# Opevňujeme webové sídlo

## Zabezpečení nejpoužívanějších webových a databázových serverů

MARTIN IGNJATOVIČ

**Webových serverů je v současnosti v síti internet mnoho tisíc. Pomalu každá firma už má v dnešní době svou prezentaci. Vlastní web má i mnoho uživatelů. Nyní se podíváme, jak je to s bezpečností webových sídel v síti. Zaměříme se na všechny aspekty, které s webovým sídlem souvisí, databázovým serverem počínaje a psaním bezpečných skriptů konče. Nejdříve si popíšeme, jak takové webové sídlo vůbec vypadá.**

## Architektura webového sídla

Mluvíme-li o webu a webovém prostředí, máme zpravidla na mysli komunikaci dvou systémů protokolem http. Protokol http nám umožňuje přenášet po sítí hypertextové dokumenty a tyto pomocí příslušných programů interpretovat a zobrazovat. Tyto dokumenty však musí být nějak vytvářeny. To je druhá strana, kterou uživatel, jenž si dané stránky prohlíží, zpravidla nevidí a ani nechce vidět. Pro správce daného systému či autora skriptu je to velmi důležité. Dokumenty mohou být tvořeny staticky (rozuměj ručně za pomoci editoru) či dynamicky za použití příslušného skriptovacího jazyka. Pokud jsou dokumenty statické, jsou k jejich tvorbě použity značkovací jazyky HTML či XHTML, u wapových stránek pak jazyk WML. Tvorba takových stránek není u rozsáhlejších projektů jednoduchá, a proto bývá nahrazována použitím skriptovacích jazyků, které práci vývojáře značně zjednodušují. Mezi nejpoužívanější skriptovací jazyky patří .NET, PHP, JSP, Perl a další. Mnoho z těchto jazyků má vlastní API a může být interpretováno přímo webovým serverem. Jazyky, které toto rozhraní nemají, pracují přes univerzální rozhraní, například CGI (Common Gateway Interface). Přes rozhraní CGI můžete na svých stránkách zobrazovat výstupy programů libovolného jazyka, jako je například Bash, C, C++, Python a mnoha dalších. Princip je podobný a spočívá ve vygenerování příslušné stránky na serveru a poté v poslání čistého (jak kdy) výstupu do klientského programu. Tyto skripty se souhrnně označují jako SSI *(Server Side Included).* Skripty pracující na straně klienta (typicky JavaScript) nejsou předmětem toho článku. Data, která slouží jako podklad pro skriptovací jazyky, však musí být odněkud čerpána a musí být někde uložena. Nejčastěji jsou data čerpána a zpracovávána z databází (relačních i nerelačních) nebo textových souborů. Databáze je umístěna na databázovém serveru, který může, ale nemusí být na jednom systému spolu s webovým serverem. Nyní si ukážeme klady i zápory obou dvou řešení.

Pokud jsou webový i databázový server na jednom stroji, má to své výhody i nevýhody. Záleží na konkrétní aplikaci a jejích požadavcích. U méně vytížených aplikací je toto řešení vhodnější, neboť se tím zjednodušuje správa systému a šetří se i finanční prostředky na hardware. Naopak u velmi vytížených systémů je zpravidla lepší rozložit zátěž na dva a více strojů. U aplikací s několika tisíci přístupy v krátkém časovém úseku se doba na vyřízení požadavku podstatně sníží, pokud je zátěž rozložena mezi více procesorů, více paměti a více disků, což jsou faktory, které nejvíce ovlivňují výkon celé aplikace. Důležitá je samozřejmě i rychlost spojení obou systémů. Bude rozdíl, pokud spolu budou komunikovat systémy po desítkovém, nebo gigabitovém Ethernetu. Další výhodou tohoto řešení je, že databázový server nemusí být vůbec vidět „zvenku", a tudíž může být lépe chráněn. Nevýhodou je složitější správa a vyšší pořizovací cena. Nyní již tedy konkrétněji k datovému skladu.

## Datový sklad

Jak již bylo řečeno, může datový sklad nabývat různých podob. U datových skladů, které mají povahu textových souborů, můžete řešit otázku oprávnění pouze na straně operačního systému a jeho přístupových práv, což může být někdy problém. Naopak u relačních databází můžete téměř libovolně určovat, kdo, kdy, odkud a kam se může podívat, a co s danými daty může dělat. Správné nastavení přístupových práv je tak vlastně to nejsložitější, co nás čeká. K určení potřeb a cílů dané aplikace je velmi důležité přesně vědět, jak příslušná aplikace funguje. Přístupová práva by měla být nastavena co nejpřísněji a každý uživatel by měl mít jen tolik práv, kolik nezbytně potřebuje. Zřejmě nejčastější chybou je používání uživatelského účtu root vždy, všude a pro všechno. Nejhorší je to, že podobné chyby se dopouštějí i autoři, kteří publikují články o programování, a setkáváme se s nimi i v jiných aplikacích, jež jsou k dispozici. Většina z nich (převážně PHP, .NET a ostatní tak rozšířeny nejsou) začíná klasickým

```
<?php
@$spojeni=mysql_connect("localhost","root")...
```

Pro útočníka mají takovéto aplikace cenu zlata. Nejenže má neomezený přístup k celé databázi, ale ještě ke všemu je heslo správce prázdné. Útočníkovi by stačilo na svém systému zadat *mysql -u root -h firma.cz,* a mohl by směle modifikovat, prohlížet a jinak libovolně upravovat. Dalším velmi rozšířeným nešvarem je používat účet root k zobrazení nějakých dat. Taková práva jsou v tomto případě naprosto nepotřebná a bohatě stačí zavést uživatele, který má přístup pouze ke čtení, a pod tímto uživatelem pak data zobrazovat. Vinu na této situaci mají nejčastěji vývojáři, kteří si takto usnadňují práci a zpravidla si nadefinují nějaký objekt nebo funkci, pomocí níž se pak k databázi připojují. Nyní se podíváme na specifické databáze, a jako první si probereme MySQL.

## MySQL

Databáze MySQL je v prostředí webu jednou z nejpoužívanějších, a to zejména kvůli výkonu a ceně (pro nekomerční použití zdarma). Tato databáze se může svým výkonem směle měřit s věhlasnými produkty známých firem, a v mnohém je dokonce předčí. Její nevýhodou je menší podpora standardů, například transakčního zpracování. V oblasti bezpečnosti je tento produkt velmi flexibilní a umožňuje definovat uživatelská práva velmi podrobně. Klíčem k bezpečnosti celého serveru je databáze MySQL a v ní definované tabulky. Tabulky jsou následující:

**user**
**db**
**hst**
**fnc**
**clumns_priv**
**tables_priv**

Nyní již k jednotlivým tabulkám.

### User

Tato tabulka obsahuje oprávnění uživatelů ke všem databázím. Lze zde specifikovat, odkud se může uživatel připojit, uživatelské jméno, heslo, zda může spouštět výběrové, přidávací, aktualizační či odstraňující dotazy, nebo zda může delegovat oprávnění ostatním uživatelům – např. chceme-li z někoho učinit pomocného správce databáze.

### Db

V této tabulce lze nastavit stejná prává jako v předešlé, s tím rozdílem, že se týkají jen dané databáze. Tato funkce je opět velmi užitečná v případě, kdy chceme někoho udělat správcem databáze, ale zároveň nechceme, aby mohl spravovat databázi jinou.

### Host

Tato tabulka obsahuje stejná pole jako tabulka db, s tím rozdílem, že zde lze specifikovat hostitelský počítač, ze kterého se daný uživatel může připojit. Jako hostitelský počítač můžeme uvést IP adresu, doménové jméno. Využití této funkce bude zřejmě nejčastější v podnikové sféře, kdy víme, který uživatel se odkud bude připojovat.

### Columns_priv a tables_priv

V těchto dvou tabulkách můžeme určit, ze kterých tabulek či sloupců bude moci daný uživatel číst data. Chcete uživateli firemní webové aplikace poskytnout seznam zaměstnanců, ale zároveň mu odepřít čtení jejich platového výměru, když jsou oba údaje v jedné tabulce? Žádný problém.

## MS SQL

Databázový server společnosti Microsoft je v prostředí operačních systémů Windows jedním z nejpoužívanějších. Server je robustní a podporuje rozšířené standardy. My se podíváme na to, jak je to s jeho bezpečností, a hlavně na to, co všechno můžeme pro zvýšení bezpečnosti udělat. Nejdříve se však zaměříme na základní bezpečnostní mechanismy, které nám server nabízí.

### Ověřování

MS SQL server nám v podstatě nabízí implementaci dvou režimů zabezpečení. Jedním z nich je ověřování SQL serveru, druhým ověřování Windows. Z názvu je jasně vidět, jak obě řešení fungují. U ověřování Windows jsou k přístupu k databázi použita hesla uživatele od systému Windows (pouze Windows NT a vyšší). Toto řešení velmi zjednodušuje správu systému i databáze, a v posledních verzích SQL serveru je již nastaveno implicitně. Samozřejmě však vyžaduje silná a bezpečná hesla a dobré zabezpečení daného systému. Režim, kdy je použit ověřovací mechanismus SQL, funguje na principu uživatelské databáze na daném SQL serveru, podle které jsou uživatelé autorizováni. Nyní si povíme, jak vypadá proces přihlášení a ověření. Prvním pojmem, který bychom měli znát, je pojem login. Login je účet, který dává uživateli oprávnění přihlásit se k databázovému serveru. Nic jiného nezaručuje. Loginy jsou uloženy v tabulce sysxlogins v databázi master. Pokud máte nastaveno zabezpečení Windows, každý uživatelský účet, který na svém systému zřídíte, je do této tabulky zapsán. Na to, abychom si mohli prohlížet konkrétní data, nám však pouhý login nestačí. K tomu potřebujeme účet user. Tabulka sysusers je součástí každé databáze a opravňuje uživatele pro konkrétní akce. Každý účet user je mapován na určitý login. Tolik tedy k procesu uživatelských účtů. Nyní se podíváme na slabá místa. Je ironií, že většina metod, které se používají ke kompromitaci SQL serveru, je založena na předpokladu špatné konfigurace a zabezpečení. Většina útoků a incidentů se točí kolem špatně nastavených uživatelských práv. Pomineme útoky typu přetečení bufferu, které je snadné eliminovat včasnou aplikací vydaných aktualizací a service packů. Podíváme se na nastavení oprávnění a uživatelských účtů. Při přidělování oprávnění k databázi by se měl každý správce řídit heslem, které říká, že nikomu by se neměla přidělovat práva vyšší, než jaká nezbytně potřebuje. To je klíčem k eliminaci mnoha bezpečnostních problémů. Bohužel se tak v drtivé většině případů neděje. Za vše může z velké části neznalost, ale ještě častěji lidská pohodlnost a lenost. Tomuto tématu jsme se ale již věnovali, a tak se raději nyní podíváme na to, jak jinak se může útočník dostat k databázovým heslům a uživatelským účtům. Jednou z prvních možností je použít síťového sniffera. Jelikož SQL server ve výchozím stavu posílá hesla po síti v nešifrované podobě, je načase poohlédnout se po implementaci nějakého šifrovacího mechanismu. Zřejmě nejpřijatelnější je použití SSL nebo IPSec. SSL je více rozšířeno a jeho implementace je podstatně jednodušší než implementace IPSec. Dalším místem, kde by mohl útočník slídit po heslech, je aplikace, která se k serveru připojuje a v jejímž zdrojovém kódu mohou být hesla uložena. Způsobů je mnoho. Předpokladem pro uchování hesel v bezpečí je implementace kryptografie a dobrá systémová politika.

Dalším dobrým prvkem pro zvýšení bezpečnosti je spustit server pod uživatelem s omezenými právy, a ne pod účtem administrátora nebo systému. Nikdy nenechávejte prázdná hesla pro jakéhokoliv uživatele. Oblíbeným trikem ke kompromitaci SQL serveru je takzvaná injekce SQL kódu. Té se můžeme vyhnout, pokud budeme mít dobře ošetřené vstupy a vhodně nastavená přístupová práva. Dalším problematickým místem MS SQL serveru jsou uložené procedury. Ty mohou být zneužity mnoha způsoby, a nejlepší obrana proti jejich zneužití je jejich vypnutí. Myslíme tím procedury, které nutně nepotřebujete a které mohou ohrozit bezpečnost systému, jako například xp_cmdshell a podobné.

## Webové servery

### IIS

IIS server je v prostředí operačních systémů Windows stále ještě nejpoužívanějším webovým serverem a je druhým nejpoužívanějším webovým serverem světa. S nástupem technologie .NET se zdá, že jeho podíl na poli webových serverů poroste nebo minimálně zůstane stejně velký. S bezpečností tohoto serveru to však zas tak slavné není. Za posledních pár let bylo objeveno několik stovek bezpečnostních děr, a to také vedlo k celosvětovým katastrofám při šíření různých červů. Zde není tak často příčinou špatná konfigurace serveru, jako spíš neaktualizace systému. Nové bezpečnostní chyby typu přetečení bufferu se objevují periodicky, a je tedy vždy třeba řádně sledovat dění v oblasti bezpečnosti a včas systém aktualizovat. Pro zvýšení bezpečnosti však

můžeme udělat více, a to zejména promyšlenou konfigurací, která pomáhá předcházet některým známým trikům. Nejznámější triky a cíle útočníka si nyní popíšeme detailněji.

Cílem útočníka bude pravděpodobně ovládnout celý systém nebo přinejmenším celou webovou aplikaci, což znamená databázi, zdrojové kódy a soubory související s danou aplikací. K tomu může použít celou škálu již známých metod a triků. Triky nemá cenu detailně popisovat, neboť většina z nich je již opravena a každou chvíli se objeví nová díra na podobném principu. Řekneme si spíše, jak ony bezpečnostní díry identifikovat a jak je eliminovat. K poznání bezpečnostních slabin můžete použít řadu nástrojů. Zkušení uživatelé sáhnou po nástroji netcat, pomocí kterého lze dělat takové věci jako získávat hlavičky webového serveru, odesílat požadavky s nekorektními daty, a tak testovat server na různé známé chyby. Zřejmě nejkvalitnějším nástrojem na testování webových serverů je Stealth http scanner, který můžete získat na většině serverů zabývajících se bezpečností. Tento nástroj obsahuje i podporu pro vlastní skripty, a není tudíž problém napsat si vlastní pravidlo pro testování bezpečnosti vašeho serveru. Nyní si povíme, jak můžeme vhodnou konfigurací zvýšit bezpečnost serveru IIS.

## Systémová jednotka

Je velmi vhodné umístit kořenový adresář webového serveru na jinou systémovou jednotku, než na které je systém Windows. To lze velice snadno provést pomocí Správce služeb. Tím zabráníte útočníkovi ve zneužití systémových příkazů, jako například cmd.exe, které prostě nejsou na této jednotce dostupné. S tím souvisí i další věc, a to jsou přístupová práva.

## Přístupová práva

Na jednotce, kde máte umístěn kořenový adresář webového serveru, nikdy nepoužívejte souborový systém FAT(32), ale vždy NTFS! U každého adresáře poté nastavte práva co nejpřísněji. Není důvodu povolit zápis do složky, ve které jsou data určená jen pro čtení. Rovněž je velmi vhodné odebrat skupinám Everyone a Users práva pro zápis či vykonání. Nastavení správných přístupových práv je poměrně složitá záležitost a vyžaduje určité experimentování. Řiďte se zásadou, že uživatel by neměl mít práva větší, než nezbytně potřebuje ke své práci. Škoda, že ji vývojáři serveru nenastavili jako výchozí při instalaci IIS.

# Apache

Webový server Apache je nepoužívanějším webovým serverem světa. Je to díky jeho výkonu, bezpečnosti a rozšiřitelnosti. Nezajímavá není ani cena (je zdarma). Ani server Apache však není imunní vůči různým útokům, a proto se něj podíváme blíže. Vše, co se týká serveru Apache, lze nastavit v jeho konfiguračním souboru. Tento soubor se jmenuje *httpd.conf* a na *nixových systémech je zpravidla umístěn někde v adresáři */etc*. Tento soubor je velmi dobře okomentován. Důležitým krokem při konfiguraci serveru Apache je volba kořenového adresáře. Tuto direktivu najdete pod položkou *DocumentRoot*. Je vhodné umístit server Apache na samostatný diskový oddíl, a to nejen kvůli bezpečnosti, ale i kvůli výkonu. Dalším důležitým krokem je volba indexového souboru. Názvy souborů můžete zvolit pod direktivou DirectoryIndex a je vhodné ji vždy nastavit, aby případný útočník nemohl procházet vaše webové sídlo zcela libovolně a beztrestně. Pokud používáte http autentizaci pomocí souborů *.htaccess* (nepříliš vhodné řešení), je vhodné zakázat zobrazování těchto souborů pomocí webového klienta (direktiva AccessFileName a související). Pokud chcete přidělovat každému uživateli systému vlastní domovské adresáře, prohlédněte si direktivu Directory *„něco ../public_html"*- Můžete zde specifikovat, kteří z uživatelů mohou mít vlastní stránky, atd; tato volba je vhodná v případě, kdy chceme povolit několika uživatelům psaní skriptů, ale zároveň máme problém s nastavením přístupových práv. Rovněž je velmi rozumné (ve většině instalací se tak děje) umístit interpret php mimo strom webového serveru. Pokud provozujete server ve víceuživatelském prostředí, zauvažujte o zapnutí volby *safe_mode*. Tato direktiva kontroluje mnoho věcí, například přístupová práva, a zároveň vypíná některé funkce (lze definovat). Pokud provozujete na jednom serveru více virtuálních webserverů, prostudujte si dokumentaci ohledně Virtual Hosts.

▲ **Nástroj na testování webových serverů Stealth můžete získat na většině serverů zabývajících se bezpečností.**

## Skripty

Povolení skriptů je důležité. Pokud např. nepovolíte skripty, které jste napsali a neopatrně je vystavíte, bude mít každý přístup k jejich zdrojovému kódu, z něhož se může dozvědět velmi mnoho zajímavých informací, strukturou aplikace počínaje a hesly konče. Není vhodné povolovat interpretaci skriptovacích jazyků, které nepoužíváte. Nejčastěji používanými skriptovacími jazyky jsou PHP a rozhraní CGI skriptu. PHP můžete používat jako modul serveru Apache nebo přes rozhraní CGI. První volba je jistě vhodnější, neboť je nejen rychlejší, ale i bezpečnější (zpravidla). Přes rozhraní CGI mohou být zobrazeny výstupy prakticky libovolného jazyka, jako například Perl, C, Bash. Bezpečnost těchto aplikací již závisí na jiných faktorech, ale o těch až někdy jindy. Velkým problémem http komunikace obecně je možnost odposlouchávání, neboť data jsou po síti přenášena v textové podobě. To lze řešit použitím kryptografie. Nejrozšířenější metodou použití šifrování je v prostředí webu implementace SSL. A právě to, jak zprovoznit server Apache s podporou SSL, si nyní ukážeme. Budeme předpokládat, že modul SSL již máte nainstalovaný. Pokud ne, stáhněte si jej buď ve formě zdrojového kódu, nebo balíčku a nainstalujte. Popis instalace je vysvětlen v dokumentaci. Nyní se podíváme na základní nastavení SSL a na to, jak vytvořit vlastní certifikát.

## Nastavení

Předně musíte serveru definovat, na jakých portech má naslouchat. Pro SSL je to nejčastěji port 443, zadejme tedy do konfiguračního souboru *Listen 443*, poté je třeba zapnout SSL engine, a to doplněním (v řadě případů odkomentováním) řádku *SSLEngine On*. Nyní již jen stačí zadat cestu k privátnímu klíči a privátnímu certifikátu, např.: *SSLCertifikateKeyFile /data/ssl/www.domena.cz.key* a *SSLCertifikateFile /data/ssl/www.domena.cz.crt*. Pokud chcete v nějakém adresáři vynutit používání SSL, přidejte do konfiguračního souboru direktivu

**<Directory /cesta/k/adresari>**
**SSLrequireSSL**
**</Directory>**

Nyní již jen zbývá vytvořit certifikáty a klíče. Pro vytvoření privátního klíče zadejte příkaz:

**openssl genrsa -des3 1024 > www.domena.cz.key**

Poté zadejte heslo (frázi). Na tuto frázi se vás server bude ptát pokaždé, když se jej pokusíte spustit. Pokud nechcete při každém spuštění heslo zadávat, vynechejte direktivu -des3. Poté nezapomeňte změnit přístupová práva souboru s klíčem na 400 (pouze čtení). Nyní nám zbývá ještě vytvořit certifikační požadavek, který můžeme odeslat certifikační autoritě nebo si jej můžeme sami podepsat (pro testovací účely). Vytvoření certifikačního požadavku docílíme příkazem

**openssl reg -new -key www.domena.cz. key -out www.domena.cz.csr**

Po spuštění tohoto příkazu budete dotázáni na několik informací, jako je adresa, e-mail atd., které budou později uvedeny ve vlastnostech certifikátu. Takto vytvořený soubor můžeme odeslat certifikační autoritě k podpisu nebo si jej můžeme podepsat sami. Toho docílíme zadáním příkazu

**openssl req -x509 -key www.domena.cz.key -in www.domena.cz.csr -out www.domena.cz.crt**

Samopodepsaný certifikát není zárukou bezpečnosti, a pokud se s ním někde setkáte, zbystřete svou pozornost. Hodí se zejména pro testovací účely; chcete-li být důvěryhodní, nechte si certifikát podepsat certifikační autoritou. Zabezpečení webového sídla vyžaduje velké úsilí a dobré znalosti. Klíčovými faktory jsou webový server, datový sklad a samotná aplikace. 3 0476/FEL ❑