ASIGURAREA SECURITĂȚII ȘI CALITĂȚII SISTEMELOR INFORMATICE PRIN TEHNICI DE INGINERIE INVERSĂ

Octavian MAGDEI

Departamentul Inginerie Software și Automatică, SI-231M, Facultatea Calculatoare Informatică și Microelectronică, Universitatea Tehnică a Moldovei, Chișinău, Republica Moldova

Autorul corespondent: Octavian Magdei, magdei.octavian@isa.utm.md

Îndrumătorul/coordonatorul științific **Rodica BULAI**, lector universitar

Rezumat. *Ingineria inversă este o tehnică utilizată de dezvoltatori software, „etical hackeri" pentru a descoperi erori, vulnerabilități, înainte ca părțile răufăcătoare să le descopere. Ingineria inversă contribuie la îmbunătățirea securității și la identificarea punctelor tari și slabe ale sistemului informatic optimizând securitatea acestora. De asemenea, facilitează crearea de noi produse și versiuni ce contribuie la asigurarea calității produsului. Totuși, cât de util nu ar fi pentru dezvoltatori ingineria inversă, aceasta poate fi folosită și în scopuri negative, cum ar fi crearea de clone de malware, versiuni trojanizare ale softurilor legitime. Articolul scoate in evidență beneficiile aduse de ingineria inversă ca unealtă care poate fi folosită atât pentru îmbunătățirea securității și calității sistemelor informatice, cât și ca recomandări privind prevenirea utilizării abuzive și neetice.*

Cuvinte cheie: *securitate cibernetică, inginerie inversă, identificarea vulnerabilităților, remedierea erorilor.*

Introducere

Ingineria inversă este procesul de analiză și înțelegere a designului, structurii și funcționalității unui produs sau sistem, lucrând înapoi de la forma sa finală [1].

Aceasta implică dezasamblarea unui obiect sau software pentru a-i descoperi funcționarea interioară și a înțelege cum a fost creat.

Cu toate acestea, este important de reținut că, deși ingineria inversă în sine este legală, utilizarea acesteia pentru a încălca drepturile de proprietate intelectuală, cum ar fi copierea produselor brevetate sau a software-ului proprietar, este ilegală.

Tehnica de inginerie inversă

Sunt cunoscute două tipuri de inginerie inversă:

Inginerie inversă statică Aceasta implica examinarea codului fără a-l executa. Cu analiza statică, experții analizează codul binar sau sursă pentru a extrage informații despre variabile, funcții și multe altele, ce le permite sa identifice vulnerabilități și să înțeleagă structura soft-ului [2].

Inginerie inversă dinamică. Implică examinarea comportamentului și funcționalității unui sistem sau software în timpul rulării, ceea ce permite experților să înțeleagă modul cum interacționează cu diferite componente și să identifice funcționalități ascunse [3].

Pașii ingineriei inverse reprezintă esența procesului de inginerie inversă, furnizând o metodologie riguroasă și sistematică pentru analiza detaliată și înțelegerea sistemelor și software-urilor:

Examinarea software-ului sau sistemului. Acest pas implică efectuarea unei examinări meticuloase a software-ului sau sistemului. Obiectivul este de a identifica și de a analiza funcționalitățile, componente și alte aspecte relevante ale sistemului, pentru a obține o înțelegere comprehensivă a funcționării sale înainte de a iniția analiza detaliată.

Analiza și cercetarea. În această etapă, se efectuează o analiză exhaustivă a componentelor și funcționalităților sistemului. Acest proces implică consultarea documentației relevante, examinarea codului sursă (dacă este disponibil), utilizarea instrumentelor de analiză statică sau dinamică și alte metode de investigare pentru a obține o înțelegere detaliată a arhitecturii și funcționalităților sistemului [4].

Dezasamblarea. Dezasamblarea este procesul de conversie a codului binar al sistemului într-o formă mai accesibilă pentru analiză. Această etapă poate implica utilizarea unor instrumente specializate pentru a traduce codul binar în limbaj de asamblare sau într-un cod sursă similar, care este mai ușor de înțeles și de analizat decât codul binar brut. Instrumentele fiind Ghidra, IDA, și altele [5].

Extragerea informațiilor. În această etapă, se extrag informațiile relevante sau semnificative din codul dezasamblat sau din alte surse de date obținute în timpul analizei. Această activitate poate implica identificarea algoritmilor esențiali, a funcțiilor critice, a vulnerabilităților de securitate sau a altor aspecte relevante ale sistemului, care pot fi valoroase pentru înțelegerea sa sau pentru realizarea de modificări sau îmbunătățiri.

Utilizarea ingineriei inverse și resurse existente

Sunt utilizate următoarele instrumente de inginerie inversă:

Debugger(dbg). Este un depanator de programare care poate fi folosit și pentru a face inginerie inversă a codului binar. Permite analiza conținutul registrelor și memoriei în timp ce se execută codul de asamblare [6]. Punctele de întrerupere pot fi plasate oriunde în program.

Disassemblers. Un „disassembler” este o unealtă software utilizată pentru a traduce codul mașină al unui program în codul asamblare corespunzător. Aceasta permite utilizatorilor să analizeze și să înțeleagă comportamentul programului, să identifice vulnerabilități și să efectueze depanarea.

Decompilatoare. Un „decompiler” este o unealtă software utilizată pentru a traduce codul mașină al unui program într-un limbaj de nivel înalt sau în cod sursă. Acesta facilitează înțelegerea și modificarea programelor.

Ingineria inversă poate fi utilizată pentru a efectua o serie de sarcini de securitate cibernetică, inclusiv găsirea defectelor sistemului, cercetarea programelor malware și a virușilor și determinarea și recuperarea algoritmilor software critici care pot ajuta la prevenirea fraudei [7]. Aceasta este, de asemenea, valoroasă pentru colectarea de informații precise, crearea de modele și prototipuri [8].

Ingineria inversă este o modalitate excelenta de a înțelege limitările unui proiect. Ca exemplu dezvoltarea unui produs pentru platformele iOS sau Android, dar nu exista aplicații pentru nici una din ele, aceasta înseamnă ca acestei tehnologii îi este greu sa imite funcționalitatea aplicației pentru aceste platforme [9].

Integrarea cu succes a api-urilor terțe. Problema constă în bibliotecile învechite sau fără documentație. Daca aplicația nu rulează cum trebuie, va fi dificil de a detecta sau remedia, aici ingineria inversă ne va permite sa analizam biblioteca și sa găsim de ce eșuează.

Concluzii și beneficii

Dintre beneficiile analizate pot fi menționate:

- Găsirea și corectarea erorilor din programele la care dezvoltatorii nu au acces la codul sursă;
- Analiza virușilor pentru a crea software antivirus;
- Îmbunătățirea funcționalității aplicației atunci când este imposibil de contactat dezvoltatorul anterior;
- Învățare: dacă un dezvoltator nu înțelege o parte din produsul său, el poate analiza alt cod și poate trage concluzii despre funcționarea acestuia.

În concluzii, ingineria inversă este o metodă eficientă de a accelera dezvoltarea unui soft și a micșora costurile. Este o metodă eficienta de a găsi limitele unui soft, de a găsi probleme de performanță și de securitate. Ingineria inversă ajută la îmbunătățirea, codului vechi, dar și poate inspira pentru a utiliza idei din acel proiect mai vechi.

Referințe

[1] M. K. Khachouch, A. Korchi, și Y. Lakhrissi, „Architecture Driven Modernization: A Review on Reverse Engineering Techniques based on Models' Approach", *WSEAS Transactions on Information Science and Applications*, vol. Volume 20, 2023, pp. 293–302, oct. 2023, doi: 10.37394/23209.2023.20.32.

[2] A. S. Basutakara, „A Review of Static Code Analysis Methods for Detecting Security Flaws", vol. 23, nr. 6, 2021.

[3] „A Deep Dive into Static vs Dynamic Code Analysis", BuildPiper. Data accesării: 15 aprilie 2024. [Online]. Disponibil la: https://www.buildpiper.io/blogs/a-deep-dive-into-static-vs-dynamic-code-analysis/

[4] S. Sotnik, V. Lyashenko, și T. Schakurova, „Modern Integrated Software Development Environments", vol. 5, nr. 10, 2021.

[5] Z. Chen, E. Brophy, și T. Ward, „Malware Classification Using Static Disassembly and Machine Learning". arXiv, 10 decembrie 2021. doi: 10.48550/arXiv.2201.07649.

[6] A. Pereberina, A. Kostyushko, și A. Tormasov, „An approach to dynamic malware analysis based on system and application code split", *J Comput Virol Hack Tech*, vol. 18, nr. 3, pp. 231–241, sep. 2022, doi: 10.1007/s11416-021-00412-z.

[7] X. He și R. Li, „Malware detection for container runtime based on virtual machine introspection", *J Supercomput*, vol. 80, nr. 6, pp. 7245–7268, apr. 2024, doi: 10.1007/s11227-023-05727-w.

[8] S. Supriyono, „Software Testing with the approach of Blackbox Testing on the Academic Information System", *IJISTECH (International Journal of Information System and Technology)*, vol. 3, nr. 2, Art. nr. 2, mai 2020, doi: 10.30645/ijistech.v3i2.54.

[9] M. Anusha și M. Karthika, „A Comprehensive Analysis on Various Deep Learning Techniques for Malware Detection in Android Mobile Devices", *SN COMPUT. SCI.*, vol. 4, nr. 5, p. 593, aug. 2023, doi: 10.1007/s42979-023-01894-y.