Mjere zaštite računalnih igara od piratstva i varanja

Kinkela, Dominik

Sveučilište u Rijeci
Fakultet informatike i digitalnih tehnologija
Sveučilišni prijediplomski studij Informatika

Dominik Kinkela

Mjere zaštite računalnih igara od piratstva i varanja

Diplomski rad

Mentor: doc. dr. sc. Vedran Miletić

Rijeka, 11. prosinca 2023.

Sveučilište u Rijeci
Fakultet informatike
i digitalnih tehnologija
www.inf.uniri.hr

Rijeka, 1. veljače 2023.

Zadatak za diplomski rad

Pristupnik: Dominik Kinkela

Naziv diplomskog rada: Mjere zaštite računalnih igara od piratstva i varanja

Naziv diplomskog rada na eng. jeziku: Piracy and cheating prevention measures in PC games

Sadržaj zadatka:

Računalne igre od samih početaka koriste brojne metode prevencije neovlaštenog kopiranja (piratstva), a od doba omasovljenja višeigračkih naslova koriste i metode prevencije varanja. Kako vremenom napreduje znanje koje omogućuje zaobilaženje postojećih metoda prevencije, tako se razvijaju i nove metode prevencije koje je teže zaobići. Cilj rada je dati pregled povijesnog razvoja metoda i na primjeru vlastite aplikacije prikazati način rada metode prevencije neovlaštenog kopiranja i njeno zaobilaženje.

Mentor:

Doc. dr. sc. Vedran Miletić

Voditeljica za diplomske radove:

Prof. dr. sc. Ana Meštrović

Komentor:

Zadatak preuzet: 1. veljače 2023.

(potpis pristupnika)

Adresa: **Radmile Matejčić 2**
51000 Rijeka, Hrvatska

Sažetak

U ovom se radu istražuje isprepleteni svijet softvera i hardvera u pitanju postupaka protiv neovlaštene izmjene izvornog koda i njegove namjene, varanja, razvoja DRM-a te načinima sprječavanja štetnog ponašanja. Stalni izazovi u zaštiti digitalne imovine od manipulacije i prijevare u brojnim sektorima zahtijevaju konstantno poboljšanje raznih sustava zaštite. Ovo istraživanje namjerava pružiti cjelovito tehničko stajalište o održavanju digitalnog integriteta i promicanju pravilnog ponašanja i poštene igre u digitalnim okruženjima ispitivanjem tehnologija, metoda i pravnih posljedica.

Ključne riječi: drm; denuvo; anti-cheat; anti-tamper; crack; licenca; piratstvo

Sadržaj

Poglavlje 1

Uvod

U eri u kojoj dominira digitalna imovina, integritet digitalnog sadržaja i aplikacija ostaje najveća briga. Uporna potraga za nezakonitim promjenama (petljanje ili tampering) i nepošteno ponašanje (varanje ili cheating) u digitalnoj sferi stvorila je složen problem koji zahtijeva kreativna rješenja. Kao odgovor na te opasnosti, područje metoda i softvera protiv petljanja i varanja postalo je ključno za prevenciju nepoželjnog i nepredviđenog ponašanja.

U ovom radu analizirati ćemo cjelokupnu sferu različitih tematika poput upravljanja digitalnim pravima, očuvanja integriteta digitalnog sadržaja, metoda probijanja hardverskih i softverskih komponenti i pravnih i moralnih posljedica istog. Kroz analizu legalnih i etičkih aspekata upravljanja digitalnim pravima obraditi ćemo legalno korištenje sadržaja u različite svrhe te utjecaj na korisnika i njegovo iskustvo. Pružiti ćemo pregled razvitka tehnologije kroz povijest do stanja kakvog poznajemo danas. Navesti ćemo razne metode zaštite protiv petljanja i zašto su one bitne te spomenuti najpoznatije implementacije navedenog i kako to utječe na ukupni opseg video igara i ostalog sadržaja. Zaronit ćemo u izazove koje predstavlja zaštita od varanja u video igrama i načinima funkcioniranja nekih od najpoznatijih više i manje sofisticiranih varki. Navigirati ćemo legalna i etička razmatranja u vezi jailbreakanja i krekiranja (cracking) te primjetiti koliko su metode lagane za izvedbu, a koliko i zašto su korisne i vrijedne rizika. U sklopu toga ćemo demonstrirati korištenje alata potrebnih za izvedbu navedenog te objasniti način njihove provedbe. Analizirati ćemo različite stavove po pitanju promjene sadržaja koje rezultiraju njegovim monetarnim iskorištavanjem. Osvrnut ćemo se na utjecaj raznih odluka na pojedinca te navesti nekolicinu slučajeva velikog razdora između kompanija i društva kao cjeline kroz preispitivanje tanke linije između legalno dozvoljenog i moralno ispravnog načina konzumiranja sadržaja.

Poglavlje 2

DRM

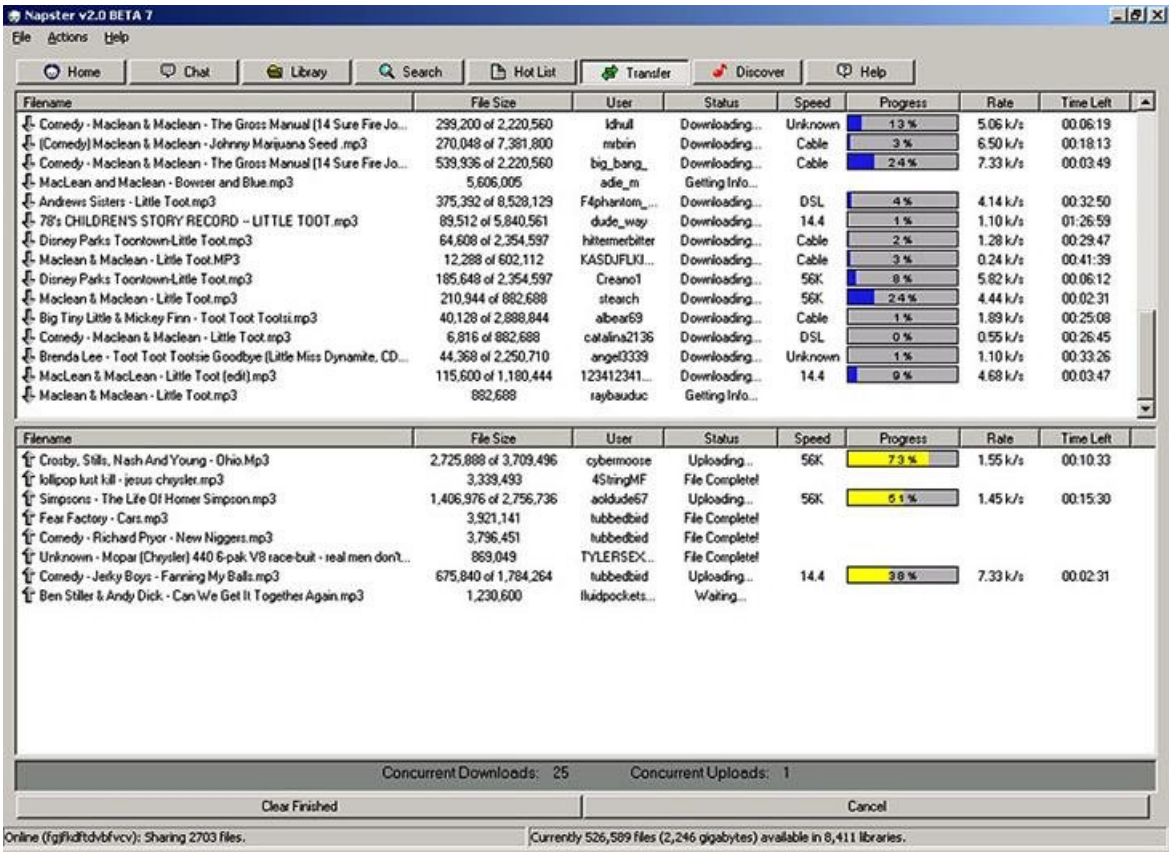
Upravljanje digitalnim pravima (digital rights management, DRM) je upravljanje legalnog pristupa digitalnom sadržaju [1, 2]. Ono koristi razne alate i tehnološke mjere zaštite poput kontrole pristupa. Bitno je pratiti podrijetlo ovih postupaka prije nego što zaronimo u detalje i zamršenost modernih tehnika protiv petljanja i varanja. Razvoj upravljanja digitalnim pravima kroz povijest pruža dobar kontekst za naše istraživanje. U suštini, DRM radi na zaštiti digitalne imovine od neovlaštene distribucije, pristupa i modifikacije. DRM taktika i tehnologija doživjele su brzu promjenu kao rezultat konstantne utrke u naoružanju između dobavljača sadržaja i neovlaštenih korisnika. Poznavanje ovih alata postavlja temelje za razumijevanje suvremenih tehnika protiv petljanja i varanja.

2.1 Povijest

Potreba za DRM-om može se pratiti još od ranih faza distribucije digitalnih informacija, kada je Internet omogućio brzo dijeljenje i kopiranje digitalnih datoteka. Distribucija i korištenje medija doživjelo je promjenu paradigme gdje korisnici sada imaju još neviđeni pristup digitalnim informacijama. Razvoj interneta i bitnije napredak u brzini veza znatno je olakšao interakciju sa sadržajem, no ova novootkrivena lakoća također je izazvala rastući problem: digitalno piratstvo.

Prava intelektualnog vlasništva pružatelja sadržaja ozbiljno su ugrožena porastom platformi za dijeljenje datoteka i peer-to-peer (P2P) mreža. Neviđene količine sadržaja ilegalno su se kopirale i distribuirale, što je muzičke izdavačke kuće koštalo značajne količine novca. Od peer-to-peer mreža za dijeljenje datoteka najpoznatiji je Napster [3].

Predstavljen 1999. godine, nudio je jednostavnu platformu za distribuciju glazbenih datoteka. Dok je tehnologija donijela novu eru pristupačnosti i otkrivanja glazbe, također je omogućila široko rasprostranjeno kršenje autorskih prava u dosad neviđenim razmjerima. Korisnici mogu lako pronaći i preuzeti glazbene zapise iz biblioteka drugih ljudi, često bez plaćanja glazbenika i nositelja autorskih prava. Sadržaj je često bio promijenjen i popunjen raznim virusnim programima, no ta činjenica nije obeshrabrila preuzimanja ljudi koji su sada taj sadržaj mogli nabaviti



Slika 2.1: Napster

bez plaćanja. Provedba autorskih prava bila je ozbiljno otežana Napsterovom decentraliziranom arhitekturom, koja je držala datoteke na osobnim računalima korisnika, a ne na centraliziranim poslužiteljima. Napster je bio tužen od strane raznih umjetnika i izdavačkih kuća te je naposlijetku zatvoren u 2001. godini zbog pritiska sudskih naloga. Tada je već bilo prekasno i način nabave sadržaja postao je previše popularan, dok su izdavačke kuće trpile značajne financijske gubitke. Primijetile su hitnu potrebu zaštite svog intelektualnog vlasništva na internetu. Ovaj pravni spor poslužio je kao katalizator za razvoj DRM-a ističući nužnost metoda zaštite digitalnog sadržaja. Tu je označena prekretnica između svijeta bez DRM-a i onoga sa neprekidnim napretkom DRM-a kakvoga poznajemo danas.

Prva generacija DRM rješenja pojavila se kao odgovor na hitnu potražnju za zaštitom sadržaja nakon preokreta digitalnog piratstva. Ti rani DRM sustavi uglavnom su koristili zaštitu lozinkom i tehnike šifriranja. Ističe se sustav kodiranja sadržaja (content scrambling system, CSS) za DVD što zahtijeva pohranu ključa za dešifriranje na uređaju za reprodukciju. Ova enkripcija služi kao prepreka protiv neovlaštenog kopiranja, međutim, pokazalo se da CSS nije siguran protiv odlučnih napadača. Obrnutim inženjeringom (reverse engineering) došlo se do potrebnih ključeva, omogućujući stvaranje softvera poput DeCSS, koji može dešifrirati i kopirati sadržaj DVD-a. Slično tome, u području zaštite softvera i dokumenata, DRM koji se temeljio na lozinci je bio vrlo popularan. Korisnici su morali unijeti lozinku za pristup zaštićenim datotekama ili aplikacijama. Iako je ovaj pristup dodao dodatni sloj sigurnosti, bio je ranjiv na

napade uzastopnim pokušavanjem (brute force napade) i druge tehnike probijanja lozinki. Unatoč ograničenjima i ranjivostima prve generacije DRM-a, oni su označili važan korak u evoluciji zaštite sadržaja. Ovi početni napori pokazali su predanost industrije za zaštitu digitalne imovine i postavili temelje za sofisticiranije DRM tehnologije u godinama koje dolaze.

U narednim generacijama razvitak DRM-a se pomiče prema specifičnijim rješenjima za svaku platformu. Kreirajući zatvorene ekosisteme i vlastite formate dobili su veću kontrolu nad sadržajem, no također su ograničili slobodu i lakoću korištenja za legitimnog korisnika. Apple-ov FairPlay, predstavljen 2003. s iTunes Storeom, imao je cilj zaštititi kupnju digitalne glazbe. Glazba kupljena u iTunes Storeu kodirana je u Appleovom vlasničkom formatu Advanced Audio Coding (Advanced Audio Coding, AAC), a DRM ograničenja se primjenjuju kako bi se reprodukcija ograničila na ovlaštene Apple uređaje. Ovaj pristup stvorio je zatvoreni ekosustav u kojem se u sadržaju može uživati samo na Appleovim proizvodima, čime se obeshrabruje neovlaštena distribucija, ali u isto vrijeme znatno limitira doseg sadržaja. Industrija igara također je usvojila noviji pristup DRM-u. Konzole za video igre, kao što su Sonyjev PlayStation i Microsoftov Xbox, implementirale su DRM mjere koje zahtijevaju igranje igara isključivo na određenom hardveru. Svaka igra povezana je s jedinstvenim serijskim brojem ili ključem za provjeru autentičnosti, a konzola komunicira s poslužiteljem za provjeru autentičnosti kako bi provjerila legitimnost kopija igre. Ovaj pristup ima za cilj borbu protiv piratstva igara i osiguravanje da su korisnici kupili legalne kopije za svoje konzole. O načinima zaobilaženja tih ograničenja poput bušenja rupe kroz konzolu kako bi se eliminirala sigurnosna igla, više ćemo pričati u kasnijim poglavljima. U području digitalnih dokumenata i izdavaštva, Adobe-ov format prenosivog dokumenta (Portable Document Format, PDF) postao je popularan format za distribuciju e-knjiga, izvješća i drugih dokumenata. Adobe je uveo DRM mehanizam za PDF datoteke, dopuštajući kreatorima dokumenata da ograniče prava pristupa, ispisa i kopiranja. Korisnici se moraju autentificirati ili unijeti lozinku za pristup PDF datotekama zaštićenim DRM-om. Iako ova metoda pruža određenu razinu sigurnosti dokumenata, također se suočava s izazovima odlučnih napadača koji su razvili alate za uklanjanje DRM ograničenja iz PDF datoteka. Zbog mnogo kritika DRM se dalje razvija sa boljom interoperabilnosti i iskustvu za korisnika.

Microsoft je odigrao središnju ulogu u razvoju nove generacije DRM-a s Windows Media DRM tehnologijom. Ovaj DRM omogućio je pružateljima sadržaja da ponude zaštićeni audio i video sadržaj koji se može reproducirati na velikom broju uređaja, ne ograničavajući se samo na Microsoftove proizvode. Windows Media DRM uveo je koncept “garantirane reprodukcije”, koji je naglasio kompatibilnost na različitim platformama i uređajima. Ovaj pristup rješava neka od ograničenja prethodnih DRM implementacija i pruža raznolikiji softver i ekosustav uređaja. Prepoznajući potrebu za standardima za cijelu industriju, organizacije kao što su World Wide Web Consortium (W3C) i Digital Living Network Alliance (DLNA) počele su razvijati otvorene standarde za DRM. Na primjer, specifikacija šifriranih medijskih ekstenzija (Encrypted Media Extensions, EME) nastojala je pružiti zajednički okvir za DRM u web preglednicima.

Mozilla kao neprofitabilna organizacija koja se zagovara za otvorenost standarda i interneta, predložila je alternativni način implementacije DRM-a korištenjem metode vodenog žiga (watermarking) [4]. Watermarking podrazumijeva ugrađivanje neprimjetnih oznaka u sam sadržaj, pomoću kojih se kasnije može otkriti izvor neovlađtenih kopija, dok se ne ograničava pristup sadržaju. Cilj ovih standarda je postići ravnotežu između zaštite sadržaja i pogodnosti korisnika dopuštajući reprodukciju sadržaja zaštićenog DRM-om na različitim preglednicima i platformama. Što se tiče digitalne glazbe, prijelaz s upravljanja digitalnim pravima na glazbu bez DRM-a bio je vrlo bitan. Velike glazbene kuće, uključujući EMI i Warner Music, počele su nuditi glazbu bez DRM-a putem internetskih glazbenih trgovina kao što je iTunes. Ova promjena je potaknuta činjenicom da ograničenja DRM-a često frustriraju potrošače i da se mogu zaobići. Dopuštajući korisnicima kupnju glazbe bez DRM-a, glazbena se industrija prilagodila promjenjivim preferencijama potrošača. Vidimo da su DRM rješenja počela smanjivati razliku između zaštite sadržaja i pogodnosti korisnika, međutim, tekuća rasprava o utjecaju DRM-a na korisnička prava i uporni izazov zaobilaženja nastavio je oblikovati njegovu evoluciju.

2.2 Kritike

Jedna od glavnih kritika DRM-a odnosi se na njegov utjecaj na iskustvo i udobnost korisnika. Ograničenja DRM-a često rezultiraju ograničenjima izbora pri načinu i mjestu pristupa upotrebe digitalnog sadržaja [5]. Korisnici nisu mogli prenositi sadržaj između svojih uređaja, dijeliti sadržaj s obitelji ili prijateljima ili stvarati sigurnosne kopije svojih digitalnih biblioteka. Ta su ograničenja izazvala frustraciju potrošača i potaknula potražnju za rješenjima koja su jednostavnija za korištenje. Iako je DRM namijenjen sprječavanju digitalnog piratstva, često postaje meta za pojedince i zajednice hakera da ga namjerno zaobiđu. Pojavili su se sofisticirani alati i tehnike za zaobilaženje DRM zaštite, omogućujući uklanjanje ograničenja i slobodno dijeljenje sadržaja zaštićenog autorskim pravima. Učestala borba između podupiratelja DRM-a i ostatka zajednice istaknula je stalni izazov balansiranja zaštite i pristupa. DRM također postavlja pitanja o ograničenjima poštene upotrebe i digitalnih prava. Neki ljudi tvrde da DRM neopravdano ograničava mogućnost korisnika da koriste svoja zakonska prava, kao što je izrada sigurnosnih kopija medija koje su kupili ili korištenje sadržaja u obrazovne svrhe. Primjeri poput posjedovanja instance sadržaja te naizgled ilegalna nabava na drugome mjestu sivo su područje te se većina slaže dok je sadržaj legalno nabavljen druge instance nabave potencijalno upitne prirode nisu važne. Pravne bitke, posebice one koje uključuju zakon o autorskim pravima u digitalnom tisućljeću (DMCA), dodatno su istaknule napetost između zaštite sadržaja i prava korisnika. Postoje i problemi s kompatibilnošću i povezanosti sa pružateljem usluga. Korisnici koji su uložili svoje vrijeme i novac u određeni sustav nalaze se vezani za taj sustav, platformu ili uređaj. To je ograničilo njihovu mogućnost da se prebace na alternativne usluge ili hardver, pridonoseći zabrinutosti o anti-konkurentskom ponašanju davatelja sadržaja i tehnoloških tvrtki. Kritika od strane korisnika protiv DRM-a ilustrirana je događajima poput “Sony rootkit skandala” [6].

Godine 2005. Sony BMG distribuirao je glazbene CD-e koji su sadržavali DRM sa rootkit-om. Ne samo da to predstavlja sigurnosni rizik za računala korisnika, već također uzrokuje katastrofu u odnosima s javnošću za Sony. Slučaj naglašava potencijalne opasnosti DRM-a i potrebu za transparentnošću u njegovoj primjeni, no također povećava nepovjerenje i nezadovoljstvo prema DRM-u, moralno olakšavajući postupke zlonamjernih pojedinaca u njihovoj borbi protiv DRM-a.

2.3 Napredak

Posljednjih godina moderna DRM rješenja su promijenila fokus, osobito s rastom usluga video/glazbenog strujanja (streaming) i servisa temeljenih na radu na oblaku. Streaming platforme, kao što su Netflix, Amazon Prime Video, Max za video i Spotify, SoundCloud, Tidal za glazbu, postale su dominantne u medijskom krajoliku. Moderni DRM sustavi prilagodili su se novim potrebama korištenjem sigurnih protokola streamanja. Ovi protokoli osiguravaju da sadržaj ostane šifriran tijekom prijenosa i da se može dešifrirati samo na ovlaštenim uređajima. Ovaj pristup smanjuje rizik od presretanja sadržaja i neovlaštene distribucije tijekom reprodukcije. DRM rješenja temeljena na oblaku uvela su mehanizme dinamičke enkripcije koji kontinuirano mijenjaju ključ šifriranja tijekom strujanja. Ovaj fleksibilni pristup napadačima izuzetno otežava presretanje i dešifriranje sadržaja. Dodatno, praćenje u stvarnom vremenu koristi se za otkrivanje neobičnih trendova ili potencijalnih kršenja korisničkog dogovora, što omogućuje brzi odgovor na prijetnje. Ove mjere ne samo da štite sadržaj tijekom distribucije, već osiguravaju i sigurnost korisničkih podataka. Moderne DRM modele karakteriziraju fleksibilne mogućnosti licenciranja. Korisnici se često pretplate na usluge strujanja s više mogućnosti pristupa, što im omogućuje da odaberu kako žele pristupiti sadržaju i koristiti onoliko koliko se osjećaju sigurno. Primjer te opcije može biti izvanmrežno preuzimanje s vremenski ograničenim pristupom, kako se radilo i u prošlosti, dodajući element pogodnosti dok i dalje sadržaj održava DRM zaštitu. Modeli licenciranja i dalje imaju za cilj postići ravnotežu između dostupnosti i zaštite. Kako bi spriječili piratstvo i pratili neovlaštenu distribuciju, moderni DRM sustavi često koriste napredniju tehnologije digitalnog otiska prsta i vodenog žiga. Ove tehnike ugrađuju jedinstvene identifikatore u sam sadržaj, omogućujući praćenje podrijetla neovlaštenih kopija. Iako nisu savršene, ove mjere služe kao sredstvo odvraćanja i pomažu u prepoznavanju potencijalnih prekršaja. Mehanizmi provjere autentičnosti također su postali moćniji u modernom DRM-u. Korisnici se moraju autentificirati kroz provjeru s više faktora, osiguravajući da sadržaju mogu pristupiti samo ovlaštene osobe. Biometrijska autentifikacija, kao što je prepoznavanje otiska prsta ili lica, ili dvofaktorska autentifikacija poput tokena, dodaje dodatni sloj sigurnosti.

Pomak prema novim oblicima zaštite uzrok je današnjeg konstantno promijenjivog okoliša. Moderna DRM rješenja prioritet daju praktičnosti korisnika uz održavanje jake zaštite. Ova poboljšanja omogućila su pružateljima sadržaja da isporuče kvalitetna i sigurna iskustva stre-

aminga, poboljšana dinamičkom enkripcijom, nadzorom u stvarnom vremenu i fleksibilnim modelima licenciranja. Zbog potrebe za većom interoperabilnošću, krajolik DRM-a doživio je pojavu standardizacije i otvorenih DRM inicijativa. Ove inicijative uključuju Encrypted Media Extensions (EME) i World Wide Web Consortiuma (W3C), čiji je cilj stvoriti zajednički DRM okvir za web preglednike, i Digital Living Network Alliance (DLNA) s fokusom na multimedijski streaming i interoperabilnost uređaja. Grupe koje zagovaraju otvorene DRM inicijative su grupe poput Electronic Frontier Foundation (EFF) i GOG (bivši Good Old Games) koje daju prioritet korisničkim pravima i distribuciji sadržaja bez DRM-a.

2.4 Današnji DRM (knjige, muzika, video, softver)

Fokusirajući se na konkretnije načine zaštite i razine uspješnosti vrijedi spomenuti knjige. E-knjige sada čine značajan dio izdavačke industrije, a DRM igra središnju ulogu u zaštiti e-knjiga od neovlaštene distribucije. Veliki prodavači e-knjiga poput Amazona i Barnes & Noble koriste vlastite DRM sustave za šifriranje i zaštitu preuzimanja e-knjiga. Ovi sustavi povezuju e-knjige s određenim uređajima ili aplikacijama, osiguravajući da korisnici mogu čitati samo kupljene knjige na ovlaštenim platformama. Velik broj ljudi zagovara da bi znanje trebalo biti svima dostupno i besplatno, te u taj argument osim znanstvenih radova vole uključiti i knjige. Tu se stvara etički i moralni diskors gdje se cijeli argument ne može primijeniti za svim primjerima istovremeno već se razlikuje od primjera do primjera. Zbog nekolicine razloga poput takvog stava, same jednostavnosti nabave i upoznanosti generalne populacije sa pdf formatom ogromna količina knjiga i radova biva piratizirana. Ono što također olakšava piratstvo knjiga je činjenica da je sama pohrana sadržaja vrlo jednostavna zbog malih veličina zapisanih datoteka, a te datoteke potom gotovo svaki uređaj može pročitati u tekstualnom ili pdf formatu.

Kao što je prethodno spomenuto glazba i video se često zaštićuje šifriranjem tijekom prijenosa dinamičkom enkripcijom, što otežava korisnicima blokiranje, kopiranje i dijeljenje sadržaja zaštićenog autorskim pravima. Ti algoritmi rade u stvarnom vremenu, a koriste se i hardverske limitacije. Modeli licenciranja često određuju razinu korisničkog pristupa, a izvanmrežna preuzimanja često dolaze s vremenski ograničenim pristupom kako bi se osigurala zaštita sadržaja.

Situacija sa zaštitom softvera je malo drugačija, zato jer mnogi softverski proizvodi zahtijevaju online aktivaciju i povremene provjere licenci. Na taj način se osigurava da se koriste samo legitimne kopije proizvoda. Registracijom i kupnjom korisnik može dobiti aktivacijski ključ koji potom može biti tradicionalno vremenski neograničen, implicirajući jednokratnu kupnju ili izdavač softvera može izdati licencu za korištenje baziranu na pretplati. Te provjere najčešće zahtijevaju online konekciju, nekada samo pri prvotnoj provjeri aktivacijskog koda, a nekada pri svakom korištenju proizvoda. Ograničenja također mogu biti povezana sa određenom hardverskom komponentom koja je prisutna samo na korisničkom uređaju. DRM softver uključuje različite mjere protiv piratstva za sprječavanje neovlaštenog kopiranja i distribucije. Ove

mjere mogu uključivati maskiranje koda, enkripciju izvršnih datoteka i tehnike za otkrivanje neovlaštenog mijenjanja softvera ili neovlaštenih izmjena (anti-tampering). Mnogi programeri softvera koriste DRM za izradu probnih ili demo verzija svog softvera. Te verzije imaju ograničene značajke ili vremenski ograničen pristup, potičući korisnike da kupe punu verziju kako bi otključali sve značajke, dok DRM osigurava da korisnici ne mogu jednostavno zaobići ograničenja kako bi dobili puni pristup. Međutim sa nizom postupaka za borbu protiv piratstva i neovlaštenih izmjena, pretjerano restriktivni DRM može omesti legitimne korisnike i dovesti do negativnih korisničkih iskustava, barem tako misle neki kritičari. Od usporavanja programa i računala do nepotrebnih i kompliciranih provjera i instalacijskih postupaka, ima pregršt nadodanih problema koje ometaju legitimne korisnike. Osim toga, neki su zabrinuti za privatnost i prikupljanje podataka pri online postupku aktivacije.

Poglavlje 3

Anti-tampering

Anti-tampering mehanizmi ili mehanizmi protiv petljanja su mehanizmi za zaštitu od neovlaštenih promjena kritičnih komponenata softverskih sigurnosnih sustava [7]. Dizajnirani su za otkrivanje, sprječavanje i zaštitu protiv napada na hardversku ili digitalnu imovinu. U kontekstu rada pričati ćemo o softverskom aspektu anti-tampering mehanizama no valja spomenuti i hardverski. Hardverske mjere zaštite često uključuju pakiranje i naljepnice zaštićene od otvaranja koje ukazuju na pokušaje prijevare. Hardverski sigurnosni moduli (HSM) i sigurni elementi koriste jake kriptografske mehanizme za zaštitu osjetljivih podataka i koda. Siguran proces pokretanja na pouzdanom hardveru, temelj je povjerenja za osiguravanje autentičnosti pokretanog uređaja i osiguravanje autentičnosti i neizmijenjenosti pokretanog koda. Mehanizmi fizičke sigurnosti, kao što su kućišta i senzori zaštićeni od neovlaštenog otvaranja, mogu oglasiti alarm ili izbrisati podatke ako se otkrije da je kučište ugroženo od strane neovlaštenih aktera. Zaštita od neovlaštenog pristupa temeljena na hardveru ključna je u industrijama kao što su zrakoplovstvo, obrana i kritična infrastruktura, gdje je fizički integritet uređaja najvažniji za cjelokupnu sigurnost i nastavak glatkog i neprekinutog rada.

Softverski su mehanizmi važni za zaštitu integriteta softvera, osiguravanje njegovog funkcioniranja kako je predviđeno i sprječavanje zlouporabe od strane zlonamjernih aktera. Jedan od temeljnih aspekata zaštite od neovlaštenog mijenjanja je osiguranje integriteta softverskog koda. To se postiže pomoću kriptografskih hash funkcija kao što je SHA-256, koja generira jedinstvenu hash vrijednost izvornog koda. Tijekom izvođenja, softver izračunava hash vašeg koda i uspoređuje ga s pohranjenom hash vrijednošću. Svako odstupanje između izračunatog hash-a i pohranjenog hash-a ukazuje na petljanje. Mehanizmi protiv neovlaštenog mijenjanja također se proširuju na zaštitu pohrane softvera. Koriste se različite tehnike kao što su sprječavanje izvršavanja podataka (data execution prevention - DEP) i randomizacija izgleda adresnog prostora (address space layout randomization - ASLR) za sprječavanje napada prekoračenjem kapaciteta međuspremnika (buffer overflow napad), oštećenja memorije i ubacivanja koda. Petljaši često koriste debugging alate za ispitivanje i manipuliranje izvršenjem programa. Sustavi zaštićeni od neovlaštenog otvaranja uključuju mjere detekcije prisutnosti debugging softvera te

sprječavanje njeovog daljnjeg korištenja. Ove mjere mogu uključivati pronalaženje prijelomnih točaka, otkrivanje artefakata povezanih s programom za ispravljanje pogrešaka i korištenje maskiranja koda kako bi se kod otežao analizirati obrnutim inženjeringom. Moderni sustavi često koriste i dinamičku analizu i praćenje ponašanja. Promatraju ponašanje softvera dok radi i zabilježe sva odstupanja od očekivanih obrazaca. Ovaj proaktivni pristup omogućuje sustavu da detektira i odgovori na pokušaje petljanja u stvarnom vremenu. Neki mehanizmi za zaštitu od neovlaštenog otvaranja rade tako da ostavljaju dokaze neovlaštenog otvaranja umjesto da u potpunosti sprječavaju neovlašteno mijenjanje. Te tehnike uključuju postavljanje oznaka ili kriptografskih potpisa na softver. U slučaju neovlaštenog mijenjanja, ove oznake mogu biti promijenjene ili poništene kako bi pokazale da je integritet softvera ugrožen. Secure boot i potpisivanje koda ključni su u slučajevima u kojima se softver distribuira na višestruke uređaje. Secure boot osigurava da se samo pouzdani kod izvršava tijekom procesa pokretanja, a potpisivanje koda provjerava da softverska ažuriranja ili zakrpe dolaze iz legitimnih izvora. Ove su mjere važne za sprječavanje neovlaštenih promjena firmvera ili softvera uređaja, pogotovo na mjestima sa većim brojem uređaja i individualaca potencijalno manjeg tehnološkog znanja i iskustva.

Strategije protiv manipulacije nisu statične, već se konstantno razvijaju i izlaze sa novim prijetnjama. Stalna ažuriranja softvera i zakrpe ključni su element zaštite od neovlaštenog mijenjanja. Neki od glavnih načina kako programeri štite svoj kod su randomizacija, enkripcija i zamagljivanje.

3.1 Metode zaštite

3.1.1 Randomizacija

Randomizacija u kontekstu računalne sigurnosti je tehnika koja se koristi za uvođenje nepredvidljivosti u različite aspekte dizajna softvera ili sustava. Cilj je odvrgnuti napadače koji se oslanjaju na fiksne obrasce i pretpostavke o ponašanju sustava. Randomizacija se može provoditi:

- Randomizacijom izgleda adresnog prostora (address space layout randomization - ASLR) [8]: ASLR je uobičajeni oblik randomizacije koji koriste operativni sustavi. Nasumično dodjeljuje lokacije na kojima se učitavaju sistemski i aplikacijski procesi. Zbog toga je napadačima teško predvidjeti memorijsku adresu ranjivosti koje žele iskoristiti, kao što je buffer overflow.
- Randomizacijom skupa instrukcija (instruction set randomization - ISR) [9]: ISR je randomizacija skupa instrukcija ili nizova kodova koji se koriste u programu. Program koristi drugačiji skup instrukcija ili putanju koda svaki put kada se izvrši, što napadaču otežava analizu ili predviđanje ponašanja programa.
- Randomizacijom podataka: Randomiziranje struktura podataka kao što su memorija hrpe

i memorija gomile može pomoći u sprječavanju buffer overflow-a i napada oštećenja memorije. Preuređivanjem redoslijeda i položaja podatkovnih elemenata smanjuje se vjerojatnost da će napadač iskoristiti određenu lokaciju za pohranu.

Randomizacija čini softver i sustave nepredvidljivijima, što napadačima otežava pronalaženje i iskorištavanje ranjivosti. Međutim, ne pruža apsolutnu sigurnost i mora se koristiti zajedno s drugim sigurnosnim mjerama.

3.1.2 Enkripcija

Enkripcija je osnovna sigurnosna tehnika koja pretvara podatke u šifrirani tekst pomoću algoritama šifriranja i kriptografskih ključeva. Samo indivitualc s odgovarajućim ključem za dešifriranje može obrnuti proces i vratiti izvorni otvoreni tekst. Kriptografija je temelj moderne kibernetičke sigurnosti, a njezina učinkovitost ovisi o jakim kriptografskim algoritmima, sigurnom upravljanju ključevima i najboljoj praksi implementacije. Neke od glavnih metoda enkripcije su:

- Zaštita podataka: Enkripcija se koristi za zaštitu osjetljivih podataka i u prijenosu (kao što je kada se podaci prenose putem interneta) i u mirovanju (kao što je kada su podaci pohranjeni na uređaju ili poslužitelju). To osigurava da čak i ako napadač dobije pristup vašim podacima, ne može ih pročitati bez ključa za dešifriranje.
- Autentifikacija: Digitalni potpisi i certifikati koriste kriptografiju za provjeru autentičnosti i integriteta digitalnih poruka i dokumenata. To sprječava neovlašteno ometanje komunikacije i krađu osobnih podataka.
- Sigurna komunikacija: Sigurni komunikacijski protokoli kao što je SSL/TLS koriste enkripciju za uspostavljanje sigurnog kanala između klijenta i poslužitelja. To osigurava da podaci razmijenjeni između dviju strana ostanu povjerljivi i da ih zlonamjerni akteri ne mogu presresti.
- Pohrana lozinki: Lozinke se često pohranjuju u šifriranom obliku kako bi se zaštitile od lakog pristupa ili krađe. Kada se korisnik prijavi, lozinka koju unese uspoređuje se s šifriranom verzijom pohranjenom u bazi podataka.

3.1.3 Zamagljivanje

Zamagljivanje koda je sigurnosna tehnika osmišljena da softver ili kod učini teškim za razumijevanje ili iskorištavanje. To uključuje transformaciju koda na način koji čuva funkcionalnost, ali sprječava razumijevanje. Zamagljivanje se često koristi za zaštitu intelektualnog vlasništva i sprječavanje obrnutog inženjeringa. Neke od glavnih tehnika zamagljivanja jesu [10]:

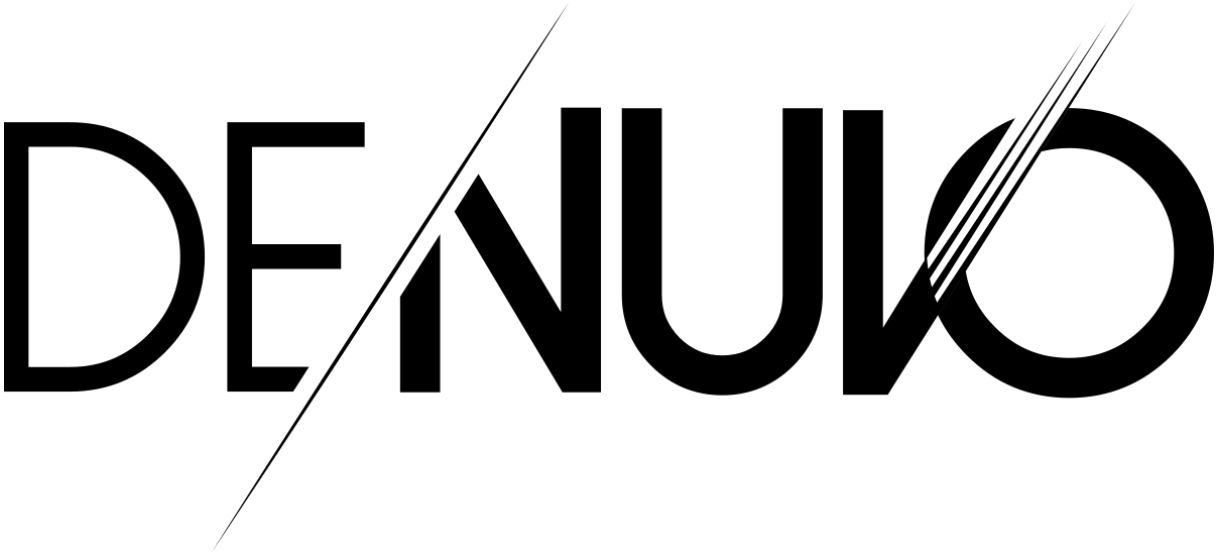
- Skrivanje imena: Ova tehnika mijenja nazive varijabli i funkcija u neopisne ili kriptične oznake, što otežava obrnutim inženjerima razumijevanje svrhe koda. Korištene varijable moraju se odgonetnuti iz konteksta što znatno otežava i usporava rad zlorabitelja softvera.

- Zamagljivanje toka kontrole: Zamagljivanje tijeka kontrole mijenja logiku programa uvođenjem dodatnih grananja, petlji ili uvjetnih iskaza. Zbog toga je još teže otkriti stvarni put izvršenja programa.
- Enkripcija niza: Enkripcija niza šifrira literale niza koji se koriste u vašem kodu. Dešifriranje se događa tijekom izvođenja, tako da napadač ne može lako izvući osjetljive nizove kao što je njezin API ključ ili lozinka.
- Podjela koda: Dijeljenje koda dijeli program na više komponenti ili modula, čineći ga složenijim i distribuiranijim. Obrnuti inženjeri moraju sastaviti fragmentirani kod kako bi razumjeli potpunu funkcionalnost.

Zamagljivanje samostalno također nije sigurno. Odlučan napadač još uvijek ima mogućnost reverznog inženjeringa maskiranog koda. No ono, podiže težinu obrnutog inženjeringa i dodaje dodatni sloj obrane. Kada se ove sigurnosne metode koriste u kombinaciji jedne s drugima, level pružene zaštite se znatno povećava. One doprinose ukupnoj otpornosti softvera i sustava na različite oblike napada i prijetnji te poprilično produljuju potrebno vrijeme i razinu sposobnosti za probijanje zaštite.

3.2 Denuvo

Denuvo [11, 12] je najpoznatija anti-tampering tehnologija i sistem za upravljanje digitalnim pravima. Istaknut je u području zaštite softvera i često kontroverzan zbog svog invazivnog pristupa. Poznat po svojoj moćnoj tehnologiji protiv neovlaštenog otvaranja, postao je i zaštitnik intelektualnog vlasništva programera igara i meta hakera koji žele poraziti njegovu obranu. Da bismo razumjeli puni utjecaj Denuva, bitno je zadubiti se u njegovu povijest, ključne značajke, kontroverzene postupke i njegovu stalno evoluirajuću ulogu u industriji igara.



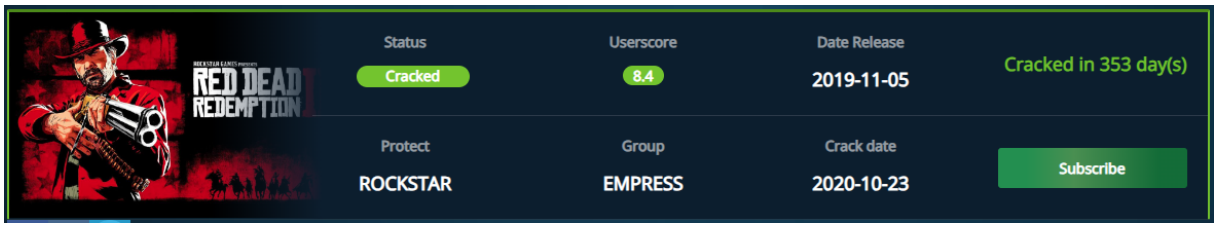
Slika 3.1: Denuvo

Austrijska tvrtka Denuvo Software Solutions GmbH pokrenula je Denuvo 2014. kako bi poku-

šala savladati stalni problem softverskog piratstva u industriji igara. Odmah su ostavili dojam, te se pokazali kao dostojan štit dovoljno moćan za obranu od pirata. Od svog početka, Denuvo je prošao kroz mnoge verzije, a svaka verzija uključuje napredne sigurnosne mjere i značajke protiv hakiranja. Ova poboljšanja imaju za cilj ostati ispred razvijajućih cracking i jailbreaking tehnika. Denuvo koristi enkripciju za zaštitu kritičnih dijelova izvršnog koda igre. Prilikom pokretanja igre, Denuvo dinamički dekodira te dijelove, provjeravajući njihovu cjelovitost i autentičnost. Ovaj mehanizam zaštite u stvarnom vremenu temelj je njegove snage. Bilo kakva sumnja na neovlašteno mijenjanje, kao što su promjene koda ili neovlaštene modifikacije, može uzrokovati da se softver odbije izvršiti ili se izvršenje i igra na neki način mogu ograničiti. Prije dolaska Denuva na scenu većina igara bi se uspjela crackati u nekolicini dana ili tjedana od izlaska. Denuvo je taj vremenski period znatno produžio, sprječavajući crack za neke igre čak preko godinu dana. Vidjevši te rezultate većina, pogotovo većih, kompanija odlučila se koristiti Denuvo softver. No iako je Denuvo uspio značajno odgoditi crackanje zaštićenih igara, nisu svi bili zadovoljni njegovim ponašanjem pogotovo uzevši u obzir velik broj nuspojava koje sa njim dolaze. Recenzenti i korisnici izrazili su zabrinutost zbog njegovog mogućeg utjecaja na performanse sustava. Izjave o povećanom korištenju CPU-a, produljenom vremenu učitavanja i navodnim negativnim učincima na iskustvo igranja potaknuli su raspravu unutar zajednice te se većina legitimnih igrača složila u svom nezadovoljstvu sa Denuvo softverom. Osim toga, Denuvova česta ažuriranja i zakrpe usmjerene na povećanje sigurnosti stvaraju neugodnosti korisnicima koji onda moraju instalirati ažuriranja kako bi nastavili igrati svoje igre. Denuvo je softver napravljen za zaštitu od pirata no pri ostvarenju tog cilja značajno je osakatio legitimne igrače koji se nakon što plate igru moraju nositi sa pregrštom problema i nepotrebnih neugodnosti.

Igra Red Dead Redemption 2, koju je razvio Rockstar Games, jedna je od značajnih igara koja je ostala necrackana duži period vremena. Kada je igra prvi put izdana na računalu u studenom 2019., uključila je Denuvo zaštitu zajedno s nekim drugim zaštitama. Igra se pokazala kao posebno težak izazov za sve cracking grupe. Dok su dani odbojavali i ljubitelji igre nestrpljivo čekali, na tajmeru se napunila gotovo cijela godina dana, sa 353 dana nakon izlaska, Red Dead Redemption 2 je konačno bio crackan. To je označilo jedno od najdužih razdoblja u kojem je igra zaštićena Denuvom ostala sigurna. Ovo produljeno razdoblje zaštite privuklo je široku pozornost i raspravu unutar industrije igara i zajednice. Naposljetku, odlučni hakeri crackali su igru, naglašavajući da nijedan sustav zaštite nije potpuno otporan na napade, već je pasti pod uporima dovoljno snažne sile. Odgoda cracka Red Dead Redemption-a pokazala se vrlo povoljnom za izdavačku kuću igre, sprječavajući nelegitimne kopije i zahtjevajući kupnju pri punoj cijeni u najkritičnijim trenucima izdanja igre. Makar svjesni činjenice da će zaštita biti probijena, izdavači igara skočili su na priliku da zaštite svoje financijske interese te je gotovo svako naredno izdanje AAA igre raznih studija došlo zajedno sa Denuvom.

Ignorirajući brojne kritike legitimnih korisnika i igrača sentiment populacije prema Denuvu je svakim trenom postajao sve lošiji. Počeo je biti viđen kao softver koji samo štiti financijske



Slika 3.2: Red dead redemption 2

i ostale interese kompanije dok oštećuje iskustva igrača igre. Naknadni testovi pokazali su znatnu razliku u performansama igre koja je zaštićena Denuvo softverom naspram one koja nije. Ista igra upakirana sa Denuvom i verzija te igre koja je crackana i Denuvo softver izbrisan ili onesposobljen ponašaju se vrlo različito pri direktnoj usporedbi. Brzina izvedbe pri početnom paljenju, manje opterećenje sustava i stabilno i konstantno poboljšanje u broju FPS-a samo su neke od brojnih i poprilično očitih posljedica što je integracija Denuvo softvera u igru imala. Sada postoji razlog da se igra cracka makar ju netko želi kupiti i legitimno koristiti. Legalni korisnici traže crackanu verziju igre jer je objektivno bolja verzija igre. Denuvo je sada u jako lošem svijetlu i generalna populacija ne podržava ponašanje kompanija koje koriste Denuvo. Crackerske scenske grupe koje se ponose time što crackaju teški softver i dokazuju kako mogu sve crackati, sada dobivaju još više podrške od pirata i legitimnih igrača. Većina njih prezire Denuvo softver i namjerno još više truda ulažu u crack samo da bi oštetile sliku softvera i kompanije koja ga je odlučila koristiti. Nedavni primjer ove odlučnosti je pokazan u vremenu za izlazak crackane verzije igre Hogwarts Legacy koja je crackana u samo 13 dana.

Denuvo također planira postaviti ograničenja i zaštitu u Unreal Engine zajedno sa njihovom kolaboracijom, što će znatno otežati kreaciju novih modova te razljutiti još više igrača.

Poglavlje 4

Anti-cheat

Varanje [13] u svijetu online igara postalo je vrlo često. S varanja u singleplayer igrama bez posljedica za druge igrače prešlo se na varanje u igrama s drugim igračima kako bi se postigla određena prednost. Događa se konstantna borba s igračima koji žele steći te nepravedne prednosti različitim nezakonitim sredstvima, dok programeri igara i zajednica neumorno rade na odvraćivanju hakera i održavanju ugodnog iskustva za većinu igrača. To dovodi do dalekosežnih posljedica jer varanje narušava integritet natjecateljskog igranja, narušava poštenu igru i smanjuje angažman igrača [14]. Osim toga, varanje u rjeđim slučajevima može poremetiti ekonomiju unutar igre, budući da ilegalno stečeni predmeti i valute mogu preplaviti tržište i narušavajući ravnotežu u igri.

4.1 Vrste varanja

- Aimbot: softverski alat koji pomaže igračima da ciljaju svoje oružje s iznimnom preciznošću tako što automatski nacilja protivnika, bez unosa igrača i na način kako to redovna osoba ne bi mogla. Aimbot se koristi u pucačinama iz prvog lica odnosno FPS igrama.
- Wallhack: softver za varanje koji omogućuje igračima da vide kroz zidove i druge prepreke u okolišu igre. Wallhack omogućuje igraču da vidi protivnike i njihovo kretanje što omogućuje dodatni level percepcije i pretpostavljanja neprijateljevih sljedećih koraka.
- Ekstra senzorna percepcija (ESP - extra sensory perception): pruža igraču inače nevidljive informacije o okruženju igre i drugim igračima u igri. Jači od wallhacka jer pokazuje lokacije, zdravlje, dostupnu opremu i još mnogo toga drugih inače skrivenih informacija.
- Speedhack: omogućuje igraču da se kreće puno većom brzinom od normalne. Speed hakeri mogu biti brži od protivnika, čineći ih teškim za napad i dajući im taktičku prednost u borbi.
- Teleportacija: omogućuje igračima instantno premještanje s jedne lokacije na drugu u svijetu igre. Teleportacijom igrači zaobileze predviđeni tok igre i mogu dobiti znatnu prednost naspram legitimnih igrača.

- Eksploatacije: metode ili tehnike koje igrači koriste kako bi iskoristili nenamjerne ranjivosti ili slabosti u mehanici igre. Mogu uključivati različite radnje, poput manipuliranja fizike igre, kopiranja predmeta ili postizanja naplaniranih rezultata radi stjecanja prednosti.
- Greške: neočekivane i često privremene pogreške ili anomalije u kodu ili ponašanju igre. Igrači mogu iskoristiti greške za postizanje neplaniranih rezultata, kao što je ulazak u nedostupna područja, svladavanje prepreka ili stjecanje prednosti koje nisu bile planirane niti pokazane razvojnim programerima igre.
- Zlouporaba makronaredbi: uključuje korištenje makronaredbi, koje su skripte za automatizirano izvršenje unaprijed određenih naredbi. Koriste se za izvršavanje složenih nizova radnji s iznimnom preciznošću. Igrači koji zlorabe makro mogu brzo i dosljedno izvoditi radnje u igri, uživajući u prednostima kao što su brza paljba, precizno mjerenje vremena i pozicije ili optimalno upravljanje resursima. To može biti posebno problematično u kontekstu natjecanja.

4.2 Načini zaštite od varanja

- Anti-cheat na strani klijenta: softver ugrađen u klijent igre za praćenje i otkrivanje sumnjivih aktivnosti na uređaju igrača. On aktivno skenira pokrenute programe, neovlaštene izmjene ili neobično ponašanje na računalu ili konzoli igrača kako bi spriječio varalički softver da utječu na igru.
- Anti-cheat na strani servera: softver koji radi na poslužitelju igre koji je najčešće server, i provjerava radnje igrača kako bi se osiguralo da su u skladu s pravilima i standardima igre. On otkriva anomalije u ponašanju igrača, kao što su neuobičajeni pokreti ili nemoguće radnje, i poduzima mjere protiv varalica baniranjem ili ograničavanjem njihovog računa.
- Analiza ponašanja: bihevioralna analitika koristi algoritme strojnog učenja za analizu i prepoznavanje obrazaca igranja koji odstupaju od norme. Otkriva anomalije u ponašanju igrača, uključujući iznenadna poboljšanja vještina ili neobične radnje, kako bi označio sumnjive račune za daljnji ljudski pregled od strane razvijača igre ili kaznene radnje.
- Hardverske zabrane: onemogućuju igračima pristup igrama pomoću prepoznavanja određenih hardverskih komponenti ili konfiguracija povezanih sa zabranjenim računom. Zabrane hardvera otežavaju varalicama da se vrate u igru pod drugim identitetom odnosno pod drugim makar plaćenim računom te time povećavaju poslijedice za osobu suočenu sa zabranom.
- Redovito ažuriranje: česta ažuriranja i zakrpe znače da programeri igara redovito objavljuju verziju softvera kako bi popravili otkrivene ranjivosti i borili se protiv novih tehnika varanja. Ova ažuriranja održavaju sigurnosne mjere igre, što otežava varalicama pronalaženje i bitnije iskorištavanje već znanih slabosti u kodu igre.
- Moderacija: mehanizmi za moderaciju i izvješćivanje zajednice omogućuju igračima da

prijave sumnju na varanje. Potom se osumnjičeni pregledavaju i izvješća istražuju, poduzimajući mjere protiv potvrđenih varalica na temelju prezentiranih ili viđenih dokaza. To promiče odgovornost igrača i pomaže u održavanju poštenog igračkog okruženja.

- Reprodukcija: reproduciranjem sadržaja video igre omogućuje se administratorima poslužitelja, moderatorima i igračima da gledaju trenutnu igru ili pregledaju prošlu sesiju igranja kako bi ustavovili je li bilo nezakonite igre. Ovi alati olakšavaju otkrivanje sumnjivog ponašanja ili varanja tijekom igranja jer promatrači mogu pomno pratiti radnje i interakcije igrača te primjetiti sve sto igrač nebi trebao znati.
- Kažnjavanje ponašanja: uključuje nametanje ograničenja, kao što su ograničenja chata, smanjenja nagrada ili privremene zabrane koje se naknadnim prekršajima eksponencijalno povećavaju, za račune koji su varali ili svojim ponašanjem kršili pravila igre. Ovaj pristup potiče poštenje, obeshrabruje varanje te je najraširenij način kažnjavanja prijestupnika.

Zbog stalne i većinski neuspješne bitke protiv hakera i varanja u igrama, neki izdavači igara odlučili su poduzeti strože dosad neviđene mjere zaštite. Implementirajući zaštitu na levelu operativnog sustava. Riotova najnovija igra zvana Valorant došla je sa dosad neviđenim levelom sigurnosti za koju je većina korisnika više upoznatih sa tehničkim aspektima sigurnosti, rekla da je pretjerivanje. Zaštita na razini OS-a proširuje mjere protiv varanja izvan same aplikacije za igranje i integrira se s operativnim sustavom. Na taj način otkriva i spriječava prijevaru na osnovnoj razini te uključuje nadzor i kontrolu interakcija na razini sustava, što uključuje pristup memoriji, aktivnostima datotečnog sustava i ulazno/izlaznim uređajima. Intervencijom na razini operativnog sustava, anti-cheat softver može učinkovitije spriječiti varalice od pokušaja manipulacije ili uplitanja u proces igre. No iako sprječavajući više pokušaja varanja, implementacija na toliko niskom nivou sustava, a da nije sam proces operativnog sustava, može dovesti korisnikov sistem u vrlo ugroženu poziciju.

Valorantov Vanguard [15] sustav protiv varanja, koji radi na razini kernela, ilustrira moć i rizike povezane s dodjeljivanjem privilegija na razini kernela, softveru koji nije napravljen od strane operativnog sustava. Iako Vanguardova duboka integracija s jezgrom operativnog sustava omogućuje otkrivanje i prevenciju prijevara bez presedana, ona također ima potencijalne nedostatke. Jedna od glavnih briga je rizik od nestabilnosti sustava, jer sve pogreške ili ranjivosti u Vanguardu mogu dovesti do pada sustava, što utječe na korisničko iskustvo. Dodatno, sigurnosne implikacije su važne jer softver sa kernel level privilegijama postaje atraktivna meta za potencijalne napadače. Potreba za stalnim oprezom, rigoroznim testiranjem i transparentnom praksom najvažniji su za smanjenje ovih rizika. Uz to, korisnici mogu imati problema s kompatibilnošću i mogućim utjecajima na performanse, što naglašava složenost upravljanja softverom koji operira na razini kernela kao što je Vanguard. Općenito Vanguardova ring 0 implementacija donosi prednosti i izazove vezane uz duboku integraciju s operativnim sustavom kako bi se održalo pošteno i sigurno igranje te zasad nije pokazala nikakvih većih problema.

Poglavlje 5

Jailbreaking

Jailbreaking kao pojam se odnosi na praksu zaobilaženja ugrađenih ograničenja nametnutih od strane proizvođača ili programera u operativnom sustavu uređaja. Imenom insinuira bijeg iz zatvora koji je figurativni set postavljenih mjera i ograničenja za tipičnog korisnika. Jailbreak korisnicima daje visok pristup i kontrolu, dopuštajući im pokretanje neovlaštenog softvera, modificiranje sistemskih datoteka i istraživanje neplaniranih značajki. Dok ćemo se u ovom poglavlju primarno fokusirati na jailbreaking konzola, jailbreaking je pojam koji prelazi granice samo igraćih konzola već uključuje i mobilne uređaje poput iPhone-a.

Jailbreaking konzola, praksa je koja je oblikovala igraći krajolik, pružajući mogućnost igranja piratiziranih igara te time omogućavajući dodir sa sadržajem, puno većem broju ljudi nego što bi inače bilo. Također mnogo regijskih ili drugačijih limitacija moglo je biti zaobiđeno korištenjem jailbreakane konzole. U doba starijih igraćih konzola, u ranim 2000.-ima, većina područja sa manje zastupljenom i razvijenom tehnologijom igre su mogli nabavljati samo ilegalno preko interneta, te ih potom snimati na DVD-e i igrati. U dućanima su se čak prodavale ilegalne kopije tih istih igara po smanjenoj cijeni, dok original igara nije bilo na vidiku. Pojave poput takvih i sama nenaučenost na trošenje većih suma novaca na igre dodatno su poticale jailbreakanje konzola, jer se bez jailbreakane konzole nije moglo igriti doli nekolicinu igara. Jailbreak najčešće se obavljao obrnutim inženjeringom raznih algoritama potrebnih za različite akcije koje je korisnik htio da se ponašaju drugačije. Otkrivanjem njihovog načina rada, vlastit softverski update sa određenim potrebnim promjenama uploadao se na uređaj te je konzola tada postala jailbreakana. Nekada se vlastita softverska zakrpa mogla direkno instalirati bez problema dok se u drugim situacijama morala zaobići još jedna prepreka provjere legitimnih zakrpa. Gotovo sve konzole tadašnjeg doba jailbreakane su u jednom trenu sa različitim levelima uspjeha. Kompanije koje su bile vlasnici tih konzola uporno su unaprijeđivale sustav i ograničavale pokušaje napada, a s obzirom da je hardver bio njihovog originalnog dizajna, ta ograničenja su jako brzo postala ozbiljna. Jedan od zanimljivijih odgovora na konstantna unaprijeđenja sigurnosti bio je takozvani kamikaze napad pri jailbreakanju xbox-a 360 [16], nakon što je Microsoft onemogućio dotadašnje metode jailbreakanja konzole. Napad se zvao tako jer je uključivao bušenje čipa

na DVD čitaču s malom bušilicom, nakon čega bi se na konzolu mogla instalirati vlastita zakrpa sa promjenama koje omogućuju jailbreak. Ako bi se pri bušenju desila makar mala pogreška ili nepreciznost, konzola bi bila potpuno neupotrebljiva. PlayStation 2 imao je također zanimljiv, ali i puno sigurniji, način jailbreakanja, tako što se na matičnu ploču zalemio dodatni čip koji bi onemogućio algoritme zaštite autorsikh prava. U današnje vrijeme jailbreakanje konzola je puno teže i sporije, no novom povezanošću interneta postoji puno veći broj ljudi koji zajedno radi kako bi provalili i nove razine sigurnosti i zaštite.

5.1 PS4 i PS5 – višedjelni jailbreak

Današnji jailbreakovi najčešće iskorištavaju niz ranjivosti različitih djelova sustava kako bi zajedno uspjeli provesti potpuni jailbreak. U slučaju PlayStation-a 4 to su userland i kernel exploitovi [17].

Proces jailbreakanja obično započinje iskorištavanjem userlanda. Userland je najmanje privilegirani dio operativnog sustava PS4, gdje se pokreću aplikacije i igre. U njemu se pokušavaju iskoristiti ciljane ranjivosti u softveru na korisničkoj razini, kao što je web preglednik konzole ili određene igre. U slučaju PlayStation-a 4 bio je to upravo web preglednik. Prethodno poznata ranjivost u webkit-u PlayStation-a, koji je dijelio istu ranjivost originalno otkrivenu za internetski preglednik Mac OS X Safarija, iskorištena je kao ulazna točka u nizu postupaka potrebnih za jailbreak. Ranjivost webkita trenutno se koristi i u najnovijoj verziji jailbreaka. Kada se userland uspješno probije, napadaču je omogućeno izvršavanje vlastitog koda u tom okruženju. Ovaj kod može uključivati prilagođene aplikacije, razne učitavače ili druge alate.

Pristup i kontrola na userland levelu najčešće nije dovoljna, te za potpuni jailbreak korisnik treba dobiti veći pristup samim funkcijama sustava. Kernel exlpoit cilja na ranjivosti u samoj jezgri PS4, odnosno osnovnoj komponenti operativnog sustava s najvišom razinom privilegija. Ako je exploit uspješan, eksploatacija kernela omogućuje napadaču preuzimanje potpune kontrole nad sustavom. Pronalazak ranjivosti kernela znatno je rjeđi nego ostale ranjivosti, te čak i kada je nađen teži je za iskoristiti. Probijanje sigurnosti kernela obično uključuje pažljivo izrađen kod koji iskorištava pronađene ranjivosti koda na levelu kernela i upravljanje memorijom sustava.

PlayStation 4 je kao što se iz prethodnog teksta može pretpostaviti, u potpunosti jailbreakan. PlayStation 5 ima parcijalni jailbreak jer su u sigurnost PS5 dodali još jedan level zaštite. Userland PS5 je već odavno probijen no zasad nema javno poznatog probijanja kernel sustava. Glavni razlog za to je što PS5 ima hipervizor koji nadzire kernel i pruža dodatnu razinu sigurnosti od kernel exploit-ova. Efektivno obe razine sigurnosti će se trebati pobijediti kako bi se ostvario potpuni jailbreak. Provalom sigurnosti userlanda, korisnik može pokretati određene programe i skripte koje ne zahtjevaju najdublji pristup sustavu.

Poglavlje 6

Cracking

Krekiranje (cracking) ili probijanje izvršne (EXE) datoteke praksa je koja obično uključuje neovlaštenu manipulaciju softverom u svrhu uklanjanja postavljenih ograničenja licenciranja, kao što su ograničenja probnog razdoblja, provjere autentičnosti i sličnog [18]. Motivacija crackera koji zlonamjerno koriste softver najčešće je novčana, ali razlikuje između pojedinaca. Laički se crackanje dijeli na dvije kategorije, lagano i teško. Pod lagano crackanje spadaju bypass-ovi i slične modifikacije samog programa. Mijenjanjem ponašanja određenih funkcija možemo dobiti željene rezultate bez prekomjerno puno utrošenog vremena. Bypass-om zaobiđemo predviđenu funkcionalnost i red postupaka aplikacije te spremimo te promjene u aplikaciji kao novu aplikaciju. U suštini zakrpamo aplikaciju da funkcionira na nama željen način. Ovaj postupak je lakši nego neki drugi, no zato nosi i određene probleme. Kompliciranije aplikacije sa većim brojem interoperabilnih komponenti ne mogu uvijek biti zakrpane na taj način. Aplikacija može zahtjevati zakrpu nekolicine datoteka ili pak provjeravati ako je datoteka mijenjana nakon prvotnog preuzimanja. Ikakva detekcija nepredviđenog ponašanja može onemogućiti korištenje aplikacije.

Stoga dolazimo na teži oblik crackanja, koji podrazumijeva korištenje aplikacije na predviđeni način bez promjena na samoj aplikaciji. Kako bi aktivirali aplikaciju najčešće trebamo neki oblik autentikacije poput serijskog ključa. Kako bi otkrili serijski ključ, trebamo obrnuto inženjerati djelove aplikacije kako bismo shvatili kako temeljni kod funkcionira, osobito te koje su zadužene za autentikaciju. Potom s tim informacijama možemo napraviti keygen.

6.1 Alati

Debugger je softverski alat koji se primarno koristi za analizu i rješavanje problema softverskih programa [19]. Njegova je glavna funkcija omogućiti programerima i obrnutim inženjerima da korak po korak prate i ispitaju izvođenje programa, postave prijelomne točke, testiraju memoriju i varijable te identificiraju probleme u kodiranju. Debuggeri su vrijedni u identificiranju grešaka u vremenu izvođenja i logičkih pogrešaka tijekom razvoja softvera. Međutim, također se često

koriste u postupcima obrnutog inženjeringa za analizu ponašanja izvršne datoteke. Omogućuju korisnicima pokretanje programa uz potpunu kontrolu, što može biti posebno korisno kada se istražuje unutarnji rad datoteke.

Decompiler je specijalizirani softverski alat dizajniran za preokretanje procesa kompilacije, pretvaranje strojnog ili binarnog koda natrag u programski jezik više razine, kao što je C ili C++. Dekompilator pokušava ponovno izgraditi izvorni izvorni kod iz kompajlirane binarne datoteke, pružajući ljudima čitljiv prikaz koji nalikuje logici izvornog programa. Međutim, dekompilacija je složen proces i dobiveni kod možda neće uvijek točno odgovarati originalnom izvornom kodu zbog optimizacija koje izvodi kompilator. Dekompilatori su korisni u situacijama kada nema pristupa izvornom izvornom kodu ili kada se obrnutim inženjeringom namjerava učinkovitije razumjeti ili modificirati program. Decompiler je često dostupan kao dio dissasembler programa.

Disassembler je softverski alat koji prevodi strojni ili binarni kod, najčešće nerazumljiv ljudima, u asemblerski jezik čitljiv ljudima ili reprezentaciju više razine. Ovaj proces omogućuje obrnutim inženjerima da razumiju funkcionalnost programa ispitivanjem njegovog asemblerskog koda. Disassembleri se naširoko koriste kako bi se bolje razumjelo unutarnje funkcioniranje izvršne datoteke na nižem levelu. Iako rastavljeni kod pruža vrijedne informacije, nedostaju mu semantičke informacije visoke razine, kao što su imena varijabli ili pozivi funkcija, što otežava razumijevanje svrhe izvornog izvornog koda. Nadalje se kod decompilerom može dalje interpretirati, ali sama interpretacija koda može biti u potpunosti pogrešna. Postoje mnogi programi koji se problemom nose na različite načine, a većina omogućava vlastoručno unošenje nekog poznatog djela koda kako bi decompiler mogao ponovo ponoviti postupak sa novijim informacijama.

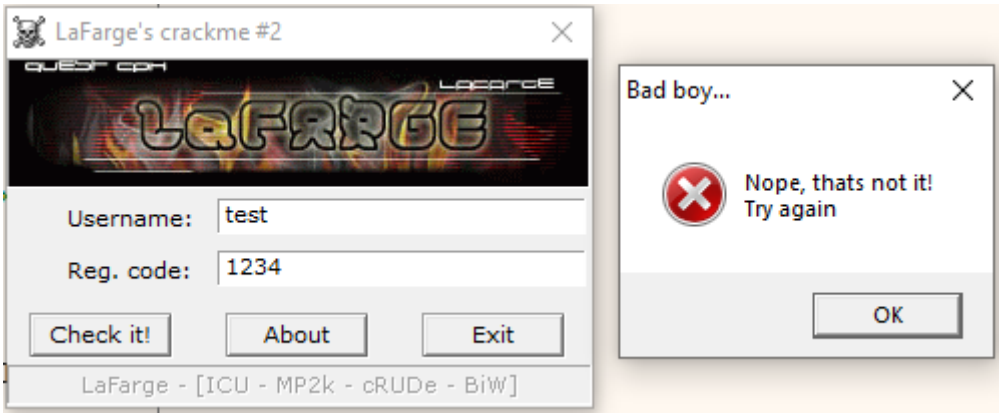
U suštini, debuggeri, edcompileri i disassembleri služe različitim svrhama u analizi softvera te imaju razne prednosti i nedostatke u odnosu na druge. Korištenjem alata primjenjenijeg za trenutnu situaciju najčešće se dolazi do traženih rezultata.

6.2 Izmjena exe datoteke

Za demonstraciju osnovnog bypass-a i kreaciju modificirane exe datoteke, koristiti ćemo LaFargeov crackme #2 softver. To je softver namijenjen isključivo za demonstraciju i vježbanje navedenih tehnika, te je zato provedba istih potpuno legalna [20].

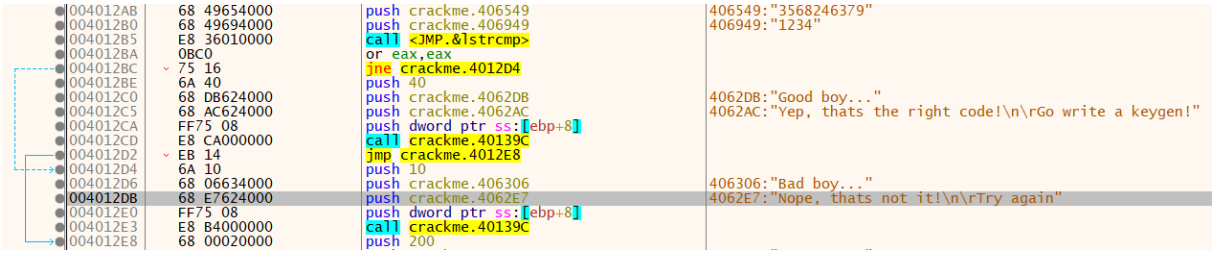
Kada pokrenemo aplikaciju sredne nas login ekran uz popratnu pozadinsku 8-bitnu glazbu koja najčešće karakterizira razne instalacijske programe crackanog softvera. Postoje dva polja za unos podataka, korisničko ime i registracijski ključ. Imamo gumbe za više informacija i za izlaz, te gumb za provjeru unosa vrijednosti u dva navedena polja.

Ukoliko upišemo probne vrijednosti te kliknemo za provjeru ispravnosti, susretne nas ekran koji nam saopći da uneseni podaci nisu ispravni. Krenimo izanalizirati aplikaciju. Upalimo



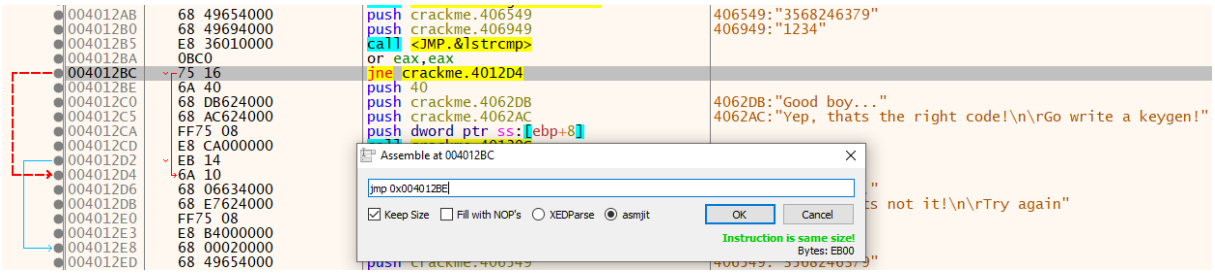
Slika 6.1: crackme 1

program otvorenog koda za debugiranje zvan x64dbg. S obzirom da nemamo puno informacija o potrebnim podacima krećemo unazad, pretraživanjem stringa poruke koja je došla kada unesemo neispravne podatke. U glavnom prozoru kliknemo desni klik te navigiramo na pretraživanje stringova te unosimo navedeni string. Dolazimo na adresu gdje se vrši naredba za ispis poruke te možemo primijetiti da do par naredbi iznad vodi jump naredba. Vidimo također da postoje gotovo identicne naredbe, sa drugom porukom uspjeha, nekoliko naredbi iznad.



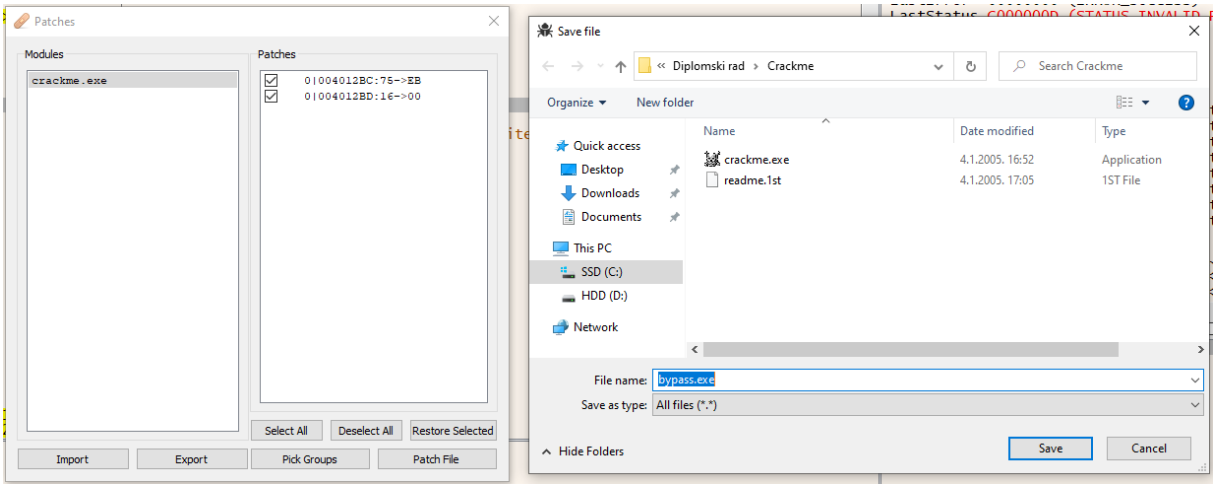
Slika 6.2: crackme 2

Ukoliko jump naredbu koja provjerava jednakost unesenih podataka, zamijenimo sa jump naredbom koja nas bez provjera vodi na adresu naredbi za uspjeh, zaobići ćemo potrebu za unosom ispravnih podataka. Stoga mijenjamo jne (jump if not equal) naredbu sa čistom jump naredbom na adresu 004012BE koja se aktivira ukoliko su uneseni ispravni podaci.



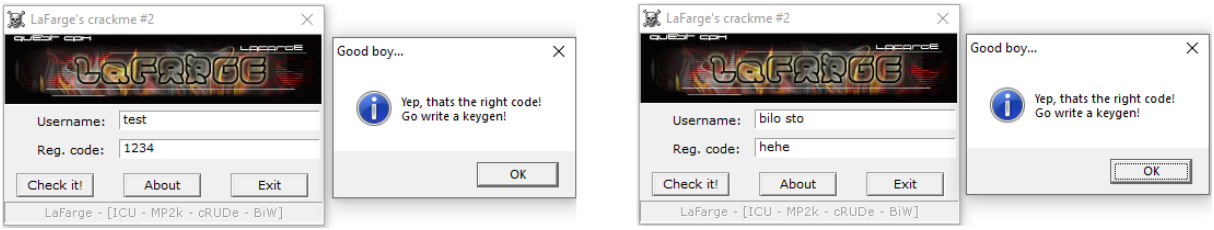
Slika 6.3: crackme 3

Zakrpamo exe datoteku te ju spremamo kao *bypass.exe*. Pobrinemo se da ju ne ometa sigurnosni sustav na vlastitom računalu, te ju pokrećemo.



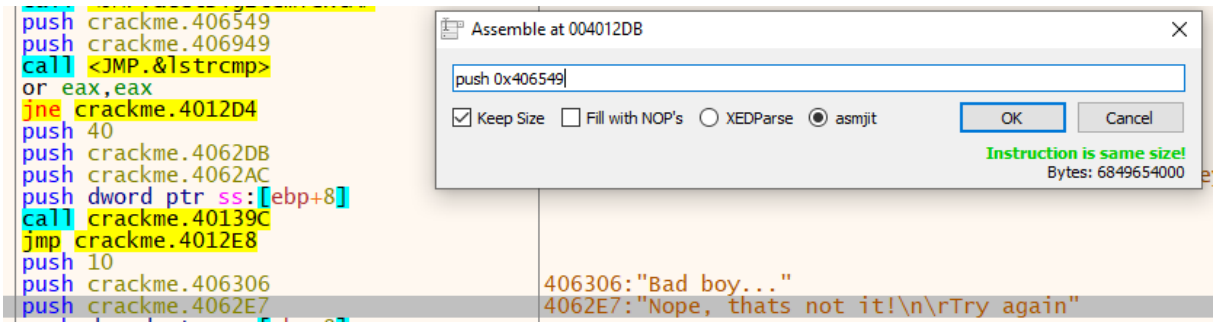
Slika 6.4: crackme 4

Pri unosu identičnih podataka kao i prije, sada dobivamo drugu poruku, poruku uspjeha. Na ovaj način smo bez znanja potrebnih podataka uspjeli doći do željenog cilja zaobilaženjem postavljenih provjera. Sada pri svakom unosu, neovisno o podacima, ispisati će se poruka uspjeha.



Slika 6.5: crackme 5+6

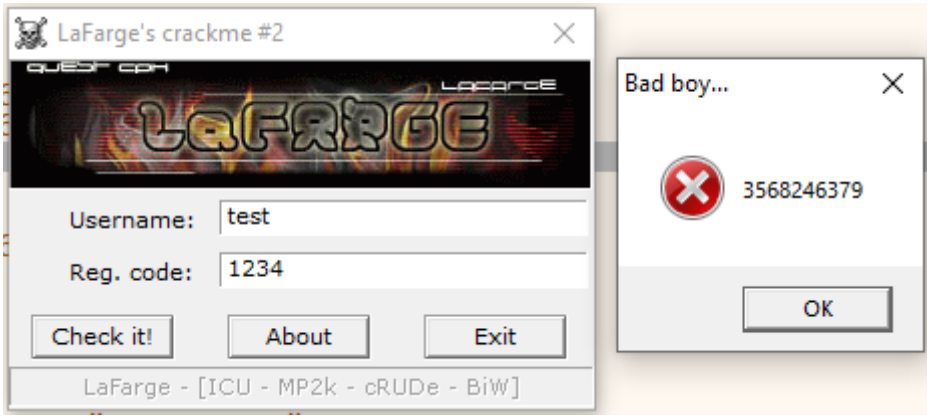
Ukoliko želimo saznati koji su to ispravni podaci te zelimo dobiti poruku uspjeha bez modificiranja aplikacije, provjerimo što se točno uspoređuje. Uspoređuju se dvije adrese te kako bi vidjeli što na njima piše zamijenimo komandu tako da umjesto ispisa poruke neuspjeha, ispiše sadržaj te adrese.



Slika 6.6: crackme 6

Primijetimo da je na jednoj od tih adresa neki broj, a na drugoj naš unos na registracijskom kodu. Možemo zaključiti da jedina provjera koju aplikacija radi je jeli uneseni broj u polju

registracijskog koga jednak zapisanom broju. Zbog toga znamo da je zapisani broj u stvari registracijski ključ.



Slika 6.7: crackme 8

Ukoliko probamo unesti dobiveni broj kao registracijski ključ uistinu dobivamo poruku uspjeha bez modificiranja aplikacije.

Slika 6.8: crackme 9

6.3 Reverse engineering i generacija ključeva

Jedinstveni aktivacijski ključ lako je nabaviti, upravo prikazanim metodama ili prikupljanjem informacija od drugih ljudi koji ga znaju. Zbog toga određeni softveri su počeli proizvoditi razne raspone aktivacijskih ključeva kako bi svaki korisnik imao drugačiji ključ koji se može samo jednom aktivirati odnosno ograničen je samo na legitimnog kupca tog ključa. Legitimni ključevi generirani su nepoznatim algoritmom koji može generirati samo određene ključeve. Time se osigurava laka provjera nelegitimnih ključeva jer ih taj algoritam nije mogao generirati. S obzirom na ogromni broj mogućih ključeva od 16 znakova, kompliciranost algoritma može biti jako velika, a raspon jako mali što rezultira malim brojem ključeva naspram svih mogućih, ali to će i dalje dati više ključeva nego što je potrebno. Šansa za pogoditi željeni radeći legitimni

ključ je izuzetno malena, otprilike kao za osvojiti jackpot na lutriji, tri puta za redom. Šansa za pogoditi ključ se eksponencijalno smanjuje, a broj mogućih ključeva eksponencijalno diže, sa povećanjem broja znakova kao u današnjim novijim 25 znakovnim ključevima. Brute force očito nije rješenje zato crackeri pokušavaju obrnuto inženjerati algoritam korišten za generiranje legitimnih ključeva kako bi ga potom mogli koristiti za izradu keygen alata.

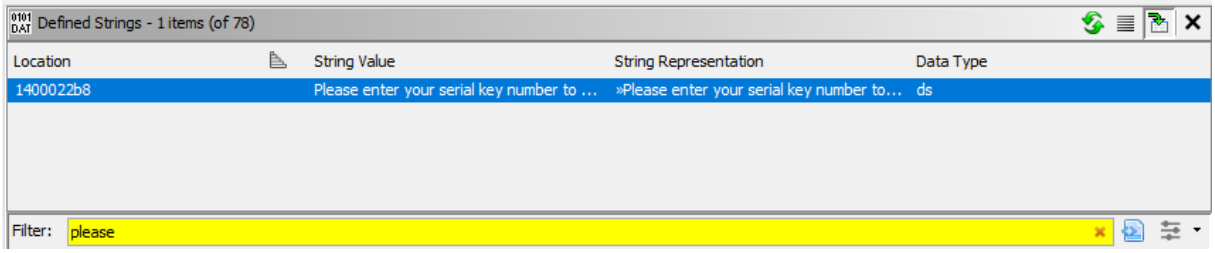
Keygen je softver ili alat dizajniran za generiranje autentifikacijskih ključeva za softver bez kupnje legalne licence. Često ih koriste ljudi koji žele zaobići ograničenja licenciranja ili koristiti softver bez plaćanja [21]. Neki ljudi poput crackera često imaju drugačije i često neopipljive ciljeve na umu poput dokazivanja sebi i svijetu da mogu probiti zaštitu softvera, i naknadna uživanja u novosteknutoj slavi zbog toga. Keygeni se često povezuju sa softverskim piratstvom i kršenjem autorskih prava jer dopuštaju korisnicima korištenje softvera bez odgovarajućeg ovlaštenja. Oni rade tako što stvaraju ključeve koji simuliraju strukturu pravih ključeva, omogućujući korisnicima da aktiviraju softver. Nakon toga što je keygen napravljen odnosno algoritam za stvaranje ključeva probijen ne postoji mnogo što se više može napraviti, stoga fokus u poboljšanju sigurnosti je na stvaranju boljih, kompliciranijih i sugurnijih algoritama za kreiranje ključeva. Keygen-ovi su opasan komad softvera i za ilegalne korisnike jer se za proizvodnju validnog ključa moraju pokrenuti na korisnikovom računalu. To korisnika dovodi u nelagodnu situaciju gdje mora riskirati sigurnost svog računala u svrhu dobitka validnog ključa jer nitko točno ne zna što je sve u keygen programu i postoji li neki nadodani virus. Prikazat ćemo jednostavan primjer analize programa i probijanja algoritma kreacije ključa, te naknadnog kreiranja keygena.

Kad pokrenemo program *product.exe* dobijemo sljedeći izlaz:

```
C:\Programski-kod\Diplomski rad\Crackme>product.exe
Please enter your serial key number to activiate the product.
e.g -> ./product 1111-2222-3333-4444
```

Slika 6.9: crackme 15

Otvorimo program u programu za obrnuto inženjeranje zvanom Ghidra [22]. Otvorimo prozor za pregled stringova te potražimo string koji je ispisan pokretanjem product.exe programa.



Slika 6.10: crackme 11

Nalazimo upravo taj string na lokaciji 1400022b2 te odlazimo na tu lokaciju.

```
                    s_Please_enter_your_serial_key_num_1400022b8    XREF[1]:    FUN_140001160:140001169(*)
1400022b8 50 6c 65        ds         »Please enter your serial key number to activi...
          61 73 65
          20 65 6e ...
```

Slika 6.11: crackme 12

Vidimo sadržaj cijele poruke, no vidimo i referencu na tu lokaciju u drugoj funkciji. Koristimo funkcionalnost programa Ghidra i otvorimo decompile prozor. Pratimo funkciju te dolazimo do funkcije koja se čini kao main funkcija.

Decompile: FUN_140001160 - (product.exe)

```
1
2 void FUN_140001160(int param_1,longlong param_2,undefined8 param_3,undefined8 param_4)
3
4 {
5   code *pcVar1;
6
7   if (param_1 != 2) {
8     FUN_140001010("Please enter your serial key number to activiate the product. ",param_2,param_3,
9                   param_4);
10    FUN_140001010("e.g -> ./product 1111-2222-3333-4444\n",param_2,param_3,param_4);
11                    /* WARNING: Subroutine does not return */
12    exit(1);
13  }
14  FUN_140001070(*(char **)(param_2 + 8),param_2,param_3,param_4);
15  pcVar1 = (code *)swi(3);
16  (*pcVar1)();
17  return;
18 }
19
```

Slika 6.12: crackme 13

Primjećujemo da je if uvjet jedan jedini i da ukoliko je zadovoljen dobivamo dosad dobiveni ispis. Taj uvijet je da pri pokretanju, programu treba pružiti argument koji služi kao serijski ključ. Ukoliko ne pružimo ključ, ili ga pružimo u krivom formatu, ponovno će nam se ispisati poruka kako da unesemo ključ. No ako unesemo ključ u pravilnom formatu dolazi nam drugačija poruka, da je ključ nevaljan.

```
C:\Programski-kod\Diplomski rad\Crackme>product.exe 1111-2222-3333-4444
Invalid serial key number.
```

Slika 6.13: crackme 19

Primjetimo u prethodnom kodu da ukoliko if uvjet nije zadovoljen, program nastavlja izvršavanje te dolazi do sljedeće funkcije, FUN_140001070.

```
void FUN_140001070(char *param_1)

{
```

```
  int iVar1;
  uint uVar2;
  char *pcVar3;
  undefined auStack_38 [32];
  char *local_18;
  ulonglong local_10;

  local_10 = DAT_140004008 ^ (ulonglong)auStack_38;
  pcVar3 = strtok_s(param_1,"-",&local_18);
  iVar1 = atoi(pcVar3);
  if (iVar1 - 0x539U < 0x7d1) {
    pcVar3 = strtok_s((char *)0x0,"-",&local_18);
    iVar1 = atoi(pcVar3);
    if (iVar1 - 0x1ca9U < 0x7d1) {
      pcVar3 = strtok_s((char *)0x0,"-",&local_18);
      uVar2 = atoi(pcVar3);
      if ((999 < (int)uVar2) && ((uVar2 & 1) != 0)) {
        pcVar3 = strtok_s((char *)0x0,"-",&local_18);
        uVar2 = atoi(pcVar3);
        if ((999 < (int)uVar2) && ((uVar2 & 1) == 0)) {
          FUN_140001010();
                    /* WARNING: Subroutine does not return */
          exit(0);
        }
      }
    }
  }
  FUN_140001010("Invalid serial key number.\n");
                    /* WARNING: Subroutine does not return */
  exit(1);
}
```

Znamo da ključ treba sadržavati 4 seta po četri broja odvojenih minusom. S tom informacijom analiziramo kod funkcije i primjećujemo kako se nekoliko puta pokreće funkcija *strtok_s* sa delimiterom -. Možemo pretpostaviti, ali i potvrditi brzom provjerom na cppreference web sjedištu, kako se radi o funkciji koja služi za odvajanje niza znakova prema danom delimiteru. Nadalje primjećujemo kako se taj odvojeni niz znakova koristi u *atoi* funkciji, koja se koristi za pretvaranje znakovnih nizeva raznih vrsta u integer vrijednost. Kada su nizovi od 4 broja zasebno odvojeni i pretvoreni u integere, nad svakim se vrši daljnja provjera, što vidimo pojavom 4 različita if uvjeta. Zbog jednostavnosti će se svi nazivi različitih varijabli brojeva ključa

prezentirati kao x.

```
x - 0x539U < 0x7d1
```

Vidimo uvijet te imamo x koji je varijabla tipa integer. Kako bismo lakse usporedili, pretvorimo heksadekadske brojeve u decimalne da dobijemo:

```
x - 1337U < 2001
```

Napravimo jednostavnu matematičku operaciju da dobijemo prvi uvjet: **x <= 3337** No primjetimo simbol U, on nam označava unsigned integer. Unsigned integer je broj isti kao i integer osim toga što nema vodeću pozitivnu ili negativnu oznaku, već može biti samo pozitivan broj ili 0. To nam je odmah drugi uvjet odnosno donja granica, ukoliko razlika x i 1337 mora biti veća od 0, znači da je drugi uvjet: **x >= 1337**.

Istom logikom dobijemo u uvjete za drugi broj serijskog koda, koji su: **x >= 7337** i **x <= 9337**.

Vidimo da su daljnje provjere drugačije:

```
(999 < (int)x) && ((x & 1) != 0)
```

Prvi dio provjerava da je **x <= 1000** dok u drugom djelu vidimo x & 1. U ovom kontekstu gledamo broj x i binarno ga množimo sa brojem 1. To rezultira vrijednosti točno ili krivo, bazirano na tome jeli zadnja znamenka binarnog broja x 0 ili 1, odnosno jeli broj x paran ili neparan (binarni brojevi koji završavaju brojem 1, su uvijek neparni i obrnuto). Nakon dobivenog rješenja i provjere nejednakosti sa nula, zaključujemo da je drugi uvjet ove provjere taj da **x mora biti neparan broj**.

Kao i prije, posljednji je uvjet sličan prethodnome te su uvjeti **x <= 1000** i **x je paran broj**.

Koristeći sve navedene uvjete, napišimo jednostavan program koji će nasumično generirati jedan od mnogo mogućih ispravnih ključeva te ga ispišimo.

```
from random import randrange

def main():
    firstPart = randrange(1337, 3338)    # 1337 <= x <= 3337
    secondPart = randrange(7337, 9338)  # 7337 <= x <= 9337
    thirdPart = randrange(1001, 10000, 2)    # x >= 1000 and x == oddNumber
    fourthPart = randrange(1000, 9999, 2)    # x >= 1000 and x == evenNumber

    key = '-'.join([str(part) for part in [firstPart, secondPart, thirdPart, fourthPart

    return f'Key: {key}'

if __name__ == '__main__':
    print(main())
```

Ukoliko probamo unesti ispisani ključ, dobivamo potvrdni rezultat da je proizvod aktiviran,

```
PS C:\Programski-kod\Diplomski rad>
ki rad/Crackme/keygen.py"
Key: 2995-8854-4065-8388
```

Slika 6.14: crackme 17

odnosno da smo uspješno uspjeli obrnuto inženjerati algoritam za provjeru serijskog ključa.

```
C:\Programski-kod\Diplomski rad\Crackme>product.exe 1869-8481-2411-1510
Congratz!, Product Activated Successfully!
```

Slika 6.15: crackme 18

Ovaj keygen sada može ispisivati mnogo uspješnih serijskih ključeva.

6.4 Modiranje i društvo

U današnje vrijeme sve je više priče o tome kako su modovi crackovi. Poistovjećenje tih dviju stvari bi u nekim slučajevima moglo biti isto, no većina ljudi se ne slaže. Modovi su modifikacije, poboljšanja ili dodaci napravljeni za promjenu iskustva sa nekim softverom, aplikacijom ili najčešće igrom. Fokusirajući se primarno na moddiranje igara, promjene mogu biti od jako malih poput mijenjanja broja nekog resursa do onih koji totalno promijene cijelu igru, mijenjajući temeljne funkcionalnosti i nadodajući vlastiti sadržaj. Najčešći modovi uključuju modove koji su “quality of life”, odnosno olakšavaju igru jer popravljaju male probleme koje su programeri igre propustili ili dodaju nove funkcionalnosti koje većina igrača prihvaća kao originalnima iako sami programeri igre to nisu tako zamislili. Također velika količina se fokusira na poboljšanja u grafičkoj vjerodostojnosti povećavajući rezoluciju tekstura i mijenjajući globalno osvjetljenje. Modovi kao takvi su ponos aktivne zajednice zato jer potiču napredak igre, povećuju djelokrug, popravljaju iskustva korisnika, potiču kreativnost i produžuju životni vijek igre.

Neki razvijači igara aktivno sudjeluju u razvitku modova tako što pružaju alate za kreiranje modova i bitnije, pružaju kod kako bi se modovi mogli puno brže i lakše kreirati. No to ne rade svi razvijači igara, a njihove se igre i dalje modiraju. Tu dolazi pitanje može li se modiranje i crackanje poistovijetiti. Površinski pogledano moderi također kao i crackeri provaljuju zaštitu softvera, mijenjaju temeljne funkcionalnosti funkcija i izvoze softver pun modifikacija, no to su gotovo sve sličnosti. Velika stvar što treba uzeti u obzir kod ove usporedbe je namjera. Dok crackeri ilegalno pokušavaju provaliti sigurnosne mjere softvera u svoju korist kako ne bi morali poduprijeti razvijača igre, moderi rade upravo suprotno i poboljšavaju iskustvo i svojim postupcima promoviraju igru. Cilj im je pružiti veće zadovoljstvo za sebe i druge igrače koji dijele njihovu strast. Također vrijedi napomenuti da oni razvijanjem modova ne krše EULA-u (end user license agreement), odnosno set pravila kojih se igrač igre legalno treba pridržavati.

6.4.1 Nepravedno naplaćivanje

Modovi su uvijek bili besplatni te su samo poticali donacije kreatoru, no razvitkom sve većih i kompliciranijih modova te dolaskom novih ljudi na scenu, kojima modiranje nije samo strast i hobi već prilika za zaradu, određeni modovi su se počeli naplaćivati. Implementacija vrlo zahtjevne značajke poput DLSS 3, korisna je i tražena od gotovo svakog igrača koji posjeduje grafičku karticu kompanije Nvidia. Moderi su modirali igru sa značajkom poboljšanja DLSS 3, dok igra još nije pružila tu funkcionalnost no u narednim zakrpama vjerojatno bi. S obzirom na to da ne postoje konkretno određeni zakoni o tome tko smije profitirati i na koji način od korištenja tuđe imovine makar indirektno, veliko pitanje je smije li kreator moda naplaćivati svoj mod bez davanja djela prihoda kreatoru igre. Na temelju već spomenutog velikog obujma različitih modova, mnogi se pitaju bi li se stvorila i različita pravila za različite modove.

Nedavno se dogodila situacija u kojoj je kompanija imena Rockstar prodavala ilegalno crackane verzije svojih igara [23]. Igre su sadržavale kod višestrukih scenskih cracker grupa te su sadržavale mnoge korisne značajke poput bolje kompatibilnosti za starije sustave. Ti primjerci igara izgrađenih bez DRM zaštite prezervijaju igre kao što su bile u svom izvornom formatu, bez prevencije korištenja zbog potrebe za zastarjelim primjerima sigurnosti poput korištenja CD-a. Iako Rockstar nikada nije odgovorio na mnoge upite i komentare na situaciju, mnogi se šale da bi crackeri trebali biti plaćeni za svoj rad koji se sada nalazi u legalno izdanoj verziji igre.

Poglavlje 7

Zaključak

U ovome radu analizirali smo susrete inovacija u sigurnosti i pokušaju njenog probitka. Složena priroda DRM-a, zaštite od neovlaštenog mijenjanja, zaštite od varanja, crackanja i jailbreakanja može se naveliko diskutirati bez dolaska na konkretni odgovor. Ono što je neupitno je to da su postupci u svakoj od navedenih sfera znatno oblikovali krajolik igranja i konzumacije digitalnog sadržaja kakvog poznamo danas. Većina obrađenih tematika spada u takozvanu sivu zonu. Makar se većina populacije slagala oko njima usmjerenih postupaka, neki od njih nisu potpuno legalni. Istraživanjem i boljim upoznavanjem u zamršenosti situacije svaka osoba može osobno za sebe donjeti ispravnu odluku, bila ona moralno ili financijski ispravna.

Literatura

[1] What is Digital Rights Management (DRM)? (The Definitive Guide): https://www.digitalguardian.com/blog/what-digital-rights-management

[2] Digital rights management: https://en.wikipedia.org/wiki/Digital_rights_management

[3] Napster: https://en.wikipedia.org/wiki/Napster

[4] Reconciling Mozilla's Mission and W3C EME: https://hacks.mozilla.org/2014/05/reconciling-mozillas-mission-and-w3c-eme/

[5] What Is DRM & Why Does It Exist If It's So Evil?: https://www.makeuseof.com/tag/what-is-drm-and-why-does-it-exist-if-its-so-evil/

[6] Sony BMG copy protection rootkit scandal: https://en.wikipedia.org/wiki/Sony_BMG_copy_protection_rootkit_scandal

[7] Anti-tamper software: https://en.wikipedia.org/wiki/Anti-tamper_software

[8] Address space layout randomization (ASLR): https://www.techtarget.com/searchsecurity/definition/address-space-layout-randomization-ASLR

[9] The Effectiveness of Instruction Set Randomization: https://www.cs.virginia.edu/feeb/usenix05.pdf

[10] What is code obfuscation?: https://cybersecurity.asee.co/code-obfuscation/

[11] Denuvo: https://en.wikipedia.org/wiki/Denuvo

[12] What Is Denuvo, and Why Do Gamers Hate It?: https://www.howtogeek.com/400126/what-is-denuvo-and-why-do-gamers-hate-it/

[13] What Is Anti-Cheat Software, and How Does It Work?: https://www.makeuseof.com/what-is-anti-cheat-software/

[14] Anti-Cheat. How does it work?: https://www.reddit.com/r/gamedev/comments/87i3p1/anticheat_how_does_it_work/

[15] Security Concerns About Kernel-Level Anti-Cheat in Video Games: https://ritcsec.wordpress.com/2022/08/03/security-concerns-about-kernel-level-anti-cheat-in-video-gam

[16] One Of The Wildest Console Hacks Ever: https://kotaku.com/one-of-the-wildest-console-hac

[17] Hacking the PS4, part 1, part 2, part 3: http://cturt.github.io/ps4.html, http://cturt.github.io/ps4-2.html, http://cturt.github.io/ps4-3.html

[18] What is game cracking and how can brands prevent it?: https://www.redpoints.com/blog/what-is-game-cracking/

[19] The Difference Between Decompilers, Disassemblers, Debuggers, and Hex Editors: https://danielmiessler.com/p/the-difference-between-decompilers-disassemblers-debuggers-and

[20] Cracking Software with Reverse Engineering: https://www.youtube.com/watch?v=Wbm-a-7zc4g&ab_channel=nang

[21] How hackers crack software // Reverse engineering a program and writing a keygen for it: https://www.youtube.com/watch?v=TfSFkNLYpos&ab_channel=LeetCipher

[22] Ghidra quickstart & tutorial: Solving a simple crackme: https://www.youtube.com/watch?v=fTGTnrgjuGA&ab_channel=stacksmashing

[23] ROCKSTAR GAMES CAUGHT SELLING CRACKED VERSIONS OF ITS OWN GAMES AGAIN: https://insider-gaming.com/rockstar-games-caught-selling-cracked-versions-