↺ Sadržaj

SAVREMENE VISOKOTEHNOLOŠKE PRETNJE: O RANJIVOSTI SOFTVERSKIH PROIZVODA I PRETNJAMA

MODERN CYBER SECURITY THREATS: ON SOFTWARE VULNERABILITIES AND THREATS

Miloš Jovanović[1], Nemanja Maček[2], Igor Franc[3], Dragan Mitić[4]

[1] OpenLink Grupa, e-mail: mjovanovic@openlink.rs

[2] SECIT Security i Visoka škola elektrotehnike i računarstva strukovnih studija u Beogradu; e-mail: nmacek@secitsecurity.com

[3] SECIT Security i Fakultet informacionih tehnologija Univerziteta Metropolitan u Beogradu, e-mail: ifranc@secitsecurity.com

[4] OpenLink Grupa, e-mail: dmitic@openlink.rs

***Apstrakt**: U ovom radu izvršena je analiza savremenih visokotehnoloških pretnji koje nastaju kao posledica sigurnosnih propusta u softveru i ranjivosti softverskih proizvoda. Ranjivosti u softverskim proizvodima često nastaju kao posledica primene metodologije brzog razvoja i predstavljaju pretnje koje napadači sa odgovarajućim znanjem i računarskim resursima mogu da iskoriste kako bi stekli neovlašćeni pristup računarskim sistemima i mrežama, a samim tim i poverljivim podacima koji se na njima nalaze. Shodno tome, u radu su date neke preporuke koje se odnose na ublažavanje posledica koje mogu nastati kao i preporuke za smanjenje broja potencijalnih ranjivost tokom razvoja softvera. U radu su takođe analizirani načini za objavljivanje informacija o otkrivenim ranjivostima i skrenuta je pažnja na neke pravne aspekte koje treba uzeti u obzir prilikom objavljivanja.*

***Ključne reči**: softver, ranjivost, pretnja, ublažavanje posledica, objavljivanje informacija*

***Abstract**: In this paper, an analysis of modern cybercrime threats that arise as a result of security flaws and vulnerabilities in software products is given. Vulnerabilities in software products often arise as a result rapid development and represent threats that adversaries with hands-on knowledge and resources can use to gain unauthorized access to computer systems and networks, and thus confidential information inside. Accordingly, the work provides some recommendations concerning the mitigation of the consequences that may arise as well as recommendations on how to reduce the number of potential vulnerabilities that may occur during software development. The paper also analyzes the ways of vulnerability disclosure and attention was drawn to some legal aspects to be taken into account when disclosing information.*

***Key words**: software, vulnerability, threat, mitigation, disclosure*

1. Uvod

Pre masovnog korišćenja Interneta, jedan od načina koji su napadači najčešće koristili da se povežu na privatnu mrežu i steknu pristup poverljivim informacijama bio je biranje telefonskog broja modemom preko javne telefonske mreže. Zato pitanju zaštite udaljenog pristupa nije posvećivano mnogo pažnje. Postoji verovanje da se kriminal seli tamo gde ima novca, a sama činjenica da je Internet danas infrastrukturna osnova elektronskog poslovanja donosi brojne sigurnosne rizike i otvara nove mogućnosti koje potencijalni napadači mogu da iskoriste [1].

Christ je još pre 15 godina u svojoj doktorskoj disertaciji naveo da je brzina rasta, odnosno širenja Interneta eksponencijalna, ukoliko se kao metrika koristi broj Web servera dostupnih preko javnih IP

adresa [2]. Ta činjenica kao posledicu nameće izazove u razvoju tehnologija koje čine okosnicu Interneta. Tehnologija mora brzo da evolvira kako bi u scenariju naglog povećanja broja korisnika ostala upotrebljiva. Tadašnji komunikacioni standardi i protokoli zamenjeni su novim verzijama, rešenjima koja su većim delom redizajnirana ili potpuno novim rešenjima. Brzo evolviranje je takođe nametnuto i komunikacionom softveru: upravljačkim programima za mrežne adaptere, softveru koji obezbeđuje funkcije rutiranja, kao i serverima koji pružaju servise na aplikacionom sloju OSI referentnog modela [3], poput Web servera i servera za elektronsku poštu. U nastavku rada navodi se nekoliko primera evolviranja protokola, standarda i softvera. Prva verzija Hypertext Transfer protokola, HTTP/1.0 zamenjena je verzijom HTTP/1.1 koja danas predstavlja dominantni metod opsluživanja korisnika Web stranicama. Protokol HTTP/2 [4], zasnovan na eksperimentalnoj Google SPDY tehnologiji [5], trenutno podržava 7,6% Web stranica [6]. Protokoli koji obezbeđuju usluge poverljivosti i autentičnosti sadržaja, poput SSL (trenutno u verziji SSLv3 [7]) i TLS (trenutno u verziji TLSv1.2 [8]) se često ažuriraju, uglavnom zbog identifikovanih sigurnosnih propusta. Najzačajnije promene, uslovljene povećanjem kvaliteta sadržaja, načinjene su na standardima koji se koriste za razvoj front-end dela Web prezentacija: HTML5 [9], CSS3 i ECMAScript 6 [10]. Važno je napomenuti da je verzija u ovom slučaju više formalne prirode, s obzirom da se često menjaju postojeće funkcije i dodaju nove na osnovu predloga, shodno potrebama prilagođavanja Web pretraživačima i standardizacije. Što se softvera tiče, značajna unapređenja načinjena su na upravljačkim programima za mrežne adaptere koje koriste serverski operativni sistemi i operativni sistemi posebne namene koji se izvršavaju na ruterima, mrežnim barijerama i drugim mrežnim uređajuma. Unapređenja se uglavnom odnose na prilagođavanje većem protoku podataka i povećanje brzine odziva bez potrebe za značajnijim povećanjem ostalih resursa. Prema istraživanjima W3Tech W3Techs Web Technology Surveys, HTTP serversko tržište je danas, i pored činjenice da Apache i dalje dominira, znatno više fragmentisano nego ranije. Nove tehnologije (uglavnom Nginx) zauzimaju više od 30% ukupnog dela tržišta.

Brza evolucija i iterativna poboljšanja reflektuju se u polje softverskog inženjerstva i nameću pravce poput brzog razvoja softvera (engl. *agile software development*) [11]. Nepotpun proces kontrole kvaliteta softverskih proizvoda ostavlja dovoljno prostora za greške, sigurnosne propuste i ranjivosti koje se mogu iskoristiti.

2. Uzroci ranjivosti u softverskim proizodima

Ranjivost se definiše kao slabost u nekoj vrednosti, resursu ili imovini koja može biti iskorišćena, tj. eksploatisana. Pretnja je protivnik, situacija ili splet okolnosti sa mogućnošću i/ili namerama da eksploatiše ranjivost. Na primer, finansijski sponzorisani protivnik sa jasno definisanim ciljem i formalnom metodologijom smatra se strukturiranom pretnjom. Ova definicija pretnje stara je nekoliko decenija i konsistentna je s načinom opisivanja terorista [1].

Ukoliko se izuzme društveni inženjering [12], proboj u računarske sisteme i mreže, sticanje neovlašćenog pristupa poverljivim informacijama i narušavanje integriteta najčešće se izvodi iskorišćavanjem ranjivosti koje nastaju tokom razvoja softvera. Ranjivosti nastaju kao posledice brzog razvoja softvera i korišćenja metodologije "sigurnost zasnovana na skrivanju" (engl. *security by obscurity*) prilikom razvoja višenivovskih, odnosno modularnih sistema.

Kriminal karakterističan za informatičko doba je znatno ozbiljnija pretnja u odnosu na to kako ga pojedini entiteti doživljavaju [13]. Za kompanije koje razvijaju i održavaju softverske proizvode prethodno pomenuta percepcija pretnje je neprihvatljiva: osim održavanja konstantne ili rastuće stope tehnološkog razvoja, od takvih kompanija se očekuje da obezbede razumni nivo sigurnosti softvera i blagovremeno sprečavanje iskorišćavanja ranjivosti koje mogu nastati tokom razvoja. Iako industrija ima mehanizme za identifikaciju pretnji, veliki broj sigurnosnih propusta i iskorišćenih ranjivosti u poslednjih nekoliko godina ukazuju na činjenicu da postoji dovoljno mesta za dalje unapređenje i pojačano nametanje tih mehanizama. Metodologija brzog razvoj softvera isključuje mogućnost dovoljno detaljne provere

postojanja sigurnosnih propusta. Softverski proizvod se na tržištu pojavljuje sa potencijalnim ranjivostima koje se naknadno identifikuju i uklanjaju. U međuvremenu, softver ostaje ranjiv a ranjivosti se mogu iskoristiti ukoliko ih napadač otkrije pre nego što ih proizvođač identifikuje i otkloni.

Metodologija "sigurnost zasnovana na skrivanju" odnosi se na činjenicu da se potencijalna ranjivost krije od javnosti. Drugim rečima, ako protivnik otkrije ranjivost, mehanizam koji bi sprečio njeno iskorišćavanje ne postoji. Ova metodologija je suprotnost "sigurnosti zasnovanoj na dizajnu" (engl. *security by design*). Uopšteno, prilikom projektovanja bilo kakvog višesnivovskog sistema ili softverskog proizvoda, potrebno je razmotriti posledice prisustva napadača na bilo kom nivou. Razlike u primeni ovih metodologija prilikom projektovanja mogu se jednostavno objasniti na primeru generičkog modularnog biometrijskog autentifikacionog sistema. Generički sistem za biometrijsku kontrolu pristupa sastoji se od senzora, modula za ekstrakciju atributa, modula za poređenje i baze podataka o identitetima korisnika i odgovarajućim biometrijskim uzorcima. Korisnik koji želi da pristupi određenim resursima navodi svoj identitet. Senzor prikuplja biometrijski uzorak korisnika. Iz uzorka se izdvajaju atributi i računa sličnost između prikupljenog uzorka i uzorka smeštenog u bazi podataka koji odgovara navedenom identitetu. Na osnovu dozvoljene granice greške sistem donosi odluku, tj. određuje da li je to zaista taj korisnik i shodno odluci dozvoljava ili blokira pristup resursima. Ovakav modularni sistem može se zaobići na nekoliko načina, izvođenjem vrlo sofisticiranih napada [14, 15]:

1. Napadač prilaže lažni biometrijski uzorak (na primer, otisak prsta) senzoru.
2. Napadač izvršava napad ponavljanjem: snimljeni signal sa izlaza senzora prosleđuje se ostatku sistema (zaobilazi se senzor).
3. Napadač upotrebljava zlonamerni softver kako bi kompromitovao modul za ekstrakciju atrubuta; kompromitovani modul generiše vektore atribute koje odabira napadač.
4. Napadač prosleđuje sintetički vektor atributa modulu za poređenje.
5. Napadač modifikuje rezultat koji generiše modul za poređenje.
6. Napadač modifikuje biometrijske uzorke legitimnih korisnika u bazi podataka; ovaj napad osim sticanja neovlašćenog pristupa rezultuje odbijanjem usluga legitimnim korisnicima.
7. Napadač presreće komunikacioni kanal između baze podataka i modula za poređenje i podmeće lažne biometrijske uzorke modulu za poređenje.
8. Napadač stiče administrativne privilegije i menja konačnu odluku sistema.

Ovaj primer ilustruje ranjivost sistema koji su projektovani primenom metodologije "sigurnost zasnovana na skrivanju". U ovom slučaju je jasno da projektant nije predvideo postojanje napadača u prethodno pomenutih osam tačaka. Projektant koji predvidi postojanje napadača primenjuje metodologiju "sigurnosti zasnovana na dizajnu" što znači da može da identifikuje potencijalne ranjivosti, implementira dodatne zaštitne mehanizme i samim ti spreči izvođenje prethodno pomenutih napada. Na primer, napad prosleđivanjem sintetičkog vektora modulu za poređenje je moguće sprečiti ukoliko se modul za ekstrakciju atributa i modul za poređenje realizuju kao jedna komponenta ili ukoliko je veza imeđu njih kriptografski zaštićena. Šifrovanje veze između baze podataka i modula za poređenje sprečava napad presretanjem komunikacionog kanala i podmetanja lažnih uzoraka modulu za poređenje [16].

3. Pregled značajnijih ranjivosti softverskih proizvoda

U ovom delu rada ukratko su opisane značajnije ranjivosti koje su otkrivene 2014 godine (*Heartbleed*, *Shellshock, POODLE*), 2015 godine (*GHOST*, *Freak*). Ranjivosti CVE-2014-0160 i CVE-2014-6271 su, zbog svoje ozbiljnosti i pažnje medija koju su privukle, detaljnije opisane od ostalih.

Tabela 1. *Preled značajnijih ranjivosti softverskih proizvoda u periodu 2014-2015 godine*

CVE	Naziv	Otkrivena	Softver	Posledica iskorišćenja
2014-0160	Heartbleed	2014	OpenSSL	Čitanje poverljivih podataka iz memorije sistema, preuzimanja kriptografskih ključeva
2014-6271	Shellshock	2014	Bash	Izvršenje proizvoljnog koda, neovlašćeni pristup udaljenom računarskom sistemu
2014-3566	POODLE	2014	SSLv3	Dešifrovanje poruka
2015-0235	GHOST	2015	glibc	Preuzimanje kontrole nad udaljenim Linux sistemom
2015-0204	FREAK	2015	SSL/TLS	Krađa podataka kriptoanalizom zasnovanom nametanjem kratkih RSA ključeva

Diskusiju vezanu za ranjivost započinjemo analizom sigurnosnog propusta CVE-2014-0160, pozantijeg u javnosti kao *Heartbleed*. CVE-2014-0160 je sigurnosni propust u softverskoj biblioteci OpenSSL, koja obezbeđuje potpuno rešenje za implementaciju SSL/TLS protokola na serverskoj i klijentskoj strani. OpenSSL na najvišem nivou, obezbeđuje transparentnu kriptografski zaštićenu komunikaciju između krajnjih tačaka u mreži i time sprečava pasivne napade prisluškivanja i aktivne napade izmene sadržaja. Da bi se to postiglo, u softveru je implementiran veliki broja kriptografskih primitiva, poput simetričnih blokovskih algoritama, kriptografije zasnovane na eliptičkim krivama i protokola za razmenu ključeva, kao i kompletna podrška za rad sa X.509 sertifikatima. Imajući to u vidi, može se zaključiti da je OpenSSL složena biblioteka sa velikim brojem elemenata koji međusobno interaguju na različite složene načine, što povećana šansu za pojavu sigurnosnih propusta. Iako se nezvanično navodi da postoji od 2011 godine, kao i da potiče od studenta doktorskih studija koji je bio zadužen za implementaciju *Heartbeat* ekstenzije, ranjivost je *Heartbleed* otkrio Neel Mehta, inženjer Google sigurnosnog tima 2014 godine. Ranjivost je posledica greške u kodu koja se iskorišćava na sledeći način: slanjem posebno oblikovanog upita serveru nazvanog *Heartbeat bug* moguće je izazvati čitanje podataka sa lokacija van memorijskog bafera [17]. Ranjivost je nazvana *Heartbleed* zato što upit iskorišćava nedostatak u proširenju *Heartbeat*, čime napadač stiče mogućnost da čita poverljive podatke iz memorije sistema zaštićenih ranjivim verzijama OpenSSL softvera. Testovima je čak pokazano da napadači mogu, bez korišćenja bilo kojih privilegovanih informacija ili akreditiva, da preuzmu privatni ključ servera i simetrične sesijske ključeve. To znači da serveri koji primene sigurnosnu zakrpu koju je ponudio OpenSSL, moraju takođe nadograditi sve svoje ključeve ili će i dalje biti ranjivi. Pošto veliki broj Web servera koristi OpenSSL biblioteku koja pruža kriptografske usluge HTTPS protokolu, otkrivanje ove ranjivosti imalo je značajne posledice [18]: na tržištu su se pojavile alternativne verzije OpenSSL biblioteke drugih proizvođača (engl. *fork*), poput LibreSSL (OpenBSD) i BoringSSL (Google), u kojima su u prečišćenom kodu implementirane drugačije metode umanjenja ovakvih sigurnosnih rizika u budućnosti.

Ranjivost CVE-2014-6271 je takođe privukla značajniju pažnju medija i dobila prepoznatljivo ime *Shellshock*, odnosno *Bash bug*. Komandni interpreter je komponenta prisutna u svim UNIX zasnovanim sistemima. GNU Bash (Bourne-again shell) je komandni interpreter objavljen 1989 godine kao zamena za tada dominantni Bourne shell, koji nije pripadao kategoriji softvera otvorenog koda. Analiza istorije izvornog koda za Bash ukazuje na činjenicu da je ova ranjivost prisutna u kodu od verzije 1.03 (septembar 1989), što znači da se u kodu nalazila 25 godina. Drugim rečima, računari na kojima se izvršava bilo koji UNIX zasnovani operativni sistem (na primer, Linux ili Mac OS X), uključujući i rutere i NAS uređaje, bez adekvatne zakrpe za Bash komandni interpreter bili su sve vreme ranjivi. Ranjivost je izazvana činjenicom da Bash nenamerno izvršava komande kada su komande dodate na kraj definicija funkcija koje se čuvaju u vrednostima promenljivih okruženja [19]. Namenski generisane promenljive okruženja mogu izazvati izvršenje proizvoljnog koda u kontekstu korisničkog naloga, odnosno procesa koji je pokrenuo komandni interpreter. Kakve su posledice ove ranjivosti? Hrvatski nacionalni CERT

objavio je u oktobru 2014 godine sledeću vest: "Napadači iz Rumunije uspešno su iskoristili Shellshock ranjivost kako bi kompromitovali servere kompanija Yahoo!, Lycos i WinZip, prema rečima sigurnosnog stručnjaka Future South Technologies-a" [20]. Stručnjaci u domenu računarske sigurnosti navode da je *Shellshock* nešto veći problem od *Heartbleed* ranjivosti: *Heartbleed* omogućava napadačima da, na primer, ukradu korisnička imena i lozinke, ali ne i da pokrenu zlonamerne programe na ranjivim sistemima. Kompanija Rapid7, koja se bavi razvojem softvera za testiranje proboja, upozorava da je ranjivost ocenjena sa "10" kada je reč o ozbiljnosti, kao i da je napadač može relativno lako iskoristiti i preuzeti kontrolu nad operativnim sistemom. Problem postaje još ozbiljniji ukoliko se u obzir uzme činjenica da je za funkcionisanje nekih serverskih aplikacija, poput Web servera koji koriste CGI, neophodan komandni interpreter. To omogućava napadačima da iskoriste ranjive verzije Bash alata i udaljeno izvrše proizvoljnih komande. Dakle, navedena ranjivost može omogućiti napadaču da dobije neovlašćeni pristup udaljenom računarskom sistemu. Broj javno dostupnih servera ranjivih u ovom kontekstu nije zanemarljiv [21]. Nakon otkrivanja Shellshock ranjivosti, usledile su vesti o načinu rešavanja i objavljene su zakrpe, a zazim su se pojavile informacije o novim dodatnim ranjivostima koje su dobijale interesantna imena, kao što su *Aftershock* i slično.

Krajem septembra 2014 godine otkrivena je još jedna ranjivost u verziji 3 SSL protokola koja se može iskoristiti za krađu poverljivih informacija. CVE-2014-3566, odnosno POODLE ("*Padding Oracle On Downgraded Legacy Encryption*") ranjivost je algoritamske prirode i omogućava izvođenje napada tipa čovek u sredini (engl. *man-in-the-middle*). SSLv3 ne sprovodi validaciju određenih delova podataka koji prate svaku poruku. Napadači mogu da iskoriste tu ranjivost sa ciljem dešifrovanja individualnog bajta u jednom trenutku, tako da se može ekstrahovati otvoreni tekst poruke dešifrovanjem bajt po bajt [22]. TLSv1.0 i novije verzije sprovode robusniju validaciju dešifrovanih podataka i kao takve nisu osetljive na isti problem. Problem postoji zato što određeni broj Web servera i Web pretraživača i omogućavaju korišćenje SSL v3 protokola s ciljem održavanja kompatibilnosti sa IE6.

Kompanija Qualys objavila je 2015. godine vest o ranjivosti u Linux GNU C biblioteci (glibc), koja je deo gotovo svih distribucija Linux operativnog sistema. Ranjivost CVE-2015-0235 dobila je ime GHOST, koje potiče od mogućeg prekoračenja bafera unutar glibc GetHOST funkcije. Pomenuta funkcija je zadužena za razrešavanje mrežnih adresa, i kao takva potencijalno ugrožava sigurnost gotovo svog softvera koji se na neki način odnosi na mrežu. GHOST ranjivost se smatra kritičnom zato što napadač može da je iskoristi i preuzme kontrolu nad ciljnim Linux sistemom bez potrebe za prethodnim znanjem sistemskih akreditiva, tj. lozinki naloga sa administrativnim privilegijama. Na primer, napadači mogu da iskoriste ranjivost, udaljeno izvrše zlonamerni kod i preuzmu kontrolu nad Web serverom. Qualys, kompanija koja je otkrila ovu ranjivost, tvrdi da korišćenjem zlonamernog koda koji iskorišćava ovu ranjivost može izvršiti prozvoljni kod preko Exim servera za elektrosku poštu [23].

U martu 2015 godine otkrivena je nova SSL/TLS ranjivost, CVE-2015-0204, koja dozvoljava napadaču da presretne HTTPS konekcije između ranjivih klijenata i servera i nametne im korišćenje slabe kriptografske zaštite, što za dalju posledicu može imati krađu osetljivih podataka [24]. Ranjivost FREAK ("*Factoring RSA Export Keys*") indirektna je posledica usaglašavanja sa kriptografskim izvoznim regulativana Sjedinjenih Američkih Država. Ove regulative ograničavaju dužine ključeva koje se koriste – cilj je omogućiti Američkoj nacionalnoj agenciji za bezbednost (NSA) da izvršiti kriptoanalitičke napade i onemogućiti druge organizacije sa manjim računarskim resursima da izvrše iste. Na primer, moduo u RSA algoritmu može biti najveće dužine 512 bita (takozvani RSA izvozni ključevi). Kriptoanaliza RSA algoritma sa kratkim ključevima izvodljiva je pomoću *Number Field Sieve* algoritma, koristeći računarske servise u oblaku za ne više od 100 dolara. Kombinovanje napada "čovek u sredini" u cilju manipulisanja incijalnog dogovora o kriptografskim algoritmima koji će se koristiti u toku sesije i prethodno pomenutog algoritma koji se izvršava na *cloud* servisima, nameće činjenicu da napad može ugroziti bezbednost bilo kog Web sajta koji omogućava korišćenje RSA ključeva dužine 512 bita uz upotrebu relativno skromnih računarskih resursa.

4. Ublažavanje mogućih posledica

Koraci koje je potrebno izvesti kako bi se iskoristila bilo koja od pomenutih ranjivosti nisu složeni i može ih izvesti svaki ozbiljniji programer ili istraživač koji se bavi računarskom sigurnošću. Kada se ove slabosti subjektivno posmatraju, lako se može steći utisak da su nastale nepažnjom programera ili projektanata. Međutim, ovakve greške i propusti dešavaju se bez obzira na to da li je do nepažnje došlo ili ne. U ovom delu rada identifikovano je nekoliko ključnih elemenata na koje treba obratiti pažnju kako bi se mogućnost pojave ranjivosti i njenog iskorišćavanja sveo na minimum: testiranje softvera, revizija sigurnosti, pravilan odabir programskih jezika i ekonomski podsticaji.

Automatsko testiranje softverskih proizvoda upražnjava se i njegov značaj je poznat duži niz godina [25]. Razvoj naprednih alata za fazi testiranje, poput American Fuzzy Lop [26] olakšava pronalaženje neočekivanih grešaka i propusta koji se ne mogu otkriti standardnim metodologijama testiranja. Značajno unapređeni alati za statičku analizu mogu da predvide i ukažu na moguće posledice grešaka koje se javljaju u ranom periodu razvoja određene komponente. Razvojni timovi, međutim, navodno ne koriste ove alate onoliko često koliko bi trebali da ih koriste [27]. Potpuna provera koda koji evolvira zahteva, s druge strane, dosta kontinualnog truda – u obzir treba uzeti i činjenicu da veliki broj programera testiranje koda doživljava kao izuzetno nemaštovitu upotrebu svog vremena. Ovaj problem se može rešiti ekonomski, odnosno stimulacijom programera koji rade na kodu koji je kritičan po pitanju sigurnosti ukoliko je kod u potpunosti tesiran i pokriven. Alternativno, ovaj zadatak se može delegirati kompetentnim inženjerima za kontrolu kvaliteta, čiji će jedini zadatak biti da proizvod detaljno testiraju nakon svake promene.

Nezavisni istraživači, vođeni ekonomskim razlozima, entuzijazmom ili radi održanja ugleda, često sprovode sigurnosne revizije popularnog zaštitnog softvera ili komponenti sistema koje u značajnoj meri mogu da naruše sigurnost ukoliko su ranjive na određene napade. Često ovakve revizije sprovode kompanije ili organizacije specijalizovane za sigurnost softvera, sprečavanje visokotehnološkog kriminala i slične oblasti, kako bi u budućnosti bile poznate kao entiteti koji su otkrili ranjivosti i ukazali na moguće napade, ili iz čisto ideoloških razloga. Polarizovanost u domenu sigurnosti, međutim, može dovesti objektivnost revizije u pitanje. Objasnićemo ovo na primeru softvera TrueCrypt. Na zvaničnoj stranici proizvođača softvera TrueCrypt naznačeno je da je razvoj prekinut u maju 2014 godine. Jedan od razloga koji je naveden je izjava NSA da je na zahtev proizvođača izvršila analizu sigurnosti softvera i otkrila sigurnosne propuste. Dokazi o sigurnosnim propustima nisu objavljeni, a pomenuti razlog je kontradiktoran sa izjavama Bruce Schneiera i Edwarda Snowdena koji su podržavali razvoj i preporučivali upotrebu softvera, zato što nije omogućavao pristup šifrovanim podacima ni jednom entitetu (uključujući NSA) koji nije imalo odgovarajući ključ. U ovakvim situacijam potrebno je uložiti veća finansijska sredstva u nezavisnu komaniju koja je specijalizovana za reviziju sigurnosti. Potrebno je takođe naznačiti da sigurnosne revizije nisu ograničene isključivo na potpune revizije, već i na nivou modula, kao i da revizija koda treba da bude obavezna, a ne opciona.

OpenSSL, biblioteka u kojoj je otkriven veći broj ranjivosti i koja je takođe bila meta velikog broja napadača napisana je u jezika C. U jeziku C su takođe napisana i jezgra svih UNIX zasnovanih operativnih sistema, koja, prema istraživanjima W3Tech W3Techs Web Technology Surveys čine okosnicu 68% web servera [28], kao i pametnih telefona, rutera i hardverskih mrežnih barijera. Jezik C je nastao 1978 godine, a osnovni standard od tada nije značajno ažuriran. Jasno je da jezik koji ne obezbeđuje zaštitu memorijskog prostora i u odnosu na savremene programnske jezike predstavlja blagu apstrakciju asemblera nije pogodan izbor za izradu softvera koji je kritičan po pitanju sigurnosti. Danas je raspoloživ veći broj bezbednijih programskih jezika koji su otporni na greške a istovremeno ne nameću degradaciju performansi softvera. Ukoliko se za dugoročni cilj postavi upotreba tih jezika umesto alternativa niskog nivoa (gde je moguće), može se očekivati znatno smanjenje broja kritičnih ranjivosti koje se danas često otkrivaju. Među jezicima koji su pogodni za upotrebu u ovom scenariju može se izdvojiti Rust [29], koji je eksplicitno dizajniran za sigurno mrežno i sistemsko programiranje, i čiji dizajn

sprečava one kategorije ponašanja koje su identifikovane kao najčešći uzrok pojave kritičnih ranjivosti (prekršaji vezani za zaštitu memorijskog prostora i stanje trke).

Softver koji "napaja" većinu savremenog Web-a je najčešće besplatan softver ili softver otvorenog koda, poput operativnog sistema Linux, Apache Web servera, MySQL baze podataka, PHP-a i dr. Prednosti softvera otvorenog koda su mnogobrojne, ali treba u obzir uzeti i činjenicu da programeri koji rade na besplatnom softveru ili softveru otvorenog koda nemaju adekvatnu kompenzaciju za svoj rad. Doniranje i podsticanje entitete koji su ostvarili korist na doniranje tim projektima dozvoliće onima koji rade na projektu da posvete više vremena razvoju, održavanju i poboljšanju kvaliteta softvera, kako po pitanju performansi, tako i po pitanju bezbednosti.

5. Načini objavljivanja informacija o otkrivenoj ranjivosti i pravni aspekti

Objavljivanje informacija o otkrivenoj ranjivosti (engl. *vulerability disclosure*) ima svoje dobre i loše strane. Drugim rečima, ukoliko nezavistan istraživač ili kompanija koja se bavi informacionom sigurnošću u javnosti iznese podatke o otkrivenoj ranjivosti, posledice su dualne prirode.

Dobre strane objavljivanja informacija o ranjivosti su sledeće:

- javnost se upoznaje sa sigurnosnim rizikom.
- sistem administratori i administratori zaduženi za sigurnost mogu blagovremeno da reaguju,
- proizvođač softvera je motivisan da kreira i objavi zakrpu i na taj način sačuva svoj ugled,
- eliminiše se bilo kakva mogućnost prelaska na princip "sigurnosti zasnovane na skrivanju".

Objavljivanje informacija o ranjivosti ima i svoje loše strane:

- šanse za napad na sisteme koji su ranjivi su znatno uvećane ukoliko sistem administratori, administratori zaduženi za sigurnost ili menadžment nisu obratili pažnju na činjenicu da su informacije o ranjivosti javno dostupne,
- ugled proizvođača softvera može biti narušen proporcionalno količini štete koja je naneta ukoliko u ugovoru o korišćenu softverskog proizvoda nije prezizno i potpuno definisano odricanje od odgovornosti.

Postoji nekoliko generičkih kategorija u koje se otkrivanje informacija o ranjivosti može svrstati [30, 31]: bez objavljivanja, potpuno i delimično. Prvu kategoriju je najjednostavnije objasniti na primeru nezavisnog istraživača koji je ranjivost identifikovao ali o tome nije obavestio proizvođača softvera ili odgovarajuće autoritete zadužene za sigurnost. Ova kategorija je iz očiglednih razloga tipična za zajednice hakera sa "crnim šeširom". U slučaju potpunog objavljivanja, nezavisni istraživač sve informacije o ranjivosti prosleđuje kako proizvođaču softvera, tako i javnosti – kako je otkrivena, koji su softverski proizvodi i koje verzije ranjive, a u nekim slučajevima čak i odgovore na sledeća pitanja: kako se ranjivost može iskoristiti i kako se sistemi mogu zaštititi od iskorišćavanja ranjivosti. Delimično objavljivanje, koje se takođe naziva i odgovornim otkrivanjem je objavljivanje informacija na način koji u najmanjoj mogućoj meri ugrožava korisnike. Drugim rečima kada je ranjivost otkrivena, istraživač obaveštava proizvođača softvera; ukoliko se proizvođač ne odazove nakon 30 dana, osnosno ne obezbedi zakrpu, pristupa se potpuom objavljivanju. Više podataka o delimičnom objavljivanju dostupno je u radu Stephen Shepherd-a [30].

Detaljnu analizu pravnih aspekata objavljivanja informacija o ranjivosti može se naći na stranici udruženja Electronic Frontier Foundation [32], a u daljem tekstu su navedene neke činjenice koje su pravno najkritičnije po istraživače.

- Što je više činjenica iznešeno u javnost, postupak je rizičniji. Potrebno je postaviti pitanje koliko savet istraživača može pomoći potencijalnom napadaču.
- Što je više funckionalnog koda dato u savetu i iznešeno u javnost, postupak je rizičniji. Potrebno je postaviti pitanje da li se kod može prevesti u izvršni kod koji može iskoristiti ranjivost.
- Objavljivanje je rizičnije ukoliko ima više entiteta koji te informacije mogu da iskoriste kako bi prekršili zakon. U ovom slučaju se postavlja pitanje da li se informacije otkrivaju javnosti ili grupi od poverenja.
- Ukoliko se sigurnosni propust odnosi na softver za upravljanje digitalnim pravima (engl. *digital rights management*, *DRM*) ili softver koji kontroliše pristup delima zaštićenim zakonom o autorskim pravima (poput autentifikacionih protokola i maskiranja koda) objavljivanje informacija može biti vrlo rizično. U ovom slučaju neophodno je zatražiti savet od pravnika da li se objavljivanjem infromacija krši *Digital Millennium Copyright Act* (DMCA).
- Ukoliko se objavljivanjem informacija prekrši zakon ili objavljivanje dokaz nezakonitih aktivnosti, istraživač se smatra krivim, bez obzira što je objavljivanje informacija o ranjivosti bilo dobronamernog karaktera.

6. Zaključak

Na osnovu izloženog može se zaključiti da u softverskim proizvodima postoje sigurnosni propusi koji napadačima pružaju mogućnost uspešnog izvođenja napada na informacione sisteme. Veliki broj ranjivosti nastaje kao nedostatak radne snage na složenim softverskim prozvodima (što je, na primer, slučaj sa OpenSSL bibilotekom) i/ili nedovoljno testiranog nasleđenog koda iz starijih verzija (što je, na primer, slučaj sa komandnim interpreterom Bash). Autori rada smatraju da se indentifikovani elementi ublažavanja mogućih posledica mogu pokazati efektivnim na duže staze, uzevši u obzir da se pomenute metode, poput automatskog testiranja softvera i revizije sigurnosti, koriste u softverskoj industriji. Iako pomenute metode zahtevaju veća finansijska ulaganja u sam proces razvoja softvera, autori smatraju da su ta ulaganja opravdana ukoliko razultuju otklanjanjem sigurnosnih propusta, a samim tim i delimično ili potpuno ublažavanje posledica nastalim iskorišćavanjem ranjivosti

Literatura

[1] Pleskonjić, D., Maček, N., Đorđević, B., Carić, M., *Sigurnost računarskih sistema i mreža*, Mikro knjiga, Beograd, 2007.

[2] Christ, H. D-K. M., *Lay Internet Usage-An Empirical Study with Implications for Electronic Commerce and Public Policy*, doktorska disertacija, Humboldt-Universität, Berlin, Nemačka 2002.

[3] ISO/IEC 7498-1:1994, *Information technology – Open systems interconnection – Basic reference model: The basic model*, 1994.

[4] Belshe, M., Peon, R., Thomson, E. M., *Hypertext Transfer Protocol Version 2 (HTTP/2)*, RFC 7540, 2015.

[5] Chromium Project, *SPDY: An experimental protocol for a faster web*, dostupno na Web lokaciji: https://dev.chromium.org/spdy/spdy-whitepaper, 2009. Poslednji put posećeno 20. maja, 2016.

[6] W3Techs Web Technology Surveys, *Usage of HTTP/2 for websites*, dostupno na Web lokaciji: http://w3techs.com/technologies/details/ce-http2/all/all. Poslednji put posećeno 20. maja, 2016.

[7] Freier, A., Karlton, P., Kocher, P., *The Secure Sockets Layer (SSL) Protocol Version 3.0*, RFC 6101, 2011.

[8] Dierks, T., Rescorla, E., *The Transport Layer Security (TLS) Protocol Version 1.2*, RFC 5246, 2008.

[9] W3C, *HTML5: A vocabulary and associated APIs for HTML and XHTML*, dostupno na Web lokaciji: https://www.w3.org/TR/2014/REC-html5-20141028/, 2014. Poslednji put posećeno 20. maja, 2016.

[10] ECMA, *ECMA-262 6th Edition: ECMAScript® 2015 Language Specification*, dostupno na Web lokaciji: http://www.ecma-international.org/ecma-262/6.0/, 2015. Poslednji put posećeno 20. maja, 2016.

[11] Cohen, D., Lindvall, M., Costa, P, *Agile software development*, DACS SOAR Report, 11, 2003.

[12] Mitnick, K. D., Simon, W. L., *The art of deception: Controlling the human element of security*, John Wiley & Sons, 2011.

[13] Wall, D., *Cybercrime: The transformation of crime in the information age (Vol. 4)*, Polity, 2007.

[14] Biggio, B., *Adversarial Pattern Classification*, doktorska disertacija, University of Cagliari, Cagliari, Italija, 2010.

[15] Ratha N. K., Connell J. H., Bolle R. M., *An analysis of minutiae matching strength*, In Josef Bigün and Fabrizio Smeraldi, editors, AVBPA, volume 2091 of Lecture Notes in Computer Science, pages 223–228. Springer, 2001.

[16] Maček, N., *Detekcija upada mašinskim učenjem / Machine Learning in Intusion Detection*, Zadužbina Andrejević, Beograd, 2015.

[17] CVE-2014-0160. Dostupno na Web lokaciji: https://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2014-0160. Poslednji put posećeno 20. maja, 2016.

[18] Durumeric, Z., Kasten, J., Adrian, D., Halderman, J. A., Bailey, M., Li, F., Paxson, V., *The matter of Heartbleed*, In Proceedings of the 2014 Conference on Internet Measurement Conference, pp. 475-488, ACM, 2014.

[19] CVE-2014-6271. Dostupno na Web lokaciji: https://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2014-6271. Poslednji put posećeno 20. maja, 2016.

[20] Nacionalni CERT, *Napadači uspješno iskorištavaju Shellshock ranjivost*, dostupno na Web lokaciji: http://www.cert.hr/node/24193, 2014. Poslednji put posećeno 20. maja, 2016.

[21] Delamore, B., Ko, R. K., *A Global, Empirical Analysis of the Shellshock ulnerability in Web Applications*, In Trustcom/BigDataSE/ISPA, 2015 IEEE, 1, pp. 1129-1135, 2015.

[22] CVE-2014-6271. Dostupno na Web lokaciji: https://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2014-3566. Poslednji put posećeno 20. maja, 2016.

[23] CVE-2015-0235. Dostupno na Web lokaciji: https://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2015-0235. Poslednji put posećeno 20. maja, 2016.

[24] CVE-2015-0204. Dostupno na Web lokaciji: https://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2015-0204. Poslednji put posećeno 20. maja, 2016.

[25] Zhu, H., Hall, P. A., May, J. H., *Software unit test coverage and adequacy*, ACM Computing Surveys (csur), 29(4), pp. 366-427, 1997.

[26] Zalewski, M., *American Fuzzy Lop*., dostupno na Web lokaciji: http://lcamtuf.coredump.cx/afl/. Poslednji put posećeno 14. marta, 2016.

[27] Johnson, B., Song, Y., Murphy-Hill, E., Bowdidge, R., *Why don't software developers use static analysis tools to find bugs?*, in Software Engineering (ICSE), 2013, 35th International Conference on (pp. 672-681). IEEE.

[29] Mozilla Foundation, *The Rust Programming Language*, dostupno na Web lokaciji: https://www.rust-lang.org/. Poslednji put posećeno 14. marta, 2016.

[30] Shepherd S., Vulnerability Disclosure: How do we define Responsible Disclosure?, SANS Institute, 2003.

[31] Vidstrom A., *Full Disclosure of Vulnerabilities – Pro/Cons and Fake Arguments*, Net Security, 2002.

[32] Electronic Frontier Foundation, *Coders' Rights Project Vulnerability Reporting FAQ*, dostupno na Web lokaciji: https://www.eff.org/issues/coders/vulnerability-reporting-faq. Poslednji put posećeno 20. maja, 2016.