Zabezpečení serveru Apache a PHP

Vložil/a pog [1], 11 Listopad, 2004 - 18:23

- Encryption [2]
- Programming [3]
- Security [4]

Řetěz je vždy tak silný, jak je silný jeho nejslabší článek. Ve světě IT to platí obzvlášť. Spousta lidí má veřejnou IP adresu, spousta z nich provozuje WWW server ať už pro svoje potřeby nebo kvůli něčemu jinému. Vysoké procento těchto lidí má výchozí konfiguraci všech (nebo některých) serverových komponent, což není zrovna šťastné řešení. Podívejme se, co se s tím dá udělat.

Dnes si uděláme krátkou exkurzi do světa konfiguračních souborů WWW serveru Apache a skriptovacího jazyka PHP, což je podle mě nejčastější kombinace aplikací u lidí, kterých se právě tento článek týká nejvíce.

Pravidlo č. 1: čím míň informací, tím líp!

V první fázi jakéhokoliv útoku přichází na řadu zjišťování informací o cíli. Podle získaných dat se útočník buď rozhodne pokračovat na jistotu, naslepo, a nebo (nepravděpodobně) se na útok vykašle. Každopádně je snad všem jasné, že čím méně se toho "hacker" dozví, tím lépe.

Co se s tím dá dělat?

Apache

- nastavením direktivy ServerTokens na Prod zmenšíte identifikaci serveru v HTTP hlavičce na strohé "Apache", namísto obvyklého "Apache <verze> <OS> <zakompilované moduly>"
- nastavením direktivy ServerSignature na Off zamezíte vkládání serverového podpisu na stránky generovaného přímo serverem (stránky 401, 403, 404 a další).
- Server Apache má ve výchozí konfiguraci nastavené to, že když ve volanén adresáři není nalezen výchozí soubor, vypíše se celý jeho obsah. To je z bezpečnostního hlediska velice "nehodící se". Pokud tuto funckci vysloveně nepotřebujete, doporučuji jí vypnout. To se dělá tak, že u vašeho DocumentRoot nastavíte Options FollowSymLinks místo výchozích hodnot (důležité je, aby tam nebylo to Indexes - to je to, co nastavuje zobrazování adresářů)

Tak, Apache jsme schovali jak jen to šlo. Sice útočník může pořád přijít na to, že máme Apache, nicméně už neví jakou máme verzi, na jakém OS běží, ani nic dalšího. Teď ještě můžeme trošku schovat PHP. Upřímně si myslím, že to moc nepomůže, ale přinejmenším by to mohlo trošku zmást vulnerability scannery.

- Můžete přinutit Apache, aby posílal PHP interpreteru soubory s libovolnou příponou (samozřejmě tím myslím PHP skripty s nestandartní příponou). To se dělá nastavením AddType application/x-httpd-php (jakakoliv pripona). Pokud toto změníte, tak nezapomeňte změnit direktivu DirectoryIndex tak, aby obsahovala i výchozí soubor s Vaší příponou. Poté samozřejmě budete muset přejmenovat všechny vaše PHP skripty a pozměnit odkazy v nich, což by mohlo být dost pracné.

PHP

- Na internetu jsem ještě našel, že v souboru php.ini se dá nějak skrýt přítomnost PHP tím, že se nastaví direktiva expose_php na Off. V praxi jsem ale neobjevil, která identifikace se tím zamezí. Pokud víte co toto nastavení způsobuje, prosím napište to do diskuze pod článkem, popřípadě mi to pošlete ve vzkazu, upravím tento článek.

Pravidlo č. 2: co nejmíň chybových hlášek!

Může se stát, že změníte část svých skriptů, vše uploadnete bez kompletního otestování a v nejhorším případě by se mohlo stát, že tím útočníkovi poskytnete cenné informace. V krajním případě (při opravdu hloupě napsaných skriptech) například poznámka, že není definovaná proměnná $prihlasen. Věřte, že žádnému trochu znalejšímu návštěvníkovi Vašich stránek to nedá, a pokusí se tuto informaci nějak využít (například zadá do adresy stranka.php?prihlasen=true).

Co se s tím dá dělat?

PHP

- PHP obsahuje možnost nastavení urovně varování ve skriptech pomocí direktivy error_reporting. Mě osobně se osvědčilo vyvíjet weby na mém počítači s nastavením E_ALL a v ostrém provozu s ~E_ALL (negace E_ALL, nezobrazuje se nic). - **Oprava:** Jakub Vrána [5] upozornil na nepřesnost - pro zakázání zobrazování chybových hlášek je třeba nastavit error_reporting = 0, viz diskuze ke článku. Nastavení hlášení chyb v souboru php.ini je globální, pokud potřebujete vypnout, zapnout nebo změnit úřoveň hlášení chyb pro jeden konkrétní skript, použijte funkci error_reporting() (najdete v manualu :)
- Pokud chcete být velice paranoidní a nebo si nechcete hrát s error_reporting, pro vypnutí hlášení chyb nastavte direktivu display_errors na Off. Výsledek bude stejný jako s error reporting = 0.

Samozřejmě vypnutím robrazování chybových hlášek se připravíte někdy o cenné ladící informace, nebo i o hlášení o podezřelých aktivitách a to by byla škoda. Nevadí, můžete si zapnout logování do souboru pomocí direktivy log_errors nastavené na On a následně v direktivě error_log definováním souboru, do kterého se logy budou ukládat. Doporučuji soubor umístit mimo oblast přístupnou z WWW (jak zakázat přístup mimo wwwroot, to se dozvíte za chvíli)

Pravidlo č. 3: nikdy nevěřte vstupním informacím!

toto téma spíš spadá do zabezpečování samotných skriptů, což přesahuje rozsah tohoto článku, nicméně nastavení PHP umožňuje zmenšit riziko, které představují neplatná data od uživatele. (zabezpečením aplikací na serverech se budu zabývat v přístím článku)

- Čas od času data získaná od uživatele obsahují tzv. nebezpečné znaky. A nemusí se jednat o úmyslně škodící údaje, prostě uvozovky a apostrofy (podle mě nejnebezpečnější) byly, jsou a budou a my se s nimi musíme nějak vypořádat. PHP obsahuje direktivu magic_quotes_gpc, která když je nastavená na On, tak způsobuje to, že veškeré nebezpečné znaky na vstupu se nahrazují tzv "escape sekvencí" (" na \", ' na \', na atd). Zapnutím této direktivy prakticky odpadá nutnost používaní funkce addslashes() pro kontrolu vstupů. To nám ale nebrání být paranoidní a tuto kontrolu do svých skriptů zahrnout. Nehledě na to, že zapnutí magic_quotes_gpc zpomaluje samotné provádění skriptu.
- Data od klienta mohou přijít třemi metodami. Metodou GET, POST a nebo mohou být odeslány v cookie. A aby se nestalo, že někdo podvrhne data z formuláře odeílaného metodou POST tím, že zavolá cílový skript s příslušnýma proměnnýma, měli byste nastavit direktivu register_globals na Off a importovat všechny proměnné ručně. Je to sice o něco pracnější, ale o dost bezpečnější. Nemyslete si však, že když odesíláte formulář metodou POST, tak nemusíe vstupy kontrolovat. Zfalšovat jdou téměř stejně lehce jako při metodě GET.

Pravidlo č. 4: ne víc, než potřebujete!

Zde bych rád začal několika otázkami. Opravdu potřebujete, aby PHP skripty mohly otevírat soubory a adresáře mimo Váš wwwroot? Opravdu potřebujete mít možnost nahlédnout na funkci phpinfo()? A hlavně: Opravdu potřebujete ze skriptů volat funkci system() nebo exec()??? Já myslím, že většina z Vás na tyto otázky odpoví záporně. Nuže co s tím?

- V PHP se vyskytuje direktiva open_basedir, která umožňuje, který nejvyšší adresář ve stromové struktuře se smí otvírat. Doporučuji nastavit na wwwroot vašeho serveru. Jakmile se někdo pokusí otevřít adresář kam nesmí (například ../../../../../../../), tak to PHP odmítne udělat a vše se nám v tichosti zaloguje.
- Jazyk PHP disponuje stovkami funkcí, z nichž většinu nikdy nejspíš nepoužijete. A některé z nich jsou potencionálně nebezpečné. Proto v souboru php.ini existuje možnost zakázat některé funkce - tím, že napíšete jejich seznam oddělený čárkou k rirektivě disable_functions. Z osobních zkušeností doporučuju vypnout například tyto funkce: "system,phpinfo,exec,ftp_connect,eval,show_source". a pokud nikde nepracujete se soubory tak ještě všechny funkce, které s touto činností souvisí.

použité zdroje: manuál php, chvilka přemýšlení

URL článku:
https://www.security-portal.cz/clanky/zabezpe%C4%8Den%C3%AD-serveru-apache-php

Odkazy:
[1] https://www.security-portal.cz/users/pog
[2] https://www.security-portal.cz/category/tagy/encryption
[3] https://www.security-portal.cz/category/tagy/programming
[4] https://www.security-portal.cz/category/tagy/security
[5] http://php.vrana.cz