Detekcija i mitigacija Apache napada simboličkim poveznicama

Molnar, Josip

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
ELEKTROTEHNIČKI FAKULTET

Stručni studij

DETEKCIJA I MITIGACIJA APACHE NAPADA SIMBOLIČKIM POVEZNICAMA

Završni rad

Josip Molnar

Osijek, 2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac Z1S: Obrazac za ocjenu završnog rada na stručnom prijediplomskom studiju

Ocjena završnog rada na stručnom prijediplomskom studiju

Ime i prezime pristupnika:	Josip Molnar
Studij, smjer:	Stručni prijediplomski studij Elektrotehnika, smjer Informatika
Mat. br. pristupnika, god.	AI4309, 29.08.2013.
JMBAG:	0236208162
Mentor:	prof. dr. sc. Marijan Herceg
Sumentor:	
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	izv. prof. dr. sc. Ratko Grbić
Član Povjerenstva 1:	prof. dr. sc. Marijan Herceg
Član Povjerenstva 2:	doc. dr. sc. Denis Vranješ
Naslov završnog rada:	Detekcija i mitigacija Apache napada simboličkim poveznicama
Znanstvena grana završnog rada:	**Telekomunikacije i informatika (zn. polje elektrotehnika)**
Zadatak završnog rada:	U sklopu teme potrebno je analiziranti symlink napade na Linux poslužitelje s naglaskom na njihovu detekciju i mitigaciju.

Datum ocjene pismenog dijela završnog rada od strane mentora:	20.09.2024.
Ocjena pismenog dijela završnog rada od strane mentora:	Izvrstan (5)
Datum obrane završnog rada:	3.10.2024
Ocjena usmenog dijela završnog rada (obrane):	Izvrstan (5)
Ukupna ocjena završnog rada:	Izvrstan (5)
Datum potvrde mentora o predaji konačne verzije završnog rada čime je pristupnik završio stručni prijediplomski studij:	03.10.2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O IZVORNOSTI RADA

Osijek, 03.10.2024.

Ime i prezime Pristupnika:	Josip Molnar
Studij:	Stručni prijediplomski studij Elektrotehnika, smjer Informatika
Mat. br. Pristupnika, godina upisa:	AI4309, 29.08.2013.
Turnitin podudaranje [%]:	3

Ovom izjavom izjavljujem da je rad pod nazivom**: Detekcija i mitigacija Apache napada simboličkim poveznicama**

izrađen pod vodstvom mentora prof. dr. sc. Marijan Herceg

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis pristupnika:

1. UVOD

Ovaj rad se bavi problemom napada koji iskorištava ranjivosti u načinu na koji *Apache* mrežni poslužitelj (engl. *server*) radi sa simboličkim poveznicama (engl. *symbolic link, symlink*) u određenim konfiguracijama. Premda su ovakvi napadi prisutni i u drugim okruženjima, primarni fokus ovog rada je specifičan oblik *Apache symlink* napada koji većinom zahvaća poslužitelje u domeni udomljavanja internetskih stranica (engl. *web hosting*) koji koriste softverske pakete upravljačkih ploča (engl. *control panel*) poput *cPanel*, *Plesk*, *InterWorx* i slično. Ovaj napad napadaču omogućava lateralan proboj iz jedne kompromitirane stranice u podatkovne prostore drugih stranica na istom poslužitelju putem ekstrakcije vjerodajnica (engl. *credentials*), tj. korisničkih imena i zaporki za pristup bazama podataka drugih stranica. Poslužitelji sa upravljačkim pločama su posebno ranjivi na ovaj tip napada iz više razloga, no specifično se mogu istaknuti: predvidljivost strukture datotečnog sustava (engl. *filesystem*) i lokacija konfiguracijskih datoteka, učestala upotreba sustava za upravljanjem sadržajem stranica (engl. *content management system, CMS*) sustava, te popularnost ovih upravljačkih ploča u tržišnom segmentu manjih i srednjih organizacija i poduzeća. Takve organizacije često nemaju osoblje sa potrebnim informatičkim i informacijsko-sigurnosnim kvalifikacijama, te se uvelike oslanjaju na pružatelje *hosting* usluga i njihovu tehničku podršku. Neadekvatno održavanje *web* aplikacija i poslužitelja, kao i rastuća kompleksnost *web* aplikacija, otvaraju iste raznim napadima putem poznatih i nepoznatih ranjivosti. Ove ranjivosti predstavljaju potencijalne ulazne točke ili vektore napada (engl. *attack vector*), a suma svih vektora napada naziva se površinom napada (engl. *attack surface*). Napadači iskorištavaju ove ranjivosti kako bi ostvarili inicijalni pristup meti, a dalje putem kompleksnijih napada poput navedenog *Apache symlink* napada vrše sekundarne proboje i osiguravaju si daljnji pristup poslužitelju.

U ovom radu su opisana rješenja za pravodobno otkrivanje ove vrste *Apache symlink* napada, postupak oporavka od proizašle štete, te rješenje za prevenciju daljnjeg neovlaštenog pristupa stranicama i poslužitelju. U periodu izrade programskih rješenja nisu pronađeni slični programi koji vrše istu funkciju. Programska rješenja su pisana u *bash* skriptnom jeziku radi neovisnosti o softverskim bibliotekama trećih strana i bibliotekama specifičnim za pojedine varijante *Linux* operativnog sustava, te zbog lakoće interakcije skripti sa datotečnim sustavom i servisima *Linux* poslužitelja. Rješenja su u trenutnoj izvedbi prilagođena za *cPanel* poslužitelje sa *WordPress* stranicama, te su na istima i testirana.

Drugo poglavlje daje pregled postojećih radove iz domene kibernetičke sigurnosti i kriminala, i opisuje najčešće vrsta kibernetičkih napada koji zahvaćaju internetske poslužitelje ili

se putem njih izvršavaju. Treće poglavlje objašnjava *symlink* napade, ranjivosti koje ih omogućavaju, i metode njihove prevencije. Također daje detaljnu analizu naslovnog napada i pojašnjava potencijalnu štetu koja može proizaći iz pojedine etape napada. Četvrto poglavlje daje prijedlog programskog rješenja za brzo otkrivanje ove vrste *symlink* napada. Ovo rješenje je orijentirano na administratore višeposlužiteljskih okruženja i pružatelje *hosting* usluga, te je osmišljeno sa ciljevima jednostavne implementacije i instalacije na velikim flotama poslužitelja. Poglavlje se nadalje bavi procesom analize i forenzike sigurnosnog incidenta i ulogom analize u pravilnom otklanjanju posljedica proboja, te opcijama za oporavak od sigurnosnih incidenata. U konačnici, poglavlje opisuje programsko rješenje namijenjeno automatizaciji i značajnom ubrzavanju postupka prevencije daljnjeg neovlaštenog pristupa kompromitiranim stranica. Izmjereno je vrijeme izvođenja ovog programskog rješenja u nekoliko poslužiteljskih okruženja, te su rezultati uspoređeni sa vremenom koje je za ručno obavljanje istog zadatka bilo potrebno nekolicini korisnika različitih razina iskustva sa zadanim poslužiteljskim okruženjem. U petom poglavlju nalazi se zaključak rada i evaluacija rezultata.

2. PREGLED POSTOJEĆIH RADOVA IZ PODRUČJA RAČUNALNE SIGURNOSTI I KRIMINALA

Rad “Vrste kibernetičkih napada na poduzeća i njihove mjere obrane” [1] daje osvrt na opće pojmove kibernetičke i računalne sigurnosti te opisuje problematiku sigurnosti u kontekstu poduzeća. Ova tema je primjenjiva i na druge tipove organizacija koje ostvaruju mrežnu tj. *online* prisutnost. Prema [1, str. 2] računalna sigurnost obuhvaća zaštitu računalnih sustava, podataka, i informacija, od štete, krađe, i neovlaštene uporabe, kao i remećenje usluga koje računalni sustavi pružaju. Računalni kriminal je definiran kao upotreba računala u nezakonite svrhe, većinom počinjenje prijevare i iznude; krađa identiteta, sredstava, i podataka; te trgovanje nedozvoljenim materijalima i intelektualnim vlasništvom i špijunaže [1, str. 3,4]. Prema [1, str. 5,6,7] izvršitelji računalnog kriminala, odnosno hakeri, se mogu ugrubo podijeliti na prema motivacijama: financijska dobit - računalni kriminalci (engl. *cybercriminal*), politička ili vjerska uvjerenja - hakeri aktivisti (engl. *hacktivists*), osveta - orijentirana na osobe ili poduzeća, samodokazivanje ili zabava - većinom među početnicima.

U radu [1, str. 9] zlonamjeran softver (engl. *malicious software*, *malware*) definira se kao bilo koji program ili kod kojemu je cilj oštetiti, onemogućiti normalan rad, ili preuzeti kontrolu nad računalnim sustavom ili mrežom. Nadalje opisuje najčešće oblike *malwarea*:

- crvi (engl. *worm*) - opći naziv za zloćudne programe koji se samostalno izvršavaju, repliciraju, i šire, bez obzira na točnu funkciju koju izvršavaju
- virusi - *malware* koji se integrira u legitimne programe ili datoteke te se uz njih i izvršava, a širenje mu ovisi o eksternim (korisničkim ili automatiziranim) akcijama
- trojanci - *malware* koji je maskiran kao legitiman softver ili datoteka
- RAT - trojanci za ostvarivanje neovlaštenog pristupa sustavu na daljinu (engl. *Remote Access Trojan*)
- *keylogger, spyware, skimmer* - *malware* za prikupljanje osjetljivih podataka
- zlonamjeran *adware* - generira prihode od oglasa bez pristanka vlasnika stranice
- *ransomware* - *malware* koji enkriptira podatke ili onemogućuje pristup sustavu žrtve u svrhu iznude pod prijetnjom gubitka ili daljnje preprodaje osjetljivih osobnih ili poslovnih podataka
- *rootkit i bootkit* - tip *malwarea* kojim se ostvaruje privilegirani, tj. *root* pristup sustavu, vrši promjene nad sistemskim datotekama te prikriva svoju prisutnost

Među ostalim vrstama napada koji se spominju ističu se:

- *phishing* [1, str. 20] (izvedenica engl. *fishing* - pecanje), vrsta kibernetičkog kriminala kojoj je cilj krađa osobnih, osjetljivih, ili financijskih podataka žrtve, najčešće radi financijske koristi.
- Mreže daljinski kontroliranih uređaja (engl. *botnet*) [1, str. 21] - zaraženi uređaj naziva se *zombie* ili *bot* (skraćeno za robot); dok je cjelokupna mreža zaraženih uređaja *botnet*, odnosno mreža robota. Zlonamjernim softverom se upravlja sa udaljenih upravljačkih, tzv. C&C ili C2 poslužitelja (engl. *Command & Control server*). Koristi se za izvršavanje drugih vrsta napada prema naredbama sa C2 poslužitelja, najčešće distribuiranih napada uskraćivanjem usluge (engl. *Distributed Denial of Service attack, DDOS*), slanja *spam* poruka, rudarenje kriptovaluta i slično.

U radu [1, str. 31, 36] je istaknuto da premda poduzeća trebaju razviti i provoditi strategiju računalne sigurnosti, u pravilnoj definiciji kibernetička sigurnost također treba obuhvaćati fizičku sigurnost, sigurnosno osviještena pravila i politike za zaposlenike, te provođenje edukacije o sigurnosnim prijetnjama. Uz to, poduzeća trebaju redovno vršiti analize ranjivosti i rizika [1, str. 33], uključujući penetracijsko testiranje, kao i osigurati mjere zaštite i oporavka od proboja [1, str. 37]. Premda ovaj rad daje opširan osvrt na sigurnosne probleme s kojima se suočavaju poduzeća i organizacije, i kvalitetno opisuje ciljeve računalne odnosno kibernetičke sigurnosti u ovim okruženjima, relativno površno raspravlja o konkretnim rješenjima koja je potrebno implementirati u svrhu prevencije i odgovora na incidente budući da takva rješenja naravno ovise o poslužiteljskim okruženjima i specifičnostima mreže organizacije.

Vodič za sustave otkrivanja i prevencije upada, "*Guide to Intrusion Detection and Prevention Systems*" [2] Američkog Instituta za Standarde i Tehnologiju (NIST) je opširan rad koji opisuje ulogu i vrste sustava za detekciju i prevenciju upada odnosno proboja (engl. *Intrusion Detection and Prevention Systems, IDPS*), te principe i metodologije detekcije koje ovi sustavi koriste.

Prema [2, str. 15], otkrivanje proboja je proces nadzora (engl. *monitoring*) događaja u računalnom sustavu ili mreži, te analiza tih događaja u potrazi za indikatorima incidenata. Incidenti obuhvaćaju već izvršene, ili direktne nadolazeće prijetnje kršenja pravila ili politika računalne sigurnosti, politika prihvatljive uporabe, i sigurnosnih standarda. Prevencija proboja predstavlja provođenje ovog nadzora te djelovanje u pokušaju sprječavanja detektiranih incidenata. IDPS tehnologije uz navedeno nužno: evidentiraju sve relevantne događaje na sustavu ili mreži, generiraju izvještaje, obavještavaju mrežnog administratora o potencijalnim

incidentima, a mogu imati mogućnosti automatske prevencije putem izmjena konfiguracija servisa, npr. vatrozida ili izmjene sadržaja napada. IDPS tehnologije se dijele prema događajima koje nadziru, i prema načinu na koji se implementiraju:

- mrežni [2, str. 35]- nadziru mrežni promet cijele ili segmenta mreže te analiziraju aplikacijski promet i korištene protokole u svrhu otkrivanja sumnjive aktivnosti
- bežični [2, str. 51] - nadziru bežične mreže i njihove protokole radi otkrivanja sumnjive aktivnosti
- analiza mrežnog ponašanja (engl. *Network Behavior Analysis*, NBA) [2, str 65] - analiziraju mrežni promet radi otkrivanja prijetnji koje se očituju neobičnim tokom prometa, npr. DDOS napadi, neke vrste *malwarea*, kršenja sigurnosnih politika (prosljeđivanje pristupa mreži i slično)
- na pojedinom uređaju (engl. *host-based*) [2, str. 73] - nadziru karakteristike i događaje unutar jednog uređaja (poslužiteljskog ili klijentskog)

Tri su najčešće metodologije koje IDPS sustavi koriste za prepoznavanje prijetnji:

- bazirane na poznatom otisku tj. uzorku (engl. *Signature-Based Detection*) [2, str. 18] - svodi se na usporedbu detektiranih događaja sa poznatim oblicima napada ili općenitije bilo kojim prepoznatljivim indikatorima napada
- bazirane na detekciji anomalija (engl. *Anomaly-Based Detection*) [2, str. 19] - normalna aktivnost između pojedinih uređaja ili korisnika na mreži se dokumentira u profile poznatog ponašanja, a detekcija se ostvaruje evidentiranjem devijacije u ponašanju
- analiza protokola i njihovog stanja (engl. *Stateful Protocol Analysis*) [2, 19] - uspoređuje mrežnu aktivnost sa unaprijed definiranim pravilima benigne aktivnosti, uključujući praćenje stanja protokola

IDPS sustavi su u širokoj upotrebi u većim organizacijama. Premda i pružatelji *hosting* usluga koriste neke oblike IDPSa na mrežnoj razini, *host-based* IDPS nije jednostavno implementirati na većem broju poslužitelja. Sama instalacija IDPS agentskog programa je najčešće ručna budući da se agentski tj. klijent program mora spojiti sa upraviteljskim poslužiteljem pomoću neke jedinstvene oznake i konfigurirati sa pravilnom kontrolom pristupa. HIDPS sustavi mogu biti efikasni u otkrivanju *symlink* i drugih ozbiljnijih napada, ali su ograničeni kompleksnošću instalacije, ovisni o redovnim ažuriranjima i nadogradnjama ali i o kompetencijama mrežnog administratora ukoliko se koristi besplatno open source IDPS rješenje poput OSSEC ili Wazuh.

Članak "Kako očistiti *AnonymousFox* napad" - "*How to Find & Clean Up the AnonymousFox Hack*" [3] predstavlja temeljitu analizu metodologije i štete uzrokovane *symlink* napadima s fokusom na *AnonymousFox/FoxAuto* maliciozne skripte. Članak također obuhvaća neke metode oporavka od štete i prevencije daljnjeg neovlaštenog pristupa.

Članak opisuje *AnonymousFox* set alata koji uključuje programe za pronalazak i analizu potencijalnih žrtvi na temelju poznatih ranjivosti CMSova i njihovih komponenti i nesigurno konfiguriranih *hosting* okruženja, ostvarivanje pristupa aplikacijama na kojima su prisutni artefakti prethodnih infekcija *malwareom*, kao i vršenje daljnjih napada kroz kompromitirane stranice. Članak također opisuje i proces ostvarivanja lateralnog proboja iz okvira kompromitirane stranice i postizanja ustrajnosti (engl. *persistence*) neovlaštenog pristupa, kao i metoda kojima napadači prikrivaju tragove poput enkripcije zlonamjernog koda ili ograničavanja pristupa određenim datotekama ili direktorijima.

Članak dalje obrađuje najčešće indikatore napada ovim *malwareom,* koji obuhvaćaju: nepoznate *email* adrese vezane na korisničke račune, prisutnost sumnjivih modula (engl. *plugin*) u CMS instalacijama, injekcije zlonamjernog koda u legitimnim datotekama, izmijenjena imena korisničkih računa, neobična dopuštenja ili vlasništva datoteka i slično. Kratko se dotiče i druge maliciozne aktivnosti koja se ne odnosi na sami proboj pojedinih stranica i računa, npr. postavljanje *phishing* stranica, slanje *spam* poruka putem kompromitiranih stranica, i pokretanje odlaznih napada kroz npr. *botnet malware*. Prema navedenom, ovakve maliciozne aktivnosti i jesu primarni cilj napadača u ovim slučajevima, a *symlink* napadi su zapravo samo metoda osiguravanja šireg i kontinuiranog pristupa kompromitiranih stranica.

Premda članak pruža kvalitetne upute za uklanjanje napada i mitigaciju štete proizašle iz njih, pa čak i mjere prevencije takvih napada, ne pruža informacije o alatima koji bi mogli olakšati proces oporavka.

3. PROCES *SYMLINK* NAPADA

Symlink je posebna vrsta datoteke u *Linux* operativnom sustavu koja funkcionira slično prečacu u *Windows* operativnom sustavu. Prema [4, str. 131] *symlink* datoteka ima vlastiti broj *inode* (engl. *inode number*), te služi kao pokazivač ili referenca na drugu lokaciju u datotečnom sustavu. Točnije, sama *symlink* datoteka sadrži putanju do ciljne datoteke ili direktorija, za razliku od tvrdih poveznica (engl. *hard link*) koje dijele broj *inode* sa ciljnom datotekom te u datotečnom sustavu predstavljaju alternativnu putanju do iste lokacije u memoriji [4, str. 129]. *Symlinkovi* su manje ograničeni od *hard linkova*, zbog čega se koriste za olakšavanje pristupa dijeljenim datotekama, programima, i softverskim bibliotekama u višekorisničkim računalnim sustavima.

Symlink napad je oblik kibernetičkog napada koji iskorištava ranjivosti softvera i datotečnog sustava kako bi putem *symlinka* ostvario neovlašteni pristup određenim datotekama. Manipulacijom *symlinka* sustav ili softver može biti takoreći prevaren da izvrši neku radnju koja ugrožava integritet sustava ili da izvršava radnje nad datotekama kojima ne bi trebao pristupati, najčešće sa ciljem:

- davanja pristupa odnosno iščitavanja sadržaja ciljanih datoteka ili direktorija neovlaštenim korisnicima ili procesima
- izvršavanja izmjena na datotekama koje sadrže osjetljive ili kritično važne informacije poput konfiguracijskih datoteka sustava i servisa, enkriptiranih zaporki, te sustavskih ili programskih izvršnih datoteka (engl. *executable binaries*)
- omogućavanja eskalacije privilegija korisnika, ili izvršavanja programa sa povišenim privilegijama

Prema [5] *Symlink* ranjivosti većinom proizlaze iz nepravilne konfiguracije sustava, nepravilno postavljenih kontrola pristupa, te iz uvjeta utrke (engl. *race condition*) u aplikacijama prilikom izvođenja sustavskih poziva (engl. *system call*) jezgri (engl. *kernel*) operativnog sustava.

3.1 *Apache Symlink* napad

Specifični *Apache symlink* napad obrađen u ovom radu u najvećoj mjeri zahvaća LAMP-*stack* (*Linux, Apache, MySQL, PHP*) *web hosting* poslužitelje koji koriste upravljačke ploče poput *cPanel, Plesk, InterWorx* i slično. Upravljačke ploče pružaju grafičko sučelje za upravljanje poslužiteljem, njegovim servisima, stranicama i korisničkim računima, osiguravaju međusobnu kompatibilnost programskih paketa, te djelomično automatiziraju konfiguriranje i

ažuriranje sustava. Ovakvi poslužitelji su višekorisnički u vidu raspodjele pojedinih *web* stranica (ili manje grupe stranica, npr. domene i njenih poddomena) na odvojene korisničke račune u upravljačkoj ploči, a ti računi su istovremeno definirani i kao korisnici na razini operativnog sustava. Pravilna konfiguracija pojedinih servisa poslužitelja, te pravilno postavljanje vlasništva i dopuštenja datoteka i direktorija su ključni za reguliranje aktivnosti korisničkih računa i programa koji se pod njima izvode, odnosno za provođenje kontrole pristupa korisničkim i sistemskim datotekama i programima. Pravilna kontrola pristupa se može smatrati najosnovnijom mjerom zaštite od prelaska ili invazije jednog korisničkog računa u prostor i podatke drugog, što uključuje i širenje *malwarea* između korisničkih računa i njihovih stranica. Budući da datoteke i direktoriji u pojedinom korisničkom računu imaju isto vlasništvo, nije moguće spriječiti širenje *malwarea* ili ikakvo zlonamjerno djelovanje između pojedinih stranica na istom računu, te stoga proboj jedne stranice na korisničkom računu znači da su i ostale stranice na istom računu potencijalno kompromitirane.

Apache symlink napad iskorištava ranjivost koja se javlja u načinu na koji *Apache web* poslužitelj u određenim konfiguracijama rukuje *symlinkovima* i vlasništvom nad njima. Preciznije rečeno, postoji vremenski odmak između *Apache*-eve provjere i korištenja ili kreiranja *symlinka*, tj. dolazi do stanja utrke tijekom kojeg je moguće vršiti izmjene nad *symlink* datotekom bez da ih *Apache web* poslužitelj primjeti [6]. Iskorištavanjem ove ranjivosti *Apache* napadaču dopušta pregledavanje i otvaranje datoteka i direktorija drugih korisnika, odnosno koristi se radi zaobilaženja ranije navedenih zaštitnih mjera i postizanja lateralnog proboja iz podatkovnog prostora jednog sustavskog korisnika u prostor drugoga. Spada u kategoriju oportunističkih napada jer je uspjeh napada uvjetovan nepravilnom konfiguracijom sustava i nedostatkom zaštitnih mjera. Također se može smatrati sekundarnim napadom, budući da uvjetuje da je zlonamjeran korisnik već ostvario pristup jednoj od *web* stranica ili njenom sustavskom korisničkom računu na poslužitelju, te istu koristi kao početnu točku za pokretanje *symlink* napada. Sami lateralni proboj ostvaruje se pomoću *symlinka* na konfiguracijske datoteke *web* aplikacija, odnosno datoteka koje sadrže informacije o bazi podataka koju stranica koristi. Putem ovakvog *symlinka*, napadač može otvoriti ove konfiguracijske datoteke u pregledniku te saznati nazive korištenih baza podataka, autoriziranih korisnika pojedine baze, te njihove zaporke. Pomoću *webshella* ili alata za administraciju baze podataka kroz preglednik tada može pristupiti bazi podataka druge stranice, što mu omogućava prikupljanje spremljenih informacija, ostvarivanje administratorskog pristupa stranici, ili ubacivanja malicioznih skripti u njen sadržaj.

Zbog rasprostranjenosti samostalnih virtualnih i dediciranih poslužitelja sa upravljačkim pločama razvijene su mnoge automatizirane skripte za iskorištavanje ranjivosti (engl. *exploit kit)*

koje izvršavaju *Apache* symlink napad i koje prvenstveno ciljaju ovaj tip poslužitelja, npr. *FoxAuto*, *Con7ext*, *Symlink404*, *AlfaShell* i sl.. Većinom su napisane u *PHP*, *perl*, *python*, i *bash* jezicima. Budući da se radi o samostalnim poslužiteljima ili manjim *clusterima*, postoji određena razina izolacije poslužitelja od infrastrukture pružatelja *hostinga*. Pružatelji *hostinga* vrše stalan nadzor statusa pojedinih poslužitelja i njihovih servisa radi osiguravanja dostupnosti (engl. *uptime, availability*), no nemaju potpun uvid u svu aktivnost na pojedinom poslužitelju dok mu ne pristupe. Detekcija maliciozne aktivnosti se može vršiti putem antivirusnog i IDS softvera na poslužitelju, no zbog kontinuiranog razvoja *malwarea* i metoda napada, niti jedno softversko rješenje ne može jamčiti potpunu pouzdanost u detekciju napada. Pružatelji *hostinga* se zbog ovoga i dalje oslanjaju na prijave od strane svojih klijenata tj. vlasnika pojedinih poslužitelja, te internih službi za prijavu zloupotreba (engl. *abuse desk*) koje od trećih strana primaju i obrađuju prijave o zloupotrebi svojih usluga. Zloupotreba (engl. *abuse*) u ovom kontekstu može predstavljati namjernu malicioznu aktivnost koju vrši vlasnik poslužitelja, ili da je poslužitelj eksploatiran od strane vanjskih napadača. Ovo doduše znači da pružatelji *hostinga* ne budu nužno obaviješteni o sigurnosnim incidentima čim se dogode, već može proći neko vrijeme prije nego što maliciozna aktivnost bude uočena.

Ranjivosti ovih poslužitelja također doprinosi i to što su poslužitelji i njihove stranice primarno pod kontrolom vlasnika poslužitelja, što ih otvara raznim vektorima napada ako vlasnik postupa neodgovorno, tj. vrši promjene u konfiguraciji sustava bez potpunog razumijevanja njihovog utjecaja, ne vrši redovno održavanje sustava i stranica ili se ne pridržava prikladnih sigurnosnih pravila i standarda. Standardizacija lokacija podataka pojedinih korisnika također doprinosi ranjivosti, tj. čini putanje do pojedinih datoteka predvidljivim. Ovo uključuje i početne direktorije (engl. *document root*) pojedinih stranica - npr. u *cPanel* upravljačkoj ploči *web* stranica *example.com* korisnika “example” po zadanim postavkama će se nalaziti na lokaciji */home/example/public_html/*. Lokacije se naravno mogu izmijeniti, te će dodatne stranice na istom korisničkom računu imati druge početne direktorije, no ostvarivanjem pristupa bilo kojoj stranici na računu napadač vrlo lako može pristupiti i drugima putem malicioznih kontrolnih skripti (engl. *webshell*) ili stražnjih ulaza (engl. *backdoor*).

Budući da je ovaj tip poslužitelja popularan među privatnim korisnicima, te malim do srednje velikim organizacijama i tvrtkama, učestala je i upotreba CMS softvera i razvojnih okvira (engl. *development framework*) - softverskih paketa za izradu *web* stranica. Prema [8] preko 68% web stranica koristi neki CMS ili *framework,* dok se *WordPress* CMS koristi u 43.2% svih web stranica na internetu. Korištenje CMS-a olakšava izradu i održavanje *web* stranice, ali čini njenu podatkovnu strukturu predvidljivom, i u slučaju neadekvatnog održavanja

čini ju podložnom napadima putem već evidentiranih i u ažurnim verzijama softvera već otklonjenih ranjivosti. Mnogi CMS-ovi koriste softverske biblioteke trećih strana, te korisniku dozvoljavaju instalaciju dodatnih softverskih komponenti poput modula, tema, i ekstenzija, koje također mogu koristiti vanjske softverske biblioteke. Sama veličina sumirane baze koda svih ovih komponenti predstavlja potencijalno povećanje površine napada, odnosno čini ove stranice ranjivima na indirektne napade putem softverskog lanca opskrbe (engl. *supply chain attack*) [9]. Čak i u slučaju da su stranica i sve njezine komponentne potpuno ažurne i da koristi proaktivne mjere zaštite, i dalje su mogući napadi putem dotad nepoznatih ranjivosti, tzv. ranjivosti nultog dana (engl. *zero-day exploit*). Široka upotreba i stalan razvoj CMS softvera također je motivirajući faktor za *white-hat* sigurnosne analitičare i *black-hat* hakere, te obje strane kontinuirano rade na otkrivanju dosad nepoznatih ranjivosti.

3.2 Tok Apache *Symlink* napada

Uzimajući u obzir informacije iznesene u prethodnom potpoglavlju, za demonstraciju napada i izvedbu programskog rješenja ovog rada korišten je poslužitelj s *cPanel* upravljačkom pločom, te nekoliko stranica koje koriste *WordPress* CMS. Detaljnije specifikacije poslužitelja nalaze se u prilogu P.3.1. U nastavku je opisan uobičajen tijek događaja prilikom oportunističkog napada na ovakvu stranicu i poslužitelj, te daje kratki pregled moguće nanesene štete u pojedinom koraku.

3.2.1 Pronalazak potencijalno ranjive aplikacije

Koristeći programe za pretraživanje i indeksiranje (engl. *crawler*) napadač pronalazi *web* aplikaciju sa potencijalnom ranjivošću - na primjer zastarjela verzija CMS-a, prisutnost određenog modula, ili prisutnost poznatog zlonamjernog koda. *Crawleri* također mogu detektirati prisutnost ranjivih elementa aplikacije poput konfiguracijskih ili informacijskih datoteka kojima se može izravno pristupiti zbog pogrešno postavljene kontrole pristupa. Popisi prethodno otkrivenih ranjivih stranica također se mogu pronaći i na hakerskim forumima.

3.2.2 Enumeracija ranjivosti aplikacije i poslužitelja

Pomoću analitičkog programa prikladnog za pojedini CMS (npr. *wpscan* za *WordPress*) [10] napadač dolazi do konkretnih informacija o aplikaciji poput precizne verzije CMS-a i instaliranih tema i modula. Dobivene informacije se potom dalje istražuju uspoređivanjem sa bazama podataka poznatih ranjivosti poput *vuldb.com* u nadi da postoje ranjivosti koje nisu već

otklonjene ili zakrpane u instaliranoj verziji aplikacije ili njenih komponenti. Ove baze često sadrže i upute za testiranje ranjivosti, koje napadač može prenamijeniti u upute za izvršavanje napada. Napadač može vršiti dodatne testove poput skeniranja portova poslužitelja, ručne provjere određenih funkcija aplikacije, ustanoviti koristi li aplikacija neki eksterni vatrozid ili obrnuti *proxy*, ili kreirati korisnički račun na stranici u svrhu testiranja privatnog dijela stranice ili u pokušaju socijalnog inženjerstva. Budući da su *exploit kit*-ovi veoma raširenи te da su imena pojedinih malicioznih datoteka iz ovih paketa napadačima često već poznata, napadač također može tražiti postojeće *backdoor* ili *webshell* skripte na predvidljivim lokacijama ručno ili pomoću tzv. *shellfinder* alata [11].

3.2.3 Eksploatacija ranjivosti - ostvarivanje inicijalnog pristupa

Kada napadač potvrdi prisutnost neke ranjivosti, iskorištava ju da bi ostvario višu razinu pristupa stranici. Točan način eksploatacije ranjivosti ovisi od njenog tipa, no neki od najčešćih oblika ovih inicijalnih napada su: direktan prijenos malicioznih programa kroz forme koje nepravilno vrše validaciju tipa prenesene datoteke, manipulacija sesijama, i iskorištavanje ranjivih ili nezaštićenih elemenata aplikacije za izvršavanje proizvoljnog koda (engl. *Arbitrary Code Execution*) - npr. SQL injekcije u polja za unos teksta koja nepravilno sanitiziraju unose ili napada pretekom međuspremnika (engl. *buffer overflow*). Bez obzira na metodu, napadačima je primarni cilj osigurati si kontinuiran pristup stranici, pa je prva datoteka koju podižu gotovo uvijek neki oblik *webshell,* ili *backdoor* skripte. Kroz takvu skriptu mogu dalje prenositi dodatne zloćudne skripte, ili injekcijama malicioznog koda u bazu podataka i legitimne datoteke stranice osigurati da se takve skripte kontinuirano ponovno kreiraju na stranici. Napadači često lančano prenose dodatne maliciozne datoteke kroz prethodno prenesene maliciozne datoteke pa obrišu one ranije kako bi prikrili prvotni ulazni vektor napada.

3.2.4 Preuzimanje kontrole nad stranicom

Nakon inicijalnog proboja napadač pokušava pristupiti administracijskom sučelju aplikacije. Administratorski pristup može biti lakše ostvariv ako je stranica izgrađena na nekom od CMS-ova ili razvojnih okvira čije su struktura i funkcioniranje poznati i javno dokumentirani. Svaka *web* aplikacija koja koristi neku bazu podataka mora u sebi imati i konfiguracijsku datoteku koja sadrži lokaciju i naziv baze podataka, naziv ovlaštenog korisničkog računa za pristup bazi, te njegovu zaporku. Relativno na početni direktorij stranice, ovi podaci se nalaze u *./wp-config.php* datoteci kod *WordPress*-a, kod *Magento*-a u *./app/etc/env.php*, kod *Joomla*-e u *./configuration.php* itd.. Koristeći ove podatke, napadač putem *webshell*-a ili alata za

administraciju baze podataka (npr. *adminer*, *Mysql Workbench*) može pristupiti bazi podataka stranice te promijeniti zaporku postojećeg administratorskog računa (ili *email* adresu na koju je račun vezan) i preoteti ga, ili jednostavno kreirati novi korisnički račun sa administratorskim privilegijama. Već u ovom koraku napadač može napraviti veliku štetu - od pristupa osjetljivim podacima stranice i njenih korisnika, malicioznih izmjena koda i sadržaja, do instalacije raznog *malwarea* i *phishing* stranica.

3.2.5 Preuzimanje kontrole nad *cPanel* korisničkim računom

U starijim verzijama *cPanel* kontrolne ploče, ukoliko administrator poslužitelja omogućava korisnicima tj. pojedinačnim *cPanel* korisničkim računima da sami resetiraju zaporke putem elektroničke pošte i ne prisiljava korisnike na korištenje višefaktorske autentifikacije, napadač može preuzeti kontrolu nad samim *cPanel* računom. Ovo je moguće jer se kontakt adresa *cPanel* računa sprema u *.contactinfo* i *.contactemail* datoteke koje se nalaze u direktoriju računa */home/$username/* i u njegovom su vlasništvu, tako da ih napadač može urediti i resetirati zaporku računa putem vlastite mail adrese. U novijim verzijama *cPanel*-a, kontakt informacije se spremaju u konfiguracijske datoteke kojima je vlasnik *root,* prikazano na slici 3.1. Pomoću *webshella* napadač i dalje može pročitati ovu konfiguracijsku datoteku budući da je čitljiva korisnikovoj grupi, ali ju ne može mijenjati. *cPanel* račun i dalje može biti kompromitiran ukoliko napadač uspije ostvariti pristup navedenom *email* računu, ili se taj račun nalazi lokalno.

```
[example@host ~]$ ls -lah /var/cpanel/users/example
-rw-r----- 1 root example 1.2K Aug 24 01:31 /var/cpanel/users/example
[example@host ~]$ grep -i contactemail /var/cpanel/users/example
CONTACTEMAIL=admin@example.com
CONTACTEMAIL2=
```

Slika 3.1: Primjer konfiguracijske datoteke cPanel računa i prikaz navedene *email* adrese

Ovaj korak nije nužan za provođenje daljnjih napada kroz kompromitiranu aplikaciju, ali ih olakšava ukoliko je ostvariv. Pristupom *cPanel* računu napadač ostvaruje izravan pristup podacima i datotečnoj strukturi stranica na računu putem samog *cPanel*-a i SSH, njihovim bazama podataka, određenim konfiguracijskim opcijama (npr. *crontab* i podržane *PHP* funkcije), mogućnost upravljanja postojećim *email* računima i kreiranju novih, ali i kreiranju novih stranica i dodavanju domena ili poddomena na račun.

3.2.6 *Symlink* napad

Bez obzira na uspjeh ili neuspjeh u preuzimanju kontrole nad stranicom ili njenim *cPanel* računom, napadač kreće u lateralni proboj na poslužitelju. Koristeći neki od dostupnih *exploit kit*-ova (npr. *FoxAuto*), napadač izvršava *symlink* napad, odnosno pomoću maliciozne skripte čita */etc/passwd* datoteku te prema navedenim korisničkim računima kreira *symlinkove* na moguće lokacije konfiguracijskih datoteka za pristup bazama podataka, većinom se vodeći strukturama raznih CMS-ova. Pojednostavljen primjer je prikazan na slici 3.2. Rezultat ovog pristupa je da se uz funkcionalne *symlinkove* koji pokazuju na postojeće konfiguracijske datoteke drugih stranica kreiraju i stotine ili tisuće nefunkcionalnih *symlinka* - ovisno od korištene skripte i samog broja *cPanel* korisničkih računa na poslužitelju. Napadač potom preko funkcionalnih *symlinkova* može pročitati pristupne podatke za baze podataka stranica drugih *cPanel* računa, te pristupiti i njima kako je objašnjeno u potpoglavlju 3.2.4.

```
#!/bin/bash
mkdir symtest;
for user in `cut -d: -f1 /etc/passwd`;
do
ln -s /home/$user/public_html/wp-config.php symtest/$user-wp-config.txt;
ln -s /home/$user/public_html/wordpress/wp-config.php symtest/$user-word-wp.txt;
ln -s /home/$user/public_html/blog/wp-config.php symtest/$user-wpblog.txt;
ln -s /home/$user/public_html/configuration.php symtest/$user-joomla-or-whmcs.txt;
ln -s /home/$user/public_html/joomla/configuration.php symtest/$user-joomla.txt;
ln -s /home/$user/public_html/conf_global.php symtest/$user-conf_global.txt;
ln -s /home/$user/public_html/inc/config.php symtest/$user-inc.txt;
ln -s /home/$user/public_html/config.php symtest/$user-config.txt;
ln -s /home/$user/public_html/whmcs/configuration.php symtest/$user-whmcs.txt;
done
```

Slika 3.2: Pojednostavljena skripta za demonstraciju *symlink* napada

Na *cPanel* poslužiteljima ovaj tip napada također može izravno dovesti do potpunog proboja poslužitelja, tj. do neovlaštenog pristupa privilegiranom *root* korisničkom računu bez korištenja dodatnog *rootkit malware*-a ili iskorištavanja drugih ranjivosti. Naime, *cPanel* softverski paket obuhvaća i sučelje za upravljanje poslužiteljem WHM (*Web Host Manager)* kojemu se isključivo pristupa *root* i opcionalnim preprodavačkim (engl. *reseller*) korisničkim računima ukoliko postoje. *Root* račun ima pun pristup sustavu, kao i na svakoj *Linux* distribuciji, dok preprodavački (koji su istovremeno *cPanel* računi) računi imaju ograničen pristup WHM-u, ali pun pristup drugim *cPanel* računima kojima su vlasnici. Određeni softverski paketi trećih strana zahtijevaju pristup nekim funkcijama sustava koje su inače dostupne isključivo root ili

preprodavačkim računima. *cPanel* ovakvim aplikacijama omogućava privilegirani pristup svojim funkcijama putem API ključeva.

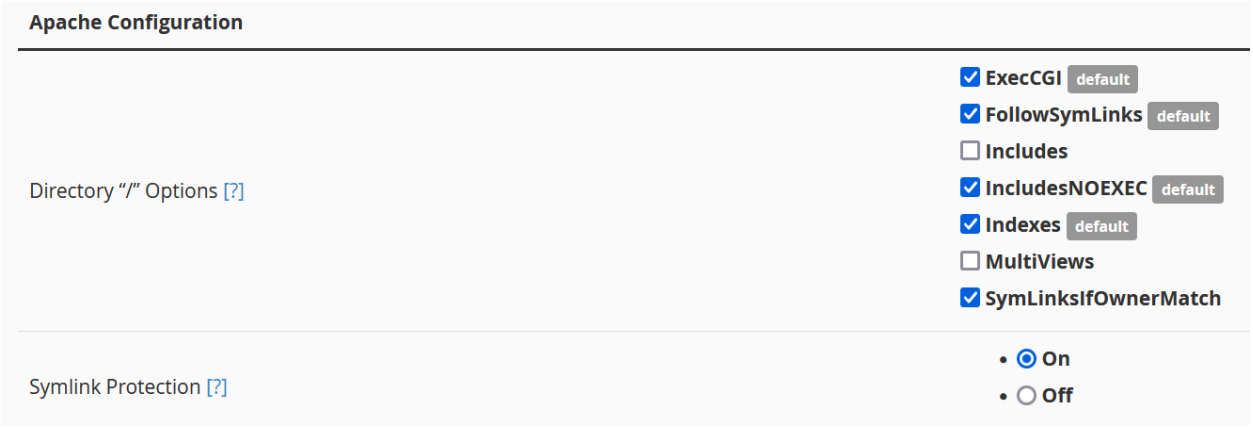
Na primjer, *WHMCS* je platforma za fakturiranje, upravljanje poslužiteljem, i komunikaciju sa klijentima [12]. Često ju koriste manji *hosting* preprodavači budući da se lako integrira sa postojećim *cPanel* i *Plesk* poslužiteljima, i instalira se unutar poslužitelja tj. na vlastiti korisnički račun, ali zahtjeva pristup nekim privilegiranim funkcijama. Ukoliko ovakav *cPanel* račun odnosno *WHMCS* ili slična aplikacija koja ima root API ključ bude probijena, napadač može promijeniti zaporku *root* računa i ostvariti pun pristup poslužitelju. Na poslužiteljima sa *cPanelom* ispod verzije 64 napadač također može pristupiti *.accesshash* datoteci *root* korisnika, te manipulacijom HTTP zaglavlja u pregledniku pristupiti WHM sučelju kao *root* [13]. Bitno je napomenuti da je u slučaju proboja *root* računa postaje nemoguće sa sigurnošću odrediti jesu li sa sustava uklonjene sve zloćudne izmjene. U ovim situacijama jedine sigurne opcije oporavka su:

- oporavak iz kopije cijelog sustava (engl. *system snapshot*), takozvani "*bare-metal restore"*
- reinstalacija operativnog sustava te oporavak pojedinih korisničkih računa i stranica iz sigurnosnih kopija
- migracija korisničkih računa i stranica na novi poslužitelj, te uklanjanje starog

3.3 Prevencija *Apache symlink* napada

U *cPanel* WHM sučelju za upravljanje konfiguracijom *Apache web* poslužiteljem *root* korisnik, odnosno vlasnik poslužitelja, može podesiti neke postavke koje utječu na izloženost poslužitelja ovom tipu napada, prikazano na slici 3.3. Ukoliko opcija "*FollowSymLinks*" nije omogućena, *Apache* neće pratiti simboličke veze već vratiti grešku. Ako je omogućena, *Apache* će pratiti *symlink* bez provjere vlasništva *symlinka* i ciljane datoteke. Premda je ova opcija iznimno nesigurna, neki korisnici ju aktiviraju u svrhu dijeljenja pristupa datotekama između pojedinih *cPanel* računa kako bi uštedjeli prostor na disku - npr. ukoliko više stranica na različitim računima treba pristup velikim datotekama poput dokumenata, medijskih sadržaja ili arhiva. Ako je uz "*FollowSymLinks*" omogućena i opcija "*SymLinksIfOwnerMatch*", *Apache* će vršiti provjeru vlasništva *symlinka* i ciljne datoteke, ali istovremeno ga ova opcija otvara ranjivosti putem uvjeta utrke [14]. Onemogućavanje opcija može spriječiti provedbu *symlink* napada ukoliko je u glavnoj *Apache* konfiguracijskoj datoteci */etc/apache2/conf/httpd.conf* pravilno podešena opcija "*AllowOverride*" koja definira koje postavke u pojedinim *.htaccess*

datotekama mogu nadjačati zadane postavke dane u *httpd.conf* i konfiguracijama virtualnih poslužitelja (engl. *virtual host - vhost*)[15]. Ako je "*AllowOverride*" opcija permisivno postavljena, napadač može urediti postojeću *.htaccess* datoteku stranice i omogućiti opcije "*FollowSymLinks*" i "*SymLinksIfOwnerMatch*".



Slika 3.3: Postavke vezane za *symlink* ranjivosti u globalnoj konfiguraciji *Apache* poslužitelja

Izuzev pravilne konfiguracije poslužitelja, također postoje nekoliko načina zaštite *symlinka* od maliciozne aktivnosti čak i kad su obje navedene opcije omogućene:

- *Bluehost* zakrpa - na slici 3.3 je u *Apache* konfiguraciji prikazana "*Symlink Protection*" opcija. Omogućavanjem ove opcije instalira se varijanta tzv. *Bluehost zakrpe* prilagođena za *cPanel* poslužitelje. Iako je ovo rješenje implementirao *cPanel*, pokazalo se neučinkovitim i ima negativan utjecaj na potrošnju resursa sustava, stoga sami *cPanel* preporuča da se ne koristi osim ako korisnik nema drugih mogućnosti.
- *KernelCare symlink* zakrpa - ova zakrpa je veoma učinkovita u sprječavanju *symlink* napada jer djeluje na razini *kernela*. Dolazi u besplatnoj i plaćenoj varijanti *kernelcare* paketa. Ključni nedostatak ove opcije je što *KernelCare*-ov raspored nadogradnji ne prati raspored nadogradnji *Linux kernela* i pojedinih distribucija *Linuxa*, pa sustav može postati izložen *symlink* napadu u vremenu između nadogradnje sustava i nadogradnje *kernelcare* zakrpe.
- *CloudLinux OS* - *CloudLinux* je *Linux* distribucija bazirana na *CentOS Linux OS*-u, razvijena specifično za višekorisnička *hosting* okruženja. Među sigurnosnim značajkama *CloudLinuxa* su *CageFS* virtualizirani datotečni sustav koji stvara dodatnu razinu izolacije među korisničkim računima, te *SecureLinks* tehnologija za sprječavanje *symlink*

napada koja djeluje na razini *kernela*. *SecureLinks* djeluje slično *KernelCare* zakrpi no vezana je za *CloudLinux kernel* pa ne može doći do *symlink* ranjivosti pri nadogradnji *kernela*. Premda je ovo rješenje najučinkovitije u sprječavanju *symlink* napada i ima mnoge druge korisne značajke, na *CloudLinux OS* se plaća pretplata pa u konačnici odluka o njegovom korištenju pada na vlasnika poslužitelja.

Mnogi poslužitelji i dalje ne koriste nijednu od ovih preventivnih mjera, bilo zbog nemara ili neosviještenosti vlasnika ili pružatelja usluga *hostinga*, tehničkih ograničenja, ili zbog nužnosti nekih od funkcija na koje bi ove zaštitne mjere mogle negativno utjecati.

4. DETEKCIJA I MITIGACIJA *APACHE SYMLINK* NAPADA

Veliki pružatelji *hosting* usluga imaju ograničene mogućnosti proaktivnog nadzora i prevencije maliciozne aktivnosti na svojim poslužiteljima jer održavaju desetke tisuća virtualnih i dediciranih uređaja. Odjeli za zaprimanje prijava o zloupotrebi većinom zaprimaju prijave o direktnim ugrozama sigurnosti, tj. očiglednim problemima poput individualnih *phishing* stranica, DDOS ili *spam* napada sa pojedinih poslužitelja, pokušajima neovlaštenog pristupa koji dolaze sa određene IP adrese i slično. Ovakvi pojedinačni incidenti nisu nužno indikativni za prisutnost opširnijeg sustavnog sigurnosnog incidenta na nekom poslužitelju. Ovisno od organizacije posla na odjelu prijava, tehničari mogu primjetiti da primaju natprosječan broj prijava za pojedini poslužitelj ili da se određeni prijavljen i prethodno riješen problem kontinuirano vraća, te iz toga otkriti prisutnost nekog sekundarnog incidenta poput *symlink* napada ili proboja *root* korisnika. Unatoč tome, može proći značajan period prije nego se na ovaj način otkrije sekundarni incident, te u tom periodu već može biti počinjena značajna šteta u vidu kompromitiranih stranica i podataka, i narušene reputacije IP adresa poslužitelja. Jedan od glavnih problema, ukoliko prođe dovoljno vremena prije otkrivanja sekundarnog incidenta, je što automatizirane sigurnosne kopije stranica već mogu sadržavati maliciozne datoteke ili injektiran maliciozan kod u legitimnim datotekama i bazi. Ako čiste sigurnosne kopije ipak postoje ali su stare, njihovo korištenje bi značilo ili značajan gubitak podataka ili potrebu za intenzivnim radom na stranici, pri tome pobijajući smisao oporavljanja iz sigurnosnih kopija. Prvi dio programskog rješenja ovog rada je stoga fokusiran na brzu i proaktivnu detekciju *symlink* napada uz minimalne tehničke zahtjeve putem skripte *foxhound.sh* opisane u potpoglavlju 4.1.

Zbog prirode *symlink* napada, odnosno štete koju takav napad može uzrokovati, nije dovoljno samo ukloniti maliciozne *symlinkove* i *malware* sa kompromitiranih *web* stranica. Mitigacija (engl. *mitigation*), odnosno ublažavanje posljedica ovakvog incidenta obuhvaća više koraka koje je potrebno pratiti kako bi se problem otklonio i kako bi se spriječio daljnji neovlašteni pristup kompromitiranim stranicama i korisničkim računima, te su koraci ovog procesa obrađeni kroz potpoglavlja 4.2, 4.3, i 4.4. Drugi dio programskog rješenja ovog rada je skripta *foxdie.sh*, namijenjena za brzo resetiranje zaporki administratorskih računa, zaporki baza podataka, i ažuriranje istih u konfiguracijama zahvaćenih stranica, opisana u potpoglavlju 4.4.

4.1 Detekcija *symlink* napada

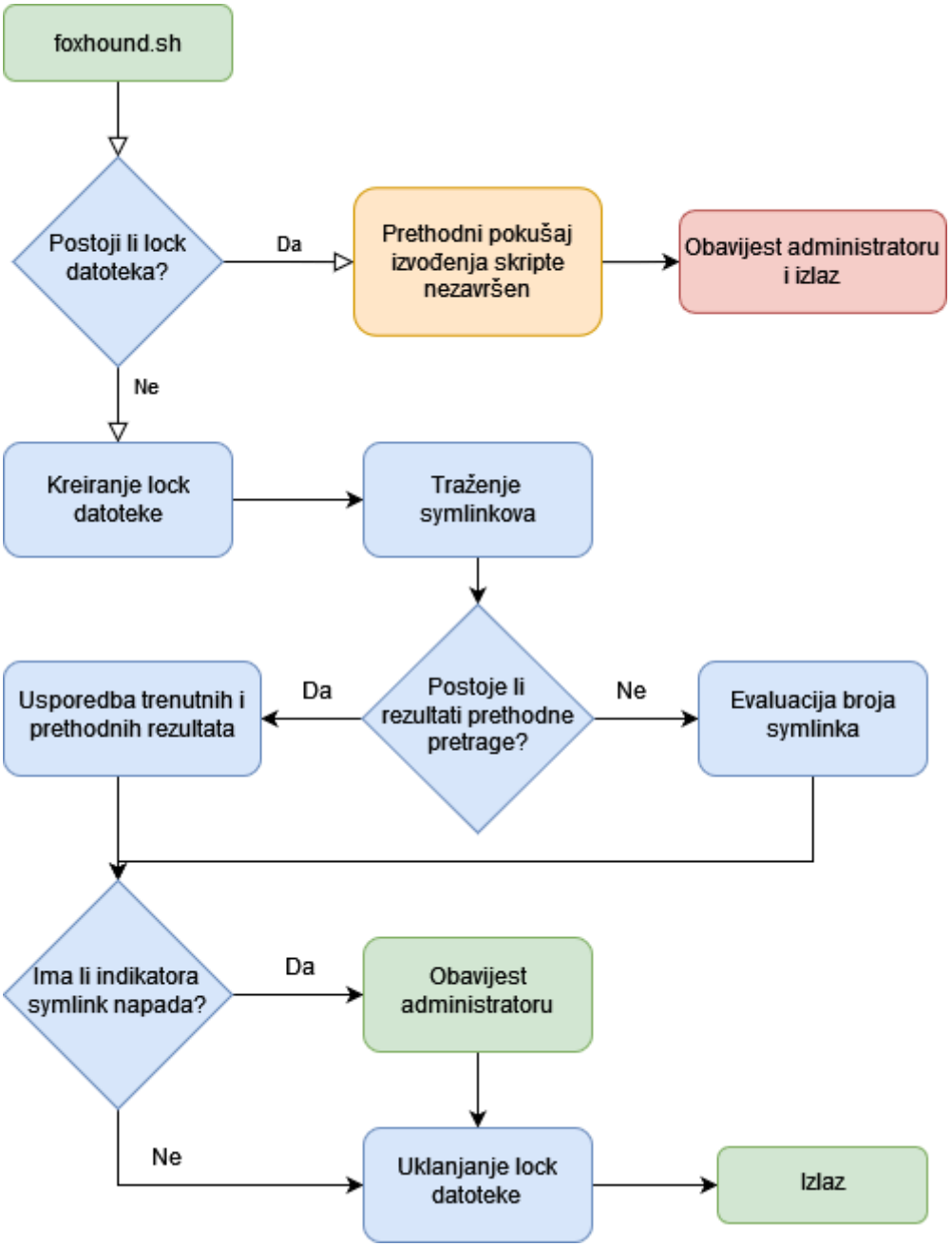
Kako bi reakcija na *symlink* napad bila što efikasnija, detekcija treba biti pouzdana i brza, a zbog neujednačenosti poslužiteljskih okruženja u flotama pružatelja *hostinga* detekcijska

skripta ne smije biti ovisna o drugim softverskim bibliotekama i paketima. Skripta također mora biti jednostavna za instalaciju kako bi ju pružatelj *hostinga* ili administrator mreže mogao implementirati u svim poslužiteljima na koje je primjenjiva bez direktne intervencije na pojedinom poslužitelju. Detekcijska skripta *foxhound.sh* dana u prilogu P.4.1 je stoga pisana u *bash* skriptnom jeziku i koristi isključivo njegove ugrađene funkcije, izvršava se kao dnevni *cron* zadatak pod *root* korisnikom, te u slučaju pravilnog izvršavanja treba detektirati *symlink* napad unutar manje od dvadeset i četiri sata od vremena incidenta. Instalacija se može vršiti uključivanjem skripte u neki već postojeći softverski paket kojim upravlja pružatelj *hosting* usluge (npr. neki namjenski RPM paket), ili udaljenim izvršavanjem *shell* skripte putem alata za automatizaciju infrastrukture poput *Ansible* ili *Salt*. Jedina izmjena koju na *foxhound.sh* skripti treba izvršiti prilikom instalacije je ažuriranje *email* adrese administratora na koju će se slati obavijesti. Slijedi primjer jednostavne *shell* skripte za instalaciju:

```
wget
https://raw.githubusercontent.com/j-molnar/foxhound/main/foxhoun
d.sh; sed -i 's/mailaddress/admin@email.addr/g' foxhound.sh;
chmod +x foxhound.sh; mv foxhound.sh /etc/cron.daily/;
```

Dijagram toka *foxhound.sh* skripte prikazan je na slici 4.1. Pokretanje skripte započinje deklaracijom i inicijalizacijom varijabli koje su potrebne u daljnjem izvođenju, te kreiranjem radnog direktorija skripte, označenog datumom i vremenom izvođenja skripte. Zatim se provjerava postoji li datoteka */home/temp/foxhound/lock* koja služi kako bi se spriječilo paralelno izvođenje više instanci skripte. Do ovakvih situacija ne bi trebalo doći na poslužiteljima koji rade unutar očekivanih parametara, ali nisu nemoguće - npr. ako se na poslužitelju nalazi iznimno velika količina podataka ili su resursi poslužitelja (CPU, RAM, disk I/O) preopterećeni. Prisutnost ove datoteke također može značiti da je prethodni pokušaj izvršavanja skripte iz nekog razloga prekinut. U oba slučaja, skripta se zaustavlja i administratoru šalje obavijest da je potrebna intervencija.

Ukoliko *lock* datoteka ne postoji, ona se kreira, te se pokreće funkcija *scan*. Ova funkcija vrši potragu za *symlinkovima* unutar početnih direktorija stranica na poslužitelju, tj direktorijima kojima se može pristupiti preko interneta. U sklopu ove skripte se također testiraju funkcionalni *symlinkovi*, tj. oni koji vode na postojeće datoteke. Rezultati pretrage i ovog testa se potom spremaju u zasebne datoteke unutar radnog direktorija skripte.



Slika 4.1 Dijagram toka *foxhound.sh* detekcijske skripte

Skripta zatim provjerava jesu li već prisutni rezultati prethodnog izvršavanja skripte na *symlinku /home/temp/foxhound/latest*, koji se kreira odnosno ažurira pri svakom uspješnom izvršavanju *foxhound.sh* skripte. Ako ovaj *symlink* postoji, vrši se usporedba broja *symlinka* detektiranih u prethodnom i trenutnom izvršavanju skripte pomoću funkcije *compare*. Ako ne postoji tj. ako se skripta izvodi prvi put, umjesto usporedbe vrši se evaluacija vjerojatnosti prisutnosti prethodno postojećih *symlink* napada prema broju prisutnih *symlinka* u odnosu na broj *cPanel* računa prisutnih na poslužitelju. Ukoliko postoje indikatori *symlink* incidenta, šalje se obavijest administratoru s informacijama o poslužitelju (ime i IP adrese) i kriteriju detekcije.

Primjer ove obavijesti prikazan je na slici 4.2. U konačnici, uklanja se *lock* datoteka, ažurira se *symlink latest*, i skripta završava izvršavanje.

Slika 4.2: Primjer *foxhound.sh* obavijesti o detektiranom *symlink* napadu

4.2 Analiza sigurnosnog incidenta

Kada se otkrije prisutnost ikakvog sigurnosnog incidenta, pa tako i *symlink* napada, prije nego što se pristupi otklanjanju problema nužno je prikupiti što više informacija o incidentu kako bi se daljnji proboji mogli spriječiti. Primarni ciljevi analize incidenta su određivanje opsega napada i vremenskog perioda u kojemu je došlo do proboja, te pronalazak izvornog ulaznog vektora - tj. ranjivosti zbog koje je do incidenta uopće došlo.

Općenito govoreći, potrebno je provjeriti koje stranice i sistemski računi su zahvaćeni napadom, te postoje li indikatori dubljeg proboja - npr. neovlaštenog pristupa *root* ili preprodavačkim računima. Poslužitelj je potrebno skenirati antivirusnim programom (npr. *maldet* ili *Imunify*), i evidentirati bitne metapodatke (engl. *metadata*) detektiranih malicioznih ili kompromitiranih datoteka. U bitne metapodatke spadaju lokacija, vlasništvo, vrijeme izmjene u inodi (engl. *change time, ctime*), vrijeme izmjene sadržaja datoteke (engl. *modify time, mtime*), te ukoliko datotečni sustav podržava tu opciju i vrijeme kreiranja datoteke (engl. *creation/birth time*). *Ctime* i vrijeme kreiranja su većinom korisniji od *mtime* jer se njihovim vrijednostima ne može manipulirati bez pristupa *root* računu, dok *mtime* može promijeniti i nepovlašteni korisnik. Osim datoteka koje sadrže maliciozan kod, također je korisno provjeriti vremena izmjene konfiguracijskih datoteka, dodavanja dodatnih domena ili poddomena na račune, i posljednjih izmjena zaporki pojedinih korisničkih računa poslužitelja jer velik broj izmjena u kratkom

vremenskom periodu također može ukazivati na proboj. Specifično za *symlink* napade, pregledom samih malicioznih *symlinkova* može se odrediti prema kojim konfiguracijskim datotekama postoje funkcionalni *symlinkovi*, odnosno koje stranice su potencijalno pogođene ovim napadom. *Symlinkove* se može testirati pristupanjem direktoriju u kojem se nalaze putem preglednika, te otvaranjem pojedinih *symlinka*. Ukoliko *symlink* otvara konfiguracijsku datoteku neke druge stranice na poslužitelju, napad je bio uspješan.

Uspoređivanjem vremenskih oznaka pojedinih datoteka sa zapisnicima (engl. *log*) servisa poslužitelja (npr. FTP, *Apache*, *PHP*) može se odrediti kako i odakle (tj. sa koje IP adrese) je napad izveden, kako je pojedina zloćudna izmjena izvršena, ili putem koje datoteke je neka druga kreirana itd.. Cilj je uspostaviti slijed događaja u incidentu, i u konačnici pronaći izvorni ulazni vektor. Treba imati na umu i da nisu sve zloćudne izmjene nužno rezultat jednog napada, tj. da je moguće da su neke datoteke ili čak stranice kompromitirane u prijašnjim incidentima koji nisu povezani sa trenutnim. Također je bitno napomenuti da napadači mogu prikriti svoje tragove, najčešće uređivanjem zapisnika ili namjernom manipulacijom metapodataka datoteka kako se ne bi mogla uspostaviti korelacija između pojedinih malicioznih datoteka i događaja, pa otkrivanje konkretnog ulaznog vektora nije uvijek moguće. Vlasnici i administratori stranica također mogu nenamjerno doprinijeti prikrivanju ulaznog vektora - npr. oporavljanjem iz kompromitiranih sigurnosnih kopija, što bi izmjenilo vremenske oznake datoteka relevantnih za istragu. U svakom slučaju, odmah po završetku analize incidenta potrebno je omogućiti neku od metoda prevencije *symlink* napada, kako ne bi došlo do daljnjih napada dok traje sljedeća faza mitigacije. Čak i u slučaju da je inicijalni ulazni vektor nepoznat, sprječavanjem daljnjih *symlink* napada postiže se da će potencijalna buduća šteta biti ograničena na manji broj pojedinih stranica i računa.

4.3 Oporavak iz sigurnosnih kopija ili uklanjanje *malwarea*

Najbrži i najefikasniji način mitigacije sigurnosnih incidenata je oporavljanje kompromitiranih stranica i njihovih računa iz čistih sigurnosnih kopija ukoliko postoje. Budući da je *foxhound.sh* skripta predviđena da prijavi ovakav incident u roku od približno jednog dana otkad se dogodio, u većini slučajeva bi oporavak iz sigurnosnih kopija trebao biti moguć i istovremeno imati minimalan negativan utjecaj na stranice na poslužitelju i njihovu dostupnost. Nakon oporavka iz sigurnosnih kopija, napadaču treba onemogućiti daljnji pristup poslužitelju kako je opisano u sljedećem potpoglavlju.

Ukoliko *foxhound.sh* odmah po instalaciji na poslužitelj detektira prisutnost prethodno postojećih *symlink* napada, pravilan oporavak iz sigurnosnih kopija vjerojatno neće biti izvediv, npr. ako je ili prošlo previše vremena od *symlink* napada, ili ako se ne može sa sigurnošću utvrditi kada se napad dogodio. U ovom slučaju, ali i u slučaju da sigurnosne kopije nisu dostupne ili nisu potpuno čiste, *symlink* napad i drugi *malware* se mora čistiti. Zavisno od vrste poslužitelja i od pružatelja usluga *hostinga* čišćenje *malwarea* može biti potpuno ili djelomično pokriveno njihovom podrškom ili uključeno u neki softverski paket. Ako nije, vlasnik poslužitelja odnosno pojedine stranice može posegnuti za uslugama trećih strana, ili čišćenje može obaviti administrator ili razvojni programer pojedine stranice. Kod mnogih CMS sustava često je jednostavnije i sigurnije stvoriti novu instalaciju stranice i uvesti potrebne podatke iz prethodne instalacije nego ih očistiti. U ovakvim situacijama, također je bolje provesti prevenciju neovlaštenog pristupa prije samog procesa čišćenja, te se uz to pobrinuti da zaražene komponente budu izolirane, tj. da se maliciozan kod u njima ne može izvršavati tijekom čišćenja ili rekreiranja stranice.

4.4 Prevencija daljnjeg neovlaštenog pristupa

Oporavak iz sigurnosnih kopija i čišćenje stranica osigurava da stranice više ne sadrže zlonamjeran kod, no potrebno je poduzeti daljnje korake kako bi se spriječilo njihovu daljnju zloupotrebu. Uz zatvaranje ulaznih vektora otkrivenih u ranijoj istrazi, također je potrebno ukloniti sve poznate ranjivosti, odnosno druge potencijalne ulazne vektore. Kod aplikacija građenim na CMS-ovima ili razvojnim okvirima, ovo većinom obuhvaća vršenje nadogradnji aplikacije i njenih komponenti na najnovije dostupne verzije, a isto pravilo je primjenjivo i na softverske biblioteke trećih strana u drugim web aplikacijama. U oba slučaja također je potrebno napraviti evaluaciju koda aplikacije te zamijeniti ranjive funkcije sigurnijim alternativama ili ograničiti pristup takvim funkcijama, te prilagoditi konfiguracije samih aplikacija i relevantnih servisa na poslužitelju (npr. postavke *Apache*-a i *PHP*-a). Skenovi za ispitivanje ranjivosti *web* aplikacija (engl. *Website Vulnerability Assessment Scan*) su jednostavan oblik penetracijskog testiranja i veoma koristan alat koji administratorima može pomoći procijeniti koje konfiguracijske promjene su potrebne, i kako ih prioritizirati na osnovu CVSS vrijednosti - Zajedničkog Sustava Ocjenjivanja Ranjivosti (engl. *Common Vulnerability Scoring System*)[16].

Nadalje, potrebno je resetirati zaporke za sve elemente *hosting* okruženja koje bi napadač mogao iskoristiti da ponovno ostvari pristup pojedinim stranicama: *cPanel* računi, FTP računi, *email* računi, administratorski računi *web* aplikacija, te računi za pristup bazama podataka.

Vlasnik ili administrator poslužitelja, odnosno pojedine stranice, također treba odraditi reviziju korisničkih računa da bi utvrdio jesu li svi računi povezani sa legitimnim *email* adresama, budući da ove informacije ne bi bile poznate samom pružatelju *hosting* usluga.

Resetiranje svih ovih zaporki može biti prilično opsežan posao ukoliko poslužitelj ima velik broj zasebnih *cPanel* računa i stranica, a pogotovo ako pružatelj usluga ovu odgovornost prebacuje na vlasnike poslužitelja ili stranice, koji često nisu toliko tehnički potkovani u usporedbi sa zaposlenicima pružatelja *hosting* usluge. Radi umanjenja štete uzrokovane zastojem rada poslužitelja i stranica, tj. kako bi se ovaj proces pojednostavio i ubrzao, idealno rješenje je automatizacija resetiranja zaporki pomoću skripti koje taj proces mogu izvesti brzo, efikasno, i uz minimalnu intervenciju. U poslužiteljskom okruženju korištenom u ovom radu već postoji skripta za masovno resetiranje zaporki *cPanel* računa i njihovih *email* i FTP računa - *lw-random-cpanel-pass* prikazana na slici 4.3, no nije bilo moguće isto učiniti za *web* aplikacije i njihove baze podataka.

```
[root@host ~]# lw-random-cpanel-pass --help

/usr/bin/lw-random-cpanel-pass 0.4.0-0

Randomizing cPanel Passwords:

/usr/bin/lw-random-cpanel-pass --cpuser all --pwlength 10 --- Sets all cPanel accounts to random 10 character
 password.
/usr/bin/lw-random-cpanel-pass --cpuser cPanelAccount --pwlength 10 --- Sets specified cPanel account to rand
om 10 character password.

Randomizing email passwords:

/usr/bin/lw-random-cpanel-pass --mailbox all --pwlength 10 --- Randomize all email account passwords to 10 ra
ndom character password for all cPanel accounts.
/usr/bin/lw-random-cpanel-pass --mailbox cPanelAccount --pwlength 10 --- Randomizes all email accounts under
cPanelAccount to random 10 character password.
/usr/bin/lw-random-cpanel-pass --mailbox cPanelAccount --pwlength 10 --singlemailbox user@domain.com --- Rand
omizes user@domain.com to random 10 character password.

Randomizing FTP passwords:

/usr/bin/lw-random-cpanel-pass --ftp all --pwlength 10 --- Randomize all FTP account passwords for all cPanel
 accounts to random 10 character password.
/usr/bin/lw-random-cpanel-pass --ftp cPanelAccount --pwlength 10 --- Randomizes all FTP accounts under cPanel
Account to random 10 character password.
```

Slika 4.3: Opcije skripte *lw-random-cpanel-pass*

U tu svrhu je kreirana skripta *foxdie.sh* koja automatizira ovaj proces, te skraćuje vrijeme potrebno za resetiranje zaporki i ažuriranje konfiguracijske datoteke pojedine stranice na najviše nekoliko sekundi. Točno vrijeme ovisi o broju individualnih administratorskih računa, općim performansama i dostupnosti resursa poslužitelja. *Foxdie.sh* u trenutnoj izvedbi podržava samo *WordPress* budući da je to najpopularnija CMS platforma, a i vrlo česta meta *symlink* i drugih napada. Kao i *foxhound.sh*, skripta *foxdie.sh* je pisana u *bash* skriptnom jeziku i koristi isključivo

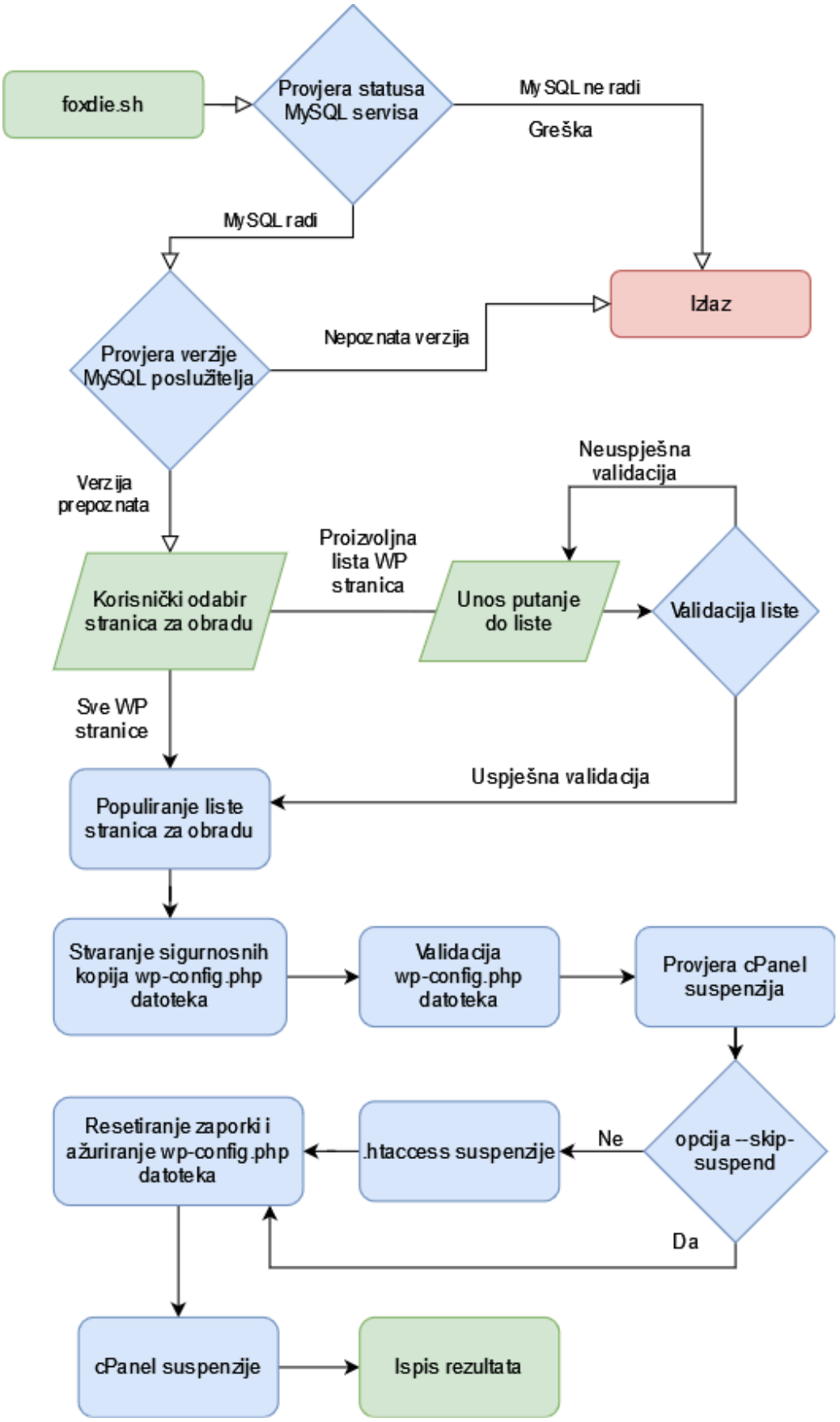
njegove ugrađene funkcije radi neovisnosti od specifičnosti pojedinih verzija cPanel softverskog paketa i *CentOS*, *Alma*, i *RHEL* operativnih sustava. Skripta *foxdie.sh* ima dva glavna načina izvršavanja: normalan tj potpun, i normalan bez suspenzije pristupa stranicama (sa *–skip-suspend* opcijom). Dijagram toka izvršavanja je dan u slici 4.4, a cjelokupni kod nalazi se u prilogu P.4.2.

Izvršavanje *foxdie.sh* započinje s deklaracijama i inicijalizacijom nužnih varijabli i kreiranjem radnog direktorija trenutne sesije, tj. trenutne instance izvršavanja skripte u */home/temp/foxdie/run.{vremenska oznaka}*. Skripta se na radni direktorij u daljnjem izvršavanju referira varijablom *$runpath*. Budući da skripta vrši izmjene u bazama podataka sustava i stranica, vrši se provjera statusa *MySQL* servisa (*mysqld*), te ako s ovim servisom postoji problem skripta se ne može izvršiti te izlazi s greškom. Ako *MySQL* servis radi, vrši se provjera verzije *MySQL* (ili *MariaDB*) poslužitelja kako bi se odredilo koju sintaksu *MySQL* upita za izmjenu zaporke korisnika baze skripta treba koristiti. Ukoliko dođe do greške u prepoznavanju verzije, skripta izlazi s greškom.

Nakon ovih inicijalnih provjera, korisniku se nude dvije opcije za odabir *WordPress* instalacija nad kojima će se skripta izvršavati - nad svima ili nad proizvoljnom listom. *WordPress* instalacija je definirana svojom *wp-config.php* datotekom. Opcija "*all*" pronalazi postojeće *wp-config.php* datoteke na temelju iščitavanja popisa svih početnih direktorija stranica na poslužitelju iz datoteke */etc/userdatadomains*. Opcija proizvoljne liste od korisnika traži da unese putanju do prethodno kreirane datoteke koja sadrži apsolutne putanje do ciljnih *wp-config.php* datoteka. Vrši se validacija korisničkog unosa prema sljedećim kriterijima:

- unos je zapravo dan
- datoteka postoji i nije prazna
- datoteka sadrži putanju barem jedne *wp-config.php* datoteke
- datoteka ne sadrži putanje do datoteka osim *wp-config.php*

U slučaju neuspješne validacije od korisnika se traži korekcija unosa. U slučaju uspješne validacije, ili selekcijom opcije "*all*", popis ciljanih *wp-config.php* datoteka se sprema u datoteku *$runpath/wp-conf.list*. Skripta zatim poziva funkciju *conf_bak* koja kreira zaštitne kopije definiranih *wp-config.php* datoteka u početnom direktoriju stranice, te u direktoriju *$runpath/wp-config-backups/*. Budući da se sve ove datoteke zovu *wp-config.php*, datotekama se dodjeljuju jedinstveni nazivi pomoću *md5sum hash* funkcije, a relacija naziva i izvorne lokacije datoteke spremaju se u *$runpath/wp-config-bak-map*.



Slika 4.4: Dijagram toka skripte *foxdie.sh*

Nakon kreiranja zaštitnih kopija, skripta započinje validaciju danih *wp-config.php* datoteka pomoću funkcije *wp_verify* koja prvo putem *MySQL* upita dohvaća popis baza podataka i korisnika *MySQL* poslužitelja. Ova funkcija zatim prolazi kroz svaku danu *wp-config.php* datoteku, dohvaćajući iz nje definirane vrijednosti varijabli DB_NAME (naziv baze), DB_USER (naziv korisnika) i DB_HOST (lokacija baze). Pomoću ovih vrijednosti funkcija determinira udovoljava li *WordPress* instalacija ovim kriterijima:

- baza podataka je na lokalnom *MySQL* poslužitelju
- naziv baze postoji u *MySQL* poslužitelju
- korisnik kojime se pristupa bazi postoji u *MySQL* poslužitelju

Ukoliko ovi kriteriji nisu zadovoljeni, skripta neće moći vršiti izmjene na bazi i njenom korisniku, te isključuje danu *WordPress* instalaciju iz daljnjih radnji. Ako su zadovoljeni, datoteka se upisuje u popis stranica za daljnju obradu *$runpath/wp-conf-valid.list.*

cPanel pruža mogućnost suspenzije tj. privremenog onemogućavanja pojedinih *cPanel* korisničkih računa i pristupu njihovim komponentama: *web* stranice, *email*, FTP, SSH, i bazi podataka. U nekim situacijama može doći do greške gdje *cPanel* nepravilno evidentira koji računi su onemogućeni, što može dovesti do daljnjih grešaka prilikom pokušaja pristupa bazi podataka. Kako bi se ove pogreške izbjegle skripta *foxdie.sh* mora osigurati da *cPanel* računi na kojima će se vršiti promjene nisu onemogućeni, odnosno u slučaju da jesu onemogućeni budu privremeno osposobljeni radi obrade. Skripta prvo sprema popis poznatih onemogućenih računa u datoteku *$runpath/suspendedaccts.* Zatim se prisilno suspendira, i odmah potom ponovno omogući *cPanel* račune na kojima su stranice navedene u *$runpath/wp-conf-valid.list.*

Kako je ranije napomenuto, *foxdie.sh* skripta ima dva glavna načina rada, bez i sa opcijom *–skip-suspend*. Ako *foxdie.sh* nije pokrenut sa ovom opcijom, izvodi se funkcija *suspend_accounts* koja blokira pristup stranicama nad kojima se skripta izvršava kreiranjem posebne *.htaccess* datoteke. Ovim putem se sprječava otkrivanje novih zaporki baze i administratorskih računa napadaču čak i u slučaju da na stranici još uvijek postoji maliciozan sadržaj. Izvorne *.htaccess* datoteke se naravno spremaju radi kasnijeg oporavka, te evidentiraju u datoteku *$runpath/suspended-htaccess.* Ova funkcija također korisniku nudi mogućnost unošenja IP adresa koje će biti izuzete od blokade pristupa, a ova funkcionalnost je osmišljena kako bi vlasniku ili administratoru poslužitelja bio omogućen pristup stranici radi izvođenja potrebnih nadogradnji, revizije korisničkih računa, čišćenju *malwarea* i slično. *Foxdie.sh* se uz dva glavna načina rada može pokrenuti i u dva pomoćna načina rada koji služe manipulaciji ranije kreiranih *.htaccess-a* zapisanih u datoteci */home/temp/foxdie/latest/suspended-htaccess*:

- –*allow-ip* - u ovom načinu rada izvršava se funkcija *update_htaccess*, koja omogućava ažuriranje *.htaccess* datoteka kreiranih zadnjim izvršavanjem *foxdie.sh* skripte, tj. dozvoljava dodavanje IP adresa izuzetih od blokade pristupa
- –unsuspend - u ovom načinu rada pokreće se funkcija *unsuspend_accts* kojom se uklanjaju *.htaccess* datoteke kreirane u posljednjoj instanci izvršenja *foxdie.sh* skripte

Pokretanjem *foxdie.sh* sa *–skip-suspend* u potpunosti zaobilazi izvođenje *suspend_accounts* funkcije, ali je bitno napomenuti da je ovaj način rada prvenstveno namijenjen za korištenje u slučajevima kada je *symlink* napad uspješno otklonjen oporavljanjem iz provjereno čistih zaštitnih kopija ili kada su već poduzete druge mjere zaštite od daljnje štete.

Završetkom ovog postupka pripreme, skripta izvršava funkciju *conf_reset* na svaku *wp-config.php* datoteku navedenu u popisu *$runpath/wp-conf-valid.list*. Ova funkcija generira novu zaporku korisnika baze podataka, vrši *MySQL* komandu koja ažurira zaporku korisnika, i upisuje novu zaporku u konfiguracijsku datoteku. Također se pomoću *curl* CLI preglednika spaja na URL *https://api.wordpress.org/secret-key/1.1/salt/* s kojeg dohvaća nove enkripcijske ključeve i tzv. soli (engl. *salt*) koji se koriste u *WordPress*-ovim funkcijama za osnaživanje enkripcije. Stari ključevi i soli u *wp-config.php* datoteci se tada zamjenjuju novima. U slučaju da je *wordpress.org* nedostupan evidentira se da ključevi i soli nisu mogli biti zamijenjeni, no ovo nije kritično pa se skripta nastavlja izvoditi.

Sljedeće se izvršava funkcija *wp_admin_pw_reset* koja iz svake dane *wp-config.php* datoteke dohvaća naziv (DB_NAME) baze podataka stranice, i tablični prefiks koji *WordPress* koristi u imenovanju tablica baze. Podfunkcijom *get_admin_users* vrši se upit *MySQL* poslužitelju kako bi se ispisali administratorski korisnički računi stranice, koji se tada upisuju u privremenu datoteku *wp-admin.temp.*

```
get_admin_users () {
     mysql $dbname --silent --raw --skip-column-names -e "SELECT u.ID, u.user_login, u.user_nicename, u.user_email FROM ${tprefix}users AS u INNER JOIN ${tprefix}usermeta AS m ON m.user_id = u.ID WHERE m.meta_key = '${tprefix}capabilities' AND m.meta_value LIKE '%administrator%' ORDER BY u.ID ASC;"
     }
```

Za svakog administratorskog korisnika se generira nova zaporka koja se uz njega sprema u *wp-admin.temp,* te se ista korisniku dodjeljuje u bazi podataka. Sadržaj privremene datoteke se

potom formatira i prebacuje u *wp-admin.info* datoteku unutar početnog direktorija stranice. Vlasnici poslužitelja ili stranice ovoj datoteci mogu pristupiti kroz *cPanel*, SSH, ili FTP, a zbog izrazito restriktivnih dopuštenja datoteke ona trećim stranama nije vidljiva čak i ako pristup stranici nije ograničen. Jedini način na koji napadači mogu pristupiti datoteci je ako i dalje imaju neki stražnji ulaz odnosno *shell* koji im omogućava pregled datotečne strukture stranice ili ponovno uspiju ostvariti pristup *cPanel* računu.

U konačnici se ponovno onemogućavaju *cPanel* računi koji su bili u suspendiranom stanju prije početka izvršenja *foxdie.sh* skripte, ažurira se *symlink /home/temp/foxdie/latest,* te izvođenje skripte završava. Primjer ispisa (engl. *output*) *foxdie.sh* skripte dan je u prilogu P.4.3.

4.5 Evaluacija performansi skripte *foxdie.sh*

Ključni zadatci koje skripta *foxdie.sh* ispunjava na svakoj ciljanoj aplikaciji, odnosno instalaciji *WordPress*-a mogu se sumirati u sljedeće točke:

1. Dohvaćanje informacija o bazi podataka
2. Izmjena zaporke za pristup bazi podataka
3. Ažuriranje zaporke u konfiguracijskoj datoteci *wp-config.php*
4. Ažuriranje kriptografskih ključeva i soli u *wp-config.php*
5. Izmjena zaporki administratorskih korisničkih računa aplikacije

Radi evaluacije korisnosti i efikasnosti programskog rješenja *foxdie.sh* , gore navedene točke su postavljene kao zadatci trojici korisnika različitih razina iskustva u radu sa *cPanel* poslužiteljima i *WordPress* CMS-om. Korisnicima je dana dokumentacija koja objašnjava pojedini korak, te je svaki od njih postupak odradio na tri različite *WordPress* instalacije kako bi se izmjerilo prosječno vrijeme (u sekundama) potrebno za obavljanje zadataka. Sve tri instalacije su podešene po zadanim postavkama i bez dodatnih modula, koriste lokalnu bazu podataka i imaju do najviše 3 administratorska korisnička računa. Rezultati mjerenja dani su u tablici 4.1.

U rezultatima prikazanim u tablici se vidi da su iskusniji korisnici konzistentniji u vremenu izvršavanja zadatka neovisno o specifičnostima pojedine aplikacije. Također se vidi da je neiskusnom korisniku trebalo značajno duže da bi obavio zadatak na prvoj *WordPress* instalacijim što je u skladu s očekivanjima, a već u drugom pokušaju smanjuje vrijeme sa preko 11 minuta na približno 8. Pretvoreni u minute, prosjeci korisnika su približno 9, 6 i pol, i 4 minute. Ukupni prosjek svih troje korisnika iznosi 388 sekundi, što je približno 6 i pol minuta po obrađenoj *web* aplikaciji. Imajući na umu da *symlink* napadi zahvaćaju veći broj *web* aplikacija

na poslužitelju, može se zaključiti da bi jednoj osobi bilo potrebno približno sat vremena rada da bi se isti postupak obavio na 10 individualnih stranica, preko 5 sati za 50 stranica itd..

#	Neiskusan korisnik	Srednje iskusan korisnik	Iskusan korisnik
1.	671 s	390 s	236 s
2.	467 s	364 s	221 s
3.	495 s	401 s	248 s
Prosjek	544 s	385 s	235 s

Tablica 4.1: Vrijeme potrebno za ručnu izmjenu zaporki i ažuriranje konfiguracijskih datoteka stranica izraženo u sekundama

Provjera vremena izvođenja *foxdie.sh* skripte vršena je pomoću programa *time* ugrađenog u *Linux* sustave, a rezultati su dani u tablici 4.2. Zbog fluktuacija u performansama i dostupnosti resursa sustava, izvršeno je po pet mjerenja na tri poslužitelja sa različitim brojem *WordPress* instalacija: 1, 5, i 10. Radi konzistentnosti i zbog toga što *foxdie.sh* skripta traži korisnički unos, skripta je pokretana sa *–skip-suspend* opcijom, te je u svakom slučaju u izborniku odabrana opcija izvršavanja skripte nad svim *WordPress* instalacijama na poslužitelju. Skripta je također uređena na način da su iz nje uklonjene *sleep* komande koje dodaju približno dvadeset sekundi vremenu izvođenja, a u normalnom izvođenju skripte služe samo radi lakše čitljivosti ispisa. Svi korišteni poslužitelji su istovjetnih konfiguracija i niskog prosječnog opterećenja.

Ispis *time* komande daje tri vremenske vrijednosti: *real* - stvarno vrijeme izvršavanja, *user* - vrijeme izvršavanja na procesoru u korisničkom načinu, i *sys* - vrijeme izvršavanja u *kernel* načinu, tj. u sistemskim pozivima. Vrijednost *real* opisuje protok stvarnog vremena, tj koliko je vremena prošlo “na satu” od početka do kraja izvršavanja programa. Zbroj vrijednosti *user* i *sys* daje ukupno vrijeme procesiranja, odnosno izvršavanja programa na procesoru. Pregledom rezultata vidljivo je da ni stvarno vrijeme izvršavanja niti ukupno vrijeme izvršavanja na procesoru nisu proporcionalni povećanju broja *WordPress* aplikacija koje se obrađuju *foxdie.sh* skriptom. Budući da postoje razlike u broju administratorskih računa na pojedinoj aplikaciji, i da trenutno opterećenje poslužitelja nije identično tijekom svake iteracije izvođenja skripte, kao i zbog relativno malog broja mjerenja, ovi rezultati nisu savršeno statistički pouzdani.

#	1 WP instalacija	5 WP instalacija	10 WP instalacija
1.	real 0m2.380s user 0m0.759s sys 0m0.196s	real 0m3.490s user 0m1.649s sys 0m0.569s	real 0m6.357s user 0m3.739s sys 0m1.121s
2.	real 0m2.300s user 0m0.758s sys 0m0.181s	real 0m4.590s user 0m1.822s sys 0m0.713s	real 0m6.643s user 0m3.819s sys 0m1.238s
3.	real 0m1.901s user 0m0.766s sys 0m0.222s	real 0m3.418s user 0m1.558s sys 0m0.528s	real 0m7.058s user 0m3.853s sys 0m1.092s
4.	real 0m2.122s user 0m0.724s sys 0m0.219s	real 0m3.304s user 0m1.595s sys 0m0.495s	real 0m7.433s user 0m3.810s sys 0m1.116s
5.	real 0m3.248s user 0m0.782s sys 0m0.183s	real 0m3.735s user 0m1.574s sys 0m0.528s	real 0m7.562s user 0m3.761s sys 0m1.088s
Prosjek	real 0m2.390s user 0m0.757s sys 0m0.200s	real 0m3.707s user 0m1.639s sys 0m0.566s	real 0m7.011s user 0m3.796s sys 0m1.131s

Tablica 4.2: Vremensko trajanje izvođenja *foxdie.sh* u odnosu na broj obrađenih *WordPress* instalacija

Proporcionalno gledano, najsporije izvršavanje tj. najgore prosječno vrijeme izvršavanja skripte je izmjereno na poslužitelju sa jednom *WordPress* aplikacijom i iznosi 2.39 sekunde. Među korisnicima, najbolje prosječno vrijeme je ostvario iskusni korisnik, koji je zadatak izvršio u 235 sekundi. Čak i pod pretpostavkom da je skripti potrebno približno 2.4 sekunde po ciljnoj aplikaciji, u prosječnom vremenu u kojem je iskusni korisnik zadatak obavio na jednoj aplikaciji, *foxdie.sh* skripta bi mogla obraditi njih približno 98. Koristeći prosječno vrijeme izvršavanja *foxdie.sh* skripte na poslužitelju sa 10 instanci *WordPress*-a (7.011 sekundi), tj. 0.701 sekunda po instanci, taj broj se penje na 335.

Naravno, ovi rezultati ovise o više faktora, prvenstveno opterećenju resursa poslužitelja na kojemu se skripta izvodi. Premda dobiveni rezultati demonstriraju važnost i korisnost automatizacije ovakvih zadataka, treba imati na umu da automatizacija nije zamjena za kompetencije administratora, već alat koji administratorima oslobađa vrijeme i dozvoljava im da se posvete onim problemima koji se automatizacijom ne mogu riješiti.

5. ZAKLJUČAK

Prvi cilj ovog rada bio je obrazložiti *Apache symlink* napade u detalje i objasniti što ih čini problematičnim čak i nakon otklanjanja inicijalnog proboja. Premda trajno adekvatna rješenja za prevenciju ovih napada nisu slobodno dostupna, programska rješenja predložena ovim radom mogu uvelike smanjiti štetu koju ovakvi napadi mogu učiniti. Prvo izrađeno programsko rješenje je skripta *foxhound.sh* koja služi za detektiranje incidenata *Apache symlink* napada na poslužiteljima pružatelja usluga *web hostinga,* a njenim bi korištenjem za to zaduženi odjeli *hosting* tvrtki imali sposobnost brze detekcije i reakcije na ove napade. Premda je detekcija implementirana na relativno jednostavan način, postignuti su ciljevi robusnosti detekcijske skripte, njenog niskog utjecaja na performanse poslužitelja, i mogućnosti jednostavne masovne instalacije skripte na velik broj poslužitelja.

Drugi cilj je bio izrada programskog rješenja koje bi administratorima poslužitelja omogućilo jednostavan i brz način resetiranja zaporki koje budu izložene u *symlink* napadima, a putem kojih napadači ostvaruju kontinuiran neovlašten pristup stranicama nakon inicijalnog napada. Slična rješenja postoje za resetiranje zaporki sistemskih korisnika i njihovih povezanih FTP i email računa, ali u periodu izrade ovog rješenja nisu pronađena rješenja koja ovo rade za pojedine *web* aplikacije. S tim ciljem izrađena je skripta *foxdie.sh* koja uspješno i konzistentno vrši izmjene zaporki baza podataka i administratorskih korisničkih računa *WordPress* stranica u svrhu prevencije neovlaštenog pristupa stranicama nakon napada. Ova skripta također uvelike smanjuje vrijeme potrebno za izvršavanje ovih promjena, budući da ih ona obavi za nekoliko sekundi, dok čak i iskusnijem korisniku ili administratoru treba u prosjeku 6 i pol minuta po zahvaćenoj stranici. U jednom konkretnom testu na poslužitelju sa otprilike 170 pojedinačnih cPanel računa i približno 150 WordPress instalacija zahvaćenih *Apache symlink* napadom, cjelokupna skripta se izvršila u manje od dvije minute.

Dana programska rješenja su u trenutnoj izvedbi ograničena na poslužitelje koji koriste *Linux* distribucije bazirane na *Red Hat Enterprise Linux* distribuciji (*CentOS*, *Alma*), *cPanel* upravljačku ploču, i na *WordPress* CMS. Ovo je bio svjestan i namjeran izbor temeljen na raširenosti poslužitelja koji odgovaraju zadanim kriterijima. Odluci su također doprinjele opširne razlike između ovih sustava i sustava drugih konfiguracija, koje bi učinile potrebne provjere i funkcionalnosti značajno kompleksnijim. Unatoč tome, opisana rješenja se se mogu smatrati dokazom koncepta i daljnjim razvojem usavršavati kako bi se poboljšali kriteriji detekcije, i omogućilo resetiranje zaporki na sustavima i stranicama drukčijih konfiguracija i CMS-ova.

LITERATURA

[1] R. Ljuban, Vrste kibernetičkih napada na poduzeća i njihove mjere obrane, Diplomski rad, Sveučilište u Zagrebu, Filozofski fakultet (Odsjek za informacijske i komunikacijske znanosti), RH, 2021

[2] K. Scarfone, P. Mell, Guide to Intrusion Detection and Prevention Systems (IDPS) - Recommendations of the National Institute of Standards and Technology, Special Publication 800-94. NIST, USA, 2007

[3] C. Anjos, How to Find & Clean Up the AnonymousFox Hack, 2022, https://blog.sucuri.net/2022/06/how-to-find-clean-up-the-anonymousfox-hack.html, pristupljeno 14.9.2024.

[4] E. Nemeth, G. Snyder, T.R. Hein, B. Whaley, D. Mackin, Unix and Linux System Administration Handbook, Addison-Wesley, Boston, USA, 2017.

[5] http://www.halfdog.net/Security/2010/FilesystemRecursionAndSymlinks/#Introduction, pristupljeno 19.6.2024.

[6] https://capec.mitre.org/data/definitions/27.html, pristupljeno 19.6.2024.

[7] https://github.com/beruangsalju/shell-backdoor/blob/main/README.md, pristupljeno 8.2.2024.

[8] https://w3techs.com/technologies/overview/content_management, pristupljeno 8.2.2024.

[9] https://arstechnica.com/information-technology/2022/01/supply-chain-attack-used-legitimate-wordpress-add-ons-to-backdoor-sites/, pristupljeno 20.6.2024.

[10] https://gitlab.com/kalilinux/packages/wpscan, pristupljeno 22.6.2024.

[11] https://www.geeksforgeeks.org/shellfinder-simple-tool-to-find-shells-and-endpoints-in-websites/, pristupljeno 22.6.2024.

[12] https://www.whmcs.com/, pristupljeno 25.6.2024.

[13] https://docs.cpanel.net/whm/clusters/remote-access-key/, pristupljeno 28.8.2024.

[14] https://docs.cpanel.net/ea4/apache/symlink-race-condition-protection/, pristupljeno 28.8.2024.

[15] https://httpd.apache.org/docs/2.4/mod/core.html#allowoverride, pristupljeno 28.8.2024.

[16] https://owasp.org/www-community/Vulnerability_Scanning_Tools, pristupljeno 1.9.2024.

SAŽETAK

U ovom radu dan je pregled najčešćih sigurnosnih problema u *web hosting* okruženjima, s fokusom na raširenu vrstu napada koji iskorištava ranjivosti u načinu na koji *Apache web* poslužitelj radi sa *symlinkovima*. Detaljno su obrazloženi: utjecaj ovakvih napada na poslužitelje koji koriste upravljačke ploče poput *cPanela*, ograničenja u prevenciji ovih napada, važnost brzine njihovog otkrivanja, i postupak mitigacije proizašle štete. Skripta *foxhound.sh* je izrađena s ciljem pravodobne detekcije ovih incidenata i jednostavne masovne implementacije na floti poslužitelja. Ova skripta traži *symlinkove* unutar početnih direktorija stranica na poslužitelju, vrši jednostavnu analizu rezultata i obavještava administratora ukoliko se incident dogodio. Također je izrađena skripta *foxdie.sh*, namijenjena automatizaciji procesa resetiranja zaporki za pristup bazama podataka kompromitiranih *web* aplikacija, zaporki njihovih administratorskih računa, te ažuriranju relevantnih konfiguracijskih datoteka. Korištenjem ove skripte uvelike se ubrzava postupak prevencije daljnjeg neovlaštenog pristupa kompromitiranim aplikacijama, te shodno tome i smanjuje šteta vlasnicima poslužitelja i stranica.

Ključne riječi: *Apache web* poslužitelj, *symlink* napad, *web hosting*, *bash* skriptiranje

DETECTION AND MITIGATION OF APACHE SYMLINK ATTACKS

Abstract

This paper provides an overview of the most common security issues present in web hosting environments, with a focus on a widespread attack that exploits the way that the *Apache* web server handles symlinks. It provides a detailed explanation of how these attacks affect servers that utilize control panel software such as *cPanel*, the limitations of symlink attack prevention, the importance of quick detection of such attacks, as well as the process of mitigating the resulting damage. The *foxhound.sh* bash script was created with the goals of timely detection of such attacks and easy deployment on a fleet of servers. It searches for symlinks within the document roots of the websites present on the server, performs a simple analysis of the results, and notifies the administrator if an incident is found to have occurred. The *foxdie.sh* script was created to automate the process of resetting the database user and CMS administrator passwords of compromised sites, as well as updating the pertinent configuration files. Using this script greatly speeds up the process of preventing further unauthorized access to compromised sites, and thus also reduces the damage done to the owners of the server and sites.

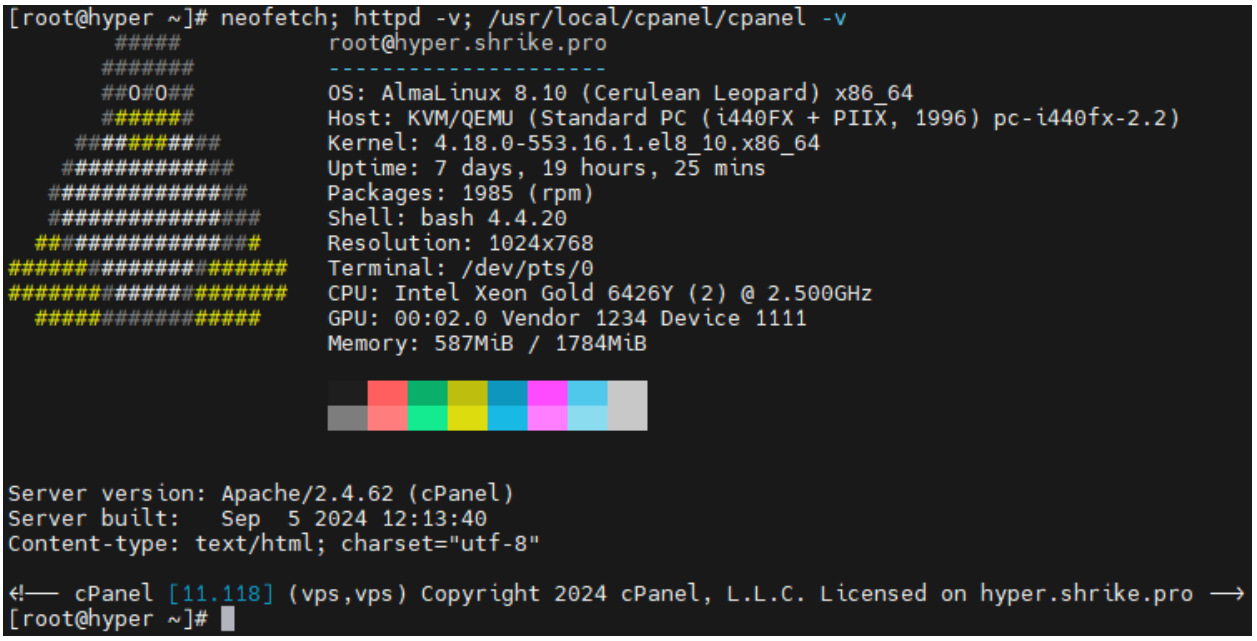
Key words: Apache web server, symlink attack, web hosting, bash scripting

ŽIVOTOPIS

Josip Molnar rođen je 6.2.1992. u Zagrebu. Srednjoškolsko obrazovanje je završio u Prirodoslovnoj Gimnaziji i Tehničkoj Školi Ruđer Bošković u Osijeku s odličnim uspjehom. Ima 6 godina radnog iskustva u web hosting industriji na pozicijama Linux System Administratora i Security Operations Center Engineera, a odnedavno radi i na administraciji Windows sustava i održavanju mrežne infrastrukture.

PRILOZI

Prilog P.3.1: Specifikacije korištenog poslužitelja



Prilog P.4.1: Programski kod skripte *foxhound.sh*

```
#!/bin/bash
# FOXHOUND - SYMLINK HACK DETECTION
# Global variable declarations and init
        # hostname - server hostname
        hostname=`hostname`
        # ips - server IP addresses
        ips=`hostname -I`
        # msg - message string for email alerts
        msg="$hostname\n$ips\n"
        # now - current datetime
        now=`date +%F_%R`
        # runpath - current run data dir
        runpath="/home/temp/foxhound/run.$now"
        # mailto - where to send email reports
        mailto="mailaddress"
        # count_ac - count of current symlinks
        count_ac=""
        # count_wc - count of working symlinks
        count_wc=""
        # count_ap - count of all symlinks in previous run
        count_ap=""
        # detection - bool, change to true if possible symlink hack is detected and alert needs
to be sent
        sendmail=FALSE

main () {
        # Create the results directory
        mkdir -p $runpath
        # check if a previous foxhound run may still be in progress
        if [ -e /home/temp/foxhound/lock ]
                then
        sendmail=TRUE
                msg+="Previous foxhound run possibly still in progress or did not complete
correctly. Please review the /home/temp/foxhound directory and check for previosly started
foxhound processes."
        mail
        else
        # Create lock file
                touch /home/temp/foxhound/lock
        # Perform symlink scan
        scan
        # Load previous results dir path if exists
        if [ -e /home/temp/foxhound/latest ]
                then
                previous="/home/temp/foxhound/latest"
                # If previous scan results already exist, run the compare function to check if
the number of symlinks has increased
                compare
        else
                # Otherwise, check if the number of symlinks greatly exceeds an estimated
"normal" number of symlinks to check for symlink hacks that predate the foxhound install.
Triggers only once.
                if (( $count_ac > (( $count_accts * 10 )) ))
                then
                        sendmail=TRUE
                        msg+="Possible symlink attack detected. Scan detected an unusual
number of symlinks relative to the number of cPanel accounts!\n   Total symlink count:
$count_ac \n   Possible working symlinks: $count_wc \n     Number of cPanel accounts:
$count_accts"
                fi
        fi
```

```
        # Unlink latest
                unlink $previous 2>/dev/null;
        # Create new link to current run as latest
                ln -s $runpath /home/temp/foxhound/latest
        # Remove lock file
        rm -f /home/temp/foxhound/lock
        fi
        if [ $sendmail = TRUE ]
                then echo -e $msg | mail -s "$hostname - foxhound alert" $mailto
        fi
}

scan () {
        # Search for symlinks in site docroots
        for docroot in `cut -d= -f9 /etc/userdatadomains`; do
        find $docroot -type l;
        done > $runpath/all-sym.txt
        # Test for working symlinks, print to separate file
        for symlink in `cat $runpath/all-sym.txt`; do
        if [ -e $symlink ]; then
                echo $symlink >> $runpath/working-sym.txt;
        fi;
        done;
        count_ac=`wc -l < $runpath/all-sym.txt`
        count_wc=`wc -l < $runpath/working-sym.txt`
        count_accts=`wc -l < /etc/trueuserdomains`
}

compare () {
        count_ap=`wc -l < $previous/all-sym.txt`
        if (( $count_ac > (( $count_ap + $count_accts )) ))
        then
        sendmail=TRUE
        msg+="Possible symlink attack detected. Number of symlinks increased significantly
since previous scan!\n       Current symlink count: $count_ac \n  Previous symlink count:
$count_ap \n   Possible working symlinks: $count_wc \n     Number of cPanel accounts:
$count_accts"
        fi
}

main;
```

Prilog P.4.2: Programski kod skripte *foxdie.sh*

```
#!/bin/bash
# Global variable declarations
        # now - current datetime
        now=$(date +%F_%R)
        # mysql_state - used for the return value of the mysql_up_check function
        mysql_state=""
        # LIMIT_IP - yes/no value, used in the scripts default behavior of suspending sites,
related function checks if an end user (ie server admin) provided an IP that should be allowed
to bypass site suspension redirects
        LIMIT_IP=""
        # ALLOW_IP - IPs allowed to bypass site suspension redirect
        ALLOW_IP=""
        # allow_ip_valid - used in function to ask the user to verify they listed the ALLOW_IP
IP addresses per the provided instructions - essentially an "ARE YOU SURE?"
        allow_ip_valid=""
        # skip_suspend - this variable is used in a check that allows the script to bypass the
default site suspend behavior
        skip_suspend="FALSE"

# Main function
main () {
        clear
        printf "\n
\t
\t
\t
\t
\t
\t
\t#### Automated WP symlink hack cleanup ####
\t##Script by https://github.com/j-molnar/ ##\n\n"
        # Define and create the session directory
        runpath="/home/temp/foxdie/run.$now"
        mkdir -p $runpath
        print_style "FoxDie working directory is $runpath\n\n" "info"
# Check if mysql/mariadb is running
        sleep 1
        print_style "Checking if MySQL server is up... " "info"
        mysql_state=$(mysql_up_check)
        if [ $mysql_state == "active" ]
        then
        print_style "Success - MySQL server is running!\n\n" "success"
        elif [ $mysql_state == "inactive" ]
        then
        print_style "Fail - MySQL server is not running! Exiting...\n\n" "danger"
        exit
        else
        print_style "Fail - unknown MySQL server error! Exiting...\n\n" "danger"
        exit
        fi
        sleep 1
        # Get mysql version
        mysqld -V > $runpath/mysqlversion
        print_style "MySQL server version:\n" "info"
        cat $runpath/mysqlversion
        sleep 0.5
        print_style "\nTesting MySQL server compatibility... " "info"
        sleep 1
        mysql_status=$(mysql_version_check)
        if [ $mysql_status == "recent" ]; then print_style "Compatible - MySQL version is
recent.\n\n" "success";
        elif [ $mysql_status == "old" ]; then print_style "Compatible - MySQL version is
```

```
old.\n\n" "warning";
        elif [ $mysql_status == "ERROR" ]; then print_style "MySQL version not
recognized!\n\n" "danger";exit;
        else print_style "Unrecognized MySQL error!\n\n" "danger";exit;
        fi
        sleep 1
        # This is a menu to select which sites to process with the script. Selection works by
populating the $runpath/wp-conf.list file. Options are to work on either ALL wp sites, or only
AFFECTED sites. List of affected accounts can be provided manually or from foxhound.
        # Selecting ALL will not actually select ALL WP installations - only those whose
docroots are defined in /etc/userdatadomains
        SELECTION=""
        print_style "Please select which sites to process:\n" "info"
        print_style "[Select 1 or 2]\n" "warning"
        printf "\t1) All WordPress sites\n\t2) Custom list\n"
        printf "Selection > "
        while [[ $SELECTION != @(1|2) ]]; do
        read SELECTION
        case $SELECTION in
                1) listwp_all > $runpath/wp-conf.list;;
                2) print_style "\nPlease provide the PATH to a file containing ONLY a full
list of ABSOLUTE PATHS to the target WP installations' wp-config.php files:\n" "warning"
                printf "> "
                custom_status=""
                while [[ $custom_status != "OK" ]]; do
                read CUSTOM
                # verify that $CUSTOM  matches these criteria:
                        # input was actually provided
                        # input file exists and is not empty
                        # input file contains at least one path to a wp-config.php
                        # input file contains NO lines other than paths to wp-config.php files
                if [[ $CUSTOM != "" ]] && [ -s $CUSTOM ] && (( `grep "wp-config.php" $CUSTOM
| wc -l` > 0)) &&  (( `grep -v "wp-config.php" $CUSTOM | wc -l` == 0 ))
                        then
                        print_style "\nValid input file. Processing.\n" "success";
                        cp $CUSTOM $runpath/wp-conf.list
                        custom_status="OK"
                        else
                        print_style "\nInvalid or missing input file, please try again:\n"
"danger"
                fi
                done;;
                *) print_style "Please choose 1 or 2\nSelection: " "danger";;
        esac
        done
        sleep 0.5
        print_style "\nSelected option $SELECTION - proceeding.\n" "info"
        sleep 1
        print_style "\nFound `wc -l < $runpath/wp-conf.list` wp-config files, listed in
$runpath/wp-conf.list \n\n" "info"
        sleep 1
        # Create backups of the target wp-config.php files.
        conf_bak
        sleep 1
        # verify wordpress databases are local and the db names and db users are correctly
defined in wp-config files
        wp_verify
        sleep 1
        # Grabbing known suspended accounts. Updated to filter out accounts that aren't on the
target list.
        touch $runpath/suspendedaccts
        for acct in `ls /var/cpanel/suspended/ | xargs basename > /dev/null 2>&1`; do
        if ( grep -q "/hom.*/$acct/*" $runpath/wp-conf-valid.list )
                then echo $acct >> $runpath/suspendedaccts
        fi
        done
```

```
        # Force suspend and unsuspend all target accounts
        sleep 1
        print_style "\nPerforming forced suspend and unsuspend of all target accounts to
avoid database errors... " "info"
        for acct in `cut -d/ -f3 $runpath/wp-conf-valid.list | sort -u`; do
        /scripts/suspendacct $acct --force > /dev/null 2>&1;
        /scripts/unsuspendacct $acct > /dev/null 2>&1;
        done
        sleep 1
        print_style "Done!\n" "success"
        sleep 1
        # Disable access to the sites to prevent the new credentials from being exposed.
        if [ $skip_suspend == "FALSE" ]
        then suspend_accts
        fi
        # Run reset function for each target site
        print_style "\nPerforming database password and wp-config.php file resets... " "info"
        for target in `cat $runpath/wp-conf-valid.list`; do
        conf_reset $target;
        done
        sleep 1
        print_style "Done!\n" "success"
        sleep 0.5
        mysql -e "FLUSH PRIVILEGES;"
        sleep 1
        print_style "\nResetting WordPress admin user passwords... " "info"
        for target in `cat $runpath/wp-conf-valid.list`; do
        wp_admin_pw_reset $target
        done
        print_style "Done!\n" "success"
        sleep 2
        # re-suspend known suspended accounts
        print_style "\nSuspending previously suspended accounts:\n" "info"
        for acct in `cat $runpath/suspendedaccts`; do
        /scripts/suspendacct $acct > /dev/null 2>&1
        print_style "$acct\n" "danger"
        done
        sleep 2
        print_style "\nSuccessfully updated WP database user password, WP admin user
passwords and reset salts and keys on:\n" "success"
        for site in `cat $runpath/api-call-success.list`
        do grep $site $runpath/site-file.map
        done
        sleep 0.5
        print_style "\nSuccessfully updated WP database user password and WP admin user
passwords, but failed to reset salts and keys on:\n" "warning"
        for site in `cat $runpath/api-call-fail.list`
        do grep $site $runpath/site-file.map
        done
        sleep 0.5
        print_style "\nEncountered issues with the following wp-config.php files and their
sites, they will need to be checked manually:\n" "danger"
        cat $runpath/wp-conf-invalid.list
        sleep 1
        print_style "\nWordPress Admin user passwords can be found in each site's docroot, in
the wp-admin.info file.\n" "info"
        print_style "\nFoxDie run completed!\n\n" "success"
        # Create "latest" symlink to the latest COMPLETE foxdie run. Will not update if
execution is interrupted.
        unlink /home/temp/foxdie/latest > /dev/null 2>&1
        ln -vs `ls -tc /home/temp/foxdie/ | head -n1` /home/temp/foxdie/latest > /dev/null
2>&1
}

############### Auxiliary functions defined below here ###############
```

```
listwp_all () {
        # Grabs paths to all wp-config.php files
        for docroot in `cut -d= -f9 /etc/userdatadomains | sort -u`; do ls
$docroot/wp-config.php; done 2>/dev/null
}

mysql_up_check () {
        mysql_up=""
        if ( grep -qi "centos.*7.*" /etc/redhat-release ) || ( grep -qi "alma"
/etc/redhat-release )
        then mysql_up=`systemctl is-active mysqld`
        elif ( grep -qi "centos.*6.*" /etc/centos-release )
        then
        if ( service mysql status | egrep -qi "OK|running" ) || ( service mysqld status |
egrep -qi "OK|running")
                then mysql_up="active"
                else mysql_up="inactive"
        fi
        else mysql_up="ERROR"
        fi
        echo $mysql_up
}

mysql_version_check () {
        # hacky mysql version check - needed for mysql user password reset query formatting
        local mysql_type=""
        local mysql_v=""
        local v_num=()
        local mysql_status=""
        # Evaluating MySQL server version and type.
        if ( grep -q -i "mariadb" $runpath/mysqlversion );
        then mysql_v="`awk '{print $3}' $runpath/mysqlversion | cut -d- -f1`";
mysql_type="mariadb";
        else mysql_v="`awk '{print $3}' $runpath/mysqlversion`"; mysql_type="mysql";
        fi
        # parse version numbers for comparison
        for n in {1..3}; do
        v_num=("${v_num[@]}" "`printf $mysql_v | cut -d. -f$n`");
        done
        # compare version to requirements
        if [ $mysql_type == "mysql" ]
        then
                if ((${v_num[0]} >= 5)) && ((${v_num[1]} >= 8))
                then mysql_status="recent"
                else mysql_status="old"
                fi
        elif [ $mysql_type == "mariadb" ]
        then
                if (( ${v_num[0]} >= 10 )) && (( ${v_num[1]} >= 2 ))
                then mysql_status="recent"
                else mysql_status="old"
                fi
        else mysql_status="ERROR"
        fi
        # return "recent" "old" or "ERROR"
        echo $mysql_status
}

wp_verify () {
        # function to verify existing database and db user for each wp-config file. Important
due to to possibility of missing databases or incorrectly configured sites which will need
manual intervention.
        print_style "\nVerifying valid database hosts, names, and users...\n" "info"
        sleep 1
        # list all existing mysql users and databases
        mysql -e "SELECT user FROM mysql.user;" > $runpath/dbusers.list
```

```
        mysql -e "SHOW DATABASES;" > $runpath/dbs.list
        #define valid DB_HOST values aside from loopback range
        for i in `hostname -I`; do echo $i >> $runpath/validhosts; done
        hostname >> $runpath/validhosts
        echo "localhost" >> $runpath/validhosts
        # compare db_host, db_name and db_user entries from wp-config.php files with the
previously defined lists
        for file in `cat $runpath/wp-conf.list`; do
        if  [ -s $file ]
                then
                dbname=`grep DB_NAME $file | cut -d "'" -f4`
                dbuser=`grep DB_USER $file | cut -d "'" -f4`
                dbhost=`grep DB_HOST $file | cut -d "'" -f4`
                sitepath=`dirname $file`
                site=`grep $sitepath /etc/userdatadomains|grep -vi "parked" | cut -d: -f1`
                printf "$site\t$file\n" | column -t >> $runpath/site-file.map
                touch $runpath/wp-conf-invalid.list
                if [[ $dbname != "" ]] && [[ $dbuser != "" ]] && [[ $dbhost != "" ]]
                then
                        if [[ $dbhost == "localhost" ]] || [[ $dbhost == "localhost:3306" ]]
|| ( echo $dbhost | egrep -q "^127\." ) || ( grep -qi $dbhost $runpath/validhosts )
                        then
                        if ( grep -q "$dbname" $runpath/dbs.list ) && ( grep -q "$dbuser"
$runpath/dbusers.list )
                                then
                                print_style "Valid username and db found for $file\n" "success"
                                echo $file >> $runpath/wp-conf-valid.list
                                else
                                print_style "Invalid DB_NAME or DB_USER values found for
$file\n" "danger"
                                echo $file >> $runpath/wp-conf-invalid.list
                        fi
                        else
                        print_style "Invalid database host value in $file - doesn't seem to be
local\n" "danger"
                        echo $file >> $runpath/remote-db.list
                        fi
                else
                print_style "Invalid DB_NAME, DB_USER, or DB_HOST values provided in $file\n"
"danger"
                echo $file >> $runpath/wp-conf-invalid.list
                fi
                else
                print_style "Non-existing or invalid file $file provided in the list!\n"
"danger"
                echo $file >> $runpath/wp-conf-invalid.list
        fi
        done
}

conf_bak () {
        # creates backups of the wp-config.php files in site docroot
        print_style "Backing up existing wp-config.php files...\n" "info"
        sleep 1
        mkdir $runpath/wp-config-backups
        touch $runpath/wp-config-bak-map
        for file in `cat $runpath/wp-conf.list`; do
        cp -va $file{,.bak_$now} 2>/dev/null;
        # adding "remote" backup functionality for safe keeping. MD5 hash of each wp-config.php
file should be a unique enough filename. Hashed filename is mapped to the original file
location in $runpath/wp-config-bak-map
        filename="`md5sum $file | cut -d " " -f1 2>/dev/null`"
        cp -a $file $runpath/wp-config-backups/$filename
        printf "$filename\t$file\n" >> $runpath/wp-config-bak-map
        done
        print_style "\nConfig files backed up locally, and to
```

```
$runpath/wp-config-backups/\nBackups in the wp-config-backups dir use different filenames.
Original file to backup relations are listed in $runpath/wp-config-bak-map\n" "info"
}

conf_reset () {
        # Declare local variables
        local tfile=$1
        local apicall=()
        local pwgen=`tr -dc 'A-Za-z0-9!#$%&()*+,-./:;<=>?@[\]^_{|}~' </dev/urandom | head -c
14`
        local pwstring="define( 'DB_PASSWORD', '$pwgen' );"
        local dbuser="`grep DB_USER $tfile | cut -d "'" -f4`"
        local dbname="`grep DB_NAME $tfile| cut -d "'" -f4`"
        touch $runpath/api-call-success.list
        touch $runpath/api-call-fail.list
        # get new salts and keys, dump into file for processing
        if [ `curl --connect-timeout 5 --retry 5 --retry-delay 5 -o $runpath/apicalltemp -s
-w '%{http_code}' 'https://api.wordpress.org/secret-key/1.1/salt/'|grep 200` ];
        then
        # grab salts and keys from file, store into array to prepare for replacement.
        for i in {1..8}; do
                apicall=("${apicall[@]}" "`sed -n "$i"p $runpath/apicalltemp`");
        done;
        sed -i "/'AUTH_KEY/c ${apicall[0]}" $tfile
        sed -i "/'SECURE_AUTH_KEY/c ${apicall[1]}" $tfile
        sed -i "/'LOGGED_IN_KEY/c ${apicall[2]}" $tfile
        sed -i "/'NONCE_KEY/c ${apicall[3]}" $tfile
        sed -i "/'AUTH_SALT/c ${apicall[4]}" $tfile
        sed -i "/'SECURE_AUTH_SALT/c ${apicall[5]}" $tfile
        sed -i "/'LOGGED_IN_SALT/c ${apicall[6]}" $tfile
        sed -i "/'NONCE_SALT/c ${apicall[7]}" $tfile
        echo $tfile >> $runpath/api-call-success.list
        else
        print_style "\nFailed to reach api.wordpress.org\n" "warning"
        echo $tfile >> $runpath/api-call-fail.list
        fi
        sed -i "/'DB_PASSWORD'/c $pwstring" $tfile
        if [ $mysql_status == "recent" ]
        then mysql -e "ALTER USER '$dbuser'@'localhost' IDENTIFIED BY '$pwgen';"
        elif [ $mysql_status == "old" ]
        then mysql -e "SET PASSWORD FOR '$dbuser'@'localhost' = PASSWORD('$pwgen');"
        fi
}

wp_admin_pw_reset () {
        # local variable definitions
        local tfile=$1
        local dbname=`grep DB_NAME $tfile | cut -d "'" -f4`
        local dbuser=`grep DB_USER $tfile | cut -d "'" -f4`
        local dbhost=`grep DB_HOST $tfile | cut -d "'" -f4`
        local tprefix=`grep '$table_prefix' $tfile | cut -d "'" -f2`
        local sitepath=`dirname $tfile`
        local acct=`echo $tfile | cut -d/ -f3`
        local admin_ids=()
        get_admin_users () {
        mysql $dbname --silent --raw --skip-column-names -e "SELECT u.ID, u.user_login,
u.user_nicename, u.user_email FROM ${tprefix}users AS u INNER JOIN ${tprefix}usermeta AS m
ON m.user_id = u.ID WHERE m.meta_key = '${tprefix}capabilities' AND m.meta_value LIKE
'%administrator%' ORDER BY u.ID ASC;"
        }
        # reset individual admin user password
        reset_password () {
        mysql $dbname --silent -e "UPDATE ${tprefix}users AS u SET
u.user_pass=MD5('${pwgen}') WHERE u.ID=${id};"
        }
        admin_ids+=(`get_admin_users | tee $sitepath/wp-admin.temp | awk '{print $1}'`)
```

```
        for id in "${admin_ids[@]}"; do
                pwgen=`tr -dc 'A-Za-z0-9!#%()*+,-.:=?@[]^_|~' </dev/urandom | head -c 18`
                sed -i "s/^$id.*$/& Password: $pwgen/g" $sitepath/wp-admin.temp
                reset_password
        done
        cat $sitepath/wp-admin.temp | column -t > $sitepath/wp-admin.info
        rm -f $sitepath/wp-admin.temp
        chmod 400 $sitepath/wp-admin.info
        chown $acct:$acct $sitepath/wp-admin.info
}

suspend_accts () {
        print_style "\nDisabling access to the sites to prevent exposing the new credentials.
Did the customer provide any IPs (IPv4) which should be allowed access to the sites?
[yes/no]\n" "info"
        printf "> "
        while [[ $LIMIT_IP != @(yes|no) ]]; do
        read LIMIT_IP
        case $LIMIT_IP in
                yes)
                allow_ip
                ;;
                no)
                print_style "No IP(s) provided by the customer. Setting up default suspend
.htaccess file.\n" "info"
                printf "### .htaccess generated via foxdie ###\nRedirectMatch .*
/cgi-sys/suspendedpage.cgi" > $runpath/.htaccess
                ;;
                *)
                print_style "\nPlease choose yes or no:\n" "danger"
                ;;
        esac
        done
        for docroot in `cat $runpath/wp-conf-valid.list | xargs dirname`; do
        if [ -e $docroot/.htaccess ]; then
                chattr -i $docroot/.htaccess
                mv $docroot/.htaccess{,.bak.$now}
                cp $runpath/.htaccess $docroot
                echo $docroot/.htaccess.bak.$now >> $runpath/suspended-htaccess
                chattr +i $docroot/.htaccess
        fi
        done
}

allow_ip () {
        while [[ $allow_ip_valid != "ok" ]]; do
        print_style "\nPlease input the IP address(es). If there are multiple, please list
them in a single line, each separated with a single space. Please make sure there are no
trailing spaces. " "warning"
        print_style "WARNING - invalid input here WILL cause all target sites to go down with
errors.\n" "danger"
        printf "> "
        read ALLOW_IP
        if [[ $ALLOW_IP != "" ]]
                then
                print_style "\nPlease verify the IP list before proceeding. Are the IPs
correct and formatted as instructed? [yes/no]\n" "warning"
                printf "> "
                read CONFIRM
                if [[ $CONFIRM == "yes" ]]
                then allow_ip_valid="ok"
                print_style "\nSetting up .htaccess files to block access from all IPs
except: $ALLOW_IP\n" "info"
                printf "### .htaccess generated via foxdie ###\n<IfModule
mod_authz_core.c>\n\t<RequireAll>\n\t\t Require ip
%s\n\t</RequireAll>\n</IfModule>\n\n<IfModule !mod_authz_core.c>\n\torder deny,allow\n\tdeny
```

```
from all\n\tallow from %s\n</IfModule>" "$ALLOW_IP" "$ALLOW_IP" > $runpath/.htaccess
                else print_style "Input IPs correctly:\n" "danger"
                fi
                else print_style "Empty input!\n" "danger"
        fi
        done
}

update_htaccess () {
        if [ -s /home/temp/foxdie/latest ];
        then
        runpath="/home/temp/foxdie/latest"
        print_style "\nReplacing .htaccess files generated by the latest foxdie sesion.\n"
"info"
        allow_ip
        for htaccess in `cat $runpath/suspended-htaccess`; do
                path=`dirname $htaccess`
                chattr -i $path/.htaccess
                rm -f $path/.htaccess
                cp $runpath/.htaccess $path
                chattr +i $path/.htaccess
        done
        else print_style "ERROR - couldn't find latest foxdie session!" "danger";exit
        fi
}

unsuspend_accts () {
        if [ -s /home/temp/foxdie/latest ];
        then
                print_style "\nRemoving all .htaccess suspends generated by the latest foxdie
session, replacing with the originals.\n" "info"
                for htaccess in `cat /home/temp/foxdie/latest/suspended-htaccess`; do
                path=`dirname $htaccess`
                chattr -i $path/.htaccess
                mv -f "$htaccess" "$path/.htaccess"
                done
        else print_style "ERROR - couldn't find latest foxdie session!" "danger"; exist
        fi
}

print_style () {
# many thanks to stack overflow for this one
        if [ "$2" == "info" ] ; then
        COLOR="96m";
        elif [ "$2" == "success" ] ; then
        COLOR="92m";
        elif [ "$2" == "warning" ] ; then
        COLOR="93m";
        elif [ "$2" == "danger" ] ; then
        COLOR="91m";
        else #default color
        COLOR="0m";
        fi
        STARTCOLOR="\e[$COLOR";
        ENDCOLOR="\e[0m";
        printf "$STARTCOLOR%b$ENDCOLOR" "$1";
}

## Setting up additional functions to perform via running foxdie.sh with --argument
case "$1" in
        # replace .htaccess files generated in the latest foxdie session with new ones. Can be
used to replace generic suspend .htaccess files with ones that allow access from particular
IPs, or to update the allowed IP list.
        --allow-ip)
        update_htaccess; exit
        ;;
```

```
        #Invoke the unsuspend_accts function. Works for both generic suspend .htaccess and
allowed IP .htaccess, scope is limited to the .htaccess files generated by the latest full
foxdie.sh run.
        --unsuspend)
        unsuspend_accts
        ;;

        # the "skip-suspend" option performs a full foxdie run but skips creating the suspend
.htaccess files. Intended to ONLY be used after sites have been restored from clean backups, so
there's no risk of exposing the new wp database and admin user credentials via symlink hacks or
backdoors.
        --skip-suspend)
        skip_suspend="TRUE"
        main
        ;;

        # no argument, run main normally
        "")
        main
        ;;

        # help menu
        --help)
        printf "\nAvailable options:\n\tNO ARGUMENT - full foxdie.sh run\n\t--skip-suspend -
ONLY use immediately after accounts are restored from clean backups. Performs full run
without suspending the sites.\n\t--allow-ip - replace standard .htaccess suspends with
.htaccess files that allow for access from specific IP addresses, or update the allowed
IPs\n\t--unsuspend - revert all suspensions performed through the most recent foxdie
run\n\n";exit
        ;;

        *)
        print_style "\nInvalid option $1\n" "danger"
        printf "\nAvailable options:\n\tNO ARGUMENT - full foxdie.sh run\n\t--skip-suspend -
ONLY use immediately after accounts are restored from clean backups. Performs full run
without suspending the sites.\n\t--allow-ip - replace standard .htaccess suspends with
.htaccess files that allow for access from specific IP addresses, or update the allowed
IPs\n\t--unsuspend - revert all suspensions performed through the most recent foxdie
run\n\n";exit
        ;;
esac
```

Prilog 4.3: Ispis pri izvršavanju *foxdie.sh*