# Cookies, Sessions and Authentication

Ahogy a web projektje egyre nagyobbra és összetettebre nő egyre fontosabb lesz a felhasználók nyomon követése. Ez akkor is érvényes ha nem ad a web oldalához bejelentkezést vagy kódot, akkor is fontos lesz adatokat tárolni a felhasználóknak az oldalon töltött idejéről meg arról hogy remélhetőleg mikor látogatnak vissza.

Többféle technológia is képes ezen problémák megoldására, mint például egy egyszerű böngésző süti vagy akár munkamenet kezelés és http hitelesítés. Ezek segíthetnek hogy a web oldalát a használói igényei szerint alakítsa és hogy biztosítson egy zökkenőmentes és élvezhető időtöltést rajta.

## Sütik használata PHP-ban

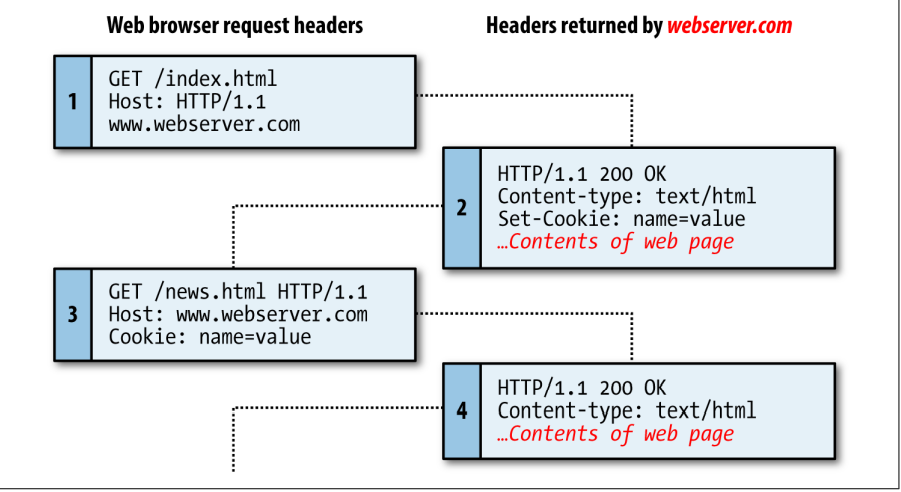
A süti egy adatdarab, amit a web szerver ment el a számítógép merevlemezében. Ezek bármilyen alfanumerikus információt tartalmazhatnak (ameddig ez 4 KB alatt van), amelyet bármikor vissza lehet szerezni a számítógéptől és visszatéríteni a szervernek. Ezek az adatok lehetnek hogy mennyi időt tölt a weboldalon, megtartani az adatokat több látogatáson keresztül, a bevásárlókosár tartalma bejelentkezési adatok és sok más.

Adatvédelmi okokból a sütiket csak a üzemeltető domain tudja olvasni. Más szavakkal, ha az üzemeltető domain például a oreilley.com, akkor csak azon a webszerveren keresztül lehet elérni, amelyik az a domaint használja. Ez megakadályozza, hogy a nem feljogosult weboldalak adatokat szerezzenek be.

Az internet működése miatt, egy weboldalon többféle elem ágyazhat be több domainhoz, és mindegyik a saját sütijét tudja lekérni. Amikor ez megtörténik harmadrendű sütiknek nevezzük. Álltalában ezeket reklámcégek alkalmazzák, hogy a felhasználókat kövessenek különböző weboldalakon.

Ebből kifolyólag, a legtöbb böngésző megengedi a felhasználónak hogy kikapcsolja ezeket a sütiket akár a szervernek domainjét, akár a harmadrendű sütiket vagy mindkettőt. Szerencsére a legtöbb ember aki kikapcsolja a sütiket, csak a harmadrendű sütiket kapcsolja ki.

A sütik kicserélődnek amikor a fejlécek átküldődnek, még mielőtt a weboldal HTML-je el lenne küldve. Ezért a sütik használatának fontos a kitervezése. Az alábbi ábra (ábra 1) bemutatja egy tipikus kérés és válasz beszélgetést a webes böngésző és a webszerver között zajlik.



Ábra 1

Ez a csere bemutatja, hogy a web böngésző megkap két oldalt:

1. A böngésző kérést ad le hogy megkapja a főoldalat, index.html, a <http://www.webserver.com> oldalon. Az első fejléc a fájlt határozza meg, és a második fejléc pedig a szervert határozza meg.
2. Amikor a webszerver a *webserver.com*-on megkapja a fejléceket, visszatérít néhányat a sajátjaiból. A második fejléc meghatározza a tartalmat, ami el lesz küldve (text/html), és a harmadik küld egy sütit a nevet *name* és az értéket *value*. Csak ezek után lesz a weboldal tartalma elküldve
3. Miután a böngésző megkapta a sütiket, vissza fogja téríteni azt minden jövőbeli kéréssel együtt egészen addig, amig a süti élettartama le nem jár vagy amíg le nem törlődik. Tehát amikor a böngésző kér egy új oldalt, *news.html,* akkor ezt visszatéríti a süti nevével és értékével.
4. Mivel a süti már be van állítva amikor a szerver megkapja a kérést hogy küldje a *news.html*-t, akkor nem kell újra elküldeni a sütit, csak visszaadja a kért oldalt

A sütiket könnyű szerkeszteni a böngésző beépített fejlesztői eszközök használatával. Mivelhogy a felhasználom meg tudja változtatni a sütik értékét, ezért nem ajánlott kulcsfontosságú adatokat tárolni a sütikben, vagy szembesülni kell azzal a lehetőséggel hogy a weboldal olyan módokon lesz manipulálva, ahogy nem is sejtenénk. A sütik legjobb felhasználási módja a nyelvi és a valuta adatok tárolása.

## Sütik beállításai

A PHP-ben süti beállítása egyszerű. Amíg még nem került HTML átvitelre, meghívhatja a *setcookie* függvényt, amely a következő szintaxissal rendelkezik(tábla 1):

|  |  |  |
| --- | --- | --- |
| Paraméter | Leírás | Példa |
| Név | A süti neve. Ez az a név, amelyet a szervere használni fog a hozzáféréshez  cookie a későbbi böngészőkéréseknél. | Location |
| Érték | A süti értéke vagy a süti tartalma. Ez legfeljebb 4 KB-ot tartalmazhat  alfanumerikus szövegben. | USA |
| Lejárat | (Opcionális.) A lejárati dátum Unix időbélyege. Általában valószínűleg fog  használati idő() plusz másodpercek száma. Ha nincs beállítva, a cookie lejár, amikor a böngésző bezárul. | time()+2592000 |
| Útvonal | (Opcionális.) A cookie elérési útjvonala a szerveren. Ha ez egy / (perjel), a  a cookie a teljes domainben elérhető, például a www.webserver.com. Ha ez egy  alkönyvtárban van, a cookie csak abban az alkönyvtárban érhető el. Az alapértelmezett  az aktuális könyvtár, amelybe a cookie be van állítva, és ez lesz a általában használni beállítás | / |
| Tartomány | (Opcionális.) A cookie internetes domainje. Ha ez a webserver.com, akkor a cookie elérhető a webserver.com összes aldomainje számára, mint például a www.webserver.com  és images.webserver.com. Ha az images.webserver.com , akkor a cookie  csak az images.webserver.com és aldomainjeihez elérhető el, mint pl sub.images.webserver.com, de mondjuk a www.webserver.com oldalon nem | webserver.com |
| Biztonságos | (Opcionális.) A cookie-nak biztonságos kapcsolatot kell-e használnia (https://). Ha ez az érték  ha IGAZ, a cookie csak biztonságos kapcsolaton keresztül továbbítható. Az alapértelmezett  HAMIS. | False |
| httponly | (Opcionális; a PHP 5.2.0-s verziója óta implementálva.) A cookie-nak kell-e használnia a  HTTP protokoll. Ha ez az érték TRUE, akkor a szkriptnyelvek, például a JavaScript nem használhatók  hozzáférjen a cookie-hoz. (Nem minden böngésző támogatja.) Az alapértelmezés FALSE | False |

Tehát egy cookie létrehozása a hely névvel és az USA értékkelaz ami elérhető aktuális domain teljes webszerverén, és eltávolítjuk a a böngésző gyorsítótárából hét napon belül, a következőket használja:

setcookie('location', 'USA', time() + 60 \* 60 \* 24 \* 7, '/');

## Cookie-hoz való hozzáférés

A cookie értékének kiolvasása nagyon egyszerű, a $\_COOKIE rendszertömb elérése történik. Mert például, ha szeretné látni, hogy az aktuális böngésző meghívja-e a cookie-nak a tárolt helyét, és ha igen, az értékének kiolvasásához használja a következőket:

if (isset($\_COOKIE['location'])) $location = $\_COOKIE['location'];

Vegye figyelembe, hogy a cookie-kat csak azután tudja visszaolvasni, miután elküldte azt egy webböngészőbe. Ez azt jelenti, hogy amikor cookie-t bocsát ki, nem tudja újra beolvasni, amíg a a böngésző újratölti az oldalt (vagy mást, amely hozzáfér a cookie-hoz) az Ön webhelyéről és a folyamat során visszaadja a cookie-t a szervernek.

## Süti megsemmisítése

Egy cookie törléséhez újra ki kell adnia, és be kell állítania egy múltbeli dátumot. Fontos hogy az új setcookie hívás összes paramétere, kivéve az időbélyeget, azonos legyen a süti első kiadásának paraméterei; ellenkező esetben a törlés sikertelen lesz. Ezért a korábban létrehozott cookie törléséhez a következőket kell használnia:

setcookie('location', 'USA', time() - 2592000, '/');

Amíg a megadott idő a múltban van, a cookie-t törölni kell. Azonban van 2 592 000 másodpercet (egy hónapot) használt a múltban arra az esetre, ha az ügyfélszámítógép a dátum és az idő nincs megfelelően beállítva. Megadhat egy üres karakterláncot is a cookie értékét (vagy FALSE értékét), és a PHP automatikusan beállítja az időt a múltban neked.

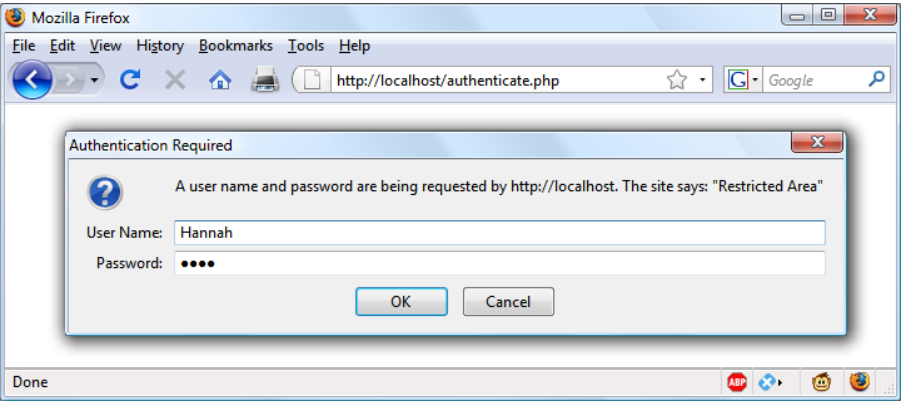
## HTTP hitelesítés

A HTTP-hitelesítés a webszerver használja a felhasználók és jelszavak kezelésére az alkalmazáson belül. Ez megfelelő egyszerű alkalmazásokhoz, amelyek bejelentkezést kérnek a felhasználóktól a legtöbb alkalmazás viszont speciális igényekkel vagy szigorúbb biztonsági követelményekkel rendelkezik amelyek más technikákat igényelnek.

A HTTP-hitelesítés használatához a PHP fejléckérést küld, amelyben kéri a hitelesítés elindítását, hogy párbeszédet indítson a böngészővel. A szerveren ez a funkciónak be kell kapcsolva hogy működjön, de mivel nagyon elterjedt, a szerver valószínűleg felajánlja ezt a funkciót

Bár általában az Apache-val telepítik, a HTTP hitelesítési modul nem feltétlenül van telepítve van az Ön által használt szerveren. Tehát ezeknek a példáknak a futtatása hibaüzenetet generálhat hogyha a funkció nincs engedélyezve, ebben az esetben ezt is meg kell tennie, telepítse a modult és módosítsa a konfigurációs fájlt a betöltéshez, vagy kérje meg rendszergazdáját, hogy végezze el ezeket a javításokat.

Miután beírta az URL-t a böngészőbe, vagy felkereste az oldalt egy hivatkozáson keresztül, a felhasználó láthassa a „Hitelesítés szükséges” felugró ablakot, amely két mezőt kér: ​​Felhasználónév és Jelszó



12-2 ábra

<? php

if (isset ($ \_ SERVER ['PHP\_AUTH\_USER']) &&isset ($ \_ SERVER ['PHP\_AUTH\_PW'])

{

echo "Welcome User:". htmlspecialchars ($ \_ SERVER ['PHP\_AUTH\_USER']).

"Password:". htmlspecialchars ($ \_ SERVER ['PHP\_AUTH\_PW']);

} else{

header ('WWW-Authenticate: Basic realm = "Restricted Area"');

header ('HTTP / 1.0 401 Unauthorized');

die ("Please enter your username and password");

}?>

12-1. példa

A program először két konkrét tömbértéket keres: $\_SERVER['PHP\_AUTH\_USER'] és $\_SERVER['PHP\_AUTH\_PW']. Ha mindkettő létezik, akkor a felhasználó által a hitelesítéshez megadott felhasználónevet és jelszót jelentik

Vegye észre, hogy a képernyőn való megjelenítéskor az értékek, amelyeknek a $\_SERVER tömbben visszakerültek, először a htmlspecialchars függvényen keresztül. Ez azért van így, mert ezek az értékek a felhasználó adta meg, és ezért nem lehet megbízni bennük, mivel egy hacker a HTML karakterek hozzáadásával keresztirányú szkriptelési kísérletet tehet. karaktereket és más szimbólumokat ad a bemenethez. htmlspecialchars lefordítja a minden ilyen bevitelt ártalmatlan HTML-egységekké.

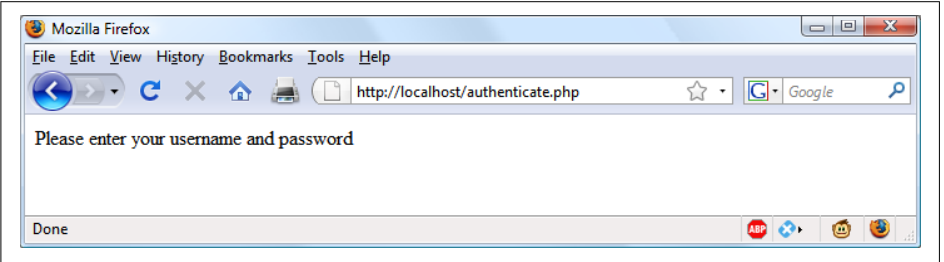
Ha valamelyik érték nem létezik, a felhasználó még nem lett hitelesítve, és megjelenik a következő üzenet a 12-2. ábrán látható felszólítást a következő fejléc kiadásával, ahol a Basic realm a védett szakasz neve, amely a felugró kérés részeként jelenik meg:

WWW-Authenticate: Basic realm="Restricted Area"

Ha a felhasználó kitölti a mezőket, a PHP-program újra lefut a tetejéről. De ha a felhasználó a Mégsem gombra kattint, a program a következő két sorral folytatódik, amely a következő fejlécet és egy hibaüzenetet küld:

HTTP/1.0 401 Unauthorized

A die utasítás hatására a "Kérjük, adja meg felhasználónevét és jelszavát" szöveget megjelenik (lásd a 12-3. ábrát)



12-3. ábra

Ha egy felhasználó egyszer már hitelesített, akkor nem lesz képes megkapni a hitelesítési párbeszédpanel újra felugrik, kivéve, ha a felhasználó bezárja és újra megnyitja az összes böngészőablakot, mert a böngésző továbbra is megmarad a ugyanazt a felhasználónevet és jelszót küldi vissza a PHP-nek. Szükség lehet néhányszor be kell zárnia és újra meg kell nyitnia a böngészőjét, miközben végigdolgozza a ezt a részt, és különböző dolgokat próbáljon ki. A legegyszerűbb módja ennek ha megnyit egy új privát vagy névtelen ablakot, hogy futtassa ezeket a programokat példák futtatásához, így nem kell bezárnia az egész böngészőt.

Most ellenőrizzük, hogy érvényes-e a felhasználónév és a jelszó. A 12-1. példa kódja nem kell sokat változtatnunk ahhoz, hogy ezt az ellenőrzést hozzáadjuk, kivéve az előző üdvözlő üzenet kódját a helyes felhasználónév és jelszó ellenőrzésére, majd a üdvözlő üzenetet. A sikertelen hitelesítés hibaüzenet küldését eredményezi (lásd 12-2. példa).

12-2. példa

<?php

$username = 'admin';

$password = 'letmein';

if (isset($\_SERVER['PHP\_AUTH\_USER']) && isset($\_SERVER['PHP\_AUTH\_PW']))

{

if ($\_SERVER['PHP\_AUTH\_USER'] === $username && $\_SERVER['PHP\_AUTH\_PW'] === $password)

echo "You are now logged in";

else die("Invalid username/password combination");

}

else{

header('WWW-Authenticate: Basic realm="Restricted Area"');

header('HTTP/1.0 401 Unauthorized');

die ("Please enter your username and password");

}

?>

A felhasználónevek és jelszavak összehasonlításakor az === (azonosság) operátort kell használni, az == (egyenlő) operátor helyett. Ez azért van így, mert azt ellenőrizzük, hogy a két értékek pontosan megegyeznek-e. Például a '0e123' == '0e456', és ez nem megfelelő egyezés sem a felhasználónév, sem a jelszó szempontjából.

A felhasználók hitelesítésére már van egy mechanizmus, de csak egyetlen felhasználónévre és jelszóval vonatkozóan. Továbbá a jelszó tiszta szövegben jelenik meg a PHP-fájlban, és ha a valakinek sikerülne feltörnie a szerverét, azonnal megtudná. Szóval nézzünk meg egy jobb módszert a felhasználónevek és jelszavak kezelésére.

## Felhasználónevek és jelszavak tárolása

Nyilvánvaló, hogy a MySQL a természetes módja a felhasználónevek és jelszavak tárolásának. De ismétlem, nem akarjuk a jelszavakat tiszta szövegként tárolni, mert a weboldalunk veszélybe kerülhet, ha egy hacker hozzáfér az adatbázishoz. Ehelyett egy ügyes trükköt fogunk használni, egyirányú függvényt használunk. Ez a fajta függvény könnyen használható, és egy szöveges karakterláncot egy látszólag véletlenszerű karakterlánccá alakít át. Egyirányú jellegük miatt az ilyen függvényeket nem lehet visszafordítani, így a kimenetük biztonságosan tárolható egy adatbázisban - és bárki, aki ellopja azt nem lesz okosabb a használt jelszavakról. A könyv korábbi kiadásaiban az MD5 hashing algoritmus használatát ajánlottam az adatbiztonság érdekében. Az idő azonban halad előre, és az MD5-öt ma már könnyedén feltörhetőnek és ezért nem biztonságosnak tartják. Sőt, még a korábban ajánlott helyettesítője, az SHA-1 is feltörhető. Így most, hogy a PHP 5.5 mindenhol a minimális szabványnak számít, a következőkkel kell foglalkoznom, áttértem a beépített hashing funkció használatára, ami sokkal biztonságosabb, és mindent szépen kezel helyetted. Korábban a jelszó biztonságos tárolásához a jelszót sózni kellett volna, ami azt jelenti, hogy a jelszóhoz olyan extra karaktereket adunk hozzá, amelyeket nem a felhasználó adott meg. (hogy tovább homályosítsa azt). Ezután az eredményt egy egyirányú függvényen, hogy egy látszólag véletlenszerű karakterkészletté alakítsuk át, ami korábban nehéz volt feltörni. Például az olyan kód, mint a következő (ami ma már nagyon bizonytalan, mert a modern grafikus feldolgozóegységek sebessége és teljesítménye): Például az olyan kód, mint a következő (ami ma már nagyon bizonytalan, mert a modern grafikus feldolgozóegységek sebessége és teljesítménye):

echo hash('ripemd128', 'saltstringmypassword');

Ezt az értéket a következő képpen írná ki:

9eb8eb0584f82e5d505489e6928741e7

Ne feledje, hogy ezt a módszert soha nem ajánlott használni. Kezelje ezt úgy, mint példát, hogy mit ne tegyünk, mivel nagyon bizonytalan. Ehelyett olvasson tovább.

Password\_hash használata

A PHP 5.5-ös verziójától kezdve sokkal jobb módja van a sózott jelszóhashok létrehozásának: a password\_hash függvény. A második (kötelező) argumentumként adja meg a PASSWORD\_DEFAULT-ot hogy a függvény a jelenleg elérhető legbiztonságosabb hashing funkciót válassza ki. A password\_hash minden jelszóhoz véletlenszerű sót is választ. (Ne essünk kísértésbe, hogy további saját sózást adjunk hozzá, mert ez veszélyeztetheti az algoritmus biztonságát.) Tehát a következő kód:

echo password\_hash("mypassword", PASSWORD\_DEFAULT);

egy olyan karakterláncot fog visszaadni, mint a következő, amely tartalmazza a sót és az összes információt a jelszó ellenőrzéséhez szükséges adatokat:

$2y$10$k0YljbC2dmmCq8WKGf8oteBGiXlM9Zx0ss4PEtb5kz22EoIkXBtbG

Ha hagyja, hogy a PHP válassza ki a hashing algoritmust, akkor a lehetővé kell tennie, hogy a visszaadott hash mérete idővel bővüljön, ahogyan a jobb biztonság megvalósul. A PHP fejlesztői azt javasolják, hogy a hash-eket olyan adatbázis-mezőben tárolja, amely legalább a következő értékre bővülhet 255 karakterig bővíthető (bár a 60-72 karakter az átlagos hossz, de ez a helyes jelenleg). Ha szeretné, manuálisan is kiválaszthatja a BCRYPT algoritmust, hogy garantálja a csak 60 karakteres hash karakterláncot, a következő adatok megadásával a függvény második argumentumaként a PASSWORD\_BCRYPT konstans megadását. Ezt azonban nem ajánlom, hacsak nem rendelkezik nagyon jó okkal.

Opciókat adhat meg (egy opcionális harmadik argumentum formájában), hogy még jobban testre szabja a hash-számítás módját, például a költséget vagy a processzoridő mennyiségét, amelyet a hasheléshez (a több idő nagyobb biztonságot jelent, de lassabb szervert). A költségnek van egy alapértelmezett értéke 10, ami a BCRYPT esetében a legkevesebb, amit érdemes használni. Nem akarom azonban összezavarni Önt több információval, mint amennyire szüksége van hogy a jelszóhüvelyeket biztonságosan, a lehető legkevesebb felhajtással tárolhassa, ezért kérem, olvassa el az alábbiakat a dokumentációban, ha további részleteket szeretne megtudni a rendelkezésre álló lehetőségekről. Még azt is megteheti, hogy választhatsz magadnak sókat (bár ez a PHP 7.0-tól kezdve elavult, mivel nem biztonságosnak tartják - tehát ne essünk kísértésbe).

## Password\_verify használata

Ha ellenőrizni szeretné, hogy egy jelszó egyezik-e a hash-sel, használja a password\_verify függvényt, átadva az alábbiakat a felhasználó által az imént megadott jelszót, valamint az adott felhasználó tárolt hash-értékét, jelszó értékét (általában az adatbázisból nyerhető). Tehát, feltételezve, hogy a felhasználó korábban a következő (nagyon bizonytalan) jelszót adta meg: mypassword, és most már megvan a jelszava hash karakterlánca (a felhasználó által létrehozott a jelszavát) a $hash változóban tárolja, ellenőrizheti, hogy egyezik-e a jelszóval:

if (password\_verify("mypassword", $hash)) echo "Valid";

Ha a helyes jelszót adtunk meg a hash-hez, a password\_verify visszaadja a TRUE értéket, így ez az if utasítás az "Érvényes" szót fogja megjeleníteni. Ha nem egyezik, akkor FALSE-t kapunk vissza, és megkérhetjük a felhasználót, hogy próbálja meg újra.

## Egy példa program

Lássuk, hogyan működnek ezek a funkciók együtt a MySQL-lel kombinálva. Először is létre kell hoznia egy új táblát a jelszóhash-ok tárolására, ezért írja be a programot a 12-3. példát, és mentsük el setupusers.php néven (vagy töltsük le a kísérő webhelyről), majd nyissuk meg a böngészőben.

<?php // setupusers.php

require\_once 'login.php';

$connection = new mysqli($hn, $un, $pw, $db);

if ($connection->connect\_error) die("Fatal Error");

$query = "CREATE TABLE users (

forename VARCHAR(32) NOT NULL,

surname VARCHAR(32) NOT NULL,

username VARCHAR(32) NOT NULL UNIQUE,

password VARCHAR(255) NOT NULL

)";

$result = $connection->query($query);

if (!$result) die("Fatal Error");

$forename = 'Bill';

$surname = 'Smith';

$username = 'bsmith';

$password = 'mysecret';

$hash = password\_hash($password, PASSWORD\_DEFAULT);

add\_user($connection, $forename, $surname, $username, $hash);

$forename = 'Pauline';

$surname = 'Jones';

$username = 'pjones';

$password = 'acrobat';

$hash = password\_hash($password, PASSWORD\_DEFAULT);

add\_user($connection, $forename, $surname, $username, $hash);

function add\_user($connection, $fn, $sn, $un, $pw)

{

$stmt = $connection->prepare('INSERT INTO users VALUES(?,?,?,?)');

$stmt->bind\_param('ssss', $fn, $sn, $un, $pw);

$stmt->execute();

$stmt->close();

}

?>

12-3. példa

Ez a program létrehozza a felhasználók táblát a kiadványok adatbázisában (vagy a 10. fejezetben a login.php fájlhoz beállított adatbázisban). Ebben a táblázatban két felhasználót hoz létre: Bill Smith és Pauline Jones. Nekik a következő felhasználóneveik és jelszavaik: bsmith/mysecret és pjones/acrobat.

A táblázat adatainak felhasználásával most módosíthatjuk a 12-2. példát, hogy megfelelően hitelesítse a felhasználókat, és a 12-4. példa mutatja az ehhez szükséges kódot. Írja be, vagy írja letöltse be a kísérő webhelyről, majd győződjön meg róla, hogy authenticate.php néven van elmentve, és hívja meg a böngészőben.

<?php // authenticate.php

require\_once 'login.php';

$connection = new mysqli($hn, $un, $pw, $db);

if ($connection->connect\_error) die("Fatal Error");

if (isset($\_SERVER['PHP\_AUTH\_USER']) &&

isset($\_SERVER['PHP\_AUTH\_PW']))

{

$un\_temp = mysql\_entities\_fix\_string($connection, $\_SERVER['PHP\_AUTH\_USER']);

$pw\_temp = mysql\_entities\_fix\_string($connection, $\_SERVER['PHP\_AUTH\_PW']);

$query = "SELECT \* FROM users WHERE username='$un\_temp'";

$result = $connection->query($query);

if (!$result) die("User not found");

elseif ($result->num\_rows)

{

$row = $result->fetch\_array(MYSQLI\_NUM);

$result->close();

if (password\_verify($pw\_temp, $row[3])) echo

htmlspecialchars("$row[0] $row[1] :

Hi $row[0], you are now logged in as '$row[2]'");

else die("Invalid username/password combination");

}

else die("Invalid username/password combination");

}

else

{

header('WWW-Authenticate: Basic realm="Restricted Area"');

header('HTTP/1.0 401 Unauthorized');

die ("Please enter your username and password");

}

$connection->close();

function mysql\_entities\_fix\_string($connection, $string)

{

return htmlentities(mysql\_fix\_string($connection, $string));

}

function mysql\_fix\_string($connection, $string)

{

if (get\_magic\_quotes\_gpc()) $string = stripslashes($string);

return $connection->real\_escape\_string($string);

}

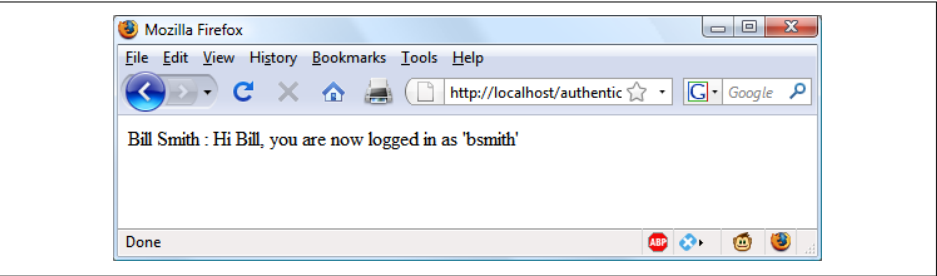
?>

12-4 példa



A HTTP-hitelesítés használata körülbelül 80 ms-os büntetést okoz, ha a password\_verify-t a következővel kódolt jelszavakkal használjuk BCRYPT-tel, az alapértelmezett költség 10. Ez a lassulás a akadályt jelent a támadók számára, hogy megakadályozza őket abban, hogy maximális sebességgel próbálják feltörni a jelszavakat. Ezért a HTTP-hitelesítés nem jó megoldás a nagyon forgalmas oldalakon, ahol valószínűleg inkább a munkamenetek használata (lásd a következő szakaszt).

Ahogy a könyvnek ezen a pontján várható volt, néhány példa már kezd egy kicsit hosszabb leszni. De ne hagyja magát elriasztani. Az utolsó 10 sor egyszerűen a 10-21. példából a 10. fejezetből. Ezek a felhasználói bemenet szanálására szolgálnak - ez nagyon fontos. Az egyetlen sor, amivel ezen a ponton tényleg foglalkoznunk kell, az a következővel félkövérrel kiemeltek. Ezek két változó, a $un\_temp és a $pw\_temp hozzárendelésével kezdődnek, a következővel a megadott felhasználónév és jelszó alapján. Ezután egy lekérdezést küldünk a MySQL-nek, hogy megkeresse a következő adatokat a $un\_temp nevű felhasználót, és ha találat érkezik, akkor az első sort a $row értékhez rendeljük. Mivel a felhasználónevek egyediek, csak egy sor lesz. Most már csak az adatbázisban tárolt hash-értéket kell ellenőrizni, ami hap-nak a negyedik oszlopban van - a 3. oszlopban, ha 0-ról indulunk. $row[3] tartalmazza az előző, a password\_hash segítségével kiszámított hash értéket, amikor a felhasználó létrehozta a jelszavát. Ha a hash és a felhasználó által megadott jelszó egyezik, akkor a password\_verify TRUE értéket ad vissza, és egy barátságos üdvözlő karakterláncot ad ki, amely a felhasználót az első névvel szólítja a felhasználót (lásd a 12-4. ábrát). Ellenkező esetben hibaüzenet jelenik meg. Ezt kipróbálhatjuk magunk is, ha a böngészőnkben meghívjuk a programot, és beírjuk- bsmith felhasználónevet és mysecret jelszót (vagy pjones és acrobat) adunk meg értékeket, amelyeket a 12-3. példa mentett el az adatbázisba.Ezt kipróbálhatjuk magunk is, ha a böngészőnkben meghívjuk a programot, és beírjuk- bsmith felhasználónevet és mysecret jelszót (vagy pjones és acrobat) adunk meg értékeket, amelyeket a 12-3. példa mentett el az adatbázisba.



12-4 ábra

Azzal, hogy a bemenetet a megjelenése után azonnal haszanáljuk, a következőket érhetjük el blokkolja a rosszindulatú HTML-, JavaScript- vagy MySQL-támadásokat, még mielőtt tovább juthatnak, és nem kell majd haszanálnia ezeket az adatokat újra. Ha egy felhasználó jelszavában olyan karakterek szerepelnek, mint a < vagy az & (például, ezeket a htmlenti &lt; vagy &amp; -ra bővíti ki. ties függvény - de amíg a kódja lehetővé teszi, hogy a karakterláncokat, amelyek a megadott bemeneti szélességnél nagyobbak lehetnek, és amíg a karakterek a jelszavakat mindig átfuttatja ezen a szanáláson, akkor minden rendben lesz.

## A munkamenetek használata

Mivel a programod nem tudja megmondani, hogy más programokban milyen változókat állítottak be - vagy akár az ugyanaz a program milyen értékeket állított be az előző futtatáskor - néha szükség van arra, hogy nyomon követni, hogy a felhasználók mit csinálnak egyik weboldalról a másikra. Ezt a következőkkel teheti meg: rejtett mezők beállításával egy űrlapon, ahogyan azt a 10. fejezetben láttuk, és a mezők értékeinek ellenőrzésével az űrlap elküldése után, de a PHP egy sokkal erősebb, több biztonságosabb és egyszerűbb megoldást kínál a munkamenetek formájában. Ezek olyan változócsoportok, amelyek a szerveren tárolnak, de csak az aktuális felhasználóra vonatkoznak. Annak biztosítására, hogy a megfelelő változókat a megfelelő felhasználókra alkalmazzák, a PHP cookie-t ment el a felhasználók böngészőjében a felhasználók egyedi azonosítása érdekében. Ez a süti csak a webkiszolgáló számára bír jelentőséggel, és nem használható fel semmilyen információt a felhasználóról. Megkérdezheti, hogy azok a felhasználók, akiknek a cookie-kat kikapcsolták. Nos, napjainkban és a mai korban, aki kikapcsolta a cookie-kat, ne számítson arra, hogy a legjobb böngészési élményt, és ha letiltva találja őket, akkor valószínűleg érdemes lenne tájékoztatni az ilyen felhasználót, hogy a cookie-kat engedélyezni kell, ha teljes mértékben ki akarja használni a böngészés előnyeit ahelyett, hogy megpróbálná megkerülni a sütik használatát, ami biztonsági problémákat okozhat.

## A munkamenetek kezdete

A munkamenet indításához a session\_start PHP függvényt kell meghívni, mielőtt bármelyik HTML kimenet előtt, hasonlóan ahhoz, ahogyan a cookie-kat a fejlécek cseréje során küldjük. Ezután a munkamenetváltozók mentésének megkezdéséhez egyszerűen hozzárendeli őket a $\_SESSION részeként tömbhöz, például így:

$\_SESSION['variable'] = $value;

A későbbi programfuttatások során ugyanilyen egyszerűen visszaolvashatók, például így:

$variable = $\_SESSION['variable'];

Tegyük fel, hogy van egy alkalmazásunk, amelynek mindig szüksége van a keresztnévhez és vezetéknevéhez való hozzáférésre, amit a users táblában tárolnak, amelyet a következő módon kell létrehoznia egy kicsit korábban. Módosítsuk tovább az authenticate.php fájlt a 12-4. példából, hogy létrehozzunk egy munkamenet létrehozását, miután egy felhasználó hitelesítése megtörtént.

A 12-5. példa mutatja a szükséges változtatásokat. Az egyetlen különbség az if(password\_verify($pw\_temp, $row[3])) szakaszának tartalma, amelyet most egy munkamenetet nyitunk, és ebbe mentjük el ezeket a változókat. Írjuk be ezt a programot (vagy módosítsuk 12-4. példa), és mentsük el authenticate2.php néven. De még ne futtassuk a böngészőben, mivel egy pillanat múlva egy második programot is létre kell hoznia.

<?php // authenticate2.php

require\_once 'login.php';

$connection = new mysqli($hn, $un, $pw, $db);

if ($connection->connect\_error) die("Fatal Error");

if (isset($\_SERVER['PHP\_AUTH\_USER']) &&

isset($\_SERVER['PHP\_AUTH\_PW']))

{

$un\_temp = mysql\_entities\_fix\_string($connection, $\_SERVER['PHP\_AUTH\_USER']);

$pw\_temp = mysql\_entities\_fix\_string($connection, $\_SERVER['PHP\_AUTH\_PW']);

$query = "SELECT \* FROM users WHERE username='$un\_temp'";

$result = $connection->query($query);

if (!$result) die("User not found");

elseif ($result->num\_rows)

{

$row = $result->fetch\_array(MYSQLI\_NUM);

$result->close();

if (password\_verify($pw\_temp, $row[3]))

{

session\_start();

$\_SESSION['forename'] = $row[0];

$\_SESSION['surname'] = $row[1];

echo htmlspecialchars("$row[0] $row[1] : Hi $row[0],

you are now logged in as '$row[2]'");

die ("<p><a href='continue.php'>Click here to continue</a></p>");

}

else die("Invalid username/password combination");

}

else die("Invalid username/password combination");

}

else

{

header('WWW-Authenticate: Basic realm="Restricted Area"');

header('HTTP/1.0 401 Unauthorized');

die ("Please enter your username and password");

}

$connection->close();

function mysql\_entities\_fix\_string($connection, $string)

{

return htmlentities(mysql\_fix\_string($connection, $string));

}

function mysql\_fix\_string($connection, $string)

{

if (get\_magic\_quotes\_gpc()) $string = stripslashes($string);

return $connection->real\_escape\_string($string);

}

?>

12-5 példa

A program egy másik kiegészítése a "Kattintson ide a folytatáshoz" link, amelynek cél URL címe continue.php. Ezt fogjuk használni annak illusztrálására, hogy a munkamenet hogyan fog átmenni a egy másik programba vagy PHP weboldalra. Hozzuk létre tehát a continue.php állományt a következő program beírásával 12-6. példában szereplő programot, és mentse el.

<?php // continue.php

session\_start();

if (isset($\_SESSION['forename']))

{

$forename = htmlspecialchars($\_SESSION['forename']);

$surname = htmlspecialchars($\_SESSION['surname']);

echo "Welcome back $forename.<br>

Your full name is $forename $surname.<br>";

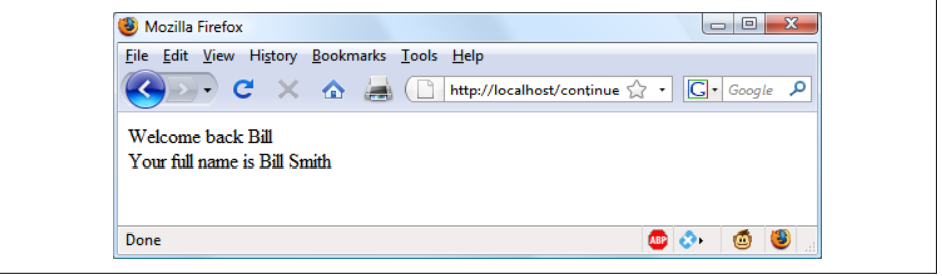
}

else echo "Please <a href=authenticate2.php>click here</a> to log in.";

?>

12-6 példa

Most már készen áll arra, hogy meghívja az authenticate2.php fájlt a böngészőjében. Adja meg a bsmith felhasználónevet és a mysecret jelszót (vagy pjones és acrobat), amikor a rendszer kéri, és kattintson a linkre a continue.php betöltéséhez. Amikor a böngésző meghívja, az eredménynek a következőnek kell lennie a 12-5. ábrához hasonlónak kell lennie.



12-5 ábra

A munkamenetek egyetlen programba korlátozzák a felhasználó hitelesítéséhez és bejelentkezéséhez szükséges kiterjedt kódot. Miután a felhasználó hitelesítése megtörtént, és létrehoztunk egy munkamenetet, a programkód nagyon egyszerűvé válik. Csak a session\_start-ot kell meghívni, és a $\_SESSION-ban megkeresni azokat a változókat, amelyekhez hozzáférésre van szükségünk. A 12-6. példában egy gyors teszt, hogy a $\_SESSION['előnév'] tartalmaz-e értéket, elegendő ahhoz, hogy megtudjuk, hogy az aktuális felhasználó hitelesített, mivel a munkamenetváltozókat a szerveren tároljuk (ellentétben a cookie-kkal, amelyeket a böngészőben tárolunk). és ezért megbízhatóak.

Ha a $\_SESSION['előnév'] nem kapott értéket, akkor nincs aktív munkamenet, így a A 12-6. példa utolsó kódsora a felhasználókat az authenticate2.php bejelentkezési oldalra irányítja.

## A munkamenet befejezése

Akkor eljön az ideje a munkamenet befejezésének, általában amikor a felhasználó kéri, hogy jelentkezzen ki a webhelyről, használhatja a session\_destroy függvényt, mint a 12-7. példában. Ez a példa egy hasznos függvény a munkamenet teljes megsemmisítéséhez, a felhasználó kijelentkezéséhez és a az összes munkamenet-változó visszaállítását.

<?php

function destroy\_session\_and\_data()

{

session\_start();

$\_SESSION = array();

setcookie(session\_name(), '', time() - 2592000, '/');

session\_destroy();

}

?>

12-7 példa

A continue.php állományt a 12-8. példa szerint módosíthatjuk, hogy ezt a gyakorlatban is láthassuk.

<?php

session\_start();

if (isset($\_SESSION['username']))

{

$forename = $\_SESSION['forename'];

$surname = $\_SESSION['surname'];

destroy\_session\_and\_data();

echo htmlspecialchars("Welcome back $forename.<br>

Your full name is $forename $surname.");

}

else echo "Please <a href='authenticate2.php'>click here</a> to log in.";

function destroy\_session\_and\_data()

{

$\_SESSION = array();

setcookie(session\_name(), '', time() - 2592000, '/');

session\_destroy();

}

?>

12-8 példa

Az első alkalommal, amikor az authenticate2.php-ból a continue.php-ba navigál, megjelenik az összes a munkamenetváltozókat. De a destroy\_session\_and\_data hívás miatt, ha a ezután a böngésző Reload gombjára kattintasz, a munkamenet megsemmisül és a rendszer felszólítja, hogy térjen vissza a bejelentkezési oldalra.

## Timeout beállítása

Vannak más esetek is, amikor a felhasználó munkamenetét saját maga szeretné lezárni, mint például a következő esetekben amikor a felhasználó elfelejtett vagy elmulasztott kijelentkezni, és azt szeretné, ha a program végezné el a kijelentkezést a saját biztonsága érdekében. Ezt úgy teheti meg, hogy beállítja azt az időkorlátot, amely után a a kijelentkezés automatikusan megtörténik, ha nem volt aktivitás.

Ehhez használjuk az ini\_set függvényt a következőképpen. Ez a példa az időkorlátot a következő értékre állítja be pontosan egy napra állítja be:

ini\_set('session.gc\_maxlifetime', 60 \* 60 \* 24);

Ha tudni szeretné, hogy mennyi az aktuális időkorlát, akkor azt a a következőkkel:

echo ini\_get('session.gc\_maxlifetime');

## Munkamenet biztonság

Bár említettem, hogy miután hitelesített egy felhasználót és létrehozott egy munkamenetet, nyugodtan feltételezheted, hogy a munkamenetváltozók megbízhatóak, ez nem egészen van így. Ennek oka az, hogy a csomagok szimatolásával (adatmintavételezéssel) lehetőség van arra, hogy a hálózaton keresztülhaladó munkamenet-azonosítók felfedezzünk. Továbbá, ha a munkamenet-azonosító átadásakor egy URL GET részében, az megjelenhet a külső webhely kiszolgálónaplóiban. Az egyetlen igazán biztonságos módja a felfedezés megakadályozásának az, ha megvalósítjuk a Transport Layer Security (TLS, a Secure Sockets Layer biztonságosabb utódja), vagy SSL), és HTTP weboldalak helyett HTTPS-t futtatni. Ez meghaladja ennek a cikknek a kereteit. könyvnek, bár a részletekért érdemes megnézni az Apache dokumentációját. a biztonságos webkiszolgáló beállításáról.

## A munkamenet eltérítésének megakadályozása

Ha az SSL nem lehetséges, akkor a felhasználók további hitelesítését az IP-címük tárolásával is elvégezheti. A címüket az egyéb adataikkal együtt egy olyan sor hozzáadásával, mint a következő, amikor tárolja a munkameneteiket:

$\_SESSION['ip'] = $\_SERVER['REMOTE\_ADDR'];

Ezután extra ellenőrzésként, amikor bármelyik oldal betöltődik, és a munkamenet elérhető, végezze el a következő ellenőrzést. Meghívja a different\_user függvényt, ha a tárolt IP-cím nem egyezik az aktuális címmel:

if ($\_SESSION['ip'] != $\_SERVER['REMOTE\_ADDR']) different\_user();

Az, hogy milyen kódot helyezel el a different\_user függvényedben, csak rajtad múlik. Én azt javaslom, hogy vagy törölje az aktuális munkamenetet, és kérje meg a felhasználót, hogy technikai hiba miatt jelentkezzen be újra, vagy ha ismeri az e-mail címét, küldjön neki egy linket, hogy megerősítse a személyazonosságát, ami lehetővé teszi, hogy a munkamenetben lévő összes adatot megőrizze.

Természetesen tisztában kell lennie azzal, hogy az azonos proxy-kiszolgálón lévő, vagy otthoni vagy üzleti hálózaton azonos IP-címen osztozó felhasználóknak ugyanaz az IP-címe. Ha ez is problémát jelent az Ön számára, használja a HTTPS-t. A böngésző felhasználói ügynök karakterláncának másolatát is tárolhatja (ez egy olyan karakterlánc, amelyet a fejlesztők a böngészőkbe helyeznek, hogy típus és verzió szerint azonosítsák azokat), ami szintén megkülönböztetheti a felhasználókat a böngészőtípusok, verziók és számítógépes platformok széles skálája miatt (bár ez nem tökéletes megoldás, és a karakterlánc változik, ha a böngésző automatikusan frissül). A felhasználói ügynök tárolásához használja a következőt:

$\_SESSION['ua'] = $\_SERVER['HTTP\_USER\_AGENT'];

És ezt használja az aktuális user-agent karakterlánc és a mentett karakterlánc összehasonlítására:

if ($\_SESSION['ua'] != $\_SERVER['HTTP\_USER\_AGENT']) different\_user();

Vagy, ami még jobb, kombinálja a két ellenőrzést, és mentse el a kombinációt egy hash hexadecimális karakterláncként:

$\_SESSION['check'] = hash('ripemd128', $\_SERVER['REMOTE\_ADDR'] .

$\_SERVER['HTTP\_USER\_AGENT']);

És ezt használja az aktuális és a tárolt karakterláncok összehasonlítására:

if ($\_SESSION['check'] != hash('ripemd128', $\_SERVER['REMOTE\_ADDR'] . $\_SERVER['HTTP\_USER\_AGENT'])) different\_user();

## Az ülésrögzítés megelőzése

A munkamenet rögzítése akkor történik, amikor egy rosszindulatú harmadik fél megszerez egy érvényes munkamenet-azonosítót (amely lehet a kiszolgáló által generált), és a saját azonosítója helyett ezzel a munkamenet-azonosítóval hitelesíti a felhasználót. Ez akkor fordulhat elő, ha egy támadó kihasználja a munkamenet-azonosító átadásának lehetőségét egy URL-cím Get részében, például így:

<http://yourserver.com/authenticate.php?PHPSESSID=123456789>

Ebben a példában az 123456789 kitalált munkamenet-azonosító kerül átadásra a kiszolgálónak. Most nézzük meg a 12-9. példát, amely érzékeny a munkamenet rögzítésére. Hogy lássuk, hogyan, írjuk be a és mentse el sessiontest.php néven.

<?php // sessiontest.php

session\_start();

if (!isset($\_SESSION['count'])) $\_SESSION['count'] = 0;

else ++$\_SESSION['count'];

echo $\_SESSION['count'];

?>

12-9 példa

Miután elmentette, hívja meg a böngészőjében az alábbi URL segítségével (a megfelelő elérési útvonalat előlegezve meg, például <http://localhost>):

sessiontest.php?PHPSESSID=1234

Nyomja meg néhányszor a Reload gombot, és látni fogja, hogy a számláló növekszik. Most próbáljon meg böngészni:

sessiontest.php?PHPSESSID=5678

Nyomja meg itt néhányszor az Újratöltés gombot, és látnia kell, hogy a számláló 0-ról újra felfelé számol. Hagyja a számlálót az első URL-től eltérő számon, menjen vissza az első URL-hez, és nézze meg, hogyan változik vissza a szám. Itt két különböző munkamenetet hoztál létre saját választásod szerint, és könnyen létrehozhatsz annyi munkamenetet, amennyit csak akarsz. Ez a megközelítés azért olyan veszélyes, mert egy rosszindulatú támadó megpróbálhatna ilyen típusú URL-eket terjeszteni gyanútlan felhasználóknak, és ha bármelyikük követné ezeket a a támadó vissza tudna térni, és átvenné az összes olyan munkamenetet, amelyik nem volt nem törölték vagy nem jártak le!

Ennek megelőzése érdekében a session\_regenerate\_id segítségével minél hamarabb változtassa meg a munkamenet azonosítóját. Ez a funkció megtartja az összes aktuális munkamenetváltozó értékét, de a munkamenet azonosítóját egy újjal helyettesíti, amelyet a támadó nem ismerhet. Ehhez ellenőrizze egy speciális munkamenetváltozót, amelyet tetszőlegesen kitalál. Ha nem

nem létezik, akkor tudja, hogy ez egy új munkamenet, így egyszerűen megváltoztatja a munkamenet azonosítóját, és a speciális munkamenetváltozót úgy állítja be, hogy a változást megjegyezze. A 12-10. példa mutatja, hogyan nézhet ki az ehhez szükséges kód, a kezdeményezett munkamenetváltozó használatával.

<?php

session\_start();

if (!isset($\_SESSION['initiated']))

{

session\_regenerate\_id();

$\_SESSION['initiated'] = 1;

}

if (!isset($\_SESSION['count'])) $\_SESSION['count'] = 0;

else ++$\_SESSION['count'];

echo $\_SESSION['count'];

?>

12-10 példa

Így a támadó bármelyik munkamenet-azonosítóval visszatérhet a webhelyére, amelyik generált, de egyikük sem fogja meghívni egy másik felhasználó munkamenetét, mivel mindegyikük újonnan generált azonosítókkal lettek helyettesítve.

## Csak cookie-kat tartalmazó munkamenetek kikényszerítése

Ha készen áll arra, hogy megkövetelje a felhasználóktól a sütik engedélyezését a weboldalán, akkor a következőket teheti meg az ini\_set függvényt használhatja, például így:

ini\_set('session.use\_only\_cookies', 1);

Ezzel a beállítással a ?PHPSESSID= trükk teljesen figyelmen kívül marad. Ha ezt használja biztonsági intézkedést használ, azt is javaslom, hogy tájékoztassa a felhasználókat, hogy a webhelye cookie-kat igényel (de csak akkor, ha a felhasználó letiltotta a cookie-kat), így tudják, hogy mi az, ami mi a baj, ha nem kapják meg a kívánt eredményeket.

## Megosztott szerver használata

Egy más fiókokkal megosztott szerveren nem szeretné, ha az összes munkamenetadata ugyanabba a könyvtárba lenne mentve, mint az övék. Ehelyett válasszon egy olyan könyvtárat a munkamenetek tárolására, amelyhez csak a saját fiókjának van hozzáférése (és amely nem látható a weben), egy ini\_set hívás elhelyezésével a programja elején, például így:

ini\_set('session.save\_path', '/home/user/myaccount/sessions');

A konfigurációs opció csak a program végrehajtása alatt tartja meg ezt az új értéket, és a program befejezésekor visszaáll az eredeti konfiguráció. Ez a munkamenetek mappa gyorsan megtelhet; érdemes lehet rendszeresen törölni a régebbi munkameneteket aszerint, hogy mennyire van elfoglalva a kiszolgáló. Minél többet használják, annál kevesebb időt kell munkamenetet akar majd tárolni.

Ne feledje, hogy a webhelyeit hacker támadás érheti és éri is. Az interneten automatizált botok garázdálkodnak, amelyek megpróbálják megtalálni a kihasználható webhelyeket. Tehát bármit is teszel, amikor olyan adatokat kezelsz, amelyek nem 100 százalékban a saját programodon belül keletkeztek, mindig kezeld azokat a legnagyobb óvatossággal.

Ezen a ponton már nagyon jól kell ismernie a PHP-t és a MySQL-t, így a a következő fejezetben a harmadik nagy technológiát mutatjuk be, amellyel ez a könyv foglalkozik. könyvben tárgyalt témát, a JavaScriptet.

## Kérdések

1. Miért kell a program indulásakor átadni egy sütit?

2. Melyik PHP-funkció tárolja a cookie-t a webböngészőben?

3. Hogyan lehet megsemmisíteni egy sütit?

4. Hol tárolják a felhasználónevet és a jelszót egy PHP-programban, amikor a

HTTP-hitelesítést használ?

5. Miért hatékony biztonsági intézkedés a password\_hash függvény?

6. Mit jelent a karakterlánc sózása?

7. Mi az a PHP-munkamenet?

8. Hogyan kezdeményezhetünk PHP-munkamenetet?

9. Mi az a munkamenet eltérítés?

10. Mi az a munkamenet rögzítés?