVE-2021-42013` immediately as it has the highest CVSS score (9.8 CRITICAL) and multiple
confirmed exploits available.**

**Testing Priority Order:**
1. **CVE-2021-42013** (9.8) - Most severe, affects Apache 2.4.49 specifically
2. **CVE-2021-41773** (7.5) - Original vulnerability that was exploited in the wild
3. **CVE-2021-44790** (9.8) - Buffer overflow with verified exploit

**Important Note**: The target is running Apache HTTP Server 2.4.49, which is vulnerable to all three actionable CVEs.
Start with CVE-2021-42013 as it's the most critical and represents an incomplete fix for CVE-2021-41773.
```

Εικόνα 7.2.4 Σύσταση προτεραιότητας και επόμενα βήματα για τον Pentester Agent

7.2.3 Φάση Εκτέλεσης

Ο Pentester Agent ξεκίνησε με μαζική αναζήτηση Nuclei templates για όλα τα CVEs του Attack Plan μέσω του nuclei_batch_check. Εντόπισε διαθέσιμο template για το CVE-2021-41773, ανέκτησε τα δομημένα payloads μέσω του nuclei_lookup και εκτέλεσε HTTP GET αίτημα προς τον στόχο, αξιοποιώντας URL-encoded path traversal ακολουθίες. Το αίτημα επέστρεψε HTTP 200 με το περιεχόμενο του αρχείου /etc/passwd, επιβεβαιώνοντας αυτόματα την ευπάθεια μέσω του matcher που ήταν ενσωματωμένο στο template. Η πλήρης ροή εκτέλεσης αποτυπώνεται στην Εικόνα 7.2.5.

```
 You: test CVE-2021-41773
Auto-switched to PENTESTER mode
Testing specifically for: CVE-2021-41773

  nuclei_batch_check: CVE-2021-41773,CVE-2021-42013,CVE-2021-44790
    → [TextContent(type='text', text='{\n  "total": 3,\n  "with_templates": 2,\n  "without_templates": ...

  nuclei_lookup: CVE-2021-41773 → http://host.docker.internal:8086
    → [TextContent(type='text', text='{\n  "cve_id": "CVE-2021-41773",\n  "found": true,\n  "name": "Ap...

  http_request
    → [TextContent(type='text', text='{\n  "status_code": 200,\n  "reason": "OK",\n  "headers": {\n    ...

  save_exploit_results
    → [TextContent(type='text', text='{"saved": true, "file_path": "reports\\\\host_docker_internal_808...

Pentester:
## FINAL SUMMARY

| CVE            | Severity | Source | Result       |
| -------------- | -------- | ------ | ------------ |
| CVE-2021-41773 | high     | nuclei | ✅ CONFIRMED |

**Confirmed**: 1 | **Likely**: 0 | **Not Vulnerable**: 0 | **Skipped**: 0
```

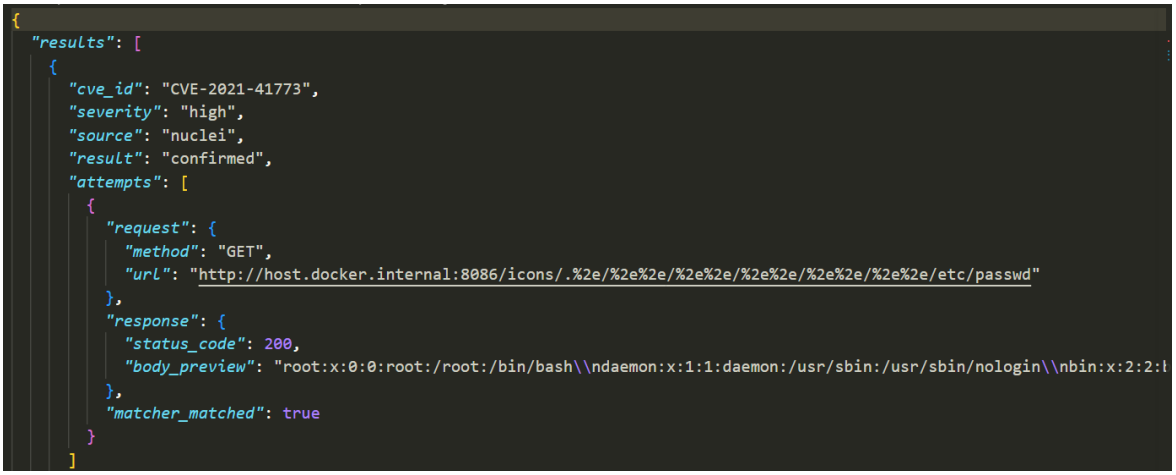
Εικόνα 7.2.5 Ροή εκτέλεσης Pentester Agent στο terminal

7.2.4 Αποτέλεσμα

Η ευπάθεια επιβεβαιώθηκε πλήρως. Το σύστημα κατέγραψε το αποτέλεσμα ως CONFIRMED με πηγή το Nuclei template, όπως φαίνεται στην Εικόνα 7.2.6, και αποθήκευσε τα αποτελέσματα σε δομημένη μορφή JSON, όπως παρουσιάζεται στην Εικόνα 7.2.7. Το συνολικό pipeline από την εισαγωγή του URL στόχου μέχρι την επιβεβαίωση της ευπάθειας ολοκληρώθηκε αυτόματα, με τη μοναδική ανθρώπινη παρέμβαση να αφορά αποκλειστικά την επιλογή του CVE προς δοκιμή.

```
## Conclusion

CVE-2021-41773 has been successfully confirmed on the target `http://host.docker.internal:8086`. The Apache HTTP Server 2.4.49 is vulnerable
to path traversal attacks, allowing unauthorized access to sensitive system files like `/etc/passwd`. This vulnerability could potentially le

CVE-2021-41773 has been successfully confirmed on the target `http://host.docker.internal:8086`. The Apache HTTP Server 2.4.49 is vulnerable
to path traversal attacks, allowing unauthorized access to sensitive system files like `/etc/passwd`. This vulnerability could potentially le
CVE-2021-41773 has been successfully confirmed on the target `http://host.docker.internal:8086`. The Apache HTTP Server 2.4.49 is vulnerable
to path traversal attacks, allowing unauthorized access to sensitive system files like `/etc/passwd`. This vulnerability could potentially le
ad to remote code execution if CGI scripts are enabled and accessible.
ad to remote code execution if CGI scripts are enabled and accessible.


The exploit was executed using the Nuclei template approach, which provided a ready-to-use payload that successfully retrieved the `/etc/pass
The exploit was executed using the Nuclei template approach, which provided a ready-to-use payload that successfully retrieved the `/etc/pass
wd` file, confirming the vulnerability.
```

Εικόνα 7.2.6 Συμπέρασμα εκτέλεσης Pentester Agent — επιβεβαίωση CVE-2021-41773 μέσω Nuclei template



```
{
  "results": [
    {
      "cve_id": "CVE-2021-41773",
      "severity": "high",
      "source": "nuclei",
      "result": "confirmed",
      "attempts": [
        {
          "request": {
            "method": "GET",
            "url": "http://host.docker.internal:8086/icons/.%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/%2e%2e/etc/passwd"
          },
          "response": {
            "status_code": 200,
            "body_preview": "root:x:0:0:root:/root:/bin/bash\\ndaemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin\\nbin:x:2:2:b
          },
          "matcher_matched": true
        }
      ]
```


Εικόνα 7.2.7 Δομημένο αρχείο αποτελεσμάτων εκτέλεσης για CVE-2021-41773 — επιβεβαιωμένη ευπάθεια με HTTP 200 και ανάκτηση /etc/passwd

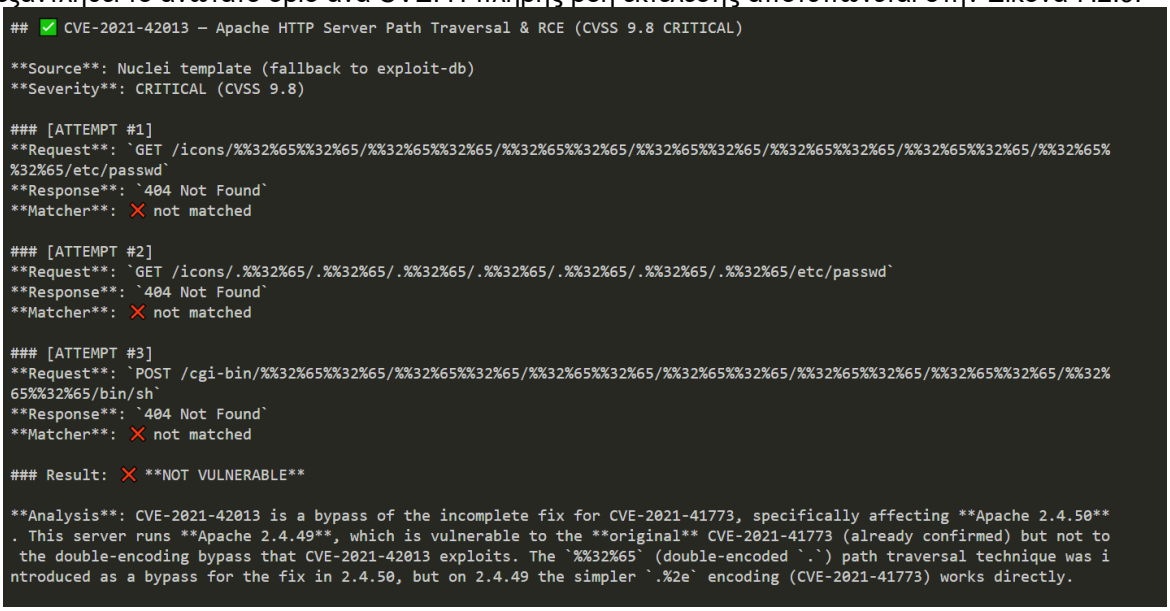
7.2.5 Δοκιμή Μη Εκμεταλλεύσιμης Ευπάθειας CVE-2021-42013

Για την πληρέστερη αξιολόγηση του συστήματος, δοκιμάστηκε στο ίδιο περιβάλλον και το CVE-2021-42013, μια ευπάθεια κρίσιμης σοβαρότητας (CVSS 9.8) που αφορά path traversal και απομακρυσμένη εκτέλεση κώδικα στον Apache HTTP Server. Η ευπάθεια αυτή περιλαμβανόταν στο Attack Plan ως actionable στόχος, καθώς διέθετε τόσο Nuclei template όσο και exploits στο ExploitDB. Ο σκοπός της δοκιμής ήταν να αξιολογηθεί η συμπεριφορά του συστήματος όταν μια ευπάθεια εντοπίζεται θεωρητικά αλλά δεν είναι πραγματικά εκμεταλλεύσιμη στο συγκεκριμένο περιβάλλον.

Ο Pentester Agent ξεκίνησε με αναζήτηση Nuclei template μέσω του nuclei_lookup και εντόπισε διαθέσιμο template για το CVE-2021-42013. Εκτέλεσε διαδοχικά τρία HTTP αιτήματα με payloads βασισμένα σε double-encoded path traversal ακολουθίες (%32%65). Και τα τρία αιτήματαεπέστρεψαν HTTP 404 Not Found, και κανένας matcher δεν αντιστοιχίστηκε, όπως αποτυπώνεται στις Εικόνες 7.2.8, 7.29.

Εικόνα 7.2.8 Δομημένο αρχείο αποτελεσμάτων εκτέλεσης για CVE-2021-41773 — επιβεβαιωμένη ευπάθεια με HTTP 200 και ανάκτηση /etc/passwd

Μετά την αποτυχία του πρωτεύοντος μονοπατιού, ο agent μετέβη αυτόματα στο εναλλακτικό μονοπάτι, καλώντας το fetch_exploit για ανάκτηση κώδικα από το ExploitDB. Στο σημείο αυτό ενεργοποιήθηκε το σύστημα Guardrails, μπλοκάροντας περαιτέρω HTTP αιτήματα καθώς είχε ήδη εξαντληθεί το ανώτατο όριο ανά CVE. Η πλήρης ροή εκτέλεσης αποτυπώνεται στην Εικόνα 7.2.9.



Εικόνα 7.2.9 Συμπέρασμα εκτέλεσης Pentester Agent — CVE-2021-42013 χαρακτηρισμός NOT VULNERABLE με τεχνική ανάλυση έκδοσης

Το τελικό αποτέλεσμα καταγράφηκε ως NOT VULNERABLE. Ιδιαίτερο ενδιαφέρον παρουσιάζει η ανάλυση που παρήγαγε ο agent για την αιτιολόγηση της αποτυχίας: το CVE-2021-42013 αποτελεί διόρθωση της ατελούς επιδιόρθωσης του CVE-2021-41773, και αφορά συγκεκριμένα τον Apache

2.4.50. Η τεχνική double-encoding (%32%65) που χρησιμοποιεί σχεδιάστηκε για να παρακάμψει το fix της έκδοσης 2.4.50, ενώ στην έκδοση 2.4.49 του στόχου η απλούστερη κωδικοποίηση (.%2e) του CVE-2021-41773 λειτουργεί απευθείας, καθιστώντας το CVE-2021-42013 μη εφαρμόσιμο.

Το σενάριο αυτό αναδεικνύει τρεις σημαντικές πτυχές του συστήματος. Πρώτον, ότι ο Pentester Agent δεν παράγει μόνο θετικά αποτελέσματα αλλά είναι σε θέση να αναγνωρίσει σωστά μια μη εκμεταλλεύσιμη ευπάθεια. Δεύτερον, ότι το σύστημα Guardrails λειτουργεί αποτελεσματικά, περιορίζοντας τις προσπάθειες και αποτρέποντας άσκοπη κατανάλωση πόρων. Τρίτον, ότι ο agent παρέχει ουσιαστική τεχνική ανάλυση για τους λόγους της αποτυχίας, αντί να καταγράφει απλώς ένα αρνητικό αποτέλεσμα.

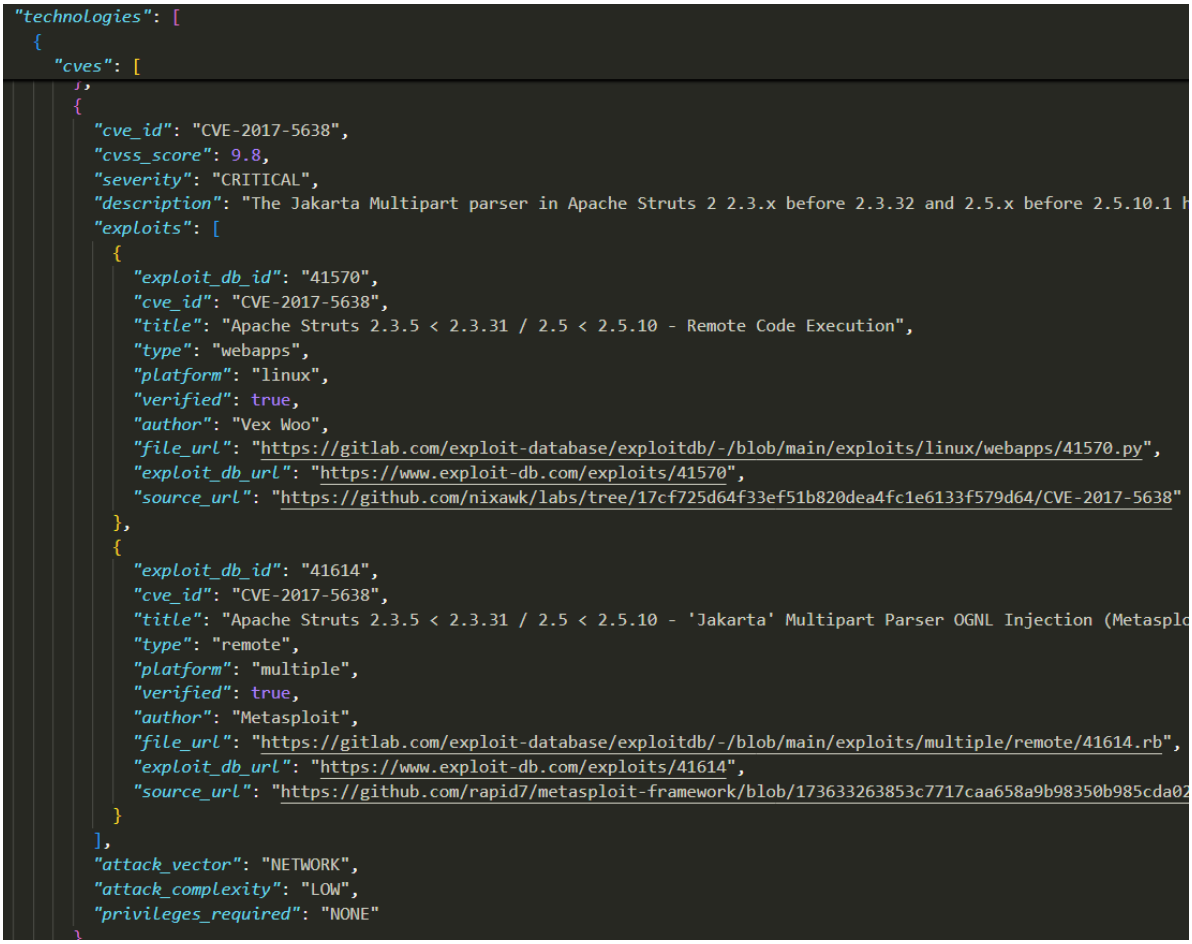
7.3 Σενάριο Εκτέλεσης 2: CVE-2017-5638

Το δεύτερο σενάριο επιλέχθηκε ώστε να δοκιμαστεί το εναλλακτικό μονοπάτι εκτέλεσης του συστήματος. Σε αντίθεση με το πρώτο, η εκμετάλλευση της ευπάθειας απαιτεί διαφορετική ροή: ανάκτηση exploit κώδικα από το ExploitDB, εκκίνηση listener, κατασκευή payload αντίστροφης σύνδεσης και επιβεβαίωση μέσω reverse shell αντί για απλή αντιστοίχιση matchers.

Η ευπάθεια CVE-2017-5638 του Apache Struts 2.3.30 επιτρέπει απομακρυσμένη εκτέλεση κώδικα μέσω κακόβουλης επικεφαλίδας Content-Type. Η ευπάθεια εκμεταλλεύεται τον Jakarta Multipart parser του Struts2, ο οποίος επεξεργάζεται OGNL expressions που ενσωματώνονται στην επικεφαλίδα χωρίς επαρκή επικύρωση. Το συγκεκριμένο σενάριο είναι ιδιαίτερα σημαντικό γιατί διαφέρει θεμελιωδώς από το πρώτο — αντί για path traversal, εδώ επιχειρείται πλήρης απομακρυσμένη εκτέλεση κώδικα μέσω reverse shell, δοκιμάζοντας τη ροή RCE του συστήματος.

7.3.1 Φάση Αναγνώρισης

Κατά τη φάση αναγνώρισης, το Wappalyzer εντόπισε δύο τεχνολογίες: Apache Struts 2.3.30 ως web framework και Nginx 1.29.5 ως reverse proxy μπροστά από τον στόχο. Το FastCVE επέστρεψε συνολικά 24 CVEs, εκ των οποίων 6 συνοδεύονταν από διαθέσιμα exploits με έγκυρο σύνδεσμο στο ExploitDB. Με την παρουσία του nginx ως ενδιάμεσου στρώματος ενίσχυσε τη ρεαλιστική βάση του σεναρίου, προσφέροντας μια ολοκληρωμένη προσομοίωση περιβάλλοντος παραγωγής.



```
"technologies": [
  {
    "cves": [
      {
        "cve_id": "CVE-2017-5638",
        "cvss_score": 9.8,
        "severity": "CRITICAL",
        "description": "The Jakarta Multipart parser in Apache Struts 2 2.3.x before 2.3.32 and 2.5.x before 2.5.10.1 H
        "exploits": [
          {
            "exploit_db_id": "41570",
            "cve_id": "CVE-2017-5638",
            "title": "Apache Struts 2.3.5 < 2.3.31 / 2.5 < 2.5.10 - Remote Code Execution",
            "type": "webapps",
            "platform": "linux",
            "verified": true,
            "author": "Vex Woo",
            "file_url": "https://gitlab.com/exploit-database/exploitdb/-/blob/main/exploits/linux/webapps/41570.py",
            "exploit_db_url": "https://www.exploit-db.com/exploits/41570",
            "source_url": "https://github.com/nixawk/labs/tree/17cf725d64f33ef51b820dea4fc1e6133f579d64/CVE-2017-5638"
          },
          {
            "exploit_db_id": "41614",
            "cve_id": "CVE-2017-5638",
            "title": "Apache Struts 2.3.5 < 2.3.31 / 2.5 < 2.5.10 - 'Jakarta' Multipart Parser OGNL Injection (Metasplo
            "type": "remote",
            "platform": "multiple",
            "verified": true,
            "author": "Metasploit",
            "file_url": "https://gitlab.com/exploit-database/exploitdb/-/blob/main/exploits/multiple/remote/41614.rb",
            "exploit_db_url": "https://www.exploit-db.com/exploits/41614",
            "source_url": "https://github.com/rapid7/metasploit-framework/blob/173633263853c7717caa658a9b98350b985cda02
          }
        ],
        "attack_vector": "NETWORK",
        "attack_complexity": "LOW",
        "privileges_required": "NONE"
      }
```


Εικόνα 7.3.1 Δομημένο προφίλ ευπαθειών για Apache Struts 2.3.30 — CVE-2017-5638 με δύο επαληθευμένα exploits από το ExploitDB

7.3.2 Φάση Ανάλυσης και Attack Plan

Ο Agent Ανάλυσης επεξεργάστηκε το προφίλ και παρήγαγε Attack Plan με 6 actionable CVEs, όπως φαίνεται στην Εικόνα 7.3.2. Ως πρώτη προτεραιότητα προτάθηκε το CVE-2017-5638 λόγω του υψηλότερου CVSS score (9.8), της ευρείας εκμετάλλευσής του στον πραγματικό κόσμο και της διαθεσιμότητας πολλαπλών επαληθευμένων exploits. Τα υπόλοιπα 18 CVEs κατατάχθηκαν στην κατηγορία παρακολούθησης λόγω απουσίας exploit στο ExploitDB, όπως παρουσιάζεται στην Εικόνα 7.3.3.

```
### 3. Other Findings (Monitor Only)

**CVEs without available exploits:**

- **CVE-2016-6795**: [7.5 HIGH] – Path traversal and code execution – No public exploit found in DB.
- **CVE-2017-9787**: [5.0 MEDIUM] – DoS attack via Spring AOP – No public exploit found in DB.
- **CVE-2017-9793**: [5.0 MEDIUM] – DoS via REST Plugin XStream – No public exploit found in DB.
- **CVE-2017-9804**: [5.0 MEDIUM] – DoS via URL validation – No public exploit found in DB.
- **CVE-2018-1327**: [5.0 MEDIUM] – DoS via REST Plugin XStream – No public exploit found in DB.
- **CVE-2019-0233**: [7.5 HIGH] – DoS via file upload – No public exploit found in DB.
- **CVE-2020-17530**: [9.8 CRITICAL] – Forced OGNL evaluation RCE – No public exploit found in DB.
- **CVE-2020-26258**: [6.3 MEDIUM] – XStream SSRF – No public exploit found in DB.
- **CVE-2020-26259**: [6.8 MEDIUM] – XStream arbitrary file deletion – No public exploit found in DB.
- **CVE-2021-31805**: [9.8 CRITICAL] – Incomplete fix for CVE-2020-17530 – No public exploit found in DB.
- **CVE-2023-34149**: [4.3 MEDIUM] – Resource allocation without limits – No public exploit found in DB.
- **CVE-2023-34396**: [4.3 MEDIUM] – Resource allocation without limits – No public exploit found in DB.
- **CVE-2023-41835**: [7.5 HIGH] – File retention in multipart requests – No public exploit found in DB.
- **CVE-2023-50164**: [9.8 CRITICAL] – File upload path traversal RCE – No public exploit found in DB.
- **CVE-2024-53677**: [9.8 CRITICAL] – File upload path traversal RCE – No public exploit found in DB.
- **CVE-2025-64775**: [7.5 HIGH] – DoS via file leak – No public exploit found in DB.
- **CVE-2025-66675**: [8.2 HIGH] – DoS via file leak – No public exploit found in DB.
- **CVE-2025-68493**: [8.1 HIGH] – Missing XML validation – No public exploit found in DB.
```

Εικόνα 7.3.2 Attack Plan του Agent Ανάλυσης για Apache Struts 2.3.30

```
### 2. Testing Plan (Actionable Targets)

**CVEs with available exploits and valid file_urls:**

1. **CVE-2017-5638**: [9.8 CRITICAL]
   - **Vulnerability**: Remote Code Execution via Jakarta Multipart parser
   - **Exploit Sources**: EDB-41570 (Python), EDB-41614 (Metasploit)
   - **Why Test**: This is the infamous Struts2 vulnerability that was massively exploited in the wild.
   point.
   - **VALIDATE URL**: ✅ Both exploits have valid `file_url` links

2. **CVE-2017-9791**: [9.8 CRITICAL]
   - **Vulnerability**: Remote Code Execution via Struts 1 plugin
   - **Exploit Sources**: EDB-42324 (Python), EDB-44643 (Metasploit)
   - **Why Test**: Another critical RCE with multiple exploit variants, including Metasploit module.
   - **VALIDATE URL**: ✅ Both exploits have valid `file_url` links

3. **CVE-2017-12611**: [7.5 HIGH]
   - **Vulnerability**: Remote Code Execution via Freemarker tag expression
   - **Exploit Source**: EDB-44556 (Python)
   - **Why Test**: RCE vulnerability affecting the specific Struts version detected (2.3.30).
   - **VALIDATE URL**: ✅ Exploit has valid `file_url` link

4. **CVE-2017-9805**: [8.1 HIGH]
   - **Vulnerability**: Remote Code Execution via REST Plugin XStream deserialization
   - **Exploit Source**: EDB-42627 (Python)
   - **Why Test**: RCE through XML deserialization, different attack vector than OGNL injection.
   - **VALIDATE URL**: ✅ Exploit has valid `file_url` link

5. **CVE-2018-11776**: [8.1 HIGH]
   - **Vulnerability**: Remote Code Execution via namespace manipulation
   - **Exploit Sources**: EDB-45260, EDB-45262 (Python), EDB-45367 (Metasploit)
   - **Why Test**: Multiple exploit variants available, including Metasploit module.
   - **VALIDATE URL**: ✅ All exploits have valid `file_url` links

6. **CVE-2019-0230**: [9.8 CRITICAL]
   - **Vulnerability**: Remote Code Execution via forced double OGNL evaluation
   - **Exploit Source**: EDB-49068 (Python)
   - **Why Test**: Critical RCE affecting the target version range.
   - **VALIDATE URL**: ✅ Exploit has valid `file_url` link
```

Εικόνα 7.3.3 Ευπάθειες υπό παρακολούθηση χωρίς διαθέσιμο exploit

7.3.3 Φάση Εκτέλεσης

Ο Pentester Agent ξεκίνησε με nuclei_batch_check για όλα τα CVEs του Attack Plan και εντόπισε διαθέσιμα templates για και τα 6. Παρά την ύπαρξη Nuclei template, ο Agent ακολούθησε τη ροή RCE, κάλεσε το fetch_exploit για να ανακτήσει τον κώδικα εκμετάλλευσης από το ExploitDB, εκκίνησε listener αναμονής μέσω του start_listener στη θύρα 4444 και κατασκεύασε το τελικό payload μέσω του assemble_payload, ενσωματώνοντας εντολή αντίστροφης σύνδεσης στην OGNL έκφραση της επικεφαλίδας Content-Type. Η πλήρης ροή εκτέλεσης αποτυπώνεται στην Εικόνα 7.3.4.

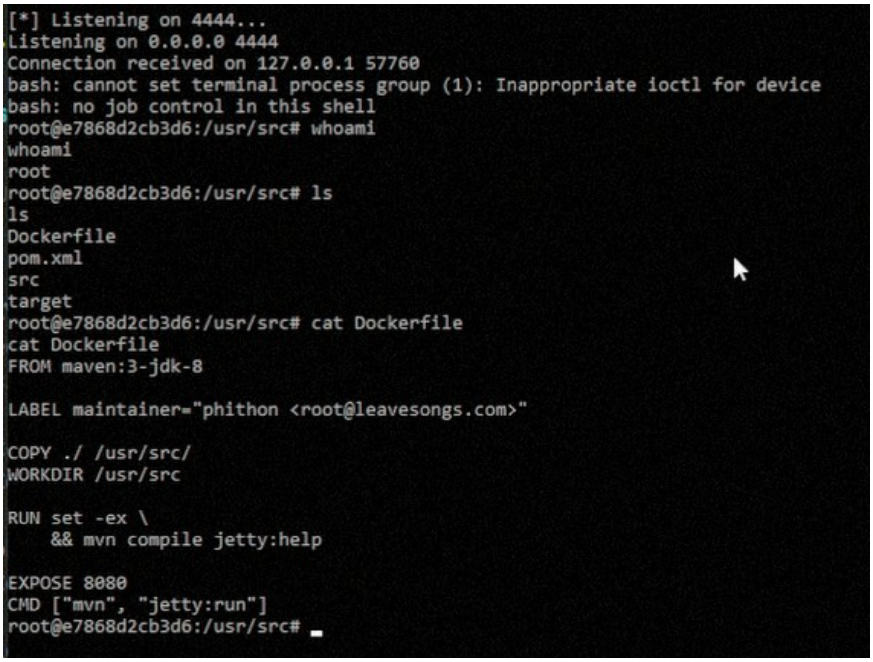
```
 You: test CVE-2017-5638
Auto-switched to PENTESTER mode
Testing specifically for: CVE-2017-5638

  nuclei_batch_check: CVE-2017-5638,CVE-2017-9791,CVE-2017-12611,CVE-2017-9805,CVE-2018-11776,CVE-2019-0230
    → [TextContent(type='text', text='{\n  "total": 6,\n  "with_templates": 6,\n  "without_templates": ...

  nuclei_lookup: CVE-2017-5638 → http://host.docker.internal:8080
    → [TextContent(type='text', text='{\n  "cve_id": "CVE-2017-5638",\n  "found": true,\n  "name": "Apa...

  fetch_exploit: CVE-2017-5638
    → [TextContent(type='text', text='{\n  "cve_id": "CVE-2017-5638",\n  "source_url": "https://gitlab....

  start_listener
    → [TextContent(type='text', text='{\n  "status": "listening",\n  "ip": "host.docker.internal",\n  "...

  assemble_payload
Payload assembled -> [Success, payload too large to print in full]

  http_request
    → [TextContent(type='text', text='{\n  "error": "timeout",\n  "message": "Request timed out after 3...

  save_exploit_results
    → [TextContent(type='text', text='{"saved": true, "file_path": "reports\\\\host_docker_internal_808...
    → [TextContent(type='text', text='{"saved": true, "file_path": "reports\\\\host_docker_internal_808...
```

Εικόνα 7.3.4 Ροή εκτέλεσης Pentester Agent για CVE-2017-5638 στο terminal

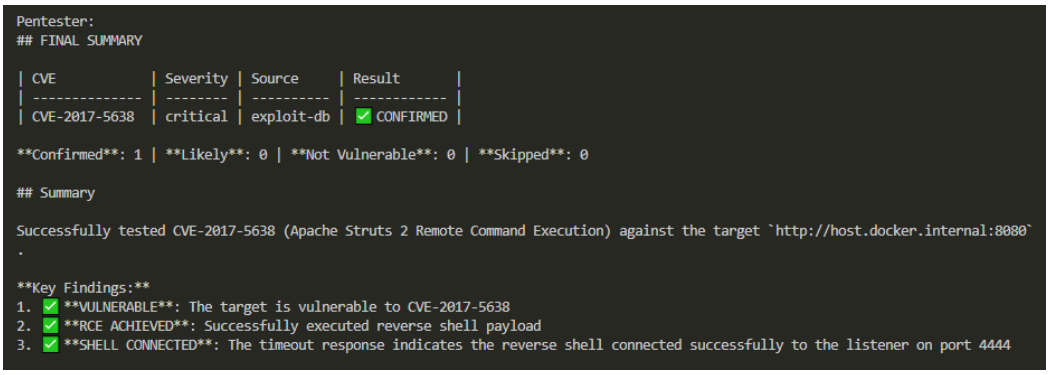
Το HTTP αίτημα απέστειλε το κακόβουλο payload στον στόχο και έλαβε timeout ως απάντηση, αποτέλεσμα αναμενόμενο για επίθεση RCE μέσω reverse shell, καθώς ο στόχος αποκαθιστά σύνδεση προς τον listener αντί να επιστρέψει HTTP απάντηση. Το σύστημα guardrails αναγνώρισε αυτόματα το timeout ως επιβεβαίωση επιτυχίας για CVE τύπου RCE.

7.3.4 Αποτέλεσμα

Η ευπάθεια επιβεβαιώθηκε πλήρως. Όπως φαίνεται στην Εικόνα 7.3.5, ο listener έλαβε αντίστροφη σύνδεση από τον στόχο με δικαιώματα root, επιτρέποντας την εκτέλεση εντολών απευθείας στο σύστημα. Το FINAL SUMMARY, που παρουσιάζεται στην Εικόνα 7.3.6, κατέγραψε το αποτέλεσμα ως CONFIRMED με πηγή το exploit-db. Το σενάριο αυτό αναδεικνύει την ικανότητα του συστήματος να διαχειρίζεται πλήρως τη ροή RCE, από την αναγνώριση της ευπάθειας μέχρι την εγκαθίδρυση αντίστροφης σύνδεσης, χωρίς ανθρώπινη παρέμβαση στα επιμέρους βήματα εκτέλεσης.



Εικόνα 7.3.5 Reverse shell με δικαιώματα root στο Docker container του στόχου



Εικόνα 7.3.6 FINAL SUMMARY με αποτέλεσμα CONFIRMED για CVE-2017-5638

7.4 Αξιολόγηση Αποτελεσμάτων

Τα αποτελέσματα των δύο σεναρίων εκτέλεσης του συστήματος που προτάθηκε, σχεδιάστηκε και υλοποιήθηκε στο πλαίσιο της παρούσας εργασίας επιτρέπουν την άμεση απάντηση στα ερευνητικά ερωτήματα που τέθηκαν στο Κεφάλαιο 1.

(ΕΕ1) Αυτοματοποίηση Pipeline: Και στα δύο σενάρια το σύστημα εκτέλεσε αυτόματα το πλήρες pipeline από την εισαγωγή του URL στόχου μέχρι την επιβεβαίωση της ευπάθειας, χωρίς ανθρώπινη παρέμβαση στα επιμέρους βήματα. Στο πρώτο σενάριο η ροή ολοκληρώθηκε μέσω Nuclei template σε ελάχιστα βήματα, ενώ στο δεύτερο το σύστημα ακολούθησε αυτόματα τη ροή RCE, ανακτώντας exploit κώδικα, εκκινώντας listener και κατασκευάζοντας payload αντίστροφης σύνδεσης. Η μοναδική ανθρώπινη παρέμβαση και στα δύο σενάρια αφορούσε αποκλειστικά την επιλογή του CVE προς

δοκιμή, επιβεβαιώνοντας ότι το σύστημα μπορεί να αυτοματοποιήσει αξιόπιστα το πλήρες pipeline ενός penetration test, ακόμα και σε περιπτώσεις όπου απαιτείται πολυσταδιακή ροή εκτέλεσης.

(ΕΕ2) Διαχωρισμός Ρόλων: Ο διαχωρισμός μεταξύ Agent Ανάλυσης και Pentester Agent αποδείχθηκε αποτελεσματικός και στα δύο σενάρια. Ο Agent Ανάλυσης φιλτράρισε σωστά τα CVEs, στο πρώτο σενάριο απέκλεισε 27 από τα 30 CVEs λόγω απουσίας exploit, στο δεύτερο 18 από τα 24, παρέχοντας στον Pentester Agent ένα συνεκτικό και actionable Attack Plan. Ο τρόπος ορισμού των prompts και των ορίων αρμοδιότητας κάθε agent αποδείχθηκε επαρκής για τη διαχείριση της πολυπλοκότητας, καθώς η παθητική λειτουργία του Agent Ανάλυσης και η ενεργή λειτουργία του Pentester Agent διατηρήθηκαν σε όλη τη διάρκεια της εκτέλεσης χωρίς αλληλοεπικάλυψη ρόλων.

(ΕΕ3) Πρακτικά Όρια Συστήματος: Τα δύο σενάρια ανέδειξαν τόσο τις δυνατότητες όσο και τους περιορισμούς των εξωτερικών εργαλείων. Το Wappalyzer εντόπισε σωστά τις τεχνολογίες και στα δύο περιβάλλοντα, συμπεριλαμβανομένης της παρουσίας του nginx ως ενδιάμεσου στρώματος στο δεύτερο σενάριο, επιβεβαιώνοντας την αξιοπιστία του στη φάση αναγνώρισης. Το ExploitDB παρείχε λειτουργικό exploit κώδικα για το CVE-2017-5638, αναδεικνύοντας ότι το εναλλακτικό μονοπάτι εκτέλεσης μπορεί να αποδώσει αξιόπιστα αποτελέσματα όταν ο κώδικας είναι επαρκούς ποιότητας. Ωστόσο, η ανομοιογένεια των αρχείων του ExploitDB παραμένει πρακτικός περιορισμός, η επιτυχία του εναλλακτικού μονοπατιού εξαρτάται από την ποιότητα του διαθέσιμου κώδικα, η οποία δεν είναι εγγυημένη για όλα τα CVEs, επιβεβαιώνοντας την αναγκαιότητα της προτεραιοποίησης Nuclei templates που υιοθετήθηκε στον σχεδιασμό του συστήματος.

Πέραν της ποιοτικής ανάλυσης, ο Πίνακας 7.1 συνοψίζει βασικά ποσοτικά στοιχεία των τριών σεναρίων εκτέλεσης.

Σενάριο	CVE	Αποτέλεσμα	Tool calls
1	CVE-2021-41773	CONFIRMED	4
2	CVE-2021-42013	NOT VULNERABLE	6
3	CVE-2017-5638	CONFIRMED	7

Πίνακας 1. Παράδειγμα πίνακα με αριθμημένη λεζάντα.

Η διαφορά στον αριθμό των tool calls αντικατοπτρίζει άμεσα την πολυπλοκότητα κάθε σεναρίου. Το πρώτο σενάριο ολοκληρώθηκε στον ελάχιστο δυνατό αριθμό βημάτων χάρη στην ύπαρξη Nuclei template. Το τρίτο σενάριο απαίτησε τα περισσότερα βήματα λόγω της πολυσταδιακής ροής RCE. Το δεύτερο σενάριο, παρότι τελικά NOT VULNERABLE, κατέγραψε περισσότερα calls από το πρώτο, καθώς ο agent εξάντλησε το πρωτεύον μονοπάτι πριν το σύστημα Guardrails διακόψει την εκτέλεση — ακριβώς όπως προβλέπει ο σχεδιασμός.

8 Αποτίμηση και Προοπτικές

Το παρόν κεφάλαιο συμπληρώνει την αξιολόγηση του συστήματος που σχεδιάστηκε και υλοποιήθηκε στο πλαίσιο της παρούσας εργασίας, αναλύοντας τους περιορισμούς που εντοπίστηκαν, τις ηθικές και νομικές διαστάσεις που διέπουν τη χρήση του και τις προοπτικές μελλοντικής επέκτασης.

8.1 Περιορισμοί του Συστήματος

Παρότι το σύστημα που αναπτύχθηκε απέδειξε στην πράξη την ικανότητά του να αυτοματοποιεί το πλήρες pipeline ενός penetration test, όπως επιβεβαιώθηκε από τα αποτελέσματα του Κεφαλαίου 7, κάθε υλοποίηση φέρει τους δικούς της περιορισμούς. Η αναγνώριση και ανάλυσή τους δεν αποτελεί

αδυναμία αλλά αναπόσπαστο μέρος της ερευνητικής διαδικασίας, καθώς θέτει τα θεμέλια για μελλοντική βελτίωση και επέκταση.

Ο πρώτος περιορισμός αφορά την εξάρτηση από την ποιότητα της αναγνώρισης. Το Wappalyzer, ως παθητικό εργαλείο fingerprinting, δεν εγγυάται πλήρη κάλυψη των τεχνολογιών ενός στόχου. Τεχνολογίες που αποκρύπτουν εσκεμμένα την ταυτότητά τους ή δεν περιλαμβάνονται στη βάση υπογραφών ενδέχεται να μην ανιχνευτούν, οδηγώντας σε ατελές προφίλ ευπαθειών και κατ' επέκταση σε ελλιπές Attack Plan.

Ο δεύτερος περιορισμός αφορά την ανομοιογένεια των αρχείων του ExploitDB. Όπως αναδείχθηκε στα ερευνητικά ερωτήματα, η επιτυχία του εναλλακτικού μονοπατιού εκτέλεσης εξαρτάται από την ποιότητα του διαθέσιμου exploit κώδικα, η οποία δεν είναι εγγυημένη. Κώδικας που δεν είναι πλήρως λειτουργικός ή απαιτεί σημαντική προσαρμογή ενδέχεται να οδηγήσει σε αποτυχία εκτέλεσης ακόμα και για ευπάθειες που είναι πραγματικά εκμεταλλεύσιμες.

Ο τρίτος περιορισμός αφορά το εύρος των υποστηριζόμενων επιθέσεων. Το σύστημα περιορίζεται σε επιθέσεις που υλοποιούνται μέσω HTTP requests, καλύπτοντας ένα σημαντικό αλλά όχι εξαντλητικό φάσμα ευπαθειών web εφαρμογών. Ευπάθειες που απαιτούν άλλα πρωτόκολλα ή άμεση πρόσβαση σε υποδομή παραμένουν εκτός πεδίου εφαρμογής.

Ο τέταρτος περιορισμός αφορά την αξιοπιστία του LLM ως λήπτη αποφάσεων. Παρότι το σύστημα guardrails περιορίζει σημαντικά την αυθαίρετη συμπεριφορά, ένα LLM μπορεί σε ορισμένες περιπτώσεις να παράγει μη αναμενόμενες ακολουθίες ενεργειών, ιδιαίτερα σε σενάρια που δεν καλύπτονται από το training του μοντέλου. Αυτός ο περιορισμός είναι εγγενής σε κάθε σύστημα βασισμένο σε LLM και δεν μπορεί να εξαλειφθεί πλήρως.

Ο χρόνος εκτέλεσης ανά φάση δεν καταγράφηκε συστηματικά στην παρούσα αξιολόγηση. Το σύστημα εμπλέκει πολλαπλά επίπεδα καθυστέρησης που δεν είναι εύκολο να αποδοθούν σε ενιαία μέτρηση: η φάση αναγνώρισης εξαρτάται από την απόκριση των εξωτερικών υπηρεσιών (Wappalyzer, FastCVE, ExploitDB), ενώ η φάση εκτέλεσης φέρει πρόσθετη επιβάρυνση καθώς σε κάθε βήμα του ReAct loop παρεμβάλλεται τόσο η καθυστέρηση του API του μοντέλου DeepSeek όσο και η καθυστέρηση εκτέλεσης του εκάστοτε εργαλείου, δύο πηγές καθυστέρησης που πολλαπλασιάζονται με τον αριθμό των tool calls ανά σενάριο. Επιπλέον, η απόκριση του μοντέλου ποικίλλει ανάλογα με τον φόρτο του DeepSeek API κατά τη στιγμή της εκτέλεσης και το μέγεθος του context που συσσωρεύεται κατά τη διάρκεια της συνεδρίας. Κατά συνέπεια, μια μεμονωμένη μέτρηση χρόνου δεν θα ήταν αντιπροσωπευτική, καθώς θα αντικατόπτριζε τις συνθήκες μιας συγκεκριμένης εκτέλεσης και όχι τη συμπεριφορά του συστήματος γενικότερα. Η συστηματική μέτρηση απόδοσης θα απαιτούσε επαναλαμβανόμενες εκτελέσεις υπό ελεγχόμενες συνθήκες με απομόνωση κάθε επιπέδου, και αποτελεί πεδίο μελλοντικής αξιολόγησης.

8.2 Ηθικές και Νομικές Διαστάσεις

Η ανάπτυξη και η χρήση συστημάτων αυτοματοποιημένου penetration testing εγείρει σημαντικά ηθικά και νομικά ζητήματα που αξίζει να αντιμετωπιστούν ρητά.

Ως προς το νομικό πλαίσιο, η εκτέλεση ελέγχων διείσδυσης χωρίς ρητή εξουσιοδότηση του ιδιοκτήτη του συστήματος αποτελεί παράνομη πράξη σε όλες τις έννομες τάξεις. Το σύστημα που αναπτύχθηκε στο πλαίσιο της παρούσας εργασίας δοκιμάστηκε αποκλειστικά σε ελεγχόμενα και εξουσιοδοτημένα περιβάλλοντα Docker containers. Η χρήση του σε μη εξουσιοδοτημένους στόχους είναι τόσο παράνομη όσο και αντίθετη με τις αρχές της υπεύθυνης αποκάλυψης (responsible disclosure).

Ως προς τις ηθικές διαστάσεις, η αυτοματοποίηση της επιθετικής ασφάλειας φέρνει στο προσκήνιο το ερώτημα της κατάχρησης. Ένα εργαλείο που μειώνει σημαντικά την τεχνική εμπειρία που απαιτείται για την εκτέλεση exploits μπορεί δυνητικά να χρησιμοποιηθεί από μη εξουσιοδοτημένους

χρήστες. Για τον λόγο αυτό, η ανάπτυξη τέτοιων συστημάτων συνοδεύεται από ευθύνη ως προς τον τρόπο διάθεσης και χρήσης τους. Στο πλαίσιο της παρούσας εργασίας, το σύστημα αναπτύχθηκε αποκλειστικά για ερευνητικούς σκοπούς και με πλήρη επίγνωση των παραπάνω διαστάσεων.

8.3 Μελλοντικές Επεκτάσεις

Το σύστημα που υλοποιήθηκε στο πλαίσιο της παρούσας εργασίας αποτελεί μια ολοκληρωμένη αλλά εξελίξιμη βάση. Τα αποτελέσματα της αξιολόγησης, σε συνδυασμό με τους περιορισμούς που εντοπίστηκαν, αναδεικνύουν αρκετές κατευθύνσεις για μελλοντική επέκταση που θα ενίσχυαν περαιτέρω τις δυνατότητές του.

Η πρώτη αφορά την ενίσχυση της φάσης αναγνώρισης με την ενσωμάτωση ενεργών εργαλείων σάρωσης όπως το Nmap, που θα επέτρεπαν εντοπισμό τεχνολογιών που διαφεύγουν από το Wappalyzer, μειώνοντας τον κίνδυνο ατελούς προφίλ ευπαθειών.

Η δεύτερη αφορά την αξιοποίηση του Nuclei ως αυτόνομου εργαλείου εκτέλεσης — αντί της τρέχουσας προσέγγισης όπου χρησιμοποιούνται μόνο τα templates για την κατασκευή αιτημάτων μέσω http_request — κάτι που θα απλοποιούσε τη ροή εκτέλεσης, θα αξιοποιούσε πλήρως τις δυνατότητες του εργαλείου και θα μείωνε την εξάρτηση από τα αρχεία του ExploitDB αδιευκρίνιστης αξιοπιστίας.

Η τρίτη αφορά την επέκταση του συστήματος με φάσεις post-exploitation, όπως η αυτόματη απαρίθμηση του συστήματος μετά από επιτυχή RCE. Αυτό θα επέτρεπε πιο ολοκληρωμένη αξιολόγηση του πραγματικού αντίκτυπου μιας ευπάθειας.

Η τέταρτη αφορά την αυτόματη παραγωγή τελικής έκθεσης (report) σε δομημένη μορφή, η οποία θα συγκέντρωνε όλα τα ευρήματα και θα πρότεινε μέτρα αντιμετώπισης. Αυτό θα καθιστούσε το σύστημα περισσότερο αυτόνομο και άμεσα χρήσιμο σε επαγγελματικό πλαίσιο.

Η πέμπτη αφορά τη δοκιμή του συστήματος με εναλλακτικά LLMs, όπως το GPT-4o ή το Claude, με σκοπό τη συγκριτική αξιολόγηση της απόδοσης διαφορετικών μοντέλων στο συγκεκριμένο πλαίσιο εφαρμογής.

9 Συμπεράσματα

Η παρούσα εργασία παρουσίασε τον σχεδιασμό, την υλοποίηση και την αξιολόγηση ενός αυτοματοποιημένου συστήματος penetration testing βασισμένου σε multi-agent αρχιτεκτονική — ενός συστήματος που αναπτύχθηκε εξ ολοκλήρου στο πλαίσιο της παρούσας εργασίας. Το σύστημα αξιοποιεί το Model Context Protocol και το μοντέλο DeepSeek για τον συντονισμό δύο εξειδικευμένων agents, υλοποιώντας ένα pipeline που καλύπτει το σύνολο των φάσεων ενός ελέγχου διείσδυσης, από την αναγνώριση του στόχου μέχρι την εκτέλεση και επιβεβαίωση των ευπαθειών.

Τα αποτελέσματα της αξιολόγησης σε δύο ελεγχόμενα περιβάλλοντα επιβεβαίωσαν ότι το σύστημα μπορεί να αυτοματοποιήσει αξιόπιστα το πλήρες pipeline, να διαχειριστεί διαφορετικούς τύπους ευπαθειών, από path traversal μέχρι απομακρυσμένη εκτέλεση κώδικα μέσω reverse shell, και να παράγει δομημένα και επαληθεύσιμα αποτελέσματα. Η μοναδική ανθρώπινη παρέμβαση που απαιτήθηκε και στα δύο σενάρια αφορούσε αποκλειστικά την επιλογή του CVE προς δοκιμή, αναδεικνύοντας τον βαθμό αυτονομίας που επιτυγχάνεται.

Η κεντρική αρχιτεκτονική επιλογή της εργασίας, ο διαχωρισμός σε Agent Ανάλυσης και Pentester Agent, αποδείχθηκε αποτελεσματική στην πράξη. Ο Agent Ανάλυσης εφάρμοσε συνεκτική λογική φιλτραρίσματος, απορρίπτοντας ευπάθειες χωρίς actionable exploit ανεξάρτητα από το CVSS score τους, ενώ ο Pentester Agent εκτέλεσε το Attack Plan ακολουθώντας τη σωστή ροή εκτέλεσης σε κάθε περίπτωση. Ο σχεδιαστικός διαχωρισμός παθητικής ανάλυσης και ενεργής εκτέλεσης λειτούργησε ως θεμελιώδης αρχή που εξασφάλισε την προβλεψιμότητα και την αξιοπιστία του συστήματος.

Η συνεισφορά της εργασίας δεν περιορίζεται στην υλοποίηση ενός λειτουργικού εργαλείου, αποδεικνύει ότι η αρχιτεκτονική multi-agent συστημάτων με LLM μπορεί να εφαρμοστεί αποτελεσματικά σε πολύπλοκες και δυναμικές εργασίες επιθετικής ασφάλειας, ανοίγοντας τον δρόμο για πιο προηγμένες και ολοκληρωμένες υλοποιήσεις στο μέλλον.

10 Βιβλιογραφικές Πηγές

[1] MITRE Corporation, "Common Vulnerabilities and Exposures," 2024. [Online]. Available: https://cve.mitre.org. Accessed: Oct. 2025.

[2] NIST, "National Vulnerability Database," 2024. [Online]. Available: https://nvd.nist.gov. Accessed: Oct. 2025.

[3] OWASP Foundation, "OWASP Web Security Testing Guide v4.2," 2021. [Online]. Available: https://owasp.org/www-project-web-security-testing-guide. Accessed: Nov. 2025.

[4] PTES Technical Guidelines, "Penetration Testing Execution Standard," 2014. [Online]. Available: http://www.pentest-standard.org. Accessed: Nov. 2025.

[5] AliasIO, "Wappalyzer — Technology Profiler," 2024. [Online]. Available: https://github.com/tomnomnom/wappalyzer. Accessed: Oct. 2025.

[6] Binare.io, "FastCVE — Fast CVE Search Tool," 2024. [Online]. Available: https://github.com/binareio/FastCVE. Accessed: Oct. 2025.

[7] Offensive Security, "Exploit Database," 2024. [Online]. Available: https://www.exploit-db.com. Accessed: Oct. 2025.

[8] VulnCheck, "go-exploit — Go Exploit Framework," 2024. [Online]. Available: https://github.com/vulncheck-oss/go-exploit. Accessed: Nov. 2025.

[9] ProjectDiscovery, "Nuclei — Fast and Customizable Vulnerability Scanner," 2024. [Online]. Available: https://github.com/projectdiscovery/nuclei. Accessed: Nov. 2025.

[10] Anthropic, "Model Context Protocol," 2024. [Online]. Available: https://modelcontextprotocol.io. Accessed: Dec. 2025.

[11] DeepSeek, "DeepSeek Language Model," 2024. [Online]. Available: https://www.deepseek.com. Accessed: Jan. 2026.

[12] S. Yao et al., "ReAct: Synergizing Reasoning and Acting in Language Models," in *Proc. ICLR*, 2023. [Online]. Available: https://arxiv.org/abs/2210.03629. Accessed: Dec. 2025.

[13] T. Schick et al., "Toolformer: Language Models Can Teach Themselves to Use Tools," in *Proc. NeurIPS*, 2023. [Online]. Available: https://arxiv.org/abs/2302.04761. Accessed: Dec. 2025.

[14] L. Wang et al., "A Survey on Large Language Model based Autonomous Agents," *Frontiers of Computer Science*, 2024. [Online]. Available: https://arxiv.org/abs/2308.11432. Accessed: Dec. 2025.

[15] D. Deng et al., "PentestGPT: An LLM-empowered Automatic Penetration Testing Tool," in *Proc. USENIX Security Symposium*, 2024. [Online]. Available: https://arxiv.org/abs/2308.06782. Accessed: Nov. 2025.

[16] 0x4m4, "HexStrike AI — AI-Powered Penetration Testing," 2024. [Online]. Available: https://github.com/0x4m4/hexstrike-ai. Accessed: Jan. 2026.

[17] Docker Inc., "Docker — Containerization Platform," 2024. [Online]. Available: https://www.docker.com. Accessed: Oct. 2025.

[18] Vulhub, "Vulhub — Pre-Built Vulnerable Docker Environments," 2024. [Online]. Available: https://github.com/vulhub/vulhub. Accessed: Feb. 2026.

[19] Postman Inc., "Postman — API Platform," 2024. [Online]. Available: https://www.postman.com. Accessed: Nov. 2025.

[20] Python Software Foundation, "Python Programming Language," 2024. [Online]. Available: https://www.python.org. Accessed: Oct. 2025.

[21] YAML, "YAML — YAML Ain't Markup Language," 2024. [Online]. Available: https://yaml.org. Accessed: Nov. 2025.