SIGURNOSNI ASPEKTI ZAŠTITE PROGRAMSKOG KODA POMOĆU ODGOVARAJUĆIH SOFTVERA

Bjelovuk Dragutin, dipl. oecc, VŠPM „PRIMUS" Gradiška

UDK 004.7.056.5
004.4:004.056.5
347.77:004.4

Sažetak: *Zloupotreba programskog koda predstavlja glavni problem za tvorce softvera. Danas je na Internetu moguće pronaći ilegalne kopije gotovo svih komercijalnih aplikacija. Iako je zloupotreba programskog koda pravno regulisana putem autorskih prava, patentata i sl., radi obezbjeđivanja dodatne zaštite u softvere se ugrađuju i različiti sigurnosni mehanizmi. Tvorci softvera takve mehanizme najčešće razvijaju sami, iako su na tržištu dostupne mnoge komercijalne aplikacije koje omogućuju zaštitu programskog koda.*

Ključne riječi: *zaštita softvera, metode zaštite softvera, zakonska zaštita softvera...*

1. Uvod

Softverska industrija je u ekspanziji i potrebe za novim aplikacijama sve su veće, a Internet postaje novi medij za jednostavnu i brzu distribuciju softvera. Ovakva situacija pruža zanimljive mogućnosti, ali istovremeno donosi i velike probleme proizvođačima softvera. Softver je proizvod koji je teško i skupo razvijati, zbog čega se često ilegalno kopira i distribuira (softversko piratstvo). Zbog toga softverske kompanije trpe velike gubitke. Prema procjenama na svaku legalnu kopiju softvera dolazi 5 ilegalnih kopija.

Kako bi se zaštitili, proizvođači softvera su počeli implementirati razne hardverske i softverske metode koje onemogućuju neovlašteno kopiranje i distribuciju programa te njihovo korištenje. Te zaštite uključuju provjeru serijskog broja isporučenog softvera, provjeru raznih parametara računara na kojem je softver instaliran, obveznu autentifikaciju svake kopije softvera, provjeru ispravnosti isporučenih licenci, enkripciju izvršnog koda programa, te razne druge metode.

Ne postoji metoda za zaštitu softvera koju bi bilo nemoguće probiti. Svaka metoda koja je do sada primijenjena u praksi na kraju je ipak razbijena te sav softver nakon nekog vremena postane dostupan u piratskoj verziji. Zbog toga se nameće pitanje smisla ulaganja vremena i novca u razvoj takvih zaštita. Razlog za to je potpuno komercijalan jer ako softver nema zaštitu postat će dostupan u besplatnoj verziji za svega nekoliko dana. Međutim, ako softver ima zaštitu, moguće je da ona neće biti probijena neko vrijeme. Na taj način je moguće spriječiti značajniju financijsku štetu softverskoj kompaniji jer nakon tog perioda softver će ionako zastarjeti te više neće donositi značajniju dobit.

U softver je moguće ugraditi višestruke zaštite i provjere u različitim dijelovima programskog koda, tako da je teško ispravno razbiti zaštitu. Kod takve zaštite je moguće da piratske verzije programa prividno rade, ali u nekim fazama rada program se jednostavno ugasi, počne se ponašati nestabilno ili nema neke funkcionalnosti koje ima legalna verzija. Ta činjenica može mnoge korisnike natjerati da kupe program, čak i nakon objavljivanja piratskih verzija, jednostavno zato da izbjegnu gnjavažu te gubitak vremena i novca, pokušavajući koristiti neispravan softver.

Piratska verzija softvera može donijeti značajne novčane uštede njegovim krajnjim korisnicima, ali pirati razbijaju softverske zaštite isključivo iz znatiželje, izazova ili ugleda. Pirati su najčešće spremni provesti mjesece pokušavajući probiti nove softverske zaštite, a programeri u pravilu podcjenjuju njihov broj i motivaciju.

2. Metode zaštite programskog koda

Najraširenije tehničke metode zaštite programskog koda (softvera) su pomoću softverskih alata. Princip rada ovih alata je da od korisnika zahtijevaju neku vrstu autentifikacije prije nego što mu omoguće normalno korištenje softvera. Softverski alati pokušavaju nekako šifrirati ili samo zakomplicirati izvršni kod programa tako da onemoguće pirate u pokušaju razbijanja zaštite.

2.1. Zaštita softvera pomoću fiksne šifre

Vjerojatno najjednostavniji oblik zaštite je ugrađivanje programske funkcije koja zahtijeva od korisnika unošenje određene šifre za identifikaciju. Ta vrijednost je uvijek ista i ne zavisi od nikakvih parametara kao što su korisnički podaci ili podaci o korisnikovom računaru. Kod ovakve zaštite je dovoljno da pirat jednom pronađe šifru i objavi je na Internetu pa da svatko može besplatno registrovati taj softver.

Usprkos svojoj jednostavnosti, ova vrsta zaštite ima neke prednosti nad ostalim mehanizmima. Najveća prednost je što ispravna šifra ne mora biti smještena na disku već je negdje u programskom kodu. Tu šifru je nemoguće saznati čitanjem programskog koda jer se obje šifre najprije nekim složenim matematičkim operacijama pretvore u nove vrijednosti koje se tek tada uspoređuju. Ako su te operacije dovoljno komplikovane može biti jako teško shvatiti ispravan postupak računanja šifre.

Najčešći način uklanjanja te šifre je prepravljanje dijela koda za provjeru šifre tako da se potpuno zaobiđe provjera i program normalno nastavi s radom. Da se spriječi takvo probijanje zaštite, moguće je koristiti šifru za dekroptovanje dijelova programskog koda. Ako je dio koda enkriptovan pomoću šifre onda će ga biti moguće dekrip-

tovati samo pomoću te iste šifre. Čak i ako pirat uspije premostiti algoritam za provjeru šifre kod upisa program još uvijek neće ispravno raditi.

Enkriptovani dio koda se može dekriptovati u memoriji neposredno prije nego korisnik pokrene tu funkciju, i kada funkcija više nije potrebna ponovno se enkriptuje. Ako program ima nekoliko enkriptovanih dijelova onda će maksimalno jedan od tih dijelova biti dekriptovan u memoriji u svakom trenutku, što piratima onemogućava snimanje dekriptovanog programskog koda iz memorije.

2.3. Zaštita softvera pomoću promjenjive šifre

Bolji način zaštite je generisanje šifre pomoću parametara koji su sakupljeni iz podataka o korisniku ili tajno prikupljeni iz parametara korisničkog računara. Ovakav program generiše šifru iz korisničkih podataka te koristi podatke poput korisničkog imena, adrese i sl. Kada korisnik upiše svoju šifru, program provjerava da li se šifra poklapa s izračunatom šifrom.

Program koji koristi podatke sakupljene iz korisničkog računara radi na sličan način samo što umjesto korisničkih podataka koristi razne podatke o korisničkom računaru. Iz prikupljenih vrijednosti program računa neku vrijednost i čuva je u skrivenom fajlu na hard disku računara. Korisnik mora proizvođaču softvera poslati generisani identifikacijski broj računara na osnovu kog mu proizvođač softvera vrati odgovarajuću šifru za registraciju softvera.

2.4. Zaštita softvera pomoću serijskog broja hardvera

Ovo je najčešće korišten način zaštite softvera. Prilikom instalacije programa generiše se pseudo-slučajni serijski broj računara. Aplikacija šifrira taj broj i sakrije ga u posebnu datoteku, a može ga zapisati i u neku od datoteka operativnog sistema. Serijski broj se generiše pomoću serijskog broja isporučene kopije softvera te iz prikupljenih podataka o hardveru računara i konfiguraciji operativnog sistema. Na taj način se postiže jedinstveni generisani broj.

Nakon instalacije program je potrebno registrovati. U procesu registracije, generisani broj se šalje proizvođaču softvera koji korisniku vraća registracionu šifru koja odgovara generisanom serijskom broju hardvera.

2.5. Zaštita softvera pomoću registracione datoteke

Registracione datoteke imaju istu ulogu kao i šifre, a prednost im je količina informacija koju je moguće snimiti u njih. Registraciona datoteka može sadržavati informacije o korisniku, šifru za autentifikaciju korisnika, ključeve za dekriptovanje enkriptovanih dijelova izvršnog koda aplikacije i sl. Registraciona datoteka je obično enkriptovana tako da je nemoguće pročitati i promijeniti njen sadržaj.

U takvoj datoteci može biti zapisan dio izvršnog koda aplikacije tako ako datoteka ne postoji ili je neispravno dekriptovana, aplikaciji nedostaje dio koda i ne može ispravno raditi. U registracionu datoteku je moguće smjestiti podatke o hardveru korisnikovog računara i na taj način je potrebna jedinstvena datoteka za svaki računar.

Kao i kod šifri, iz aplikacije je moguće ukloniti dio koda koji provjerava ispravnost registracione datoteke. Kako bi se to otežalo, provjera ispravnosti datoteke se ugrađuje u više mjesta u programu i informacije u datoteci se koriste za šifriranje dijelova izvršnog koda.

3. Metode za probijanje softverske zaštite

Postoje dva glavna pristupa probijanju softverske zaštite.

Prvi pristup je pokretanje programa unutar nekog alata za ispravljanje pogrešaka u kodu (debugging), a drugi pristup je reverzno prevođenje programa iz izvršnog koda natrag u asemblerski kod (disassembling) ili neki od viših programskih jezika (decompiling).

3.1. Debugging

Alati za ispravljanje grešaka u kodu omogućuju interaktivno pokretanje programa. Program za debugging može zaustaviti izvršavanje programa između svake dvije instrukcije. Nakon što se program zaustavi, programer može vidjeti u kojem dijelu izvornog koda programa se proces trenutno nalazi. Osim dijela izvornog koda, programer u svakom trenutku može vidjeti sadržaj radne memorije, sadržaj procesorskih registara, podatke koji se zapisuju u datoteke, informacije o programskim nitima i mnoge druge informacije. Takve alate koriste programeri za vrijeme razvoja softvera jer im omogućuju detaljan uvid u rad programa i uklanjanje pogrešaka u programiranju. Sličan alat je ugrađen u svaki softverski paket koji je namijenjen razvoju aplikacija, poput Microsoft Visual Studia ili Borland C++ Buildera.

Alati za debugging se često rade i kao samostalne aplikacije i takvi alati su obično puno moćniji. Najbolji debugging alati mogu komunicirati direktno s jezgrom operativnog sistema i zaobići njegove pozive. Na taj način su puno stabilniji u radu, mogu prepoznati više informacija u stanju sistema te čak omogućuju analizu hardverskih upravljačkih programa.

Korištenjem takvih alata, pirati mogu zaustaviti program u trenutku kad on zahtijeva autorizaciju korisnika. Nakon što se program zaustavi, moguće je vidjeti dio koda koji je odgovoran za provjeru autorizacije i izmijeniti ga tako da prihvaća svaku šifru.

Najpoznatiji alat za debugging je SoftIce kompanije Compuware, program može uhvatiti mnoge pogreške koje drugim alatima promaknu, pa čak prikazati pogrešku nakon blokiranja Windows operativnog sistema.

3.2. Disassembling

Disassembling i decompiling su postupci koji izvršni kod programa pretvaraju u izvorni kod, tako da ga je moguće analizirati. Alati za disassembling pretvaraju izvršni binarni kod u asemblerski kod. Kako je asembler jezik koji se direktno (jedna instrukcija asemblera odgovara jednoj binarnoj instrukciji) prevodi u binarni kod, disassembling alati mogu prevesti svaku aplikaciju, bez obzira na programski jezik u kojem je ona izvorno napisana.

Najbolji alati mogu generirati strukturiran kod koji je moguće lako pratiti i analizirati pa čak kreiraju i komentare za lakše snalaženje u programu. Najčešće korišten alat za disassembling je IDA Pro kompanije NuMega.

3.3. Decompiling

Alati za decompiling pretvaraju izvršni binarni kod programa u izvorni kod nekog od programskog jezika većeg nivoa kao što je C++ ili Java. Ovi alati mogu prevesti samo izvršni kod nekog određenog jezika za koji je taj alat razvijen. Na primjer, alat napravljen za programski jezik C++ ne može prevesti program napisan u Javi. Najveća prednost ovih alata nad alatima za disassembling je činjenica da većina programera i pirata puno bolje poznaje jezike većeg nivoa od asemblera i takav kod im je puno lakše analizirati.

4. Komercijalni softveri

4.1. FlexLM

FlexLM je softver kompanije Macrovision. FlexLM je izuzetno kompliciran sistem nadgledanja softverskih licenci i predstavlja danas najraširenije komercijalno rješenje za zaštitu softvera. FlexLM softver uključuje zaštitu izvršnog koda aplikacije, kontrolu ispravnosti registracionih datoteka na disku, softver za implementaciju na poslužiteljima koji nadziru korištenje licenci u lokalnoj mreži, sistem za lako obnavljanje licenci i centralnu administraciju cijelog sistema.

Za zaštitu samih aplikacija FlexLM softver koristi enkripciju izvršnog koda aplikacije, enkriptovanu registracionu datoteku koja zavisi od hardvera korisničkog računara, kontrolu servera kojom se neovlaštenim korisnicima zabranjuje pokretanje mrežnih aplikacija, a u nekim verzijama se koristi i dongle.

Sistem nadgleda korištenje softvera i ispravnost licenci, a moguće je izabrati različite metode licenciranja softvera. Softver podržava nekoliko različitih sistema licenciranja, a najpoznatiji su:

- ***Node locked*** – softver je moguće pokrenuti samo na računaru za koji je licenca izdana;
- ***User based*** – softver može pokrenuti samo korisnik koji je vlasnik licence;
- ***Site licensing*** – softver mogu pokrenuti svi korisnici na određenoj lokaciji;
- ***Floating license*** – softver mogu pokrenuti svi korisnici u lokalnoj mreži, ali postoji ograničenje na maksimalni broj korisnika koji mogu istovremeno raditi sa programom.

Softver podržava i licenciranje prema vremenu utrošenom u radu sa zaštićenim programom. FlexLM server prati vrijeme provedeno u radu s softverom za sve korisnike u mreži te se na osnovu toga korisniku isporučuje račun.

FlexLM sistem se sastoji od 4 glavna dijela i to su:

- aplikacijski softver;
- registraciona datoteka;
- server za kontrolu licenci;
- server proizvođača softvera.

Aplikacijski softver je softver koji je zaštićen FlexLM sistemom. Softver ima ugrađene funkcije koje provjeravaju postojanje i ispravnost registracione datoteke. Datoteka se mora nalaziti u određenom direktoriju, a u datoteci je naveden naziv i verzija softvera, podaci o vlasniku licence te period trajanja licence.

U registracionoj datoteci je zapisan i put do servera za kontrolu licenci čiji je zadatak kontrola licenci u lokalnoj mreži. Aplikacija se spaja na tserver i od njega saznaje lokaciju softverske kompanije. Server proizvođača softvera nadgleda korištenje licenci za sve korisnike tog proizvođača.

S obzirom da je FlexLM jako popularan softver koji koriste mnogi proizvođači softvera za zaštitu svojih aplikacija, moguće je da korisnik na svom računaru ima više aplikacija od različitih proizvođača, a koje su sve zaštićene FlexLM sistemom. U tom slučaju korisnik mora instalirati FlexLM softver samo jednom i taj sistem upravlja licencama za sve aplikacije za koji je odgovoran.

Iako je ovaj sistem poznat i koriste ga stotine proizvođača softvera širom svijeta, njegova komplicirana zaštita je ipak probijena i vrlo brzo se pojavljuju piratske verzije softvera koji koristi ovakvu zaštitu.

FlexLM softver se instalira kao dijeljena DLL biblioteka funkcija i aplikacija koja je zaštićena FlexLM sistemom komunicira s tom bibliotekom svaki puta kada provjerava ispravnost licenci. Takva implementacija ustvari olakšava probijanje zaštite.

4.2. HASP

HASP predstavlja sistem za zaštitu aplikacija koji primjenjuje dongle ključeve. Softver podržava dongle ključeve koji se spajaju na skoro sve moguće komunikacijske portove u računaru. Postoje izvedbe donglea za paralelni port, USB port, PCI sabirnicu te kao PCMCIA kartica.

Svi dongle uređaji imaju vlastitu EPROM memoriju i procesor koji vrši enkripciju i dekripciju podatka u memoriji. Kako je gotovo nemoguće pročitati podatke iz memorije bez poznavanja ispravnog ključa, postiže se nivo zaštite koji je jako teško razbiti. Dongle ima dovoljno mjesta u memoriji za 112 različitih licenci za, a s istim uređajem je moguće štititi aplikacije na Windows, Unix i Macintosh platformama. Softver koji se isporučuje s ovim uređajima ima dva načina rada. Korisničku aplikaciju je moguće zaštititi korištenjem takozvane omotnice ili promjenom izvornog koda aplikacije. Omotnica štiti korisničku aplikaciju tako da enkriptuje izvršnu datoteku i doda dio koda koji omogućuje izvršavanje aplikacije. Taj način zaštite je jednostavniji jer ne zahtijeva promjenu izvornog koda aplikacije, ali je manje siguran. Drugi način zaštite je da se prilikom programiranja u softver ugrade posebne funkcije iz biblioteke koja se isporučuje s Aladdin softverom.

4.3. PC Guard

PC Guard je softver kompanije Sofpro, namijenjen zaštiti softvera i kontroli licenci na Windows platformama. PC Guard štiti softver pomoću omotnice ili API poziva u izvornom kodu aplikacije, slično kao i kod HASP softvera. Zaštita se postiže enkripcijom izvršnog koda i detekcijom pokušaja analize koda.

Za autentifikaciju korisnika PC Guard koristi šifru koja zavisi od hardvera korisničkog računara. PC Guard može detektovati registracione oznake hard diska, CD/DVD-a, BIOS-a, Ethernet kartice, procesora i operativnog sistema na korisničkom računaru i te informacije koristi za kontrolu korisnikove šifre.

Postoji posebna verzija PC Guard softvera za klasične Windows aplikacije te posebna verzija za .NET aplikacije. Rad s programom PC Guard je veoma jednostavan. Aplikaciju je moguće zaštiti tako da se upiše ime aplikacije, njena registraciona oznaka te izabere način licenciranja.

Zaključak

Zaštita softvera (programskog koda) predstavlja veliki problem u modernom društvu. To je proces povezan ne samo s informatičkom tehnologijom, već i sa zakonodavstvom i ekonomijom.

Moderni sistemi za kontrolu softverskih licenci su narasli u velike, distribuirane, mrežne sisteme kojima je zadaća ne samo borba protiv softverskog piratstva, već i stvaranje infrastrukture koja će omogućiti efikasniji način naplate korištenja softvera. U ne tako dalekoj budućnosti se softver vjerojatno više neće plaćati unaprijed kao danas, već prema vremenu korištenja, poput naplate struje ili vode, a sistemi za kontrolu licenci to moraju omogućiti.

S druge strane, sistemi za zaštitu softvera još uvijek ne uspijevaju obaviti svoju primarnu zadaću, a to je zaštita softvera od izrade piratskih verzija. Iako je na Internetu moguće pronaći kompanije koje za svoje sisteme zaštite tvrde da nikad nisu probijeni, praksa pokazuje da je razbijanje bilo kakve softverske zaštite samo pitanje vremena. Često je isplativije u softver ugraditi vlastiti mehanizam za zaštitu, nego koristiti komercijalni. Iako komercijalni sistem vjerojatno ima bolje algoritme za zaštitu, iskusni pirati imaju uhodane procedure za njihovo probijanje. Odluka o ugradnji zaštite u softver je uvijek ekonomska i potrebno je naći kompromis između cijene ugradnje takve zaštite i potencijalnih gubitaka od prodaje softvera ako se zaštita ne ugradi

. Rijetko se isplati ulagati u najskuplje metode zaštite softvera jer svaka zaštita će s vremenom biti probijena i bitno je samo da se to odgodi dovoljno dugo kako bi proizvođač softvera ipak ostvario zadovoljavajući profit od prodaje softvera.

Literatura

1. Boško Rodić, Goran Đorđević, *Da li ste sigurni da ste bezbedni*, Produktivnost AD, Beograd, 2004.

2. Douwe Croft, *Data protection laws in the European Union*, Brussels: Federation of European Direct Marketing, 2005.

3. Boško Rodić, Radovan Marjanović, Dragutin Bjelovuk, *Poslovna informatika I*, VŠPM „PRIMUS", Gradiška, 2011.

4. Boško Rodić, Radovan Marjanović, Dragutin Bjelovuk, *Poslovna informatika II*, VŠPM „PRIMUS", Gradiška, 2011.

5. Mike Pastore, Emmett Dulaney, *Security+: studijski priručnik*, Kompjuter biblioteka, Čačak, 2007.

6. Goran Jović, *Zaštita informacionih sistema*, Zbornik radova ISSN 0352-6542, Br. 17, 2007.

7. Slobodan P. Petrović, Zaštita podataka u automatizovanim informacionim sistemima, Naučna knjiga, Beograd, 1986.

8. Dragutin Bjelovuk, Diplomski rad: „*Maliciozni softver – detekcija i odbrana*", Gradiška, 2008.