WordPress – kako varen je?

20 nasvetov, ki jih velja upoštevati pri varovanju WordPress spletnih strani

Vladimir Ban

VSEBINA

PRVO POGLAVJE

WORDPRESS – UVOD

Kaj sploh je WordPress

WordPress je platforma za oblikovanje in izdelavo spletnih strani.

Njen glavni namen je poenostaviti, pohitriti in na nek način standardizirati izdelavo spletnih strani za skrbnike in razvijalce.

Sodobne spletne strani so lahko s stališča programske kode, modulov in arhitekture zelo kompleksne. Priprava takšnih strani bi zahtevala veliko znanja in časa. Tu pride do izraza platforma WordPress, ki **z vnaprej pripravljenimi elementi bistveno pohitri izdelavo spletnih strani**, hkrati pa to omogoča tudi tistim skrbnikom, ki nimajo dovolj znanja, da bi posamezne elemente pripravili oz. razvili sami.

Preden se poglobimo v tematiko, je na mestu še pojasnilo glede izrazov »spletna stran« in »spletna aplikacija«. V uporabniškem smislu si lahko marsikdo ta dva izraza predstavlja ločeno. Spletna stran je urejen nabor spletnih strani, spletna aplikacija pa je bolj aplikacija v smislu, da uporabnikom ponuja celo vrsto funkcionalnosti. S stališča varnosti pa sta ta dva pojma praktično enaka. Tako spletne strani kot spletne aplikacije so razvite na podlagi istih spletnih tehnologij. In čeprav bi lahko upravičeno trdili, da je pojem »spletna stran« neka vrsta enostavne »spletne aplikacije«, se pri vprašanjih varnosti ta dva pojma prekrivata. Napačno je predvidevanje, da enostavne spletne strani predstavljajo zgolj enostavno tveganje. Četudi spletna stran ne omogoča funkcionalnosti dostopa do zalednih podatkovnih zbirk, posebnih podatkov ipd., grožnje še vedno presegajo tradicionalno grožnjo (zgolj) nepooblaščenega spreminjanja vsebine spletne strani. Z malce pretiravanja bi lahko rekli, da vsaka spletna stran, pa naj bo še tako enostavna, v okolje prinaša popolnoma enake grožnje, kot jih prinaša najbolj kompleksna spletna aplikacija, zato bomo v tem dokumentu ta dva pojma enačili.

Okolje WordPress deluje po načelu predlog in vtičnikov.

Predloga je skupek že pripravljenih skript in dokumentov, ki določajo osnovni videz spletne strani. Na primer, spletna trgovina je lahko že v osnovi videti povsem drugače kot pa, recimo, spletni portal z novicami.

Vtičniki pa so pripravljeni programi oz. skripte, ki v spletno stran vnašajo posamezne funkcionalnosti.

WordPress je torej skupek skript PHP, js, html in podobnih programov, ki skupaj določajo videz in funkcionalnosti spletnih strani. Nekaj teh skript in programov najprej v okolje prinese že sama platforma WordPress, drugi del skript in programov prinašajo izbrane predloge, tretji (praviloma največji) del skript in programov pa v okolje prinašajo posamezni uporabljeni vtičniki.

To, kar dela WordPress tako posebnega, je dejstvo, da se je v zadnjih letih ta platforma s pomočjo različnih razvijalcev po svetu (platforma deluje po načelu odprtokodnega sistema) razvila v zelo kakovostno in raznovrstno platformo. Nabor dostopnih predlog in vtičnikov, ki jih lahko razvijalec uporabi, namreč danes omogoča izvedbo kakršnekoli vsebine in funkcionalnosti, ki bi si jo razvijalec lahko zamislil. Funkcionalnosti in videz spletnih strani WordPress tako v resnici niso omejene z naborom predlog in vtičnikov, ki so na voljo, ampak prej z domišljijo razvijalcev.

Posledično se danes WordPress uporablja po celem svetu in če bi podrobneje preverili razne svetovno znane portale in spletne aplikacije, bi presenečeno lahko ugotovili, koliko jih v resnici temelji na tej platformi.

Zakaj je WordPress lahko zelo varen?

WordPress v resnici prinaša veliko prednosti glede varnostnih izzivov.

Prednosti izhajajo predvsem iz dveh dejstev.

1. WordPress je široko uporabljen odprtokodni sistem. To pomeni, da so vse skripte in programi, ki jih vnašamo v svoj sistem in od katerih teoretično vsak posebej prinaša določene varnostne grožnje, že uporabljeni tudi drugje. Bolj kot se določen vtičnik, predloga ali posamezna skripta uporablja v drugih okoljih, bolj se lahko zanesemo na javne vire, ki govorijo o varnosti teh elementov. Vse te skripte imajo lahko še vedno enake varnostne ranljivosti kot skripte, ki jih razvijemo sami. Ampak zaradi razširjenosti obstaja velika možnost, da bo ranljivost v javnosti slej ko prej odkrita in bo posledično razvijalec teh skript slej ko prej pripravil popravek.

Če torej v okolju WordPress uporabljamo priljubljene predloge in vtičnike ter hkrati skrbimo za sprotno nameščanje nadgradenj in popravkov, smo za varnost že veliko naredili. Seveda zgolj nameščanje nadgradenj ni dovolj. V nadaljevanju dokumenta si bomo ogledali še drugo plat zgodbe in predstavili razloge, zakaj ravno to na drugi strani prinaša določena tveganja.

2. Drugo dejstvo pa se nanaša na same vtičnike. Omenili smo že, da je velika prednost okolja WordPress v razširjenosti nabora vtičnikov. Za praktično vsako funkcionalnost, ki si jo skrbnik zamisli, obstaja cela vrsta vtičnikov, ki to funkcionalnost rešujejo. Enako seveda velja tudi za področje varnosti.

Praktično vse metode povečevanja varnosti spletnih aplikacij, ki jih poznamo (poenotenje sporočil o napakah, ustrezne nastavitve vrednosti piškotkov, onemogočanje napadi z grobo silo (brute-force attack) na prijavna okna, mehanizmi WAF, spremljanje dnevniških zapisov ipd.), so izvedljive preko različnih vtičnikov. Torej se lahko tudi pri zagotavljanju varnosti skrbnik ali razvijalec enostavno nasloni na že razvite vtičnike, s čimer si lahko pri vzpostavitvi varnostnih mehanizmov prihrani veliko časa. Hkrati pa lahko pravilno postavi veliko varovalk, četudi sam morebiti nima dovolj znanja (programiranje skript).

Zagotavljanje varnosti okolij WordPress je tako zelo podobno zagotavljanju varnosti druge priljubljene programske opreme. Niti ni tako zelo pomembno, da skrbnik podrobno pozna konkretne tehnične specifične lastnosti sistema, temveč je bistveno bolj pomembno, da redno spremlja javno objavljene ranljivosti vtičnikov in sistema WordPress, skrbi za vpeljavo ti. varnostnih vtičnikov, ki poskrbijo za različne vidike varnosti spletnih aplikacij, ter seveda skrbi za sprotno namestitev vseh objavljenih varnostnih popravkov.

Nekaj koristnih virov za skrbnike spletnih strani WordPress:

https://wpvulndb.com/

https://WordPress.org/news/category/security/

https://www.cvedetails.com/vulnerability-list/vendor_id-2337/product_id-4096/

http://resources.infosecinstitute.com/7-best-WordPress-security-plugins/

Zakaj je WordPress lahko zelo nevaren?

Seveda je lahko sistem WordPress tudi zelo nevaren.

Nevarnost izhaja predvsem iz štirih dejstev.

1 Prvo dejstvo je neupoštevanje priporočil iz prejšnjega poglavja. V prejšnjem poglavju smo razložili in utemeljili, da je WordPress lahko zelo varna platforma, ker uporablja skripte, vtičnike in elemente, ki se uporabljajo po celotnem svetu in se tako v resnici dnevno varnostno preverjajo v resničnem svetu. Če pa skrbniki ne spremljajo informacij o novoodkritih ranljivostih in popravkov ne namestijo pravočasno, lahko varnost hitro postane nevarnost. Brez nameščenih popravkov lahko hitro postanemo tarča napadov, ki so bili v resnici ugotovljeni na drugih spletnih straneh. Še več, prav zaradi uporabe WordPressa brez popravkov lahko postanemo zanimiva tarča napadalcev, četudi sicer vsebinsko zanje nismo zanimivi. Če torej spremljamo novosti in pravočasno nameščamo popravke, smo v resnici z okoljem WordPress bolj varni kot z »doma narejenim« okoljem. Če pa tega ne delamo, pa smo v resnici bistveno manj varni.

2 Drugo dejstvo je predpostavljanje, da so vsi elementi široko uporabljeni. Ker je WordPress izjemno razširjen, za vsako funkcionalnost obstaja cela vrsta vtičnikov. Posledično vedno obstajajo vtičniki, ki se redkeje uporabljajo ali je njihov razvoj obstal. Pri tovrstnih vtičnikih se ne moremo v celoti zanašati na javno objavljene ranljivosti. Podatek o tem, koliko spletnih strani vtičnik uporablja, ter podatek o tem, kdaj je bila objavljena zadnja nadgradnja vtičnika, je viden.

> *Ko govorimo o varnosti spletne strani se izogibajte vtičnikom, ki niso pogosto uporabljeni ali kjer ni vidno, da se še razvijajo.*

3 Tretje dejstvo je zanemarjanje preostalih varnostnih načel. Sprotno posodabljanje strani WordPress in vtičnikov je vsekakor koristno, ni pa dovolj. Tipičen primer napada je namreč napad na skrbnika. Skrbnik spletne strani lahko poskrbi za vse popravke in nadgradnje, če pa pri tem zanemari varnost svojih gesel, varnost vstopnih točk za skrbniški dostop ipd., je lahko varnost spletne strani še vedno zelo ogrožena.

4 Četrto dejstvo se nanaša na zmožnost prilagajanja posameznih elementov. Prednost sistema WordPress je namreč v odprtokodnosti. To med drugim pomeni, da lahko skrbnik vedno tudi sam spremeni kakšno malenkost v vtičnikih/sistemu ter tako naredi celotno aplikacijo še uporabnejšo in še bolj prilagojeno svojim potrebam. A takšni posegi lahko prinašajo tudi nevarnosti. Priporočamo, da se posegi izvajajo zgolj takrat in zgolj na tak način, da se s tem ne vnaša novih varnostnih groženj.

Sistem WordPress je torej lahko tudi nevaren, posebej če ne upoštevate naslednjih 4 pomembnih priporočil:

- ne skrbite za spremljanje novo objavljenih ranljivosti ter sprotno nameščanje popravkov,
- ne uporabljate zgolj vtičnikov, ki so pogosto uporabljeni in za katere je razvidno, da razvoj ni zamrl
- zanemarite varnost skrbniškega dostopa (podrobneje se bomo tega vprašanja lotili v nadaljevanju),
- v kodo skript in elementov posegate samo, če ste prepričani, da s tem ne uvajate novih varnostnih tveganj.

Ali lahko skrijem, da je spletna stran narejena v okolju WordPress?

Preden se lotimo konkretnih ranljivosti in načinov napadov, najprej odgovorimo na zelo pogosto vprašanje: »Ali lahko pred napadalci skrijem dejstvo, da je moja spletna stran narejena v okolju WordPress?«

Kratek odgovor je: »Ne.«

1. Najbolj enostaven način, da »napadalec« ali uporabnik ugotovi prisotnost okolja WordPress, je s pregledom določenih privzetih spletnih naslovov. Primeri tipičnih tovrstnih naslovov:

http://spletna.stran/wp-login.php

http://spletna.stran/licence.txt

http://spletna.stran/wp-json/wp/v2

V resnici različni varnostni napotki govorijo o tem, da je smiselno te posamezne naslove URL skriti. Niti ne zato, da bi napadalcem skušali prikriti obstoj okolja WordPress, temveč zato, ker lahko preko teh naslovov napadalec izvaja različne nepooblaščene aktivnosti. Teoretično naslove lahko skrijemo pred napadalci. Res pa je, da jih je veliko, tako da je uspeh vprašljiv (podrobneje bomo o posameznih naslovih govorili v nadaljevanju pri posameznih konkretnih ranljivostih).

2. Drugi način ugotavljanja prisotnosti okolja WordPress je s pomočjo **videza in preko raznih podatkov**, ki nam jih ponuja **Google**. Dober poznavalec okolij WordPress bo namreč z veliko natančnostjo prepoznal določeno predlogo. Različni privzeti načini izpisa avtorjev strani, privzeti načini izpisa besedil ipd. lahko hitro izdajo, da gre za okolje WordPress. Hkrati lahko že enostavno iskanje z Googlom jasno pokaže, da je določena spletna stran povezana z določenim vtičnikom ali okoljem WordPress. Nenazadnje obstajajo tudi dodatki spletnim brskalnikom, ki z večjo ali manjšo natančnostjo ugotovijo, v katerem okolju je spletna stran izdelana.

3

Tretji način, ki pa je najkonkretnejši, je enostavno s pregledom kode HTML. Uporabnik/napadalec lahko pri obisku določene spletne strani enostavno izbere opcijo za prikaz izvorne kode. S tem se prikaže izvorna koda spletne strani, kjer se nahaja ogromno neposrednih ali posrednih navedb uporabe sistema WordPress in vse točke je praktično nemogoče skriti.

Če se torej vrnemo nazaj na izhodiščno vprašanje, je odgovor, da je **prisotnost sistema WordPress nemogoče prikriti.**

Varnosti sistema WordPress v resnici ne zagotavljamo s skrivanjem, ampak na druge načine, ki jih bomo podrobneje opredelili v naslednjih poglavjih.

DRUGO POGLAVJE

GLAVNI NAČINI NAPADA NA SPLETNE STRANI WORDPRESS

Glede na cilj napadalca jih lahko razdelimo v 4 skupine, razvrščene po pomembnosti.

1. Pridobitev skrbniških pravic na spletni strani
2. Vnos škodljive kode brez prevzema skrbniških pravic
3. Zloraba raznih znanih varnostnih ranljivosti sistema in vtičnikov
4. Uporaba vseh preostalih načinov napadov, ki jih poznamo pri spletnih aplikacijah

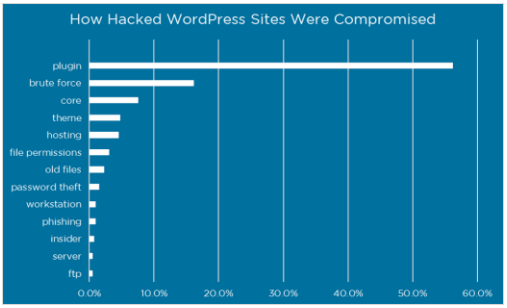
1 **Prevzem skrbništva strani** je za napadalce vedno zanimiv izziv. S prevzemom skrbništva lahko napadalci neovirano posegajo v vsebino spletne strani, na voljo jim je vpogled v vse skripte, hkrati pa lahko na spletno stran prosto nalagajo svoje zlonamerne programe.

Skrbništvo se lahko prevzame na dva načina. Prvi način je napad na prijavno okno. Napadalec mora najprej prijavno okno identificirati, nato pa mora uganiti ali na drugačen način ugotoviti pravilno uporabniško ime skrbnika in njegovo geslo.

2 Drugi način je preko **podatka o sejnem piškotku**. S stališča uporabnikov je preverjena pristnost tistega uporabnika, ki pravilno vpiše svoje uporabniško ime in geslo. S stališča aplikacije pa je v resnici preverjena pristnost tistega uporabnika, ki v svojih paketih pošilja pravilne podatke o ID seje. Napadalci lahko torej ciljajo na krajo ali zlorabo tega podatka iz piškotka z ID seje, ki bi jim na enak način kot uporabniško ime in geslo aplikacije omogočil skrbniški dostop.

Ena izmed logičnih posledic uspešnega napada na skrbniški dostop je, da lahko napadalec na spletno stran naloži poljubno škodljivo kodo. Lahko pa napadalec izbere obratno pot. Preko ranljivosti v skriptah ali drugih načinov v spletno stran najprej vnese zlonamerno kodo, ki mu nato omogoči dostop do skrbništva. Možne načine takšnih zlorab si bomo podrobneje ogledali v naslednjih poglavjih.

3 Tretji način napada je preko **znanih ranljivosti različnih vtičnikov.** Tipično te ranljivosti bodisi slonijo na uporabi ranljivih ukazov PHP, preko katerih si lahko napadalec zagotovi možnost vnosa svoje kode ali kako drugače zlorabi strežnik, bodisi gre za ranljivosti, ki omogočajo vrivanje zlonamernih podatkov, s katerimi se nato izvedejo napadi, kot



so vrivanje SQL (SQL injection), Cross-site scripting, Cross-site forgering ipd. Seveda so vedno aktualne tudi vse ostale ranljivosti, ki jih srečujemo pri spletnih aplikacijah.

4 Četrti način napada je z uporabo **metodologije napadov, ki velja za napade na neznano spletno aplikacijo**. Ta metoda na prvi pogled nima smisla, saj daje vtis, kot da je napadalec spregledal, da gre za spletno stran, izdelano v sistemu WordPress. Napadalec izvaja preizkuse oz. napade na funkcionalnosti, za katerimi stojijo vtičniki, pri katerih so bili takšni napadi že neštetokrat poizkušeni. A vseeno obstajajo primeri, ko imajo taki poizkusi smisel. Če je razvijalec posegal v kodo skript (na primer, da bi izboljšal funkcionalnosti), potem ima smisel preveriti vse od začetka. In nenazadnje, enkrat je ranljivost treba odkriti prvič, za kar je ta metoda najprimernejša.

Obstaja še **peti način napada**, ki je enako pomemben in nevaren, a presega doseg tega dokumenta. Spletna stran WordPress namreč deluje na strežniku in napad na ranljivosti samega strežnika ali pa zloraba obhodnih dostopov (recimo skrbnik ima za lažji prenos datotek vzporedno odprt še dostop FTP do strežnika) lahko enako uničujoče pripelje do uspešne zlorabe.

Kako lahko napadalec prevzame skrbništvo spletne strani?

Obstajata dva glavna načina napada na skrbništvo.

1 Prvi je preko prijavnih oken skrbnika, drugi pa preko podatkov v piškotku z ID seje. Ker je WordPress dokaj dobro zavarovan pred krajo podatkov v piškotku z ID seje (vsaj če uporabljamo varne in posodobljene različice vtičnikov), se bomo **osredotočili predvsem na napade preko prijavnih oken**.

Razmišljanje napadalca se pri tem vrti okrog naslednjih petih vprašanj:

- Ali je prijavno okno sploh dostopno?
- Ali prijavno okno omogoča ugibanje pravilnih uporabniških imen?
- Ali prijavno okno omogoča ugibanje gesel?
- Ali je možno prisluškovati prometu in posledično odkriti skrbnikovo geslo?
- Ali je možno skrbnika zavesti z lažnim prijavnim oknom, da nam tako nehote izda svoje geslo?

Najbolj pogosti napadi na prevzem skrbništva se naslanjajo na prva tri vprašanja. Če se torej napadalec želi v sistem WordPress prijaviti kot skrbnik, mora poznati tri podatke:

- točen naslov prijavnega okna,
- uporabniško ime skrbnika in
- geslo skrbnika.

Kje je prijavno okno? Privzeta namestitev sistema WordPress ima točno določeno mesto za prijavno okno. To je http://spletna.stran/wp-login.php (ali nekaj zelo podobnega). Ta naslov je mogoče pred napadalci zelo učinkovito skriti (več o tem v nadaljevanju), vendar praksa kaže, da veliko skrbnikov tega ne naredi.

Katero je pravilno uporabniško ime skrbnika? Ta del je za napadalca še najbolj enostaven. Tudi če ne štejemo zelo pogoste prakse, ko skrbniki uporabljajo uporabniška imena , ki jih je enostavno uganiti (admin, tone - napadalec na primer točno ve, da gre za skrbnika ipd.), ima napadalec še

vedno na voljo veliko možnosti, da ugotovi pravilni podatek. Najbolj pogosta napaka skrbnikov je izdajanje pravilnosti uporabniškega imena ob vpisu v prijavno okno (prijavno okno »prijazno« opozori napadalca, da uporabniško ime ni pravilno) ali pa uporaba funkcije pozabljenega gesla (Forgot password), ki je samodejno vklopljena na prijavnih oknih. Četudi prijavno okno neposredno ne izda, ali je vpisano uporabniško ime pravilno ali ne, funkcija pozabljenega gesla izda napadalcu, ali določen uporabnik sploh obstaja ali ne. Poleg tega ima napadalec na voljo še kar nekaj trikov, kako lahko preko spletne strani ugotovi pravilna uporabniška imena.

Najbolj pogosto uporabljeni triki so:

- Napadalec napade spletno stran »spletna.stran«. Že enostavno iskanje z Googlom s ključnimi besedami »spletna.stran author« bo zelo verjetno podalo pravilno uporabniško ime.
- Vpis naslova URL http://spletna.stran/?author=0, kjer napadalec spreminja vrednost spremenljivke author=1, author=2 ipd., kar prav tako zelo verjetno poda pravilna uporabniška imena.
- Klic funkcije JSON http://spletna.stran/wp-json/wp/v2/users. WordPress od različice 4.7 dalje nudi klice JSON, ki so namenjeni pridobivanju podatkov iz spletne strani za razne skripte in aplikacije. Znotraj teh klicev so tudi klici, ki podajajo informacijo o avtorjih posameznih zapisov, med temi avtorji pa se praviloma nahajajo tudi skrbniki.
- Klic funkcije http://spletna.stran/feed. WordPress ponuja funkcijo virov RSS, ki so namenjeni avtomatizaciji spremljanja novih objav na spletni strani. Ta izpis med drugim v polju Creator izdaja uporabniška imena avtorjev, med katerimi so praviloma tudi skrbniki.

Nekaj načinov ugotavljanja pravilnih uporabniških imen je možno relativno enostavno preprečiti (podrobneje o tem bomo govorili v nadaljevanju dokumenta). Vendar jih ostane dovolj, da lahko predpostavimo, da bo napadalec tako ali drugače pravilno ugotovil uporabniško ime skrbnika.

Kakšno je geslo skrbnika? Vedno obstaja možnost, da napadalec do tega podatka pride s tradicionalnimi metodami, ki veljajo za vse spletne aplikacije:

- Prisluškovanje prometu. Podrobneje bomo o tem govorili v nadaljevanju dokumenta. Pomembno je razumeti, da je treba pri dostopih skrbnika uporabljati protokol SSL in to na način, ki ne odpira ranljivosti SSL.

- Napad z lažnimi stranmi (phishing). Napadalec lahko vedno sproži napad na skrbnika z lažnim e-poštnim sporočilom, s katerim ga prepriča k obisku lažnega portala in tako pridobi geslo. Tehnično gledano takšne napade težko v popolnosti preprečimo, velja pa tukaj računati na večjo varnostno osveščenost skrbnikov, ki ne bi smeli kar tako dopustiti uspešnosti takšnih napadov.
- Geslo se ugotovi nekje drugje. Tipičen primer je, ko skrbnik na različnih sistemih uporablja enaka ali zelo podobna gesla. Napadalec bi lahko z različnimi metodami geslo ugotovil na drugem sistemu, ki je manj zavarovan, ter ga nato s pridom uporabil na sistemu WordPress.

Oglejmo si podrobneje problematiko ugibanja gesel v sistemu WordPress

V tem pogledu je WordPress lahko dokaj varen, saj pri ustvarjanju uporabnika ponudi 24-mestno geslo, ki ga praktično ni mogoče uganiti. Takšno geslo že samo po sebi zmanjšuje možnost ugibanja, vendar ga v praksi uporabljajo le redki skrbniki.

Najbolj očitno mesto za ugibanje gesel je **prijavno okno skrbnika**. V privzeti namestitvi to okno ne ponuja mehanizmov za preprečevanje ugibanja. Skrbniki imajo sicer na voljo kar nekaj mehanizmov, s katerimi lahko uganjevanje preprečijo ali vsaj bistveno otežijo – funkcionalnost CAPTCHA, zaklepanje dostopa ob večkratnih napačnih prijavah ipd. To so učinkoviti mehanizmi, ki pa so v praksi le redko uporabljeni.

Še najbolj resen napad na gesla skrbnikov je z uporabo metode xmlrpc.php. Xmlrpc.php je skripta, ki se pri privzeti namestitvi nahaja na sistemu WordPress. Ta skripta je namenjena klicanju spletnih storitev. Med napadalci sistemov WordPress je postala zelo priljubljena, saj v resnici omogoča izvajanje napadov z grobo silo na gesla skrbnikov. Ta skripta namreč kliče tudi določene funkcije, pri katerih sistem zahteva vpis pravilnega gesla. S klicanjem teh funkcij lahko napadalec torej ugiba pravilno geslo. Takšen napad je za napadalce posebej zanimiv, saj:

- Zaobide določene varovalke ugibanja, ki jih je skrbnik morebiti postavil na vstopno okno (CAPTCHA, blokiranje naslovov IP ipd.).
- V en klic lahko napadalec vstavi večje število funkcijskih klicev, tako da je tak napad bistveno hitrejši od tradicionalnega ugibanja na vstopnem oknu, kjer je vsak klic vezan le na en poizkus.

Kako točno se v praksi zavarujemo pred takšnimi metodami napadov, bomo podrobneje opisali v nadaljevanju dokumenta.

Zakaj je prevzem skrbništva spletne strani velika grožnja?

Zakaj naj nas prevzem skrbništva sploh skrbi.

Skrbniki in lastniki spletnih strani **o grožnjah velikokrat razmišljajo v kontekstu vloge**, ki jih spletna stran v informacijskem sistemu sploh ima.

Če je spletna stran preko raznih funkcionalnosti neposredno povezana z informacijskim sistemom (povezava s podatkovnimi zbirkami, dostop za stranke, e-trgovina ipd.), se skrbniki praviloma zavedajo, da je prevzem skrbništva velika grožnja.

Če pa je spletna stran enostavna, namenjena predstavitvi podjetja ter gostuje pri tretji osebi (torej izven internega informacijskega sistema), pa skrbniki ter lastniki velikokrat zanemarjajo tveganja. Zelo pogosto naletimo na naslednje izjave in pomisleke:

- Saj vdor na spletno stran še ne pomeni vdora v moje pomembne podatke.
- Moja spletna stran ni ključna za poslovanje podjetja.
- Na spletni strani zgolj predstavljamo splošno vsebino. Zakaj bi bilo komu zanimivo, da v to sploh posega …

V resnici so **nameni napadalcev** zelo pogosto popolnoma drugačni. Vsaka, še tako enostavna in »nedolžna« spletna stran, napadalcem omogoča naslednje:

- Napad na notranje uporabnike podjetja
- Napad na ostale uporabnike spletne strani (stranke, dobavitelje ipd.)
- Izkoriščanje spletnega prostora za nelegalne dejavnosti tretjih oseb

1 Oglejmo si nekaj primerov. Napadalec prevzame skrbništvo nad spletno stranjo in nanjo namesti zlonamerno kodo. S ponarejenimi e-poštnimi sporočili ali podobnimi prijemi socialnega inženiringa nato prepriča interne uporabnike podjetja, da obiščejo lastno spletno stran. Ker uporabniki spletni strani zaupajo, bodo zelo verjetno brez zadržkov klikali na podstrani in odpirali datoteke. Pogosto se zgodi, da napadalec zgolj čaka, da jo uporabniki obiščejo sami, in

se ne izpostavlja z zavajanjem uporabnikov k obisku. Takšen namen napadalca je enako nevaren, tako za notranje uporabnike podjetja kot za ostale obiskovalce spletne strani. Na primer, znan je napad na poljske banke konec leta 2016. Napadalci so takrat prevzeli skrbništvo nad »nedolžno« spletno stranjo poljske vladne agencije za regulacijo bančnega trga (www.knf.gov.pl). Agencija je imela tradicionalno spletno stran, na kateri so se nahajale zgolj nekatere formalne informacije o agenciji. Napadalci so s prevzemom skrbništva na spletno stran podtaknili zlonamerno kodo. Spletno stran so med drugim občasno obiskovali tudi uporabniki različnih bank in napadalci so postopoma preko te spletne strani uspeli vnesti zlonamerno kodo v več kot 20 poljskih bank. Napad še danes velja za enega največjih napadov na Poljskem.

2 Drug primer je namera napadalcev, da postavijo lažni strežnik za zlorabo sistema PayPal. Obstajajo razlogi, zakaj napadalci lažni strežnik raje postavijo na vašo spletno stran. Na ta način lažje skrijejo sledi in se izognejo raznim črnim listam sumljivih spletnih strežnikov. Naša spletna stran še vedno nemoteno deluje, tako da je velika verjetnost, da napada kar nekaj časa sploh ne opazimo. Seveda si lahko vsak predstavlja celo vrsto tehničnih, pravnih in poslovnih težav, ki jih takšna zloraba povzroči podjetju.

Kako zavarovati skrbniško prijavno okno?

11 stvari, ki jih mora pri zavarovanju prijavnega skrbniškega okna urediti skrbnik

1. Prijavno okno za skrbnika naj bo napadalcu nedostopno. To lahko dosežemo s kombinacijo dveh nastavitev:
    - Naslov URL za prijavno okno lahko skrijemo za določen edinstven naslov URL, ki ga pozna zgolj skrbnik. Recimo, http://spletna.stran/izmišljeniznaki.
    - Dostop do prijavnega okna omejimo na točno določene naslove IP. To sicer ni vedno možno, a skrbnik bi moral biti dovolj strog, da dostopa do prijavnega okna ne dopušča od kjerkoli zgolj zato, ker bi morebiti kdaj v prihodnosti želel dostopati iz neznane lokacije.
2. Uporabniško ime skrbnika se spremeni v nestandardno ime. Privzeti skrbnik v sistemu je »admin«, kar seveda ni varno.
3. Uporabnike se loči na skrbnike in urednike. Za vpis nove vsebine v spletno stran skrbniške pravice niso potrebne. WordPress namreč omogoča veliko načinov, s katerimi lahko napadalec izve ime avtorja določene novice. Če je to hkrati tudi ime skrbnika, je varnost deloma že ogrožena.
4. Dejanska imena za prijavo naj se razlikujejo od prikazanih imen, saj WordPress ločuje prijavno ime od prikazanega imena. Napadalec lahko ime avtorja ugotovi na različne načine. Če je to ime v resnici različno od imena za prijavo, si s tem napadalec ne more veliko pomagati.

5. V primeru večkratnih zaporednih napačnih prijav se dostop za uporabnika/napadalca na prijavnem oknu za določen čas zapre.
6. Na prijavno okno se vpelje sistem CAPTCHA.

7. Datoteko xmlrpc.php bodisi izbrišite iz sistema bodisi jo preimenujte. Pri tradicionalnih spletnih straneh datoteka praviloma sploh ni potrebna. Potrebna je pri bolj naprednih spletnih straneh, kot so tiste, ki uporabljajo tudi aplikacijo za mobilne telefone. Datoteko je mogoče preimenovati v nekaj, kar napadalci težko uganejo.
8. Možnost pozabljenega gesla (Forgot Password) se po možnosti izklopi.
9. Skrbniški dostop mora potekati po povezavi SSL, ki je nastavljena na varen način. Zgolj privzeta povezava SSL še vedno vsebuje ranljivosti v smislu uporabe slabih nastavitev ali algoritmov, ki v resnici napadalcem kljub uporabljeni zaščiti SSL omogočajo prebiranje vsebine prometa.
10. Pri geslih se uporabljajo kompleksna gesla, dolga najmanj 10 znakov. Pri tem se lahko uporabi funkcionalnost WordPress, ki kompleksno geslo določi namesto nas.
11. Potrebno je spremljanje uspešnih in neuspešnih prijav v sistem in po potrebi ustrezno ukrepati.

Vse navedene metode je možno izvesti s pomočjo vtičnika WP Security, ki ga bomo podrobneje predstavili v nadaljevanju dokumenta. Pri tem je treba razumeti, da popolnoma vse navedene varovalke niti niso nujno potrebne, saj nekatere hkrati rešujejo več varnostnih izzivov (če uporabljamo kompleksna gesla, potem sistem CAPTCHA za prijavo ni potreben). Vsekakor naj skrbnik sledi filozofiji »Če nisi popolnoma prepričan, je bolje uporabiti več varovalk kot manj«.

Zakaj je vrednost piškotka z ID seje enako pomembna pri zavarovanju skrbniškega dostopa do aplikacije?

Druga metoda napadalcev za prevzem skrbniških pravic je preko prevzema dokumenta piškotka z ID seje. Spletne aplikacije (ne glede na to, ali gre za WordPress ali ne) imajo namreč specifično metodo preverjanja pristnosti.

Pri preverjanju pristnosti se praviloma pogovarjamo o uporabniških imenih in geslih, kar poudarjamo tudi v tem dokumentu. V resnici preverjanje pristnosti s stališča aplikacije poteka malce drugače.

Osnovni mehanizem pri komunikaciji preko protokola IP je seja TCP. TCP ima vgrajen mehanizem, ki zagotavlja, da je komunikacija med uporabnikom in strežnikom na nek način varna. Ko je seja vzpostavljena, napadalec težko poseže vanjo in jo prevzame. Aplikacija bi tako zgolj morala poskrbeti, da na začetku seje TCP uporabnik ustrezno potrdi svojo pristnost, kar pomeni, da napadalec ne bo mogel nepooblaščeno poseči v aplikacijo. Za spletne aplikacije pa velja neka posebnost - komunikacija uporabnika s strežnikom ni ena sama seja TCP, ampak je sestavljena iz mnogo sej. Enak princip preverjanja pristnosti bi bil tako zelo nepraktičen, saj bi moral uporabnik vsakih 5 sekund ponovno potrditi svojo pristnost. Spletne aplikacije zato uporabljajo mehanizem piškotkov. Ob začetni uspešni potrditvi pristnosti aplikacija brskalniku uporabnika posreduje enolično in vrednost piškotka z ID seje, ki je ni mogoče uganiti. Brskalnik nato ob začetku vsake seje TCP v zaglavje paketov http doda to vrednost. Tako aplikacija ve, da nova seja TCP prihaja od uporabnika, ki je že potrdil svojo pristnost.

Sliši se enostavno in tudi je, vendar tak način dela prinaša dodatne varnostne izzive, saj imajo napadalci novo priložnosti za možne napade. Napadalec lahko uporabniku

poizkusi vrednost piškotka ukrasti. Če pri tem uspe, lahko do aplikacije dostopa kot potrjen uporabnik, ne da bi moral poznati uporabniško ime in geslo.

Pri zavarovanju skrbniškega dostopa je treba poskrbeti tako za varnost prijavnega okna, kar smo opisali v prejšnjem poglavju, kot tudi za varnost vrednosti piškotka z ID seje.

Teoretično bi lahko napadalec vrednost piškotka z ID seje enostavno uganil, a pri straneh WordPress ta možnost ni realna. Tako mu preostaneta dva načina, da pride do tega podatka:

- piškotek z ID seje pridobi s prisluškovanjem povezavi med skrbnikom in aplikacijo,
- piškotek z ID seje ukrade s pomočjo zlonamernih skript v jeziku JavaScript.

Kot smo že omenili, prisluškovanje preprečimo z uporabo protokola SSL. Vendar SSL pri varovanju piškotkov ne zadostuje. Napadalec lahko skrbniku posreduje naslov URL s spletne strani, ki namesto na https kaže na http. Če napadalec uspe skrbnika zavesti, da ta klikne na povezavo, bo brskalnik skrbnika proti aplikaciji poslal nešifriran paket. Aplikacija bo ta paket zavrnila, saj pričakuje povezavo SSL, napadalec pa ga lahko prestreže in iz njegovega zaglavja enostavno prebere vrednost piškotka z ID seje

Zlonamerne skripte v jeziku so vezane na ranljivosti XSS (Cross-Site Scripting), ki so pri spletnih straneh vedno iskane in sodijo med resnejše ranljivosti.

Kaj točno je ranljivost XSS in kako se zavarujemo v primeru obeh navedenih napadov, bomo podrobneje obrazložili v naslednjem poglavju.

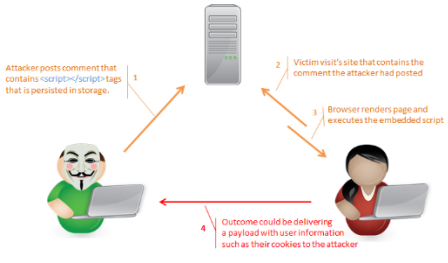
Kaj je XSS napad in kako sploh zavarujemo vrednosti piškotka?

V prejšnjem poglavju smo navedli dva primera napada, katerih cilj je ukrasti piškotek z ID seje.

1 Prvi omenja možnost, da napadalec do vrednosti v piškotku pride s prisluškovanjem. Ta napad preprečimo z naslednjimi mehanizmi:

A. Uporaba protokola SSL. To mora veljati za vsaj tisti del aplikacije, ki obravnava dostop skrbnika.
B. Uporaba varnih protokolov SSL. Kaj to pomeni, podrobneje razlagamo v naslednjem poglavju. Skrbniki se morajo zavedati, da protokol SSL sam po sebi ni dovolj. Če ga uporabljamo napačno, so povsem pričakovani predvsem napadi, ki kljub SSL omogočajo krajo piškotka z ID seje.
C. Uporaba nastavitve piškotka z ID seje. To je dodaten parameter v zaglavju, ki brskalnikom predpiše pravilo, da vrednosti piškotka z ID seje nikoli ne pošljejo preko nezavarovane povezave http. Napad, opisan v prejšnjem poglavju, s tem učinkovito preprečimo.

2 Drug napad na piškotek z ID seje poteka preko ranljivosti XSS. Kaj sploh je ranljivost XSS? Spletna aplikacija deluje tako, da ob vsakem klicu uporabniku posreduje kodo HTML, ki jo brskalnik pretvori v uporabniku vidno spletno stran. Stranem HTML so dodani tudi klici JavaScript, ki stranem zagotavljajo dinamične možnosti in celo vrsto funkcionalnosti. Brez teh klicev si danes uporabe spletnih aplikacij ne moremo več predstavljati.



Problem nastane v primerih, ko je vsebina strani HTML (z vključenimi ukazi JavaScript) odvisna od tega, kaj je uporabnik v prejšnjih klicih posredoval aplikaciji (na primer: v iskalnik vpišemo besedo »test« in aplikacija vrne stran HTML z besedilom »rezultati iskanja besede 'test'«). Če bi torej napadalec uspel aplikaciji namesto pričakovanih besed posredovati zlonamerne ukaze JavaScript, bi jih aplikacija morebiti uporabila v svojem odgovoru HTML, ki bi se nato izvedli v brskalniku uporabnika/žrtve. Imajo pa ukazi JavaScript vseeno omejitve glede zmožnosti povzročanja škode. Neposredno delovni postaji žrtve ne morejo škodovati,

lahko pa dostopajo do piškotka z ID seje in prav zato se v 99 odstotkih primerov napad XSS osredotoča na poizkus kraje tega podatka.

Kako skrbnik prepreči takšne napade na spletno aplikacijo?

1. Spletna koda mora vse vhodne podatke uporabnika vedno in brez izjeme podrobno preveriti, preden jih uporabi v nadaljnjih aktivnostih.
2. Vrednost piškotka z ID seje se označi kot »http-only«. gre za dodaten mehanizem, ki je zelo učinkovit, a ga spletne aplikacije na žalost ne uporabljajo redno. Ta oznaka enostavno in učinkovito sporoči brskalniku, da piškotek z ID seje pač ni dosegljiv skriptam v jeziku JavaScript.
3. Izklopiti je treba metodo TRACE na spletnih strežnikih. Gre za metodo, ki se praviloma uporablja za razhroščevanje spletnih strežnikov in je dokaj priročno orodje med razvojem aplikacije. Do težave pride, ker imajo določene spletne strani to vklopljeno tudi na produkcijskih sistemih. Znani so namreč načini, kako lahko napadalec vseeno pride do piškotka z ID seje vrednosti pri vklopljeni metodi TRACE, četudi je spletna stran uporabljala mehanizem http-only.

Zavarovanje pred napadi na piškotek z ID seje se torej izvaja preko pravilne postavitve prometa SSL (podrobneje o tem govorimo v naslednjem poglavju), z ustreznimi nastavitvami vrednosti piškotka (podrobneje o tem govorimo v poglavju Ostali vtičniki za varnost), ter preko zagotavljanja, da naši vtičniki in programska koda WordPress na splošno ne vsebuje ranljivosti XSS (to dosežemo tako, da pozorno spremljamo takšne ranljivosti pri uporabljenih vtičnikih – kar je podrobneje obrazloženo v poglavju Glavni viri za spremljanje varnosti sistema WordPress.

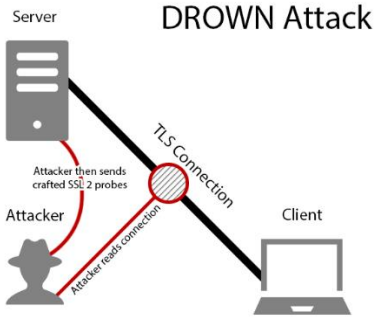
Kako pravilno zavarovati promet SSL?

SSL je mehanizem, ki poskrbi za šifriranje prometa. Prisluškovanje prometu med uporabnikom in aplikacijo sicer ni enostavno, zaradi česar v praksi vidimo, da marsikatera spletna stran zgolj vzpostavi povezavo SSL zato, da je vzpostavljena. Vendar je resnica drugačna.

Obstajajo namreč okoliščine, kjer je prisluškovanje lahko zelo realna možnost. Tipični primeri takšnih okoliščin so:

1. Prisluškovanje na javnih brezžičnih dostopnih točkah (Wi-Fi). S tehničnega vidika je zmožnost prisluškovanja na takšnih točkah sicer odvisna od nastavitev vstopnih točk Wi-Fi in ravno v tem tiči problem. Marsikatera vstopna točna je nastavljena tako, da lahko napadalec brez večjih težav promet drugega uporabnika preusmeri na svojo napravo.
2. Prisluškovanje v hotelskih sistemih. Logika in težave so podobne kot pri javnih brezžičnih dostopnih točkah. Marsikateri hotel ima vstopno točko v internet nastavljeno na način, ki spretnim napadalcem omogoča, da promet nekega uporabnika preusmerijo preko svoje naprave.
3. Prisluškovanje znotraj lokalnih omrežij. Tipičen primer je prisluškovanje sodelavcu v službenem omrežju. Znotraj lokalnega omrežja lahko spletni napadalec promet relativno enostavno preusmeri preko svojih naprav.

Pri zmožnostih prisluškovanja poznamo dve vrsti zlorab. Prva se nanaša na stanje, ko lahko napadalec zgolj pride do šifriranega prometa, druga pa, ko se lahko napadalec dejansko vmeša v promet (postavi se med uporabnika in aplikacijo). V vseh zgornjih primerih je pri resnejših napakah v nastavitvah omrežne opreme mogoče izpeljati drugo vrsto zlorab. Če pa napake niso tako resne, lahko napadalec morebiti izvede zgolj prvo vrsto zlorabe.



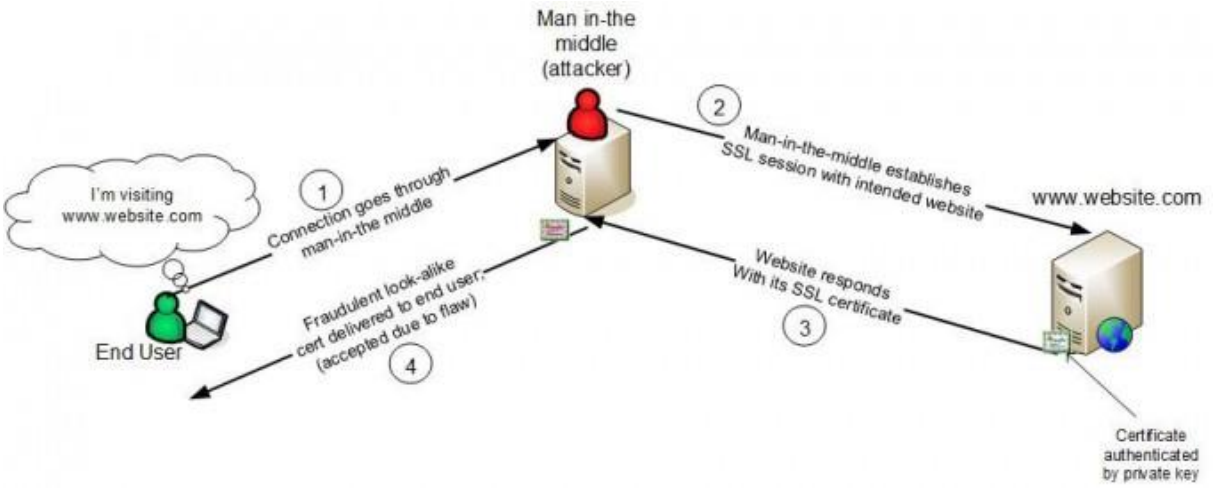
V primerih, ko govorimo zgolj o možnosti prisluškovanja brez dejanskega aktivnega vmešavanja v promet, je treba zagotoviti, da šifriranja povezav SSL dejansko ni mogoče razbiti.

Napake, ki lahko naredimo pri tem, so naslednje:

1. Uporaba slabih protokolov, kot so TLS 1.0, SSLv3 in SSLv2. Znani so napadi, kot so DROWN ali PODDLE, ki omogočajo dešifriranje prometa ob uporabi teh protokolov.
2. Uporaba slabih šifrirnih algoritmov. Določeni algoritmi so znani kot ranljivi. Po eni strani gre bolj za matematično ranljivost, saj današnja procesorska moč računalnikov omogoča realno dešifriranje določenih starejših algoritmov, posebej če se pri tem uporabljajo prekratki ključi.
3. Uporaba napačnih nastavitev. Tipične tovrstne nastavitve so:
    a. TLS compression. Obstajajo določeni napadi (napada s paradoksom rojstnih dni ali birthday attack, na primer), ki so lahko uspešni, če je vklopljeno stiskanje.
    b. TLS renegottionation. Obstajajo napadi, kjer bi lahko napadalec v že vzpostavljeni seji SSL povzročil, da se algoritmi in protokoli v seji ponovno opredelijo (pri čemer je namen seveda, da se uporabijo slabši algoritmi oz. protokoli).
    c. TLS fallback SCVS. Praviloma se brskalnik in aplikacija vedno dogovorita za čim boljše protokole in algoritme, ki jih podpirata obe strani. Napačna rezervna nastavitev pa v določenih primerih napadalcu omogoča, da to pravilo zaobide.

Kako zagotoviti navedene nastavitve oz. kako sploh preveriti, katere od teh težav ima naša aplikacija, je navedeno v zadnjem poglavju.

V primerih, ko se lahko napadalec dejansko aktivno postavi v komunikacijo med uporabnikom in aplikacijo, pa je stanje bolj resno. V takšnih primerih napadalec celotno komunikacijo razdeli na dva dela. Prvi del poteka med uporabnikom in napadalcem, drugi del pa med napadalcem in aplikacijo. Ob tem je uporabnik prepričan, da komunicira neposredno z aplikacijo in obratno. V takšnih primerih lahko napadalec z dobro zastavljenim napadom pregleduje vsebino prometa, četudi je protokol SSL vzpostavljen brez napak. Napadalec je namreč v takšnem primeru pri komunikaciji prisoten že ob vzpostavitvi povezave SSL, tako da verjetno v vsakem primeru lahko pridobi vse potrebne tehnične podatke in parametre, s katerimi lahko povezavo enostavno dešifrira, dobi vpogled v vsebino in jo po potrebi šifrira nazaj.



Kako lahko preprečimo tovrstne zlorabe?

Osnovno načelo je sicer že vgrajeno v protokole SSL, vendar se v praksi izkaže kot neučinkovito. Praviloma imajo vsi strežniki svoje digitalne certifikate, namen teh certifikatov pa je predvsem omogočiti uporabniku, da se prepriča, da dejansko dostopa do pravega strežnika. Protokol SSL tako na eni strani vidi, kaj je ciljna stran (recimo www.website.com), na drugi strani pa se povezava SSL vzpostavlja s strežnikom, ki ima določen digitalni certifikat. Znotraj tega digitalnega certifikata je navedeno ime DNS strežnika, ki mu je bilo določeno ob dodelitvi certifikata. Če se ta dva podatka ujemata, potem je vse v redu, sicer pa uporabnik dobi običajno obvestilo, da povezava ni varna. Do težav prihaja, ker številni skrbniki in marsikatera spletna stran teh nastavitev nimajo pravilno uporabljenih (uporablja se napačen certifikat ipd.), zaradi česar so uporabniki tovrstnih opozoril že več kot vajeni in za marsikoga ni nič nenavadnega, če takšno opozorilo z nekaj kliki na V redu In Sprejmi izjemo enostavno zaobide.

V primeru zgoraj opisanega napada, kjer se napadalec v resnici nahaja med uporabnikom in aplikacijo, gre za ravno tak primer. Uporabnik želi dostopati do spletne strani www.website.com, vendar se bo njegova povezava SSL v resnici zaključila na računalniku napadalca. Ker ta računalnik seveda nima digitalnega certifikata, ki bi bil vreden zaupanja in bi trdil, da je to računalnik z naslovom www.website.com, se bo uporabniku ob vzpostavitvi povezave pojavilo opozorilo, da »gre za nezaupanja vredno povezavo«.

Obstaja glava STSH (Strict Transport Security Header). To je nastavitev na strani aplikacije, ki brskalnikom učinkovito pove, naj aplikacije ne obiskujejo preko sumljivih povezav. S takšno nastavitvijo bo brskalnik uporabniku enostavno preprečil dostop do

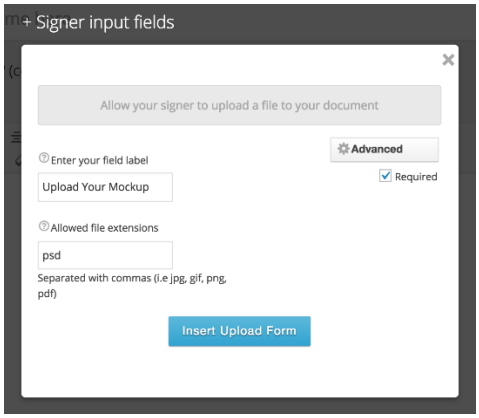
nezaupanja vredne povezave. Res je, da to v praksi (lahko) pomeni, da uporabnik tudi v »nedolžnih« primerih ne bo mogel dostopati do aplikacije. Do tega lahko pride, na primer, ko bo uporabnik poizkušal dostopati do aplikacije preko hotelskega sistema proxy, ki bo po nepotrebnem posegal v povezavo SSL, ne da bi to dejansko izkoriščal napadalec v hotelu. Toda če gre za zelo pomembne aplikacije in pomembno vsebino, je to v resnici pravilen način delovanja aplikacije.

Kako lahko napadalec v spletno stran svojo kodo vnese brez prevzema skrbništva?

Predhodna poglavja so bila predvsem namenjena napadom na spletno stran, kjer je cilj napadalca prevzeti skrbništvo. Razloženo je bilo, da je eden izmed pomembnih razlogov, zakaj napadalci sploh želijo priti do skrbništva, nepooblaščeno nalaganje programov, skript in kode na spletni strežnik. Kako pa lahko napadalec naloži svojo kodo, ne da bi pred tem dejansko prevzel skrbništvo?

Tipičen primer je z uporabo skript PHP. Programski jezik PHP vsebuje tudi ukaze, ki dejansko dostopajo do datotečnega sistema strežnika. Seveda je takšna funkcionalnost ob običajni rabi več kot potrebna (potrebna je za možnost nalaganja fotografij ipd.). Ob nepravilni uporabi teh ukazov v skriptah PHP pa se skriva priložnost, da napadalci takšne funkcije zlorabijo.

WordPress takšne funkcije uporablja že sam po sebi, saj je razumljivo, da mora imeti skrbnik možnost, da na strežnik nalaga datoteke. Uporabljajo jih tudi posamezni vtičniki, ki spletnim stranem dodajajo razne funkcije, kot je možnost nalaganja fotografij uporabnikov.



Pri vtičnikih se lahko zgodi, da slabo napisana koda vtičnika ne preveri dovolj natančno datoteke, ki jo želi uporabnik naložiti na strežnik. Če ta namesto pričakovane fotografije v resnici vsebuje zlonamerne ukaze PHP, je spletna stran v resnih težavah. Preverjanje vsebine datotek sicer ni enostavno, saj ni dovolj, da skripta, na primer, preverja zgolj končnice datotek ali osnovne podatke v glavi datoteke. Zato ne preseneča dejstvo, da se pri vtičnikih pogosto pojavijo tovrstne ranljivosti. Že nekajkrat smo poudarili, da je torej naloga skrbnika, da budno spremlja odkrite ranljivosti na vtičnikih, ki jih uporablja. Če se v javnosti pojavi tovrstna ranljivost, mora vtičnik seveda nemudoma posodobiti ali odstraniti.

Velika tovrstna ranljivost, pa se je pred kratkim pojavila tudi v samem sistemu WordPress. Namreč z različico 4.7 je WordPress predstavil klice JSON, ki so v prvi vrsti namenjeni avtomatizaciji dostopov do vsebin (za razne vmesnike, aplikacije ipd.). Tako lahko recimo preko klica JSON http://spletna.stran/wp-json/wp/2/posts/{post_id} (glede na dejansko namestitev, je lahko naslov URL malce drugačen) dostopamo do vsebine objave. Ti klici omogočajo tudi ukaze POST, kar pomeni, da so namenjeni tudi urejanju vsebin in ne zgolj pregledovanju. **Tu pa se je skrivala velika ranljivost.**

Na kratko, klic JSON je mogoče izpeljati na dva načina. Prvi je http://spletna.stran/wp-json/wp/2/post/123, drugi pa http://spletna.stran/wp-json/wp/2/post/123?id?123. Izkazalo se je, da WordPress v prvem primeru dovolj natančno preveri vsebino klica, v drugem pa ne. Dodatno se je izkazalo tudi, da WordPress daje prednost drugemu načinu klica. Če bi napadalec torej poklical ukaze JSON s sintakso http://spletna.stran/wp-json/wp/2/post/123abc, bi WordPress znake »abc« ignoriral in nadaljeval delo z vrednostjo »123«. Če napadalec pokliče ukaz JSON s sintakso http://spletna.stran/wp-json/wp/2/post/123?id=123abc, pa se vrednost »abc« ne ignorira in se v nadaljnjih funkcijah uporabi id=123abc. To nadaljnje funkcije »zmede« do te mere, da WordPress napadalcu omogoči urejanje objave z ID 123, kar pomeni, da lahko ta v zapis naloži svoje ukaze PHP.

Ta ranljivost je bila odpravljena v WordPress sistemu različice 4.7.2. Zaradi svoje enostavnosti in hudih posledic je takoj postala zelo priljubljena med napadalci in čeprav je bila ranljivost javno objavljena skupaj s popravkom, marsikateri skrbnik popravka seveda ni pravočasno namestil in v zadnjih 6 mesecih smo videli že ogromno napadov na strani WordPress, ki so izkoriščali prav to pomanjkljivost.

TRETJE POGLAVJE

KAJ VSE ŠE OGROŽA SPLETNO STRAN WORDPRESS?

Do sedaj smo govorili o napadih, katerih cilj je prevzeti lastništvo nad spletno stranjo, ter o napadih, katerih cilj je na strežnik naložiti zlonamerno kodo. Te nevarnosti grozijo prav vsem spletnim stranem, od najbolj enostavnih do najbolj kompleksnih.

Kaj vse še ogroža naše spletne strani? Možnih napadov in zlorab je še ogromno, so pa odvisne od samih funkcionalnosti in namena spletnih strani.

1 Prvi sklop ranljivosti se nanaša na nezadostno preverjanje vhodnih podatkov. V resnici smo grožnje s tovrstnim izvorom v tem dokumentu že obravnavali (napadi XSS, nalaganje datotek ipd.). Gre za primere, ko aplikacija posredno ali neposredno od uporabnika prejme neko informacijo (bodisi jo v določeno polje neposredno vpiše uporabnik bodisi jo aplikaciji posreduje brskalnik uporabnika), nato pa to informacijo aplikacija posreduje funkcijam in ostalim zalednim sistemom, ne da bi pred tem podrobno preverila njeno vsebino. Tipični primeri takšnih ranljivosti so vrivanje ukazov SQL (SQL injection), vrivanje datotek (file injection), vrivanje kode HTML (HTML-code injection), ipd. Vsi ti napadi so lahko za aplikacijo zelo škodljivi.

Aplikacija mora torej vse vhodne podatke pred uporabo podrobno preveriti.

2 Drugi sklop ranljivosti se nanaša na nezadostno preverjanje pooblastil. Tovrstna ranljivost pride do izraza predvsem na spletnih straneh, kjer imamo različne registrirane uporabnike. Vsak lahko dostopa do svojih podatkov in izvaja določene svoje funkcije. Grožnja izhaja iz namena napadalca, ki namerava preko enega registriranega uporabnika izvajati nepooblaščene funkcije pri drugem uporabniku. Pri spletnih straneh, kjer imamo večje število uporabnikov, je takšna grožnja vedno prisotna, saj moramo izhajati iz predpostavke, da napadalec spletne strani ne napada kot neregistriran uporabnik, temveč je vseeno uspel pridobiti geslo

vsaj enega uporabnika. Poglejmo si tipičen (poenostavljen) primer ranljivosti. Recimo, da aplikacija uporabnikom omogoča dostop do pomembnih datotek, pri čemer naj bi seveda vsak uporabnik dostopal le do svojih datotek. Uporabnik se registrira in nato v določenem oknu vidi seznam svojih datotek. Če želi odpreti datoteko A in klikne na to datoteko, se v smeri proti aplikaciji sproži klic http://spletna.stran/dostop-do-datotek/datoteka-A. Ta klic se morebiti sproži celo tako, da običajen uporabnik v spletnem brskalniku tega klica niti ne vidi. Najprej je napačno morebitno sklepanje aplikacije, da napadalec ne bo ugotovil pravilnega zapisa klica. Napadalci namreč aplikacije ne napadajo zgolj z brskalnikom, ampak z različnimi orodji, kjer lahko dejansko vidijo vse klice med uporabnikom in aplikacijo in ne samo tiste, ki jih prikaže brskalnik. Še več, s temi orodji lahko te klice tudi spreminjajo, četudi tega v samem brskalniku ne bi bilo mogoče narediti. Recimo, da želi napadalec dostopati do datoteke B, ki pripada drugemu uporabniku. Za dostop do te druge datoteke bi napadalec uporabil klic http://spletna.stran/dostop-do-datotek/datoteka-B. Kot rečeno, lahko napadalec tak klic sproži, četudi brskalnik prek svojih menijev niti ne omogoča takšnega klica. Tipična napaka aplikacij je, da pri takšnih klicih ne preverijo, ali je uporabnik res upravičen do datoteke B, ampak se zgolj zanašajo na dejstvo, da tak klic iz brskalnika v nobenem primeru ne bi mogel biti sprožen. V takem primeru seveda ni dovolj zgolj varovalka, da je dostop do funkcije datotek http://spletna.stran/dostop-do-datotek na voljo le registriranim uporabnikom.

Aplikacija mora zato pri vsakem dostopu dejansko preveriti, ali je uporabnik upravičen do tega dostopa ali ne, čeprav je bila pristnost uporabnika preverjena že ob vstopu v aplikacijo.

3 Tretji sklop ranljivosti je nekakšna izpeljanka ranljivosti iz drugega sklopa. Imenujejo se ranljivosti CSRF (Cross-site-forgery). Predstavljajmo si podobno stanje kot zgoraj, le da sedaj govorimo o funkciji brisanja datotek. Podobno kot zgoraj obravnavamo grožnjo, ali lahko uporabnik A nepooblaščeno pobriše datoteko uporabnika B. Kot smo se že naučili zgoraj, mora aplikacija ob klicu http://spletna.stran/dostop-do-datotek/brisi-datoteko-B preveriti, ali je klic res poslal uporabnik B, in se prepričati, da ga ni sprožil uporabnik A. To lahko učinkovito preveri preko preverjanja piškotka z ID seje, a žal to ni dovolj. Uporabnik A namreč ve, da se datoteka uporabnika B ne bo izbrisala, če bo aplikaciji poslal ta klic, saj njegov klic ne bo vseboval pravilnega piškotka z ID seje. Napadalec lahko v tem primeru ubere drugačno pot. Ta klic spretno skrije v e-pošto z napadom phishing in z zavajajočo vsebino prelisiči uporabnika B, da sam klikne na tako povezavo. Tako bo uporabnik B kar sam izbrisal datoteko, ne da bi to sploh vedel. Torej mora pri posebnih akcijah

aplikacija preverjati še kaj drugega. Aplikacije to običajno rešujejo tako, da pri klicih, ki bi lahko bili sporni, doda še nek edinstven parameter. Ko bo torej uporabnik B res želel izbrisati datoteko, bo to naredil v aplikaciji, takrat pa se bo ob klicu v vsebini zahtevka nahajal še nek edinstven parameter (imenujemo ga žeton), s katerim bo aplikacija vedela, da je klic veljaven. Če bi namreč napadalec želel izvesti napad, kakršen je opisan zgoraj, ta ne bi deloval, saj ne pozna tega edinstvenega podatka in bi aplikacija ta klic zavrnila, četudi bi ga v nekem ponarejenem e-poštnem sporočilu nehote kliknil uporabnik B.

Aplikacija mora torej pri vsaki potencialno škodljivi akciji poleg preverjanja pooblaščenosti za izvedbo takšne akcije vedno uporabiti še različne edinstvene podatke, ki bodo preprečili okoliščine, v katerih bo pooblaščen uporabnik nehote sprožil pravilen klic, ki mu ga bo napadalec podtaknil.

4

V četrti sklop ranljivosti bi lahko postavili vse ranljivosti, ki v resnici neposredno ne pomenijo zlorabe aplikacije, ampak so bolj namenjene pridobivanju podatkov, s katerimi bi bili nadaljnji napadi bolj izvedljivi in uspešnejši. Med te ranljivosti tipično sodijo:

1. Preveč zgovorna in informativna sporočila o napakah, ki napadalcu izdajo marsikatero podrobnost o strežniku (različica programske opreme, informacijo o tem, kdo je skrbnik ipd.)
2. Dostopne datoteke Readme ter podobne informacije, ki izdajajo različico programske opreme in podobne lastnosti aplikacije
3. Seznam map, ki vnaprej izdaja seznam skript, seznam vtičnikov ipd.
4. Dostop do strežniških dnevniških datotek in datotek za razhroščevanje
5. Možnost nepooblaščenega prenosa datotek. Pri spletnih straneh, ki ne temeljijo na platformi WordPress, si napadalci močno želijo prenesti skripte PHP (ali druge), saj lahko s tem pregledajo konkretno programsko kodo skripte in poiščejo ranljivosti. Skripte, klicane preko brskalnika, nikoli ne pokažejo programske kode, temveč to kodo samo izvajajo. Pri spletnih straneh WordPress je ta želja na videz manjša, saj za 99,9 odstotka WordPressovih skript PHP napadalec tako ali tako pozna izvorno kodo (saj si lahko WordPress namesti sam in podrobno preuči kodo). Kljub temu še vedno obstajajo datoteke, ki so za napadalca zanimive. WP-config.php, na primer, vsebuje tudi podatke o tem, kako WordPress dostopa do lastne zbirke, v kateri so shranjena gesla, vsebina spletne strani ipd. S klicem datoteke wp-config.php do teh podatkov ni mogoče priti, če pa bi napadalcu nekako uspelo zagnati prenos datoteke, pa bi do teh podatkov lahko prišel.

V podrobnosti vseh teh ranljivosti žal v tem dokumentu ne bomo zahajali. Velja pa pri straneh WordPress **zlato pravilo rednega spremljanja varnosti vtičnikov ter samega sistema WordPress**. Takoj ko spoznamo, da se je za določen vtičnik ali sistem WordPress pojavila katera izmed zgoraj navedenih težav, moramo vtičnik nemudoma nadgraditi oz. zamenjati z bolj varnim.

Kako zagotoviti varnost celotnega ekosistema spletne strani?

V prejšnjih poglavjih smo obdelali celo vrsto različnih napadov in ranljivosti. Žal pa s tem še nismo obdelali vseh možnih groženj.

Obstaja še en pomemben vidik zagotavljanja varnosti strani WordPress. To je ekosistem, v katerem naša spletna stran deluje. Ekosistem je lahko zelo širok pojem, običajno pa pod tem imenom mislimo na strežnik, na katerem deluje stran WordPress, **spletni strežnik Apache, okolje PHP ter nenazadnje požarna pregrada**, ki dostop do strežnika sploh varuje.

Če začnemo na koncu. Strežnik, na katerem deluje WordPress, je v 99,99 odstotkih primerov varovan z neko požarno pregrado. Skrbniki se morajo zavedati, da ta požarna pregrada praktično v nobenem pogledu ne varuje strani WordPress pred zlorabami, ki so bile do sedaj opisane v tem dokumentu (razen če ne gre za požarno pregrado za spletne aplikacije – Web Application Firewall). Vse zlorabe, ki smo jih opisali do sedaj, se namreč dogajajo znotraj protokola http in običajna požarna pregrada nove generacije v resnici v ta protokol ne posega, razen v nekaterih osnovnih primerih. **Glavni namen požarne pregrade je zaščititi strežnik** pred vsemi ostalimi dostopi. In tu se lahko skriva nova skupina groženj: spletna stran ima lahko, na primer, zagotovljeno 100-odstotno varnost sistema WordPress (vsi najnovejši popravki so implementirani, vsi vtičniki so varni, SSL je pravilno postavljen, varnostni vtičniki, ki varujejo skrbniški dostop do WordPressa so nameščeni, ipd.). Če pa ima skrbnik do svoje spletne strani vzpostavljen tudi dostop FTP (recimo, zaradi lažjega prenosa datotek), je lahko varnost spletne strani resno ogrožena. Če bi napadalec pridobil dostop do storitve FTP, bi lahko na strežnik enostavno naložil svojo skripto PHP, s čimer bi spletna stran prišla pod praktično popolno napadalčevo kontrolo. Skrbnik mora torej poskrbeti, da ustrezno zavaruje tudi vse ostale dostope (ki ne temeljijo na protokolu http) do strežnika, na katerem teče sistem WordPress. Če

so določene storitve res potrebne, jih je treba vsaj omejiti na točno določene naslove IP, pri čemer pa ne smete pozabiti na močna gesla.

Apache in programski paketi PHP so osnova sistema WordPress. Različne ranljivosti teh dveh programskih paketov lahko povzročijo ranljivosti v posameznih vtičnikih. Vtičniki (ali WordPress sam) so lahko ranljivi, če napačno uporabljajo funkcije PHP. V določenih primerih pa so ranljive že same funkcije PHP in zgodi se, da so določeni vtičniki v starih različicah okolja PHP ranljivi, v novejših pa ne. Skrbniki dobro vedo, da sistemov Apache in PHP ni mogoče posodabljati vsak dan, saj lahko vse te posodobitve hitro vplivajo na delovanje aplikacije. Vseeno pa morajo skrbniki vsaj občasno izpeljati tudi takšne posodobitve.

Podobno velja tudi za operacijski sistem strežnika. Teoretično lahko sicer velja, da v primerih, ko je do strežnika odprta zgolj povezava http, morebitne ranljivosti operacijskega sistema niti ne pridejo do izraza. Potrebne popravke je treba kljub temu redno nameščati tudi na operacijskih sistemih.

Nenazadnje lahko v kompleksnih aplikacijah del ekosistema predstavlja tudi zbirka s posebnimi podatki, do katerih dostopa aplikacija, imeniške storitve LDAP ali podoben imenik uporabnikov ipd. Na vseh takšnih sistemih moramo poskrbeti, da je **programska oprema posodobljena**, ter jih ustrezno zavarovati pred nepooblaščenimi dostopi iz interneta.

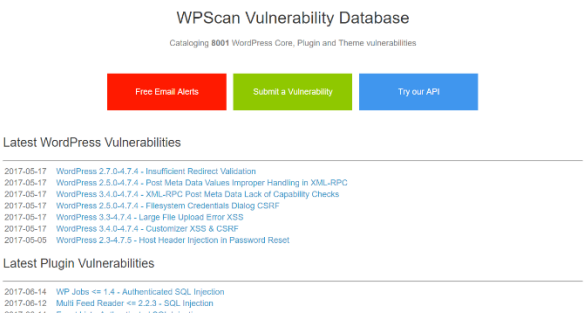
> *Če so to zaledni sistemi, morajo takšni tudi ostati in torej dostopi do njih iz interneta ne bi smeli biti omogočeni.*

Kako vem, da so vtičniki, ki jih uporabljam, varni?

Načeloma tega zares ne vemo nikoli. **Vtičniki so lahko zelo kompleksni sklopi skript** PHP in v njih se vedno lahko nahaja skrita ranljivost. Nenazadnje so bile vse javno dostopne ranljivosti nekoč skrite.

Vendar lahko v 99,9 odstotkih primerov vseeno računamo na razširjenost sistema WordPress. Če pri izdelavi spletne strani vsaj večinoma uporabljamo priljubljene vtičnike, je teh namestitev po vsem svetu toliko, da se res lahko zanesemo na dejstvo, da je določen vtičnik varen, dokler zanj ni javno znane ranljivosti.

Seveda to manj drži v primerih, ko uporabljamo manj priljubljene vtičnike ali pa vtičnike ali stran WordPress stran spreminjamo sami. Razlogi, zakaj bi uporabnik želel vtičnike ali stran WordPress spreminjati sam, so jasni. Na prvi pogled je mogoče tako zagotoviti najboljšo prilagojenost lastnim potrebam. Toda v praksi pogosto vidimo, da različne »doma narejene« predloge ali vtičniki niso boljši od tistega, kar je že na voljo v sistemu WordPress, tako da lahko za marsikateri primer ocenimo, da popravljanje kode ali pisanje lastne kode predstavlja nepotrebno izpostavljanje morebitnim ranljivostim.



Če torej uporabljamo preverjene vtičnike (z večjim številom namestitev) ter **redno spremljamo novosti** na področju varnosti vtičnikov, smo naredili že zelo veliko. Ob tem vas opozarjamo na nedoslednosti. Ko je spletna stran WordPress postavljena, bo skrbnikom sama ponujala obvestila o nadgradnji vseh uporabljenih vtičnikov. Redno obiskovanje lastne skrbniške konzole ter redno nameščanje vseh ponujenih nadgradenj je seveda dobra praksa, ki pa ni nujno dovolj. Še vedno se dokaj pogosto srečujemo s stanjem, ko je ranljivost na določenem vtičniku znana, še preden je popravek na voljo. Skrbnik mora zato spremljati tudi znane ranljivosti. Če za določen vtičnik ugotovi, da se je pojavila ranljivost, popravek pa še ni na voljo, mora resno razmisliti o stopnji ogroženosti in po potrebi vtičnik začasno celo izklopiti ali zamenjati.

Nekaj zelo koristnih strani za spremljanje vseh varnostnih novosti pri vtičnikih in sistemu WordPress:

Ta povezava URL kaže na ranljivosti sistema WordPress. V iskalniku na tej spletni strani lahko poiščete tudi znane ranljivosti posameznih vtičnikov. >>

https://www.cvedetails.com/vulnerability-list/vendor_id-2337/product_id-4096/

Iskanje s ključno besedo »WordPress« pokaže vse znane načine izrabe sistema WordPress in posameznih vtičnikov. >>

https://www.exploit-db.com/search/

Na tej spletni strani lahko nastavite obveščanje po e-pošti o novih ranljivostih, ki vas zanimajo. >>

https://wpvulndb.com/

Vtičnik WP Security

Gre za enega izmed zelo koristnih in učinkovitih vtičnikov, ki **poskrbi za veliko ranljivosti**, omenjenih v tem dokumentu. Namenjen je predvsem **varovanju skrbniškega dostopa**, uporabi pa se ga lahko še za marsikaj drugega.

Vtičnik je dokaj preprost. Meniji so pregledni in sprehod po menijih ter zavihkih bolj ali manj nazorno prikazuje, kaj lahko z določeno nastavitvijo dosežemo. Vtičnik nas opozarja tudi na namestitev zaščit, kjer moramo biti bolj pozorni (bodisi zato, ker lahko s pretirano varnostjo vplivamo na posamezne funkcionalnosti spletne strani, bodisi zato, ker lahko s preveč strogimi varnostnimi nastavitvami tudi sebi onemogočimo dostop do skrbništva).

Vseh funkcionalnosti vtičnika ne bomo podrobneje predstavljali. Vtičnik je brezplačen, več podrobnosti o njem pa je dosegljivih na spletnem naslovu https://www.tipsandtricks-hq.com/WordPress-security-and-firewall-plugin.

Glavne funkcionalnosti, ki jih skrbnikom toplo priporočamo, so naslednje:

1. Dostopen je seznam skrbniških ali uporabniških prijav v sistem.
2. Dostopen je seznam neuspešnih skrbniških oz. uporabniških prijav v sistem.
3. Omogoča ureditev, da je prikazano ime avtorja/skrbnika/uporabnika v resnici drugačno, kot je uporabniško ime, s katerim se avtor/skrbnik/uporabnik prijavlja v sistem.
4. Omogoča ustvarjanje in uporabo zahtevnih gesel.
5. Preprečuje napade z grobo silo z večkratnim ugibanjem gesel.
6. Omogoča pregled nad trenutno prijavljenimi skrbniki oz. uporabniki ter izklop dostopov.
7. Dostopen je seznam vseh neuspešnih klicev na spletno stran s kodo 404.
8. Poskrbi, da je različica okolja WordPress napadalcu težje razpoznavna.
9. Omogoča vzpostavitev sistema CAPTCHA v prijavnih oknih.
10. Preprečuje registracijo uporabnikov različnim skriptam.
11. Omogoča sistem ročne odobritve posameznih registriranih uporabnikov.
12. Omogoča dodatno zavarovanje zbirke sistema WordPress s spreminjanjem privzetih klicev in imen.
13. Nadzira, če se je nepričakovano spremenila vsebina pomembne datoteke.

14. Skrbnikom preprečuje urejanje datotek PHP preko skrbniškega vmesnika.
15. Omogoča vpogled v dnevniške zapise strežnika (če to dovoli skrbnik strežnika).
16. Omogoča začasno ali stalno blokiranje dostopov ali zgolj prijav za določene naslove IP.
17. Omogoča blokado dostopov do datoteke XMLRPC.
18. Omogoča blokado dostopov do različnih datotek za razhroščevanje.
19. Omogoča enostavnejše urejanje datotek .htaccess.
20. Omogoča blokado klicev, značilnih za ranljivosti XSS.
21. Preprečuje izpisovanje map.
22. Omogoča blokado dostopov do raznih privzetih licenčnih datotek ali datotek Readme, ki bi lahko napadalcu izdale preveč podatkov.
23. Omogoča začasno zaustavitev spletne strani (ki je tako dostopna zgolj skrbniku).
24. Preprečuje različne načine (opisane v tem dokumentu) za odkrivanje pravih uporabniških imen uporabnikov.

S pravilno nastavljenim vtičnikom WP Security lahko torej skrbnik vnese veliko varnostnih mehanizmov.

Ostali vtičniki in mehanizmi za zagotavljanje varnosti

V prejšnjem poglavju smo predstavili vtičnik WP Security. Ni pa to edini vtičnik, ki je na voljo za zagotavljanje varnosti. Sledi predstavitev nekaj priljubljenih in kakovostnih vtičnikov:

1. **Wordfence.** V primerjavi z vtičnikom WP Security ponuja celo nekatere dodatne funkcionalnosti. Te dodatne funkcionalnosti so sicer plačljive, so pa dokaj koristne.
    a. Možnost blokiranja dostopa obiskovalcem iz določenih držav
    b. Bolj podrobno spremljanje sprememb datotek v datotečnem sistemu, predvsem v smislu iskanja znanih datotek s škodljivo programsko opremo ali skritimi vrati (backdoor)
    c. Vzpostavitev dvonivojskega preverjanja pristnosti
    d. Več funkcionalnosti v okviru požarne pregrade za spletne aplikacije, s katerimi lahko dejansko preprečimo določene zlorabe, četudi bi bili vtičniki ranljivi. Res pa je, da za ta del skrbnik potrebuje določena znanja.
2. **Bulletproof security** (https://WordPress.org/plugins/bulletproof-security/)
3. **Sucuri security** (https://WordPress.org/plugins/sucuri-scanner/)
4. **Better WP security** (https://WordPress.org/plugins/better-wp-security/)
5. **WP Security scan** (https://WordPress.org/plugins/wp-security-scan/)
6. **6scan security** (https://WordPress.org/plugins/6scan-protection/)

Specifične varnostne funkcionalnosti, ki v zgoraj naštetih vtičnikih niso zajete ali pa niso zajete na način, ki skrbniku ustreza, lahko najdemo tudi v nekaterih drugih vtičnikih. Pri tem priporočamo, da se skrbnik iskanja vtičnikov ne loti po imenih, ampak najprej uporabi Google iskalnik, kjer bo na različnih forumih in podobnih portalih podrobneje spoznal vtičnike in priporočila uporabnikov. Ti forumi so polni dobronamernih in kakovostnih priporočil različnih skrbnikov iz vsega sveta.

Dodatno raven varnosti predstavlja tudi požarna pregrada za spletne aplikacije – **WAF oz. Web Application Firewall**. Požarna pregrada za spletne aplikacije ni enaka **požarni pregradi naslednje generacije** (Next Generation Firewall). Prav tako velja, da požarna pregrada

naslednje generacije ni požarna pregrada za spletne aplikacije. Čeprav imata ta dva sistema podobni imeni in se v obeh omenja pojem aplikacije, sta sistema v resnici zelo različna. **Požarna pregrada naslednje generacije** varuje spletno aplikacijo tako, da ne dopušča ostalih dostopov do strežnika, spremlja, če želi kdo zlorabiti določene ranljivosti spletnega strežnika Apache, spremlja, če želi kdo s preizkušanjem ugotoviti prisotnost spletnega strežnika, ipd. Požarna pregrada za spletne aplikacije pa je namenjena zaščiti dostopov do spletne aplikacije v globino. V bistvu poskrbi za varovanje pred vsemi napadi, ki smo jih opisali v tem dokumentu. Namesto spletne aplikacije lahko poskrbi za ustrezne nastavitve vrednosti piškotkov, ustrezne nastavitve povezave SSL, ustrezne podatke v glavah in podobno. Prav tako lahko preprečuje zlonamerne vhodne klice, ki so vezani na vrivanje kode SQL, napade XSS in podobne ranljivosti.

WAF se v praksi na žalost redko uporablja. Dobri skrbniki morda sistema WAF sploh ne potrebujejo, saj jim marsikatere funkcionalnosti prinesejo že zgoraj našteti vtičniki, a vsaj v primeru večjih spletnih aplikacij je uporaba sistema WAF koristna in priporočena.

Alternativa temu, da podjetje samo vzpostavi sistem WAF je ti. **zunanje izvajanje sistema WAF**. Najbolj znan sistem je verjetno **Cloudflare**. Gre za zunanjega ponudnika sistema WAF (seveda proti plačilu), kjer promet do naše aplikacije preusmerimo preko sistema Cloudflare, ki promet ustrezno prečisti in zavaruje. Sistemi WAF so učinkoviti le, če njihovi skrbniki poznajo tehnične lastnosti spletne aplikacije. Zato so sistemi WAF verjetno še toliko bolj koristni pri aplikacijah, razvitih na sistemu WordPress, saj so tehnične lastnosti aplikacije sistemu WAF večinoma znanje vnaprej.

ČETRTO POGLAVJE

ALI JE MOJA SPLETNA STRAN VARNA?

Kako lahko preverim, če moja spletna aplikacija izpolnjuje vse nasvete, navedene do sedaj?

Zgolj budno spremljanje pravilnosti nastavitev in zanašanje na pravilnost uporabljene kode pri spletnih aplikacijah nikakor ni dovolj. **Nikoli ne smemo podcenjevati znanja in domišljija napadalcev.**

Varnost lahko preverimo s **podrobnim varnostnim pregledom**. To pomeni, da se nekdo »prelevi« v napadalca in s podobnimi oz. enakimi metodami, kot bi jih uporabil napadalec, preizkusi vse možne načine zlorabe. Pri tem velja **zlato pravilo, da tega skrbnik NE more delati sam**. Ne glede na to, da se s to izjavo marsikateri skrbnik ne strinja, obstajata dva glavna razloga, zakaj je tako:

1. Lastne napake težje opazimo.
2. Vsaj na določenih področjih je naše znanje omejeno v primerjavi z znanjem napadalca (čeprav se to sliši kot huda kritika, se moramo zavedati, da imajo napadalci na voljo bistveno več časa, motivacije, orodij in informacij, vezanih na konkretne tehnologije, kot skrbnik sam).

Kako pa nam pri tem lahko pomagajo različna orodja za skeniranje?

Orodja so nekaj, kar bi lahko skrbnik uporabljal sam, zaradi česar mu ročnih varnostnih pregledov morda sploh ne bi bilo treba opravljati. Na žalost orodja danes še ne morejo ponuditi ustrezne zamenjave za ročne preglede. Spletne aplikacije so namreč preveč kompleksne in edinstvene, da bi lahko določeno orodje z enim klikom samostojno preverilo vse ranljivosti.

Za osnovni pregled pravilnosti nastavitev spletnih sistemov ter za morebitno odkrivanje enostavnih ranljivosti so primerna naslednja orodja:

1. Nikto (del sistema Kali Linux)
2. Burp Suite (del sistema Kali Linux, pregledovanje omogoča zgolj plačljiva različica)
3. Netsparker
4. Acunetix
5. Rapid7 Appspider

Koristijo vam lahko tudi orodja, ki preverjajo robustnost protokola SSL. Na primer:

1. Online Qualis test (https://www.ssllabs.com/ssltest/)
2. SSLscan, ki je del sistema Kali Linux

Ker pa so strani WordPress vseeno bolj predvidljive od ostalih spletnih aplikacij, so koristna tudi orodja, ki so bolj **usmerjena v specifične lastnosti sistema WordPress**. Ta orodja običajno preverjajo običajno pomanjkljive nastavitve sistemov WordPress ter programske različice vtičnikov in sistema samega.

1. Spletna orodja WPScan (https://hackertarget.com/WordPress-security-scan/)
2. Orodje WPScan, ki je del sistema Kali Linux

*Skrbniki imajo torej na voljo nekaj koristnih orodij za pregledovanje, s katerimi lahko pregledujejo robustnost in varnost lastnih spletnih strani. Pri tem se morajo zavedati, da ta orodja ponudijo zgolj **površinski vpogled v dejansko varnost**. Primerna so za hitre in sprotne preglede, nikakor pa ne morejo nadomestiti pravih ročnih varnostnih pregledov, ki jih izvajajo za to pooblaščene in usposobljene osebe.*

Kaj narediti, če sem postal žrtev uspešnega napada?

Kako čim prej po napadu vzpostaviti neoporečen sistem?

Forenzična analiza ugotavljanja vzrokov napada je lahko enako (ali pa še bolj) kompleksna kot samo razmišljanje o zagotavljanju varnosti. Vse podrobnosti bi tako zahtevale popolnoma nov dokument, še posebej, če bi razmišljali o iskanju sledi, s katerimi bi želeli priti na sled napadalcu ali bi jih lahko uporabili pri točni analizi orodij in identifikacije napadalca. Marsikdaj skrbniki po napadu ne razmišljajo o tem, kako primer prijaviti ustreznim službam in kako bi izvedli tehnično forenziko napada (kar po našem skromnem mnenju ni pravilno), ampak bolj razmišljajo o tem, kako čimprej vzpostaviti stanje spletne aplikacije, ki bo delovala običajno in do katere napadalec ne bo več imel dostopa.

Pri tem lahko skrbnik zelo hitro naredi napako, da sprožene akcije niso dovolj natančne. Dobri napadalci si namreč marsikdaj ob vdoru postavijo ti. **skrita vrata** (na primer, vstavijo kratko kodo PHP v določeno obstoječe sistemsko datoteko, ustvarijo uporabnika z lahkim geslom ipd.). Če jih odkrijemo in jim preprečimo nadaljnji dostop, bodo zelo verjetno preko vzpostavljenih skritih vrat do spletne strani še vedno lahko prišli kasneje in bodo takrat še nevarnejši, saj vedo, da so bili odkriti in da morajo biti bistveno bolj pazljivi.

Če želimo torej ob odkritju napada ponovno vzpostaviti **neoporečen sistem**, do katerega napadalec ne bo več imel dostopa in ga tudi ne bo mogel ponovno vzpostaviti in v katerem bodo vsi nepooblaščeno spremenjeni podatki povrnjeni nazaj v prvotno stanje, mora skrbnik točno vedeti:

1. Kako je napadalec sploh prišel do zlorabe?
2. Kdaj se je to zgodilo?
3. Kaj je potem naredil?

Teorija je enostavna, praksa pa žal ne. Kaj sploh imamo na voljo? Na voljo so **dnevniški zapisi aplikacije**, če je ta povezana s sistemom WordPress (recimo podatkovna zbirka ipd.). Še najbolj pogosto so na voljo **sistemski dnevniški zapisi**

na strežniku. Hkrati je na voljo tudi vpogled v **datotečni sistem spletne aplikacije**, ki lahko marsikaj posredno izda z datumi sprememb datotek in podobnih podatkov.

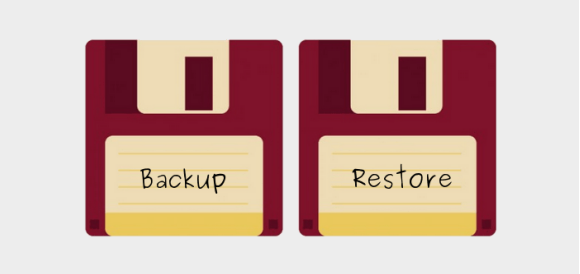
Verjetno je še najbolje najprej pregledati sistemske dnevniške zapise strežnika. Pri tem priporočamo, da se tega vedno lotite ročno. Pri pregledu dnevniških zapisov nas omejujeta dve stvari: prva je dejstvo, da je internet prava džungla zlorab in se vsak dan v dnevniških zapisih dostopov zbira cela poplava raznoraznih poizkusov zlorab. Drugi dejavnik, ki nas omejuje, pa je dejstvo, da dnevniški zapisi praviloma ne zapisujejo celotne vsebine vseh klicev, ampak zgolj zapise v zaglavju prometa http.

Kakorkoli že, smiselno je najprej poiskati nepooblaščene skrbniške dostope. Denimo, dostop do naslova http://spletna.aplikacija/wp-admin.admin.php s klicem POST, ki se konča s kodo odgovora 200, je zagotovo skrbniški dostop. Poiščemo lahko vse tovrstne zapise, ki prihajajo iz neznanih naslovov IP. Če najdemo zadetek, imamo osnovo za nadaljevanje preiskave. Sedaj vsaj točno vemo, kateri je najzgodnejši datum, od katerega dalje je imel napadalec dostop ter naslov IP, ki je bil del napada. Sedaj lahko sicer preiskavo nadaljujemo preko naslova IP, saj je dnevniške datoteke bistveno lažje pregledovati po naslovih IP, vendar je treba pri tem upoštevati nasvet, da lahko napadalec do strežnika dostopa iz določenega nabora naslovov IP, tako da je namesto naslova 12.34.2.244 smiselno iskati naslov 12.34.2.x ali kaj podobnega. Prav tako je možno, da je napadalec dostopal do spletne strani preko popolnoma različnih naslovov IP.

Smiselno je iskati vse dostope iz teh podobnih naslovov IP. Pri tem se osredotočamo na primere, ko iz klicanega naslova URL lahko zaznamo točko, ko napadalec skrbniškega dostopa verjetno še ni imel. Na primer, iščemo dostope POST do skripte wp-login.php, ki jim ne sledi dostop do skrbniškega dela aplikacije. Ali pa iščemo dostope POST do skripte xmlrpc.php, saj lahko sklepamo, da je takrat napadalec še ugibal pravilna gesla (če je recimo poizkušal v aplikacijo vdreti tudi na tak način) …

Dober nasvet je, da preiskave začnemo v sistemskih dnevniških zapisih strežnika. Najprej iščemo po vsebini, ki se nanaša na to, kar vemo (vemo, da je imel napadalec dostop do skrbništva; vemo, da je imel dostop do določenega uporabnika; ipd.). Če ne vemo veliko, je lahko dober začetek iskanje dostopov URL do skript PHP, kjer se uporablja dostop POST in se klici zaključijo s kodo HTML 200. Na ta način dobimo kandidate za sumljive naslove IP. Nato iskanje spremenimo v iskanje po teh naslovih IP (pomembno je, da ne jemljemo točnih naslovov IP, temveč zgolj njihove dele). Tako dobimo novo idejo, katere »čudne« klice je napadalec izvajal, iz česar bodisi že dobimo sliko o časovnem poteku bodisi se vrnemo nazaj na iskanje drugih sumljivih naslovov

IP, le da imamo sedaj v roki nove namige glede naslovov URL, ki jih napadalec uporablja. Pri vsem tem si lahko pomagamo z dnevniškimi zapisi napak na strežniku in ostalimi dnevniškimi zapisi, ki so morda na voljo. Cilj je vsaj približno ugotoviti, kdaj in kako se je vse skupaj začelo.



Pri takšnem iskanju smo lahko uspešni ali neuspešni.

V vsakem primeru pa **nato sledi ponovna vzpostavitev stanja.** Ko govorimo o sistemu WordPress, je stanje kombinacija namestitve okolja WordPress (z vtičniki) ter podatkovne zbirke sistema WordPress. Pri namestiti WordPressa z vtičniki ne bi smelo biti vprašanj. Nujno je treba sistem namestiti »na sveže«. Kakršnokoli popravljanje obstoječe namestitve ni dovolj, saj obstaja velika verjetnost, da so v morju skript PHP še vedno skrita vrata napadalca. Večji problem je z zbirko, saj ta morebiti vsebuje nekatere podatke, ki jih skrbnik težko postavi na novo. V zbirki je vsa vsebina spletne strani, vsi registrirani uporabniki, vsi vneseni podatki ipd., tako da je zelo verjetno, da je teh podatkov preveč za ponovni vnos ali pa jih sploh ni mogoče več dobiti). Če skrbnik na voljo nima varnostne kopije zbirke in je ni mogoče vzpostaviti od začetka, je edini način za zagotovitev neoporečnosti zbirke ročen pregled. Če pa ima skrbnik na voljo varnostno kopijo zbirke, pa je pomemben podatek, ki smo ga ugotavljali v prvem koraku, in sicer kdaj se je napad sploh pričel oz. od katerega datuma naprej smo lahko prepričani, da je baza oporečna.

Če skrbniku uspe pravilno vzpostaviti stanje sistema, bodo torej morebitni dostopi preko skritih vrat onemogočeni. Uspešnost pregledovanja dnevnikov v prvem koraku nam ponudi vsaj namig o tem, kako je do vdora sploh prišlo. Če tega namiga ni, je edina preostala možnost skrbnikov, da izvedejo **podroben varnostni pregled svoje aplikacije**, kot je opisan v prejšnjem poglavju.

Če se je določena ranljivost enkrat izkazala kot usodna, se bo pri istem ali drugem napadalcu zagotovo ponovila, tako da je vzpostavitev spletne strani brez stare ranljivosti ključnega pomena. Po zaključeni ponovni vzpostavitvi spletne strani je priporočljivo vsaj nekaj časa pozorneje spremljati dogajanje in dnevniške zapise, saj obstaja velika verjetnost, da bo napadalec tako ali drugače ponovno dostopal do spletne aplikacije, vsaj dokler ne ugotovi, da dostop ni več možen.

Koristne povezave vezane na WordPress

Dodatno koristno branje

https://hackertarget.com/attacking-WordPress/

https://www.codeinwp.com/blog/secure-your-WordPress-website/

https://sl.WordPress.org/plugins/

https://geekflare.com/WordPress-x-frame-options-httponly-cookie/

https://digital.com/blog/WordPress-security-tips/

https://www.shellandco.net/WordPress-secure-cookies-httponly/

https://www.cvedetails.com/vulnerability-list/vendor_id-2337/product_id-4096/

https://www.exploit-db.com/search/

https://wpvulndb.com/

https://www.ssllabs.com/ssltest/

https://hackertarget.com/WordPress-security-scan/

https://wpscans.com/

http://scanwp.net/

http://resources.infosecinstitute.com/7-best-WordPress-security-plugins/#gref

https://WordPress.org/plugins/wp-security-scan/

https://WordPress.org/plugins/6scan-protection/

https://WordPress.org/plugins/better-wp-security/

https://www.tripwire.com/state-of-security/featured/5-best-WordPress-security-plugins-to-keep-your-site-secure/

https://geekflare.com/online-scan-website-security-vulnerabilities/

Želite še več varnostnih napotkov?

Naročite se na mesečne varnostne novice in bodite obveščeni o aktualnem ter relevantnem dogajanju s področja kibernetske varnosti.

PRIJAVA