Hacking & Internet Security

Γιώργος Γκίρτσος

Κυριακή 20 Μαΐου 2007

Σκοπός της παρουσίασης

- Σε αυτή την παρουσίαση θα σας ενημερώσουμε για το hacking, τις τεχνικές του hacking, αλλά και πως μπορείτε να προστατεύσετε μία εφαρμογή από κακόβουλους χρήστες.
- Αρχικά, θα κατανοήσουμε τον όρο hacking και hacker και θα σας δείξουμε πόσο εύκολο είναι να «εισβάλλετε» σε μία εφαρμογή/σύστημα αν ο server δεν είναι σωστά ρυθμισμένος.
- Θα ενημερωθείτε για όλες τις τεχνικές του hacking και θα κάνουμε αναλυτική αναφορά για κάθε τεχνική. Επίσης θα σας πούμε πώς μπορείτε να ρυθμίσετε τον server σας σωστά ώστε να μην κινδυνεύετε από κακόβουλες επιθέσεις.
- Θα μάθουμε ποιες τεχνικές ασφαλείας πρέπει να χρησιμοποιούμε για να κάνουμε μία ασφαλή εφαρμογή και φυσικά θα δούμε παραδείγματα κώδικα για να καταλάβουμε τι πρέπει να αποφεύγουμε.

Προσοχή!

- Η παρουσίαση δημιουργήθηκε αποκλειστικά με σκοπό την ενημέρωσή σας για τις τεχνικές του hacking και πώς να ρυθμίσετε τον server σας ώστε να είναι πιο ασφαλής. Δεν εγγυόμαστε πως αν ακολουθήσετε πιστά τις οδηγίες μας δεν κινδυνεύετε καθόλου, διότι πάντα υπάρχουν τρόποι για να «εισβάλετε» σε μια εφαρμογή.

- **Δεν φέρουμε καμία ευθύνη σε περίπτωση που χρησιμοποιήσετε τις τεχνικές αυτές εναντίον κάποιου δικτυακού τόπου. Χρησιμοποιήστε αυτές τις τεχνικές hacking με δική σας ευθύνη!**

Hacking & Hacker

Το hacking αποτελεί σήμερα μία έννοια πολύ παρεξηγημένη. Οι περισσότεροι (σχετικοί και μη) έχοντας ή μη πρόσβαση στο διαδίκτυο είναι της άποψης ότι ο χάκερ είναι ένας υποχθόνιος τύπος που περνάει τις 20 απ' τις 24 ώρες της ημέρας του μπροστά στον υπολογιστή του, προσπαθώντας να σπάσει κωδικούς και να σπείρει καταστροφή. Οι περισσότεροι πιστεύουν ότι το hacking γενικά είναι η τέχνη της καταστροφής.

Η αιτία είναι ο κινηματογράφος και η τηλεόραση. Η κακή πληροφόρηση δηλαδή. Η απόκτηση τόσης γνώσης (όπου όποιος είναι άσχετος με τον χώρο, δεν μπορεί να διανοηθεί τι ξέρουν και τι δεν ξέρουν οι χάκερς) χρησιμοποιείται για να καταστρέψει ή να γελοιοποιήσει.

Ο χάκερ είτε δουλεύει μόνος, είτε σε ομάδες: τα hacking groups. Έχει γίνει μεγάλη σύγχυση του χάκερ με τον κράκερ (cracker) ή τον «πειρατή» που ξεκλειδώνει commercial προγράμματα και μετά τα πουλάει έναντι αδράς αμοιβής. Καμία σχέση. Στο Διαδίκτυο κυκλοφορούν πολλές απόψεις. Στο hacker.org, διαβάζουμε ότι «Ο χάκερ πρέπει να ~εισβάλλει~ σε έναν server, να βρίσκει τα κενά ασφαλείας και μετά να ειδοποιεί τον system administrator για να τα διορθώσει». Στην πραγματικότητα, πολύ λίγοι το κάνουν αυτό. Γι' αυτό και το hacking έχει πάρει κακό όνομα.

Σύμφωνα με το παραπάνω site, ο σωστός χάκερ δουλεύει αθόρυβα, υπόγεια και δεν βγαίνει να κομπάσει ότι «διέλυσα το τάδε σύστημα ή ότι έσβησα το τάδε σύμπαν». Άρα, hacking είναι η γνώση που χρησιμοποιείται για καλό σκοπό αλλά και ενίοτε για εκδικητικούς σκοπούς.

Το παραπάνω κείμενο βρέθηκε στον δικτυακό τόπο «Kybernografoi Magazine».
Αν θέλετε να διαβάσετε περισσότερα για τους hackers και τους crackers, μπορείτε να ρίξετε μια ματιά στην ιστοσελίδα του Κέντρου ΠΛΗΝΕΤ Φλώρινας.

Εισαγωγή στις τεχνικές του hacking

- Όσοι δεν γνωρίζετε τι είναι η PHP, είναι μία γλώσσα που χρησιμοποιείται αποκλειστικά για προγραμματισμό στο Internet. Όταν ο χρήστης προβάλει την σελίδα, ο server (κεντρικός υπολογιστής δικτύου) στέλνει την σελίδα στον γλωσσικό διερμηνέα και έτσι ο κώδικας μπορεί να εκτελεστεί. Αμέσως μετά η σελίδα επιστρέφεται στον χρήστη. Επίσης, δεν μπορούμε να δούμε τον κώδικα μιας σελίδας PHP διότι έχει γίνει η μεταγλώττιση και εμείς βλέπουμε το αποτέλεσμα.

Τα χαρακτηριστικά της PHP:

- → Γρήγορη Εκτέλεση.
- → Ενσωματωμένος μεταγλωττιστής.
- → Πλήθος προκαθορισμένων εντολών και συναρτήσεων.
- → Εύκολη στη σύνταξη.

- Όλες οι τεχνικές που περιγράφονται σε αυτή την παρουσίαση πραγματοποιήθηκαν στην προεπιλεγμένη εγκατάσταση της PHP (έκδοση: 4.0.4pl1) με mySQL, PostgreSQL, IMAP και ενεργοποιημένη υποστήριξη για OpenSSL. Όλα αυτά ήταν στο πακέτο του Apache Web Server (έκδοση: 1.3.19) σε ένα Linux μηχάνημα.

- Αυτό σημαίνει ότι κάποιες από τις τεχνικές που θα παρουσιαστούν *πιθανόν* να μην δουλεύουν αν ο server είναι καλά ρυθμισμένος ή η έκδοση της PHP είναι νεότερη.

Εισαγωγή στις Γενικές Μεταβλητές

Η εμβέλεια των μεταβλητών καθορίζεται από το περιεχόμενο μέσα στο οποίο ορίζεται. Για την πλειοψηφία των μεταβλητών της PHP υπάρχει μόνο ενός είδους εμβέλεια. Αυτή η εμβέλεια περιέχει τόσο τα αρχεία που περιέχονται όσο και αυτά που απαιτούνται.

Για παράδειγμα:

```
<?php
$a = 1;
include "b.inc";
?>
```

Εδώ η μεταβλητή $a θα είναι διαθέσιμη μέσα στο εμπεριεχόμενο b.inc script. Πάντως, μέσα σε συναρτήσεις που ορίζονται από το χρήστη εισάγεται εμβέλεια τοπικής συνάρτησης (local function scope). Οποιαδήποτε μεταβλητή χρησιμοποιείται μέσα σε μια συνάρτηση είναι εκ των προτέρων περιορισμένη σε εμβέλεια τοπικής συνάρτησης.

Για παράδειγμα:

```
<?php
$a = 1; /* γενικός σκοπός */

function Test()
{
   echo $a; /* αναφέρεται σε μεταβλητή με τοπική εμβέλεια */
}

Test();
?>
```

Εισαγωγή στις Γενικές Μεταβλητές

[συνέχεια]

Αυτό το script δε θα δώσει κάποιο αποτέλεσμα επειδή η echo δήλωση αναφέρεται σε μια τοπική έκδοση της $a μεταβλητής, και δεν της έχει ανατεθεί μια τιμή μέσα σ' αυτή την εμβέλεια. Ίσως παρατηρήσετε ότι είναι λίγο διαφορετική από τη C γλώσσα από την άποψη ότι οι global μεταβλητές στη C είναι άμεσα διαθέσιμες σε συναρτήσεις εκτός και αν επικαλύπτονται αυτόματα από κάποια τοπική αναφορά. Αυτό μπορεί να προκαλέσει μερικά προβλήματα αν κάποιοι, ακούσια, αλλάξουν μια global μεταβλητή. Στην PHP οι global μεταβλητές πρέπει να οριστούν ως global μέσα στη συνάρτηση αν πρόκειται να χρησιμοποιηθούν σ' αυτή τη συνάρτηση.

Ένα παράδειγμα:

```
<?php
$a = 1;
$b = 2;

function Sum()
{
  global $a, $b;
  $b = $a + $b;
}

Sum();
echo $b;

?>
```

Το παραπάνω script θα εμφανίσει το αποτέλεσμα «3». Δηλώνοντας την $a και την $b ως global μέσα σε μια συνάρτηση, όλες οι αναφορές σε οποιαδήποτε από τις μεταβλητές θα αναφέρονται στην global έκδοση. Δεν υπάρχει περιορισμός στον αριθμό των global μεταβλητών που μπορεί να χειριστεί μια συνάρτηση.

Γενικές Μεταβλητές

Οι μεταβλητές στην PHP δεν χρειάζεται να δηλωθούν μέσα στον κώδικα αφού μπορούν να δημιουργηθούν την πρώτη φορά που θα χρησιμοποιηθούν. Όταν δημιουργηθεί η μεταβλητή μπορεί να χρησιμοποιηθεί οπουδήποτε μέσα στο πρόγραμμα (εκτός από τις συναρτήσεις που πρέπει να ζητήσουμε από την PHP να περάσει μία μεταβλητή μέσα σε μία συνάρτηση χρησιμοποιώντας την συνάρτηση global).

Τελικά, όταν δημιουργούνται είναι κενές (δηλ. ""). Συνήθως ο κύριος σκοπός μίας εφαρμογής γραμμένης σε PHP είναι να πάρει κάποια δεδομένα από τον χρήστη (στοιχεία φόρμας, «ανέβασμα» αρχείων, cookies, περίοδοι εργασίας κλπ), να επεξεργαστεί τα δεδομένα και να επιστρέψει το αποτέλεσμα. Προκειμένου να το καταστήσει όσο το δυνατόν απλούστερο για την PHP, παρέχει την χρήση των global μεταβλητών.

Ας υποθέσουμε ότι έχουμε τον παρακάτω HTML κώδικα:

```
1.  <form method="GET" action="test.php">
2.  <input type="text" name="hello">
3.  <input type="submit">
4.  </form>
```

Ο παραπάνω κώδικας θα εμφανίσει ένα text box και ένα κουμπί υποβολής. Όταν υποβληθεί η φόρμα θα στείλει το αίτημα στο αρχείο test.php για να επεξεργαστεί τα στοιχεία. Όταν εκτελεστεί το αρχείο η μεταβλητή $hello θα περιέχει το κείμενο που έγραψε ο χρήστης στο text box.

Είναι πολύ σημαντικό να κατανοήσετε τις επιπτώσεις αυτής της φόρμας. Ένας κακόβουλος χρήστης μπορεί να ορίσει όσες μεταβλητές θέλει. Επίσης αντί να χρησιμοποιήσει την παραπάνω φόρμα, μπορεί να δώσει ένα url σαν αυτό: http://server/test.php?hello=hi&setup=no. Αυτό όχι μόνο θα έδινε την τιμή «hi» στο $hello, αλλά θα έδινε και την τιμή «no» στην μεταβλητή $setup.

Γενικές Μεταβλητές

[συνέχεια]

Ένα παράδειγμα ως προς το πως αυτό μπορούσε να είναι πρόβλημα θα ήταν ένα πρόγραμμα που σχεδιάστηκε για να κάνει έλεγχο αν ο χρήστης έχει κάνει είσοδο στο site (authentication script) πριν την εμφάνιση σημαντικών πληροφοριών.

Για παράδειγμα:

```
<?php
if ($pass == "hello")
$auth = 1;
...
if ($auth == 1)
echo "μερικές σημαντικές πληροφορίες";
?>
```

Σε κανονική λειτουργία ο παραπάνω κώδικας θα ελέγξει τον κωδικό για να αποφασίσει αν ο χρήστης έχει εισέλθει στο site. Αν έχει εισέλθει ο χρήστης, τότε εμφανίζει τις σημαντικές πληροφορίες. Το πρόβλημα είναι ότι ο κώδικας υποθέτει ότι η μεταβλητή $auth θα είναι κενή εκτός αν έχει οριστεί. Θυμηθείτε πως ο κακόβουλος χρήστης μπορεί να δημιουργήσει μεταβλητές με ένα URL σαν αυτό: http://server/test.php?auth=1

Το παραπάνω URL θα αποτύχει στον έλεγχο του κωδικού αλλά το script θα πιστεύει ότι ο χρήστης έχει εισέλθει επιτυχώς στο site, με αποτέλεσμα την εμφάνιση των σημαντικών πληροφοριών!

Για να συνοψίσουμε, ένα PHP script δεν μπορεί να εμπιστευτεί ΚΑΜΙΑ μεταβλητή ότι έχουν την τιμή που πρέπει (για μεταβλητές που προέρχονται από εξωτερικές πηγές). Όταν έχετε πολλές μεταβλητές, αυτό μπορεί να είναι πολύ πιο δύσκολο απ’ ότι φαίνεται.

Γενικές Μεταβλητές

[συνέχεια]

Η πιο κοινή προσέγγιση στην προστασία ενός script είναι να ελέγξουμε αν η μεταβλητή δεν είναι στο array GET/POST[] (εξαρτάται από την μέθοδο που χρησιμοποιήθηκε για να υποβληθεί η φόρμα, GET ή POST). Όταν η PHP έχει την ρύθμιση track_vars ενεργή (όπως είναι και προεπιλεγμένο) οι μεταβλητές που υποβλήθηκαν από τον χρήστη είναι διαθέσιμες και σαν global μεταβλητές αλλά και ως στοιχεία στα array GET/POST[].

Εντούτοις είναι σημαντικό να θυμόμαστε ότι υπάρχουν 4 διαφορετικά arrays για αποθήκευση στοιχείων από τους χρήστες. Τα arrays είναι:
$_GET[] → για την άντληση δεδομένων από το URL
$_POST[] → για την άντληση δεδομένων από φόρμες που έχουν υποβληθεί με την μέθοδο POST
$_COOKIES[] → για μεταβλητές που υποβλήθηκαν ως μέρος των headers μέσα στο HTTP request
$_FILES[] → που αντλεί πληροφορίες σχετικά με το «ανέβασμα» αρχείων.

Είναι αποκλειστικά επιλογή του χρήστη ποια μέθοδο θα χρησιμοποιήσει για να ορίσει μεταβλητές. Ένα αίτημα μπορεί να εισάγει μεταβλητές και στα 4 arrays. Ένα ασφαλή script πρέπει να ελέγξει και τα 4 arrays (το $_FILES[] δεν είναι πρόβλημα, εκτός από ορισμένες περιπτώσεις).

Χρήση Αρχείων

Θα το αναφέρω πολλές φορές σε αυτή την παρουσίαση: Η PHP είναι μία «πλούσια» γλώσσα με ένα πλήθος ενσωματωμένων εντολών και προκαθορισμένων συναρτήσεων. Επίσης, προσπαθεί σκληρά να καταστήσει την ζωή του προγραμματιστή όσο το δυνατόν ευκολότερη.

Ένα παράδειγμα, είναι ο τρόπος που η χειρίζεται τα αρχεία. Το ακόλουθο script είναι σχεδιασμένο για να ανοίγει ένα αρχείο:

```
<?php
if (! ($fd = fopen($filename, "r") )
{
  echo ("Αδύνατο το άνοιγμα του αρχείου: {$filename}<br />\n");
}
?>
```

Ο κώδικας προσπαθεί να ανοίξει ένα αρχείο για ανάγνωση και αν αποτύχει, εμφανίζει ένα σφάλμα. Το όνομα του αρχείου είναι αποθηκευμένο στην μεταβλητή $filename. Προφανώς αυτό είναι ένα κενό ασφαλείας αν ο χρήστης μπορεί να ορίσει την μεταβλητή $filename και να αναγκάσει το script να αποκαλύψει τα δεδομένα του φακέλου /etc/passwd (όπου εκεί είναι αποθηκευμένος ο κωδικός του server).

Οι συναρτήσεις για την διαχείριση αρχείων μπορούν να λειτουργήσουν και για απομακρυσμένα αρχεία μέσω HTTP και FTP. Για παράδειγμα αν η μεταβλητή $filename είχε την τιμή http://target/scripts/..%c1%1c../winnt/system32/cmd.exe?/c+dir η PHP θα έκανε ένα HTTP αίτημα στον server-στόχο. Σε αυτή την περίπτωση θα προσπαθούσε να εκμεταλλευτεί το κενό ασφαλείας unicode.

Χρήση Αρχείων

[συνέχεια]

Αυτό γίνεται πολύ πιο ενδιαφέρον όταν χρησιμοποιηθεί μέσα στο script μία από τις 4 συναρτήσεις include(), require(), include_once() ή require_once(). Αυτές οι συναρτήσεις ανοίγουν το αρχείο και εκτελούν τον κώδικά του μέσα στο τρέχον script! Τυπικά, χρησιμοποιούνται για την υποστήριξη βιβλιοθηκών κώδικα, όπου τα κοινά κομμάτια κώδικα είναι αποθηκευμένα και χρησιμοποιούνται όταν είναι απαραίτητο.

Τώρα, ας ρίξουμε μια ματιά στον παρακάτω κώδικα:

```
<?php
include($libdir . "/languages.php");
?>
```

Προφανώς η μεταβλητή $libdir είναι μία μεταβλητή ρυθμίσεων, που σημαίνει ότι πρέπει να έχει δηλωθεί σε ένα σημείο του script. Η μεταβλητή $libdir δείχνει που είναι αποθηκευμένα τα αρχεία του κώδικα. Αν ο κακόβουλος χρήστης μπορεί να αναγκάσει την μεταβλητή να αλλάξει στο script (κάτι που δεν είναι χαρακτηριστικά τρομερός στόχος), τότε θα μπορέσουν να τροποποιήσουν την διαδρομή του καταλόγου της βιβλιοθήκης. Κάνοντας κάτι τέτοιο δεν θα κέρδιζαν τίποτα, δεδομένου του ότι καταλήγουν στο languages.php.

Αν όμως το αρχείο languages.php έχει τοποθετηθεί σε κάποιον server που ανήκει στον κακόβουλο χρήστη και περιείχε τον ακόλουθο κώδικα:

```
<?php
passthru("/bin/ls /etc");
?>
```

Θα μπορούσε να εκτελέσει την παραπάνω εντολή (θα αποκάλυπτε τα αρχεία του καταλόγου /etc).

Upload Files

Η PHP είναι μία γλώσσα αρκετά επικίνδυνη για scripting διότι υπάρχουν πολλοί τρόποι να «εισβάλλουμε» σε μία εφαρμογή, όπως είδαμε μέχρι τώρα. Επίσης κάνει την ζωή του attacker πιο εύκολη διότι παρέχει υποστήριξη για ανέβασμα αρχείων για RFC 1867.

Ας υποθέσουμε ότι έχουμε τον παρακάτω HTML κώδικα:

```
<FORM METHOD="POST" ENCTYPE="multipart/form-data">
<INPUT TYPE="FILE" NAME="hello">
<INPUT TYPE="HIDDEN" NAME="MAX_FILE_SIZE" VALUE="10240">
<INPUT TYPE="SUBMIT">
</FORM>
```

Η φόρμα θα επιτρέψει στον χρήστη να επιλέξει ένα αρχείο από τον υπολογιστή του και μετά την υποβολή της φόρμας, το αρχείο θα τοποθετηθεί στον server (upload file). Είναι σίγουρα πολύ χρήσιμο, αλλά ο τρόπος που το χειρίζεται η PHP είναι που το κάνει επικίνδυνο.

Όταν η PHP δέχεται το αίτημα, πριν ακόμα ΑΡΧΙΣΕΙ να εκτελεί τον κώδικα, το script θα λάβει το αρχείο αυτόματα από τον χρήστη και μετά θα ελέγξει αν το μέγεθος του αρχείου είναι μεγαλύτερο από την μεταβλητή $MAX_FILE_SIZE (10kb σε αυτή την περίπτωση). Αμέσως μετά θα κοιτάξει στο αρχείο ρυθμίσεων της PHP για να ελέγξει το επιτρεπόμενο μέγεθος για το file uploading (η προκαθορισμένη τιμή είμαι 2MB). Μέχρι να περάσει αυτούς τους ελέγχους, το αρχείο αποθηκεύεται στον server σε έναν προσωρινό κατάλογο.

Παρακαλώ διαβάστε το ξανά αυτό αν δεν το καταλάβατε γιατί είναι πολύ σημαντικό: Ο χρήστης μπορεί να στείλει ένα οποιοδήποτε αρχείο σε ένα μηχάνημα που υποστηρίζει PHP και πριν το script αποφασίσει αν θα αποδεχθεί το αρχείο, το αποθηκεύει στον σκληρό δίσκο του server σε έναν προσωρινό κατάλογο!

Upload Files

[συνέχεια]

Δεν θα αναλύσουμε παραπάνω τις επιθέσεις στα upload systems διότι κατά τη γνώμη μου οι «ευκαιρίες» για επίθεση σε ένα upload system είναι τελείως περιορισμένες. Αρχικά, ας υποθέσουμε πως αυτό το script είναι σχεδιασμένο για να λαμβάνει αρχεία. Όπως περιγράψαμε και προηγουμένως, το αρχείο αποθηκεύεται στον server (στον κατάλογο που έχει οριστεί από το αρχείο ρυθμίσεων – συνήθως είναι ο κατάλογος /tmp), με ένα τυχαίο όνομα (πχ: «phpxXuoXG»).

Το PHP script χρειάζεται πληροφορίες σχετικά με το αρχείο που αποθηκεύτηκε στον server για να μπορέσει να το χειριστεί. Αυτό ουσιαστικά γίνεται με δύο τρόπους: Ο ένας χρησιμοποιούνταν στις πρώτες εκδόσεις της PHP 3 και ο άλλος τρόπος εισήχθη μετά από προτάσεις προγραμματιστών.

Το πρόβλημα είναι πως πολλές εφαρμογές χρησιμοποιούν ακόμη τον παλιό τρόπο, ο οποίος είναι λίγο επικίνδυνος. Η PHP ορίζει τέσσερις γενικές μεταβλητές για να περιγράψει το αρχείο που αποθηκεύτηκε.

Για παράδειγμα (από την προηγούμενη upload form μας):

$hello = Όνομα αρχείου (το όνομα του αρχείου στον server [πχ. «/tmp/phpxXuoXG»])
$hello_size = Το μέγεθος του αρχείου σε bytes (πχ: 1024)
$hello_name = Το αρχικό όνομα του αρχείου από τον υπολογιστή του χρήστη (πχ: «C:\\temp\hello.txt»)
$hello_type] = Τύπος MIME του αρχείου (πχ: «text/plain»)

Upload Files

[συνέχεια]

Ύστερα το script επεξεργάζεται το αρχείο που είναι αποθηκευμένο στον server. Αυτό ακριβώς είναι το πρόβλημα: δεν είναι αμέσως ορατό ότι η μεταβλητή $hello πρέπει να έχει οριστεί από τον προγραμματιστή, αλλά πολύ εύκολα μπορεί να οριστεί από τον attacker. Ας δούμε το παρακάτω URL:
file.php?hello=/etc/passwd&hello_size=10240&hello_type=text/ plain&hello_name=hello.txt

Αυτό δημιουργεί τις παρακάτω γενικές μεταβλητές:

```
$hello = "/etc/passwd";
$hello_size = 10240;
$hello_type = "text/plain";
$hello_name = "hello.txt";
```

Αυτή η φόρμα θα ορίσει ακριβώς τις μεταβλητές που το script περιμένει να οριστούν από την PHP, αλλά αντί να δουλεύει στην αποθήκευση του αρχείου, θα επεξεργάζεται τον κατάλογο /etc/passwd (συνήθως καταλήγει στην αποκάλυψη των δεδομένων του). Αυτού του είδους η επίθεση μπορεί να εμφανίσει τα δεδομένα όλων των κρυφών αρχείων (όπως και αρχεία ρυθμίσεων με στοιχεία σύνδεσης σε μία βάση δεδομένων).

Παραπάνω ανέφερα πως η PHP παρέχει διαφορετικούς τρόπους για την διαχείριση στο ανέβασμα των αρχείων. Αυτός ο τρόπος είναι το $_FILES[] array που αναφέρθηκε νωρίτερα. Επίσης παρέχει ένα πλήθος συναρτήσεων για να αποφύγουμε αυτό το πρόβλημα, για παράδειγμα μια συνάρτηση που θα ελέγξει αν ένα αρχείο υπάρχει ήδη στον server.

Ως μία εναλλακτική επίθεση σχετικά με το file uploading, ας κοιτάξουμε τον παρακάτω κώδικα:

```
if (file_exists($theme)) include($theme);
```

Upload Files

[συνέχεια]

Αν ο attacker μπορεί να ελέγξει την μεταβλητή $theme, τότε προφανώς αυτό μπορεί να κάνει include αρχεία από οποιοδήποτε σύστημα (φυσικά το αρχείο που θα βρίσκεται στον server του attacker δεν πρέπει να περιέχει τα tags <?php και ?> διότι θα δημιουργήσει σφάλμα στο script κατά την μεταγλώττιση του κώδικα). Καθώς αυτό είναι πρόβλημα, ο στόχος των attackers είναι να εκτελέσουν εντολές οι οποίες θα αποκαλύψουν κρυφά αρχεία ή θα προκαλέσουν ζημιά στον server.

Επομένως πρέπει να διαβάσουν τον κώδικα PHP του script. Αυτό φαίνεται ακατόρθωτο, αλλά εδώ είναι που έρχεται το file uploading. Αν ο attacker δημιουργήσει ένα αρχείο στον server του που περιέχει PHP κώδικα για να εκτελεστεί (για παράδειγμα η εντολή passthru που είδαμε νωρίτερα), δημιουργήσει μία φόρμα που περιέχει ένα πεδίο με όνομα "theme" και χρησιμοποιήσει την φόρμα για να τοποθετήσει το αρχείο στον server χρησιμοποιώντας το file uploading, τότε θα μπορέσει να εμφανίσει είτε τον κώδικα των αρχείων που επιθυμεί είτε να κάνει άλλου είδους ζημιές.

Η PHP θα είναι αρκετά ευγενική, έτσι ώστε να αποθηκεύσει το αρχείο και να θέσει την μεταβλητή $theme στην διαδρομή που επιθυμεί ο attacker.

Έχοντας δώσει την δυνατότητα εκτέλεσης εντολών στον server, ο attacker προφανώς θα θέλει να αποκτήσει πρόσβαση στον server ως διαχειριστής για να κάνει επιθέσεις σε άλλους servers ή να καταστρέψει τον δικό μας server, για κάτι το οποίο θα χρειαστεί κάποια εργαλεία.

Η λειτουργία τοποθέτησης αρχείων σε έναν server, κάνει αυτό το μηδαμινό κενό ασφαλείας ένα πολύ μεγάλο κενό ασφαλείας δίνοντας στον attacker την δυνατότητα να ανεβάσει όλα τα εργαλεία που χρειάζεται, και ύστερα με την χρήση της εντολής chmod() θα μπορούσε να εκτελέσει τον κώδικα των αρχείων και να δημιουργήσει σοβαρά προβλήματα στον server. Για παράδειγμα θα μπορούσαν να γίνουν διαχειριστές ενός server ή να κλείσουν το firewall (τείχος προστασίας) – κάτι που σίγουρα δεν θέλουμε.

Αρχεία Βιβλιοθήκης

Αναφέραμε τις συναρτήσεις include() και require(). Επίσης θα ήθελα να σας υπενθυμίσω πως η χρήση αυτών των συναρτήσεων είναι για να συμπεριλάβουμε κομμάτια κώδικα σε ένα php αρχείο μας. Αυτό γίνεται όταν χρησιμοποιούμε κοινό κώδικα σε κάποια αρχεία.

Συγκεκριμένα, όταν οι προγραμματιστές ξεκίνησαν την ανάπτυξη της PHP, χρησιμοποιούσαν στις εφαρμογές τους αρχεία βιβλιοθήκης με την επέκταση ".inc". Σύντομα κατάλαβαν πως αυτή η τακτική είναι λάθος διότι αρχεία με τέτοια επέκταση δεν περνούσαν από μεταγλώττιση και ως αποτέλεσμα, εμφανιζόταν ο κώδικας των αρχείων!

Αυτό γινόταν γιατί η PHP μεταγλώττιζε αρχεία μόνο με τις ακόλουθες επεκτάσεις:
.php, .php3, .php4.

Ήταν πραγματικά πολύ μεγάλο πρόβλημα, κυρίως όταν ευαίσθητα δεδομένα (όπως στοιχεία σύνδεσης με την βάση δεδομένων) ήταν τοποθετημένα σε τέτοια αρχεία. Η πιο απλή λύση (η οποία είναι και η προτιμώμενη), είναι να δώσουν σε ΟΛΑ τα αρχεία την επέκταση .php. Έτσι ο server δεν επιστρέφει τον κώδικα του αρχείου, αλλά εκτελεί την σελίδα όπως όλες τις άλλες σελίδες PHP.

Αρχεία Βιβλιοθήκης

[συνέχεια]

Ας δούμε ένα παράδειγμα:

Αρχείο **main.php**:

```
<?php
$libDir = "/libdir";
$langDir = "{$libdir}/languages";
...
include("{$libdir}/loadlanguage.php");
?>
```

Αρχείο **libdir/loadlanguage.php**:

```
<?php
...
include("{$langDir}/{$userLang}");
?>
```

Όταν καλέσουμε το αρχείο libdir/loadlanguage.php μέσα από το main.php, δεν θα έχουμε κανένα πρόβλημα. Αν όμως το αρχείο κληθεί μόνο του (χωρίς να το έχουμε κάνει include από το main.php), τότε ο attacker μπορεί να ορίσει τις τιμές των μεταβλητών $langDir και $userLang σε ότι επιθυμεί.

Αρχεία Sessions

Οι εκδόσεις της PHP 4 (και άνω) παρέχουν ενσωματωμένη λειτουργία για περιόδους εργασίας (sessions). Ο κύριος σκοπός των sessions είναι να αποθηκεύσουν πληροφορίες και να τις μεταφέρουν από σελίδα σε σελίδα σε μια PHP εφαρμογή. Για παράδειγμα, όταν ένας χρήστης εισέρχεται ως εγγεγραμμένο μέλος σε μία σελίδα, το γεγονός ότι έχει κάνει είσοδο μπορεί να αποθηκευτεί στα sessions (όπως όνομα χρήστη, ID χρήστη κλπ).

Κατά την πλοήγηση του χρήστη στο site, αυτές οι πληροφορίες είναι διαθέσιμες σε όλες τις σελίδες. Αυτό που ακριβώς συμβαίνει είναι πως όταν ένα session ξεκινάει[1] ένα τυχαίο ID για την περίοδο εργασίας δημιουργείται και η περίοδος εργασίας δεν διαγράφεται αν ο χρήστης κάνει κλικ μέσα στην σελίδα[2]. Φυσικά αυτό μπορούμε να το πετύχουμε πολύ πιο εύκολα με την χρήση cookies.

Το session είναι μια μεταβλητή αποθήκευσης και ένα PHP script μπορεί να ορίσει μια μεταβλητή που η τιμή της αποθηκεύεται στο αρχείο του session.

Υποσημειώσεις:

[1]: Στο αρχείο ρυθμίσεων της PHP υπάρχει η επιλογή αυτόματης δημιουργίας session. Αν είναι ενεργοποιημένη, προτρέπει την PHP να ξεκινήσει ένα session (δηλαδή αυτόματα, χωρίς να έχουμε βάλει session_start() στον κώδικα).

[2]: Αν ο χρήστης συνεχίσει να κάνει κλικ μέσα στην σελίδα, το session δεν θα διαγραφεί, αλλά θα παραμείνει στον server. Στο αρχείο ρυθμίσεων της PHP υπάρχει η επιλογή που ορίζει το χρονικό όριο διάρκειας ενός session αν ο χρήστης δεν έχει κάνει κλικ πουθενά μέσα στο site. Η προεπιλεγμένη τιμή του είναι 15 λεπτά.

Αρχεία Sessions

[συνέχεια]

Ας δούμε ένα παράδειγμα:

```
<?php

session_start();

$session_auth = "my_session";
session_register("session_auth");

?>
```

Το παραπάνω παράδειγμα θα δημιουργήσει ένα session με τιμή την τιμή της μεταβλητής $session_auth (δηλ. my_session). Κατά την πλοήγηση του χρήστη στο site, αν ένα script αλλάξει την τιμή ενός session, τότε οι υπόλοιπες PHP σελίδες θα δεχθούν την τροποποιημένη τιμή του session.

Είναι πραγματικά πολύ χρήσιμο, αλλά το πρόβλημα είναι ότι αρκετά PHP scripts δεν ελέγχουν την μεταβλητή αν ήρθε όντως από το session. Ας υποθέσουμε ότι το αρχείο με τον παραπάνω κώδικα είναι το αρχικό μας αρχείο και έχουμε ένα ακόμη αρχείο με τον παρακάτω κώδικα:

```
<?php
if (!empty($session_auth)) // Πρόσβαση στο site ως διαχειριστής!
…
?>
```

Αυτός ο κώδικας υποθέτει πως το $session_auth έχει οριστεί από την PHP και όχι από τον χρήστη. Αν ο attacker μπορούσε να ορίσει την μεταβλητή $session_auth σε ένα πεδίο φόρμας, τότε θα είχε πρόσβαση στο site ως διαχειριστής.

Αρχεία Sessions

[συνέχεια]

Τα δεδομένα των sessions αποθηκεύονται σε αρχεία (session files) και τοποθετούνται στον κατάλογο /tmp[1]. Το όνομα ενός αρχείου session είναι: sess_<session_id>[2]. Σε αυτό το αρχείο αποθηκεύονται τα ονόματα των μεταβλητών και οι τιμές των μεταβλητών.

Στους servers που έχουν καταχωρημένα πολλά hosts (multi-server) αυτό μπορεί να είναι πρόβλημα διότι τα αρχεία αποθηκεύονται στον server καθώς ο χρήστης χρησιμοποιεί τον server. Ένας κακόβουλος χρήστης μπορεί να δημιουργήσει ένα session που πολύ εύκολα θα του δώσει διαχειριστική πρόσβαση σε ένα άλλο site (δηλ. σε ένα host που έχει αυτός ο server) ή ακόμη να διαβάσει πληροφορίες από τα αρχεία των session.

Παραπάνω, ο attacker έπρεπε να αποθηκεύσει ένα αρχείο στον server, και αν δεν μπορούσε να χρησιμοποιήσει το file uploading, τότε θα χρησιμοποιούσαν την εφαρμογή και θα προσπαθούσαν να δημιουργήσουν μία μεταβλητή session με την τιμή που επιθυμούν ώστε να αποκτήσουν διαχειριστική πρόσβαση. Ύστερα, μπορούν να μαντέψουν την διαδρομή του αρχείου session (ή γνωρίζουν ήδη το όνομα του αρχείου[3]) και το μόνο που τους μένει είναι να βρουν τον κατάλογο που είναι αποθηκευμένο το αρχείο (συνήθως είναι ο κατάλογος /tmp).

Τελικά, μια τεχνική που δεν έχει βρεθεί ακόμη, είναι πώς ένας attacker μπορεί να δημιουργήσει ένα αρχείο session με το όνομα που επιθυμεί (για παράδειγμα: 'hello'). Αν μπορούσαμε να το κάνουμε αυτό, τότε η διαδρομή αρχείου θα ήταν αυτή: /tmp/sess_hello. Το ID του session μπορεί να συμπεριλαμβάνει μόνο χαρακτήρες με αριθμούς και γράμματα, κάτι που μερικές φορές μπορεί να φανεί χρήσιμο.

Υποσημειώσεις:

[1]: Επίσης μπορεί να ρυθμιστεί από το αρχείων ρυθμίσεων του server (httpd.conf)

[2]: Όπου <session_id>, είναι το τυχαίο ID του session που δημιούργησε ο server.

[3]: Για να γνωρίζουν το όνομα του αρχείου, πρέπει να γνωρίζουν και το session ID που έχει δημιουργήσει ο server. Αυτό ίσως είναι λιγάκι δύσκολο για να το βρούμε, αν δεν γνωρίζουμε τον τρόπο. Μερικές εφαρμογές περνάνε το session ID σε όλα τα URLs: file.php?PHPSESSID=711dc... Επίσης, μπορούμε να το βρούμε και από τα cookies. Η PHP δημιουργεί αυτόματα ένα cookie με όνομα PHPSESSID και ορίζει ως τιμή του cookie το session ID.

Loose Typing and Associative Arrays

Δυστυχώς, δεν κατάφερα να μεταφράσω τον τίτλο αυτού του κεφαλαίου, διότι θα χανόταν η σημασία της λέξης. Έτσι, θα προσπαθήσω να σας δώσω μία εξήγηση του τι είναι το «Loose Typing». Σε ελεύθερη μετάφραση σημαίνει «Χαλαρή δακτυλογράφηση». Αυτό σημαίνει ότι στην PHP έχουμε την δυνατότητα να κάνουμε το παρακάτω:

```
$x="text";

$x=5;
```

Όπως βλέπετε η πρώτη μεταβλητή είναι χαρακτήρες και η δεύτερη είναι ακέραιος αριθμός. Σε άλλες γλώσσες προγραμματισμού δεν μπορούμε να το κάνουμε αυτό, αλλά μπορούμε στην PHP. Με λίγα λόγια, στην PHP έχουμε την δυνατότητα να ορίσουμε μια μεταβλητή με όνομα x και να δώσουμε ως τύπο δεδομένο χαρακτήρες. Ύστερα, μπορούμε να αλλάξουμε την τιμή της μεταβλητής x με έναν ακέραιο αριθμό. Αυτό είναι το «Loose Typing» (δηλαδή αλλάζουμε τον τύπο δεδομένων της μεταβλητής).

Σχετικά με τα Associative Arrays, δεν υπάρχει κάποια ακριβής μετάφραση, γι' αυτό θα προσπαθήσω να εξηγήσω τι είναι. Ένα associative array στην επεξεργασία/ερώτηση ενός δείκτη ή ενός δείκτη τιμών είναι ένας αφηρημένος τύπος στοιχείων που αποτελείται από μια συλλογή κλειδιών και μια συλλογή τιμών, όπου κάθε κλειδί συνδέεται με μια αξία.

Η λειτουργία της εύρεσης της αξίας συνδεμένης με ένα κλειδί ονομάζεται «εύρεση», και αυτό είναι η σημαντικότερη λειτουργία που υποστηρίζεται από τα associative arrays. Αυτό μας δίνει μία γενική ιδέα του τι είναι τα associative arrays. Αν θέλετε περισσότερες πληροφορίες, μπορείτε να ρίξετε μια ματιά εδώ: Associative arrays

Loose Typing and Associative Arrays

[συνέχεια]

Αφού καταλάβαμε περίπου τι είναι, θα κάνουμε μία σύντομη σημείωση γι' αυτούς τους παράγοντες.

Η PHP δίνει την δυνατότητα να ορίσουμε μεταβλητές με τύπο δεδομένων κείμενο και μετά να την αλλάξουμε την τιμή της μεταβλητής σε άλλο τύπο δεδομένων (πχ ακέραιος). Παραδείγματος χάριν η μεταβλητή $hello όταν αξιολογείται ως κείμενο, έχει την τιμή "" (δηλ. είναι κενή) ενώ όταν αξιολογείται ως αριθμός, έχει την τιμή 0.

Πολλές φορές, αυτό μπορεί να οδηγήσει σε προβλήματα που δεν γίνονται εύκολα αισθητά με σκοπό την διόρθωσή τους. Στο δημοφιλές phpMyAdmin εντοπίστηκε ένα τέτοιο σφάλμα που ύστερα του δόθηκε ο κωδικός SRADV00008. Αν η μεταβλητή $x ήταν ορισμένη σε "000" δεν είναι ίση με το 0 ούτε η συνάρτηση empty() θα επιστρέψει TRUE (αληθής).

Τα arrays της PHP είναι συνειρμικά, δηλαδή ο δείκτης στη σειρά είναι μια ΣΕΙΡΑ και μπορεί να τεθεί οποιαδήποτε αξία σειράς, δεν αξιολογείται αριθμητικά. Αυτό σημαίνει ότι η είσοδος σειράς $x[000] δεν είναι η ίδια με την μεταβλητή $x[0]. Μην ελέγξετε αν το 0 είναι ίσο με ένα στοιχείο και κάπου αλλού στον κώδικά σας χρησιμοποιήσετε την empty() για να διαπιστώσετε αν είναι κενό.

Λειτουργία Συναρτήσεων

Κοιτάζοντας για κενά ασφαλείας σε εφαρμογές (όταν έχετε τον κώδικα) είναι χρήσιμο να έχετε μία λίστα με συναρτήσεις (functions) που χρησιμοποιούνται κατ' άσχημο τρόπο ή συχνά είναι καλοί στόχοι ώστε να χρησιμοποιηθούν κατά τρωτό τρόπο στην εφαρμογή. Αν ο χρήστης μπορεί να επηρεάσει τις παρακάτω παραμέτρους, τότε ο βαθμός επικινδυνότητας είναι πολύ μεγάλος!

Εκτέλεση κώδικα PHP:

require() και **include()** – Και οι δύο συναρτήσεις «διαβάζουν» ένα αρχείο και εκτελούν τα δεδομένα του αρχείου ως PHP κώδικα.
eval() – Ερμηνεύει ένα κείμενο χαρακτήρων ως PHP κώδικα.
preg_replace() – Όταν χρησιμοποιείται με /e, η συνάρτηση ερμηνεύει το κείμενο που αντικατέστησε ως κώδικα PHP.

Εκτέλεση Εντολών:

exec() – Εκτελεί μία εντολή και επιστρέφει την τελευταία γραμμή από το αποτέλεσμα του προγράμματος.
passthru() – Εκτελεί μία εντολή και επιστρέφει όλο το αποτέλεσμα κατευθείαν στον χρήστη.
system() – Είναι περίπου ίδια με την passthru() αλλά δεν χειρίζεται τα binary δεδομένα.
popen() – Εκτελεί μία εντολή και συνδέει το αποτέλεσμα μέσα στον κώδικα του αρχείου.

Κοινοποίηση αρχείων:

fopen() – Ανοίγει ένα αρχείο.
readfile() – Διαβάζει ένα αρχείο και εμφανίζει κατευθείαν τα δεδομένα του αρχείου στον browser του χρήστη.
file() – Διαβάζει ένα ολόκληρο αρχείο μέσα από ένα array.

Προστατεύοντας την PHP

Όλες οι επιθέσεις που περιγράψαμε παραπάνω δουλεύουν τέλεια σε μία προεπιλεγμένη εγκατάσταση της PHP 4. Εν τούτοις, έχω αναφέρει ότι η PHP μας δίνει την δυνατότητα να κάνουμε αλλαγές στις ρυθμίσεις της. Με ορισμένες αλλαγές στο αρχείο ρυθμίσεων[1] μπορούμε πολύ εύκολα να κάνουμε την PHP πολύ πιο ασφαλή και να κάνουμε την δουλειά του κακόβουλου χρήστη απίστευτα δύσκολη.

Πάντα όμως, υπάρχει ένα τίμημα για την ασφάλεια, γι' αυτό αποφάσισα να ταξινομήσω τις ακόλουθες ρυθμίσεις ανάλογα με την ζημία που μπορούν να προκαλέσουν:
* = Σχετικά ασήμαντο
** = Σοβαρό κενό ασφαλείας
*** = Βλάπτει σοβαρά
**** = Μπορεί να οδηγήσει σε τρομερούς μπελάδες!

Οι εκτιμήσεις μου είναι υποκειμενικές, γι' αυτό μην με κατηγορήσετε γι' αυτές. Αν χρησιμοποιήσετε όλες τις παρακάτω επιλογές, θα έχετε μία πραγματικά ασφαλή εγκατάσταση της PHP στον server σας, ακόμη και αν ο κώδικας που υπάρχει στον server σας είναι εύκολος στόχος!

**** - Απενεργοποιήστε την επιλογή register_globals
Αυτή η επιλογή θα σταματήσει την PHP από την δημιουργία γενικών μεταβλητών από τις κινήσεις των χρηστών. Αυτό είναι όταν ένας χρήστης υποβάλει μια φόρμα και υπάρχει ένα πεδίο με όνομα 'hello', η PHP δεν θα δημιουργήσει μία μεταβλητή $hello. Αυτή είναι η κύρια αιτία των προβλημάτων σε όλες τις εγκαταστάσεις της PHP. Επίσης είναι το καλύτερο που μπορείτε να κάνετε για την ασφάλεια στον server σας. Επίσης δεν δημιουργεί μεταβλητές αυτόματα, ακόμη και αν ο attacker δώσει ένα URL σαν αυτά που είδαμε προηγουμένως.

Προστατεύοντας την PHP

[συνέχεια]

*** - Ενεργοποιήστε την επιλογή safe_mode
Θα ήθελα να περιγράψω ακριβώς τι κάνει το safe_mode, αλλά δεν αναφέρεται αναλυτικά στο Εγχειρίδιο της PHP.

Οι περιορισμοί που ορίζει το safe_mode στην PHP:
-> Περιορίζει ποιες εντολές μπορούν να εκτελεστούν (όπως exec() κλπ).
-> Περιορίζει την πρόσβαση σε αρχείο ανάλογα με τα δικαιώματα χρήστη (ownership).

** - Ρυθμίστε το open_basedir
Αποτρέπει όλες τις ενέργειες αρχείων σε απομακρυσμένα αρχεία. Δηλαδή αν ο attacker δώσει ως διαδρομή ενός αρχείου την διεύθυνση του server του, η επίθεση θα αποτύχει, γιατί το αρχείο δεν βρίσκεται στον server, αλλά βρίσκεται σε απομακρυσμένο server – δηλαδή στον server του attacker.

** - Απενεργοποιήστε την επιλογή display_errors και ενεργοποιήστε την επιλογή log_errors
Αποτρέπει την PHP από την εμφάνιση σφαλμάτων κατά την εκτέλεσή της σελίδας στην ιστοσελίδα. Αυτό μπορεί αποτελεσματικά να περιορίσει τις επιθέσεις, διότι οι attackers δεν θα έχουν στοιχεία ως προς το πώς δουλεύει μία εφαρμογή. Επίσης μπορεί να κάνει την αποσφαλμάτωση μάταιη.

* - Απενεργοποιήστε την επιλογή allow_url_fopen
Σταματά όλες τις διαδικασίες στα απομακρυσμένα αρχεία.

Μπορεί να υπάρχουν και άλλες επιλογές που μου διαφεύγουν, γι' αυτό συμβουλευτείτε την τεκμηρίωση της PHP στο http://www.php.net

Υπευθυνότητα – Γλώσσα & Προγραμματιστής

Υποστηρίζω ότι είναι πολύ δύσκολη να γραφεί μία ασφαλή εφαρμογή PHP (στην προεπιλεγμένη εγκατάσταση της PHP), ακόμη και αν προσπαθήσετε. Δεν είναι ότι η PHP δύσκολη γλώσσα, είναι απίστευτα εύκολη για να προγραμματίσετε και υποστηρίζει ένα πλήθος προκαθορισμένων εντολών και συναρτήσεων που κάνει την ζωή του προγραμματιστή πολύ πιο εύκολη. Εν τούτοις, η PHP έχει ραγδαία ανάπτυξη (από τότε που ξεκίνησε μέχρι σήμερα) και στα χαρακτηριστικά της όπου δύο πράγματα συμβαίνουν:

Οι Web designers και άλλοι που δεν είναι προγραμματιστές, καταλήγουν να γράφουν εφαρμογές στην PHP! Δεν έχουν καμία ιδέα των επιπτώσεων του κώδικα που γράφουν. Αυτό γίνεται γιατί η νοοτροπία τους δεν είναι αυτή που πρέπει για να γράψεις μία εφαρμογή.

Τυπικά, μία εφαρμογή τρέχει σε ένα εκτεθειμένο σύστημα και είναι μια παγκοσμίως προσιτή σελίδα σε έναν server. Αυτό σημαίνει ότι όλες οι εφαρμογές είναι εκτεθειμένες γι’ αυτό πρέπει πάντα να κινούμαστε σαν να ήμασταν οι attackers. Δηλαδή, σε μία εφαρμογή το κύριο χαρακτηριστικό της δόμησης του κώδικά σας, πρέπει να είναι οπωσδήποτε η ασφάλεια.

Μόλις τελειώσετε την εφαρμογή και δείτε ότι δουλεύει όπως ακριβώς θέλετε, πρέπει να ξεκινήσετε την αποσφαλμάτωση και τις επιθέσεις στην ίδια σας την εφαρμογή για να ελέγξετε αν υπάρχουν προβλήματα. Για να κάνετε αποσφαλμάτωση πρέπει να χρησιμοποιήσετε το error_reporting(E_ALL) το οποίο θα σας εμφανίσει σφάλματα στον κώδικα, μη ορισμένες μεταβλητές κλπ.

Συνιστώ κατά την δημιουργία της εφαρμογής να χρησιμοποιείτε error_reporting(E_ALL), για να μην χρειάζεται να αλλάξετε πολύ κώδικα στην αποσφαλμάτωση.

Υπευθυνότητα – Γλώσσα & Προγραμματιστής

[συνέχεια]

Πολλές φορές ένα μικρό κενό ασφαλείας μπορεί να γίνει εφιάλτης, κάτι που θα θέσει την ασφάλεια των δεδομένων του server σε κίνδυνο! Για παράδειγμα μια δήλωση include() που χρησιμοποιεί τα δεδομένα μιας μεταβλητής για να κάνει include ένα αρχείο, θα ήταν ασφαλής, διότι η PHP πολύ απλά θα κάνει include το αρχείο και **δεν** θα δημιουργήσει ένα αρχείο. Όταν όμως ο το αρχείο ρυθμίσεων της PHP επιτρέπει την εισαγωγή απομακρυσμένων αρχείων (αρχεία σε άλλους servers), τότε αυτό μπορεί να εξελιχθεί σε πραγματικό εφιάλτη!

Πολλοί άνθρωποι κατηγορούν τους προγραμματιστές ότι δεν γράφουν σωστό κώδικα, προσωπικά εγώ πιστεύω ότι η γλώσσα το κάνει δύσκολο για τον προγραμματιστή να γράψει καλό κώδικα. Έτσι την ευθύνη πρέπει να την πάρει και η PHP.

Είναι λάθος να πει κάποιος ότι ο προγραμματιστής έπρεπε να το ξέρει καλύτερα αυτό. Το λέω αυτό, γιατί σε όποιες εφαρμογές και αν έχω διαβάσει τον κώδικα, οι προγραμματιστές _προσπάθησαν_ να το κάνουν σωστά αλλά επειδή ο attacker κατάλαβε το σκεπτικό του προγραμματιστή (και γνώριζε και τα κενά ασφαλείας), πολύ απλά κατάφερε να κάνει ζημία στην εφαρμογή.

Πνευματική ιδιοκτησία

Αυτή η παρουσίαση έγινε στα πλαίσια της 11η Συνάντηση Εκπαιδευτικών Πληροφορικής στη Δυτική Μακεδονία - Κοζάνη, Κυριακή, 20 Μαΐου 2007 . Όλα τα δικαιώματα, συμπεριλαμβανομένου του δικαιώματος πνευματικής ιδιοκτησίας επί του περιεχομένου της παρουσίασης, βρίσκονται υπό την κατοχή ή τον έλεγχο του συγγραφέα. Απαγορεύεται η αντιγραφή, αναδημοσίευση, φόρτωση, αποθήκευση (κάθε είδους), μετάδοση, εμφάνιση ή παρουσίαση σε κοινό, προσαρμογή ή αλλοίωση κάθε είδους του περιεχομένου της παρουσίασης για οποιοδήποτε λόγο χωρίς την προηγούμενη γραπτή έγκριση του συγγραφέα.

Στην Ελλάδα θεμελιώδης είναι ο νόμος 2121/1993 (ΦΕΚ 25, 4/3/93) ο οποίος, εμπνεόμενος από τον Παγκόσμιο Οργανισμό Διανοητικής Ιδιοκτησίας, αναθεωρεί την προηγούμενη νομοθεσία του 1920 και συμμορφώνεται με τις οδηγίες της Ευρωπαϊκής Κοινότητας.

N. 3184/2003, Κύρωση της Συνθήκης του Παγκόσμιου Οργανισμού Διανοητικής Ιδιοκτησίας για την Πνευματική Ιδιοκτησία