BUCHAREST UNIVERSITY OF ECONOMIC STUDIES

Faculty of Economic Cybernetics, Statistics and Informatics

Department of Economic Informatics and Cybernetics

APPLIED REVERSE ENGINEERING AND VULNERABILITY RESEARCH IN CYBERSECURITY

INGINERIE INVERSĂ ȘI ANALIZA VULNERABILITĂȚILOR APLICATE ÎN SECURITATEA CIBERNETICĂ

– PhD Thesis –

Sinteza tezei de doctorat

Ph.D. Candidate: ȘTEFAN SABIN NICULA

Scientific Supervisor: Prof. RĂZVAN DANIEL ZOTA, Ph.D.

ABSTRACT

Această teză explorează metodologii avansate de securitate cibernetică, cu un accent pe ingineria inversă, cercetarea vulnerabilităților și tehnicile de exploatare în cadrul sistemelor de operare Windows, în special în contextul Internetului Lucrurilor (IoT). Studiul subliniază provocările analizei mediilor software complexe, cu sursă închisă, cu un accent special pe sistemele bazate pe Windows, care diferă semnificativ de mediile bazate pe Unix în ceea ce privește arhitectura kernelului și mecanismele de securitate.

Teza oferă o metodologie cuprinzătoare care integrează ingineria inversă, fuzzing-ul și analiza malware pentru a identifica și atenua vulnerabilitățile în medii software complexe. Aspectele cheie includ explorări detaliate ale securității kernelului Windows, exploatarea browserelor și aplicarea tehnicilor de fuzzing pentru a descoperi defecte de securitate. Lucrarea propune, de asemenea, soluții inovatoare, cum ar fi utilizarea honeypot-urilor pentru detectarea în timp real a exploatărilor, depanarea automată a browserelor pentru a descoperi vulnerabilități de tip zero-day și integrarea învățării automate în analiza malware pentru a eficientiza procesele de detectare. Windows Host Honeypot with Hypervisor Monitoring este o metodă inovatoare pentru detectarea exploatărilor în mediul real prin configurarea unui honeypot pe un host Windows, combinată cu monitorizarea bazată pe hipervizor. Automated Browser Exploit Detection reprezintă soluția care implică utilizarea depanării automate și a instrumentelor de igienizare a memoriei pentru a monitoriza exploatările browserului. Automated Malware Analysis with Machine Learning abordează provocarea analizei în masă a malware-ului, această abordare integrând învățarea automată cu depanarea virtuală a kernelului. Reverse Engineering Techniques for Cybersecurity Applications face parte din cercetare și oferă un ghid detaliat privind aplicarea tehnicilor de inginerie inversă la diverse provocări de securitate cibernetică, inclusiv analiza malware, cercetarea vulnerabilităților și analiza binară.

Aceste soluții contribuie la domeniu prin oferirea unor abordări inovatoare de securitate cibernetică proactivă, în special în medii în care metodele tradiționale pot fi insuficiente. Cercetarea subliniază necesitatea integrării tehnicilor avansate de inginerie inversă cu cadrele de securitate existente pentru a proteja mai bine împotriva amenințărilor emergente.

CUVINTE CHEIE: REVERSE ENGINEERING, VULNERABILITY RESEARCH, EXPLOIT DEVELOPMENT, WINDOWS SECURITY, MALWARE ANALYSIS, FUZZING, BROWSER EXPLOITATION, INTERNET OF THINGS (IOT)

Cuprins

INTRODUCERE

Procesul de inginerie inversă poate fi aplicat cu succes în multe domenii diferite ale securității cibernetice. Acesta reprezintă un flux de lucru și o combinație de tehnici pentru a înțelege software-ul cu sursă închisă, având la dispoziție o documentație minimă sau inexistentă. Rezultatele așteptate includ informații clare despre modul de funcționare internă a software-ului, rutinele interne și scopul eșantionului analizat. Procesul include dezasamblarea unui fișier binar, decompilarea, depanarea, analiza statică și dinamică. Adesea, acest proces este folosit în acțiuni precum analiza malware și cercetarea vulnerabilităților.

Domeniul cercetării vulnerabilităților este un subiect în continuă creștere, care a început să atragă multă atenție în ultimul deceniu. Poate fi comparat cu programele de tip bug-bounty, bine cunoscute, în care cercetătorii își concentrează eforturile pe testarea aplicațiilor web, a aplicațiilor mobile, a rețelelor sau pe revizuirea codului sursă. Cu toate acestea, cercetarea vulnerabilităților în fișiere binare și aplicații desktop necesită abordări complet diferite, iar această diferență se reflectă și în exploatările și clasele de vulnerabilități. Totuși, cele două domenii au același scop, și anume descoperirea și raportarea vulnerabilităților în diferite soluții software.

Ingineria inversă poate fi aplicată în multe domenii diferite, nefiind neapărat legată de software-ul în sine, ci mai degrabă de un concept de abordare a unui mediu închis cu scopul de a-l înțelege prin testare constantă și observarea rezultatelor obținute. Ca o descriere generală, ar putea fi definită ca o înțelegere profundă a unei tehnologii auto-conservatoare, care a fost destinată să fie păstrată privată sub forma unei documentații inexistente sau a unui suport minim în depanarea soluției prezentate.

Similar, în ultimul deceniu, putem observa clar o tendință ascendentă în industria Internetului Lucrurilor (IoT), pe măsură ce tot mai multe dispozitive ajung pe piață în fiecare an. Maturitatea securității în domeniul IoT este în creștere, totuși, judecând după numărul de vulnerabilități raportate public care afectează aceste soluții, putem observa o probabilitate ridicată de identificare a vulnerabilităților ușor de prins, chiar și la cei mai mari furnizori.

În ceea ce privește dezvoltarea dedicată pentru software-ul și hardware-ul IoT, cercetarea și

dezvoltarea continuă au dus la soluții personalizate destinate anumitor dispozitive. Cu toate acestea, la un nivel general, majoritatea soluțiilor de pe piață se bazează pe sisteme de operare similare, biblioteci publice și software general care este utilizat în diverse industrii și nu este neapărat dezvoltat pentru programe legate de IoT.

Teza abordează o varietate de subiecte legate de tehnicile de exploatare, structurile interne ale sistemului de operare, procesul de inginerie inversă, depanare și instrumentele utilizate, incluzând capitole dedicate fiecărui subiect cu exemple practice. Scopul principal al cercetării este axat exclusiv pe sistemul de operare Windows cu suport pentru CPU Intel x86 și x64. Spre deosebire de mediile bazate pe Unix, Microsoft adoptă o abordare diferită în ceea ce privește dezvoltarea kernelului și implementarea mecanismelor asociate acestuia. În cele din urmă, conceptele la nivel înalt sunt aceleași ca în cazul oricărui alt sistem de operare major. Afinitatea mea pentru software-ul bazat pe Windows și provocarea componentelor kernel cu sursă închisă au fost unul dintre principalele motive pentru realizarea acestei cercetări.

Nivelul actual de documentare și cunoștințele generale despre exploatările bazate pe Windows sunt dispersate în diverse surse. Dezvoltarea exploatărilor în software și pentru ținte de mare valoare reprezintă încă un proces de învățare bazat exclusiv pe curiozitatea și dorința de auto-învățare a individului. Mai mult, putem observa o diferență semnificativă în cercetarea publică, prezentările și cursurile de formare care oferă suport pentru tehnologiile bazate pe Windows, în comparație cu mediile bazate pe Unix, o diferență semnificativă fiind observată în industria IoT. Acest lucru nu ar trebui corelat cu o lipsă de resurse sau de cercetări semnificative pe partea Windows, deoarece în ultimii 5 ani au fost publicate multe lucrări care abordează acest subiect cu cercetări tehnice bazate pe diferite scenarii exclusive și altele care au avut o abordare mai generală.

Fiind o nișă extrem de specializată în securitatea software, aceasta reprezintă o provocare atunci când vine vorba de colectarea materialelor și a referințelor pentru a dezvolta un flux de lucru solid și o fundație către atingerea unui nivel complex de înțelegere a procesului de inginerie inversă a unei soluții foarte complexe, cum ar fi un kernel de sistem de operare sau un browser. Mai mult, tehnicile de exploatare și mecanismele de descoperire a vulnerabilităților sunt extrem de personalizate în funcție de vectorii de atac și de mediul exploatat. Odată cu creșterea implementării mecanismelor de protecție și atenuare, aceste provocări devin tot mai mari, deoarece unele tehnici devin învechite, în timp ce altele suferă îmbunătățiri majore pentru a ocoli protecțiile.

OBIECTIVELE TEZEI

Obiectivul principal al tezei de cercetare este identificarea, definirea, testarea și aplicarea conceptelor de inginerie inversă pe un spectru larg de testare a software-ului la nivel scăzut, cu suport pe sistemul de operare Windows pe arhitecturi x86 și x64, dintr-o perspectivă de cercetare a vulnerabilităților. Acest lucru poate fi realizat prin înțelegerea suprafeței de atac, a abordării de securitate ofensivă a actorilor externi în căutarea unei vulnerabilități de tip 0-day în soluții complexe și a procesului de utilizare a tehnicilor de exploatare pentru a obține un cod de exploatare complet funcțional. Un alt punct cheie este evaluarea abuzului și a utilizării în medii reale a unor astfel de exploatări și aplicarea tehnicilor identificate pentru a cerceta și crea o propunere relevantă care să poată preveni sau detecta comportamentul malițios pe software-ul sau pe mașina victimei. Un ultim aspect al cercetării este furnizarea de exemple și o bază de cercetare aplicabilă, care poate fi utilizată pentru a identifica proactiv vulnerabilitățile și pentru a le raporta furnizorilor, cu scopul de a limita expunerea și de a crește nivelul general de securitate.

Necesarul cercetării actuale este determinat de lipsa unei legături între diverse scrieri și lucrări publice care se întind pe industrie, inclusiv prezentări la conferințe, postări de bloguri de afaceri și tehnice, și lucrări academice. Scopul cercetării este de a încadra mai multe contexte ale ingineriei inverse, aplicând aceleași cunoștințe de bază în ceea ce privește tehnicile de analiză utilizate, mentalitatea în abordarea perspectivei de depanare și înțelegerea software-ului țintă. Deși multe dintre exemplele prezentate, cum ar fi exploatările de browser, atacurile asupra kernelului și eșantioanele de malware, folosesc abordări foarte diferite și au propriile cunoștințe interne, procesul de înțelegere, depanare și consolidare rămâne în mare parte același. Cunoașterea modului de aplicare a cercetării vulnerabilităților într-un mod proactiv poate ajuta la elaborarea unor tehnici de apărare care mai târziu pot atenua încercările viitoare de exploatare sau, în unele cazuri, pot refuza complet o anumită tehnică.

Analizând nivelul de cunoștințe în domeniul cercetării vulnerabilităților de tip 0-day, adâncimea tehnică și de cercetare este foarte mare; majoritatea exploatărilor și cercetărilor lansate sunt adesea legate de o înțelegere vastă a funcționării interne a acestor sisteme, până la nivelul kernelului, incluzând procesele de nivel scăzut și protecțiile de securitate care ar putea intra în conflict cu procesul de dezvoltare a exploatării. Fiecare țintă reprezintă o provocare tehnică în sine. O înțelegere amplă a procesului general echipează un cercetător cu cunoștințele

de bază necesare pentru a aborda fiecare proiect cu încredere și poate contribui la realizarea unui progres semnificativ atât dintr-o perspectivă ofensivă, cât și defensivă. În ceea ce privește căutarea vulnerabilităților de tip 0-day în mediul real, majoritatea soluțiilor active pe piață au sisteme proprietare, sursă închisă și au posibilități limitate sau inexistente de personalizare sau de adăugare a detecției la nivel local.

ORGANIZAREA ȘI STRUCTURA TEZEI

Teza este structurată în 10 capitole principale, după cum urmează:

Capitolul 1: Introducere

Acesta este primul capitol prezentat în teză și deschide subiectul cercetării cu o privire de ansamblu asupra structurii și metodologiei de cercetare utilizate. Introducerea acoperă subiecte generale legate de ingineria inversă și scopul principal al lucrării. De asemenea, explică nivelul actual de documentare în domeniu și motivația personală pentru cercetare, evidențiind necesitatea acesteia. În plus, este prezentat un scurt rezumat pentru fiecare soluție originală prezentată în teză.

Capitolul 2: Impactul economic și modelul de analiză

Continuă cercetarea inițială prin examinarea impactului economic al atacurilor desfășurate pe Internet asupra peisajului software în general. Se descriu subiecte precum campaniile de malware, cum ar fi atacurile de tip ransomware, și daunele economice create, analizând, de asemenea, diferitele puncte de pornire ale campaniilor. Printre mecanismele comune de livrare, cum ar fi atacurile de inginerie socială, se remarcă și utilizarea exploatărilor de tip zero-day sau n-day pentru vulnerabilitățile cunoscute public. Există, de asemenea, o corelație directă între cercetarea vulnerabilităților freelance și divulgarea publică prin diferite programe de securitate. Mai mult, se analizează daunele economice directe între 2021 și 2022, observându-se o creștere considerabilă.

Capitolul 3: IoT și cercetarea vulnerabilităților

Explorează peisajul Internetului Lucrurilor și oferă statistici interpretate de la dezvoltatori, incluzând riscurile de securitate, utilizarea sistemelor de operare și tendințele. Capitolul reprezintă legătura între tema principală a IoT în mediul de afaceri și scopul tezei de cercetare,

care este axat pe aspectul de securitate al dezvoltărilor specifice IoT cu suport pentru Windows. Identificăm că o parte din segmentul IoT utilizează Windows Embedded (Windows IoT) ca sistem de operare principal, care împărtășește multe similitudini arhitecturale și interne cu kernelul principal Windows NT.

Capitolul 4: Cercetarea vulnerabilităților software

Extinde ideea de a efectua cercetări asupra diferitelor vulnerabilități care afectează sistemul de operare, fișierele binare și alte soluții complexe, cum ar fi browserele. Se enumeră cele mai comune erori de securitate și particularitățile lor asociate. Perspectiva exploatării binare este comparată cu mecanismele de protecție disponibile la nivel de fișier binar și de sistem de operare. De asemenea, există o clasificare bazată pe cele mai eficiente tehnici de atac și exploatare, cu accent pe ocolirea acestor controale de securitate. Fiecare tip de vulnerabilitate este prezentat în detaliu, și, în contrast, caracteristicile de atenuare implementate la diferite niveluri. Metodele mai avansate de ocolire a protecției necesită adesea utilizarea mai multor erori de securitate care pot fi utilizate în mod conjunctiv.

Capitolul 5: Securitatea internelor Windows

Este unul dintre capitolele de suport principal care detaliază internalele Windows și arhitectura din spatele celor mai importante rutine și funcții întâlnite pe parcursul cercetării tezei. Punctele cheie prezentate includ mecanismele de ridicare a privilegiilor unui proces Windows, o caracteristică ce va fi menționată în capitolele următoare în faza de exploatare și post-exploatare. Securitatea internelor Windows este un subcapitol care urmează arhitectura kernelului Windows prin diverse subiecte precum Managerul de Obiecte și Descrierele de Securitate, tokenurile de acces ale proceselor și nivelurile de integritate, și proprietățile de execuție ale proceselor. Această cercetare oferă cunoștințe de bază pentru implementarea și analiza tehnică în cadrul evaluării practice a tezei. De asemenea, cunoașterea prealabilă a proceselor și rutinelor interne a ajutat la înțelegerea modului în care apar diferite erori de securitate în kernelul Windows și în alte sisteme software complexe, cum ar fi browserele.

Capitolul 6: Securitatea virtualizării Windows

Este un subiect specific despre internalele Windows, descris pe scurt în capitolul anterior și analizat în detaliu. Securitatea bazată pe virtualizare reprezintă un avantaj în protecția împotriva

diferitelor tehnici de exploatare care vizează sistemul de operare Windows. Oferă un strat suplimentar de privilegii și adaugă metode de restricție care sunt mai greu de ocolit, deoarece pot deveni foarte situaționale. De asemenea, explorăm suprafața de atac a kernelului Windows și analizăm modul în care driverele Windows pot fi evaluate dintr-o perspectivă de securitate. Driverele reprezintă legătura dintre modul utilizator și modul kernel, fiind întâlnite frecvent în sfera IoT, unde dispozitivele și senzorii sunt încorporați în soluția IoT pentru logica de afaceri și funcțională.

Capitolul 7: Analiza malware

Oferă detalii despre procesele de analiză malware dintr-o perspectivă defensivă, valorificând cunoștințele despre sistemul de operare pentru a monitoriza activitatea și a inginerie inversă cu succes un eșantion dat. Se prezintă detalii despre diferite mecanisme de evaziune, împreună cu soluțiile posibile pentru a ocoli mecanismele de protecție pe care un malware le-ar putea dezvolta. Primele două subsecțiuni abordează modelul de analiză statică și dinamică cu diverse exemple practice, întâlnite frecvent în analiza malware. Mai mult, deoarece limbajul de programare C/C++ este unul dintre cele mai frecvent utilizate în dezvoltarea de malware, analizăm tehnici utile pentru a facilita o sesiune optimă de inginerie inversă. Folosind cunoștințele dobândite din capitolele anterioare și din cel despre analiza malware, se prezintă o soluție bazată pe hipervizor pentru detectarea automată a exploatărilor în mediul real. Sistemul poate fi folosit și pentru depanarea probelor de concept, o tehnică care poate oferi informații valoroase ce pot ajuta la crearea de detecții generice.

Capitolul 8: Fuzzing

Este o componentă cheie a domeniului de cercetare în securitatea ofensivă, care poate fi utilizată pentru a identifica automat erori de securitate în codul software-ului analizat. Există diferite tipuri de procese și metode de fuzzing. În acest capitol, trecem în revistă câteva dintre principalele metode de implementare a fuzzing-ului în cazurile în care cercetătorul are acces la cod sau dintr-o abordare black-box. Rezultatele arată că unele dintre componentele cheie ale fiecărei campanii de fuzzing sunt metodele de instrumentare și colectare a corpusului, combinate cu un mod optim de harnessing.

Capitolul 9: Exploatarea browserului

Prezintă o exploatare practică a unui CVE public care afectează browserul Chromium, care este o vulnerabilitate de tip use-after-free într-un proces Chromium sandboxat. Analizăm codul sursă public disponibil și tragem concluzii bazate pe procesul intern de depanare pentru a obține primitive de citire și scriere prin valorificarea rutinelor și structurilor interne V8. În plus, WebAssembly este utilizat pentru a crea un shellcode și a-l plasa în memoria heap folosind tehnici de heap spraying. În final, exploatarea este executată, iar rezultatele arată că procesul Chromium rulează într-un mediu sandboxat, iar o altă vulnerabilitate situată la nivelul sistemului de operare este necesară pentru a crea un bypass. Rezultatele întăresc ideea că securitatea unui sistem este strâns legată de toate componentele acestuia, iar sistemul de operare de bază joacă un rol major în securitatea unui produs.

Capitolul 10: Concluzii

Este capitolul final care prezintă concluziile tezei, toate contribuțiile rezultate din cercetare, diseminarea publicațiilor pe parcursul perioadei de cercetare și lucrările viitoare.

Această teză explorează un cadru integrat de analiză a securității, care combină ingineria inversă, fuzzing-ul, analiza malware-ului și cercetarea vulnerabilităților. Focalizată pe sistemele IoT ce utilizează sisteme de operare Windows, studiul își propune să ofere fundamentele și metodele necesare pentru descoperirea și minimizarea vulnerabilităților, îmbunătățind postura de securitate împotriva amenințărilor cibernetice sofisticate.

Creșterea incorporării dispozitivelor IoT în diverse sectoare necesită o securitate sporită datorită interconectivității lor complexe și a naturii critice a aplicațiilor lor. Această lucrare subliniază importanța unei abordări holistice a securității prin integrarea ingineriei inverse cu practicile tradiționale de securitate pentru a proteja sistemele IoT împotriva vulnerabilităților și atacurilor, în special în mediile Windows. Structura metodologiei urmate în cercetarea tezei este prezentată astfel:

1. Realizarea unei revizuiri a literaturii și obținerea cunoștințelor tehnice

- Provocările de securitate ale IoT și software-ului: Problemele actuale și considerațiile de securitate specifice sistemelor IoT și software-ului.
- Cercetarea vulnerabilităților în Windows și internalele acestuia: O privire de ansamblu

asupra vulnerabilităților inerente sistemului Windows utilizat în contexte IoT și detalii despre procesele relevante de securitate interne ale Windows aplicate subiectului tezei.

- Tehnici de fuzzing: Rolul fuzzing-ului în testarea proactivă a securității și campanii de fuzzing aplicate.
- Analiza malware-ului: Examinarea tehnicilor de analiză a malware-ului și impactul atacurilor de malware asupra sistemelor Windows.
- Ingineria inversă: Perspective asupra modului în care ingineria inversă poate ajuta la înțelegerea și securizarea software-ului, concentrându-se pe tehnici și metode de inginerie inversă.

2. Metodologia cercetării

- Configurarea sistemului: Configurarea unui mediu experimental IoT care include sistemul de operare Windows. Obținerea cunoștințelor tehnice despre practicile de inginerie inversă și evaluarea proceselor sistemului de operare.
- Inginerie inversă: Analiza software-ului pentru a identifica defecte de securitate în funcționalitățile nedocumentate.
- Analiza vulnerabilităților: Utilizarea unor instrumente specializate pentru a efectua ingineria inversă în browsere.
- Implementarea fuzzing-ului: Adaptarea instrumentelor de fuzzing pentru a identifica vulnerabilități în cadrul sistemului de operare.
- Simularea și analiza malware-ului: Introducerea de malware pentru a observa interacțiunile cu vulnerabilitățile existente și cu mecanismele de apărare ale sistemului.

3. Cercetarea vulnerabilităților și a ingineriei inverse

- Identificarea vulnerabilităților prin inginerie inversă: Tehnici de descoperire sau depanare a vulnerabilităților ascunse și a slăbiciunilor de securitate folosind ingineria inversă.
- Evaluarea impactului și strategiile de atenuare: Evaluarea riscurilor asociate cu vulnerabilitățile identificate și propunerea de strategii robuste de atenuare și detectare.

4. Rezultate și discuții

- Beneficiile integrării: Evaluarea modului în care combinarea ingineriei inverse cu

fuzzing-ul și analiza malware-ului oferă o înțelegere mai profundă a provocărilor de securitate în mediul Windows.

- Soluții propuse: Prezentarea soluțiilor originale care combină toate tehnicile discutate în teză.
- Concluzii: Concluzii rezumative care sintetizează descoperirile critice și oferă recomandări concrete pentru îmbunătățirea securității. Schițarea protocoalelor și măsurilor de securitate sugerate pe baza analizei tezei.

5. Lucrări viitoare

- Tehnici avansate de inginerie inversă: Dezvoltarea ulterioară a metodelor de inginerie inversă pentru a îmbunătăți detectarea și atenuarea amenințărilor sofisticate.
- Analiza securității pe platforme multiple: Extinderea cadrului de cercetare pentru a include sistemele IoT care operează pe platforme diferite.

CONTRIBUȚII ORIGINALE

Următoarele idei sunt prezentate ca posibile soluții concepute pentru a aduce îmbunătățiri sau pentru a rezolva problemele identificate în timpul etapei de cercetare, concentrându-se în special pe mecanisme automate pentru a integra tehnici de inginerie inversă programatică cu interpretări complexe:

1. Identificarea exploatărilor utilizate în mediul real printr-un honeypot Windows host și un proces de monitorizare bazat pe hipervizor. Procesul de depanare a vulnerabilităților kernelului Windows poate fi utilizat împreună cu sanitizatoare de adrese pentru a capta exploatări de tip zero-day prin expunerea unor servicii specifice pe internetul public. Jurnalele Windows sunt foarte ample, iar vizarea tuturor serviciilor la nivel de sistem poate crea o problemă de stocare și procesare; totuși, limitarea la caracteristici specifice, cum ar fi protocolul RDP, SMB sau RPC, va reduce volumul de informații recepționate și va permite depanarea automată fără a afecta performanța sistemului de operare. Porturile importante, precum cele menționate anterior, pot fi candidați ideali pentru monitorizarea împotriva atacurilor care vizează mașinile Windows expuse publicului. De asemenea, este esențial să se facă logare în afara configurației honeypot-ului, deoarece mașina virtuală de depanare poate fi considerată compromisă.

2. Identificarea exploatărilor care vizează browserele prin utilizarea capacităților de depanare automată și a sanitizatoarelor de memorie. În mod similar, utilizarea capacităților de depanare și a instrumentelor de instrumentare în timpul fuzzing-ului pe browser poate crea un mediu pentru monitorizarea exploatărilor active și a zero-day-urilor. Se poate configura un mediu similar cu instrumentarea manuală utilizată în campaniile de fuzzing, cu sanitizatoare de memorie activate într-o mașină virtuală care încarcă automat URL-uri. URL-urile pot fi colectate din depozite publice de malware, cu domenii cunoscute ca fiind malițioase. Încărcarea acestor URL-uri în mai multe browsere, având debuggere și sanitizatoare active, creează un mediu în care pot fi observate vulnerabilități potențial nedivulgate. Mașina virtuală internă utilizată pentru această automatizare trebuie să fie sandboxată într-un mediu strict, deoarece procesul de detonare va compromite întregul sistem de operare.
3. Analiza automată a malware-ului folosind învățarea automată și depanarea automată a kernelului virtual pe Windows. Analiza malware-ului în masă este una dintre problemele întâlnite în timpul procesului de triere a eșantioanelor, deoarece un cercetător este întotdeauna limitat de resurse și timp atunci când se confruntă cu munca manuală. Totuși, acest lucru poate fi automatizat folosind algoritmi de învățare automată bazată pe similaritatea fișierelor, combinată cu depanarea kernelului utilizând drivere înregistrate pentru a ocoli metodele de anti-detectare la nivelul utilizatorului. Prin monitorizarea și introspecția la nivelul kernelului, comportamentul complet al eșantionului analizat poate fi studiat prin implementarea de hooks la nivelul kernelului pe apelurile API Windows efectuate de eșantion. De asemenea, diverse steaguri statice și dinamice pot fi extrase din fișierul executabil pentru a crea o imagine detaliată a eșantionului și pentru a oferi mijloacele necesare pentru a crea detecții folosind o combinație de steaguri și proprietăți comportamentale. Atunci când se folosește învățarea automată, corpusul oferit pentru feedback negativ și pozitiv este foarte important, la fel ca și procesul de reînvățare al algoritmului, care poate fi bazat pe utilizator.
4. Un ghid despre cum și unde să se aplice tehnicile de inginerie inversă pentru diverse scopuri în securitatea cibernetică, incluzând analiza fișierelor binare PE, un subiect de cercetare care se ramifică în analiza malware-ului și cercetarea vulnerabilităților. În plus, având o înțelegere profundă a sistemului de operare subiacente care oferă suport

direct pentru subiectul analizat, este prezentată ca o abilitate obligatorie care va îmbunătăți procesul de analiză, va adăuga noi tehnici arsenalului unui cercetător și va ajuta la traducerea tehnicilor de depanare în procese automatizate care pot ușura procesul de analiză binară.

CONCLUZII

Ingineria inversă este considerată un subiect extrem de tehnic atunci când se lucrează cu software cu sursă închisă, deoarece necesită o investiție considerabilă de timp, iar complexitatea țintei analizate crește direct proporțional cu efortul necesar pentru a înțelege și a analiza cu succes domeniul de aplicare. Totuși, metodologia poate fi aplicată în multe domenii diferite, deoarece conceptele rămân în mare parte aceleași, dar tehnicile de depanare, instrumentele și cunoștințele interne diferă în fiecare caz. Rezultatul ingineriei inverse poate produce totuși rezultate foarte semnificative, care pot fi ulterior gestionate în funcție de scopul evaluării. Cercetarea realizată își propune să abordeze o gamă largă de scenarii aplicabile în care ingineria inversă poate fi utilizată în contextul cercetării vulnerabilităților și depanării software-ului, atât în scopuri ofensive, cât și defensive.

Mediul IoT conectează multe dintre tehnologiile care fac parte din procesul de creare a unui produs destinat consumatorilor. Postura generală de securitate a acestor dispozitive și aplicații de afaceri este definită de structura de securitate a întregului ecosistem. Aceasta include motoare puternice precum browserele și kernelul sistemului de operare, toate expunând diferite zone care ar putea fi vizate de atacatori. Analizând starea actuală a atacurilor și utilizarea vulnerabilităților în mediul real, putem observa o tendință tot mai mare în utilizarea exploatărilor de tip zero-day pentru a lansa campanii malware puternice împotriva sistemelor bazate pe Windows sau atacuri care exploatează vulnerabilități în browsere, uneori în combinație cu exploatări ale sistemului de operare pentru a ocoli protecțiile browserului. Cunoștințele și capacitățile necesare pentru a investiga astfel de probleme sunt extrem de tehnice; cu toate acestea, rezultatele au un impact mare asupra bazei de clienți, deoarece produsele afectate sunt utilizate de miliarde de utilizatori.

Teza combină o serie de subiecte legate prin tehnici de inginerie inversă, înțelegerea internelor și interdependențele care permit analiza și cercetarea atât din perspective ofensive, cât și defensive. Una dintre soluțiile prezentate oferă o configurație care poate fi implementată cu hipervizoare și algoritmi euristici capabili să detecteze exploatări executate pe sistem. Este

prezentat un exemplu de campanie de fuzzing, iar metricile sunt înregistrate în timp real pentru a arăta ajustările făcute pe diferite componente, cum ar fi corpusul utilizat și harnasamentul de fuzzing. Un exemplu de exploatare a unui browser este cercetat, iar pașii necesari pentru a obține execuția codului începând de la o primitivă de tip use-after-free sunt analizați. Toate cazurile practice și soluțiile abordate în această teză sunt susținute de cercetările privind internalele Windows și cunoștințele despre protecția cu hipervizorul Windows, care au ajutat la înțelegerea suprafeței de atac și a indicatorilor disponibili.

REFERINȚE

[1] Chadha, R., Shalom, G.S., Anand, V.K. and Goel, A., 2022, November. A Study on Exploit Development. In 2022 7th International Conference on Computing, Communication and Security (ICCCS) (pp. 1-7). IEEE.

[2] Tandon, A. and Nayyar, A., 2019. A comprehensive survey on ransomware attack: A growing havoc cyberthreat. Data Management, Analytics and Innovation: Proceedings of ICDMAI 2018, Volume 2, pp.403-420.

[3] Digging into the numbers one year after Log4Shel, (Dec.2022), https://www.scmagazine.com/feature/third-party-risk/digging-into-the-numbers-one-year-after-log4shell, retrieved Dec.2022

[4] 2022 hacker powered security report, (Feb.2022), https://www.hackerone.com/resources/i/1487910-2022-hacker-powered-security-report/3, retrieved March 2022

[5] What is the Cost of a Data Breach in 2023?, https://www.upguard.com/blog/cost-of-data-breach, Aug.2023, retrieved Aug.2023

[6] The latest 2023 Ransomware Statistics (updated August 2023), https://aag-it.com/the-latest-ransomware-statistics, retrieved Aug.2023

[7] National Institute of Standards and Technology, ICAT Metabase, https://nvd.nist.gov/general/visualizations/vulnerability-visualizations/cwe-over-time

[8] Zerodium Exploit Acquisition Program, https://zerodium.com/program.html, retrieved July 2022

[9] Householder, A.D., Chrabaszcz, J., Novelly, T., Warren, D. and Spring, J.M., 2020, August. Historical Analysis of Exploit Availability Timelines. In CSET@ USENIX Security Symposium.

[10] Goel, A.K., Rose, A., Gaur, J. and Bhushan, B., 2019, July. Attacks, countermeasures and security paradigms in IoT. In 2019 2nd international conference on intelligent computing, instrumentation and control technologies (ICICICT) (Vol. 1, pp. 875-880). IEEE.

[11] Pelaez, A., 9 IoT Operating Systems to Use in 2021 [List & Comparison] (May 2021), https://ubidots.com/blog/iot-operating-systems/, retrieved Nov.2021

[12] 2020 IoT Developer Survey Key Findings (Dec.2020), https://f.hubspotusercontent10.net/hubfs/5413615/2020%20IoT%C2%A0Developer%20Survey%20Report.pdf, retrieved March 2021

[13] 2022 IoT Developer Survey Key Findings (Sep.2022), https://f.hubspotusercontent10.net/hubfs/5413615/2020%20IoT%C2%A0Developer%20Survey%20Report.pdf, retrieved January 2024

[14] 2023 IoT Developer Survey Key Findings (Oct.2020), https://f.hubspotusercontent10.net/hubfs/5413615/2020%20IoT%C2%A0Developer%20Survey%20Report.pdf, retrieved January 2024

[15] Nawir, M., Amir, A., Yaakob, N. and Lynn, O.B., 2016, August. Internet of Things (IoT): Taxonomy of security attacks. In 2016 3rd international conference on electronic design (ICED) (pp. 321-326). IEEE.

[16] Chen, J., Diao, W., Zhao, Q., Zuo, C., Lin, Z., Wang, X., Lau, W.C., Sun, M., Yang, R. and Zhang, K., 2018, February. IoTFuzzer: Discovering Memory Corruptions in IoT Through App-based Fuzzing. In NDSS.

[17] The Security Development Lifecycle (Chapter 1), May 2006, https://download.microsoft.com/download/f/c/7/fc7d048b-b7a5-4add-be2c-baaee38091e3/9780735622142_SecurityDevLifecycle_ch01.pdf, retrieved Dec.2020

[18] Verizon's "Data Breach Investigations Report (DBIR) 2020", January 2020, https://enterprise.verizon.com/resources/executivebriefs/2019-dbir-executive-brief.pdf, retrieved January 2020

[19] Veracode State of Software Security Report, February 2024, https://www.veracode.com/sites/default/files/2024-02/SOSS-Report-2024.pdf, retrieved February 2024

[20] 2019 Thales Data Threat Report, 2019, https://cpl.thalesgroup.com/resources/encryption/2019/data-threat-report, retrieved Feburary 2020

[21] The Cisco 2020 Data Privacy Benchmark Study, January 2020, https://www.cisco.com/c/dam/global/en_uk/products/collateral/security/2020-data-privacy-cybersecurity-series-jan-2020.pdf, retrieved January 2020

[22] Ponemon Institute's "Cost of a Data Breach Report 2020", 2020, https://www.ibm.com/security/digital-assets/cost-data-breach-report/1Cost%20of%20a%20Data%20Breach%20Report%202020.pdf, retrieved March 2020

[23] CYBERSECURITY The new source of competitive advantage for retailers, May 2018, https://www.capgemini.com/fi-en/wp-content/uploads/sites/27/2018/05/cybersecurity-in-retail-report_v2-10.pdf, retrieved April 2020

[24] The Global State of Information Security Survey 2018, January 2018, https://www.pwc.com/sg/en/publications/assets/gsiss-2018.pdf, retrieved April 2020

[25] The Cost Of Fixing An Application Vulnerability, May 2009, https://www.darkreading.com/cyber-risk/the-cost-of-fixing-an-application-vulnerability, retrieved May 2020

[26] Sushil Kumar, Avinash Kaur, Ashish Jolly, Mohammed Baz, Omar Cheikhrouhou, 2021, Mathematical Problems in Engineering

[27] Mitre CVE Buffer Overflow search result, https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=Buffer+Overflow, retrieved May.2019

[28] Erick Leon, Ștefan D. Bruda, Counter-measures against stack buffer overflows in GNU/Linux operating systems., The International Workshop on Parallel Tasks on High Performance Computing, Procedia Computer Science 83, 2016, Volume 83, pages 1301 – 1306

[29] A detailed description of the Data Execution Prevention (DEP) feature in Windows XP Service Pack 2, Windows XP Tablet PC Edition 2005, and Windows Server 2003, (Jul.2017)

https://support.microsoft.com/en-us/help/875352/a-detailed-description-of-the-data-execution-pr evention-dep-feature-in, retrieved Dec.2018

[30] Daniel, M., Honoroff, J. and Miller, C., 2008. Engineering Heap Overflow Exploits with JavaScript. WOOT, 8, pp.1-6.

[31] Liu, D., Zhang, M. and Wang, H., 2018, October. A robust and efficient defense against use-after-free exploits via concurrent pointer sweeping. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (pp. 1635-1648).

[32] Sood, A.K. and Enbody, R.J., 2011. Browser exploit packs-exploitation tactics. VIRUS BULLETIN, Barcelona, Spain.

[33] Farah, T., Shelim, R., Zaman, M., Hassan, M.M. and Alam, D., 2017. Study of race condition: A privilege escalation vulnerability. In WMSCI 2017-21st World Multi-Conference Syst. Cybern. Informatics, Proc (Vol. 2, pp. 100-105).

[34] Yan Fen, Yuan Fuchao, Shen Xiaobing, Yin Xinchun, Mao Bing, A New Data Randomization Method to Defend Buffer Overflow Attacks, International Conference on Applied Physics and Industrial Engineering, Physics Procedia 24, Volume 24, Part C, 2012, pages 1757-1764

[35] The magic gadget, (Sep.2016), https://github.com/m1ghtym0/magic_gadget_finder, retrieved Apr.2019

[36] Cheng, L., Ilbeyi, B., Bolz-Tereick, C.F. and Batten, C., 2020, February. Type freezing: exploiting attribute type monomorphism in tracing JIT compilers. In Proceedings of the 18th ACM/IEEE International Symposium on Code Generation and Optimization (pp. 16-29).

[37] Ding, Y., Wei, T., Wang, T., Liang, Z. and Zou, W., 2010, December. Heap taichi: exploiting memory allocation granularity in heap-spraying attacks. In Proceedings of the 26th Annual Computer Security Applications Conference (pp. 327-336).

[38] Gadaleta, F., Younan, Y. and Joosen, W., 2010, January. BuBBle: A Javascript Engine Level Countermeasure against Heap-Spraying Attacks. In ESSoS (pp. 1-17).

[39] Position Independent Executables (PIE), (Nov.2012), https://access.redhat.com/blogs/766093/posts/1975793, retrieved Jan.2019

[40] Address Space Layout Randomization, (Mar.2003), https://pax.grsecurity.net/docs/aslr.txt, retrieved Feb. 2020.

[41] Bypassing ASLR - Part I, (May 2015), https://sploitfun.wordpress.com/2015/05/08/bypassing-aslr-part-i/, retrieved Dec.2018

[42] Hardening ELF binaries using Relocation Read-Only (RELRO), (Jan.2019), https://www.redhat.com/en/blog/hardening-elf-binaries-using-relocation-read-only-relro, retrieved Jan.2019

[43] Buffer overflow protection, (Jun.2018), https://en.wikipedia.org/wiki/Buffer_overflow_protection#Canaries, retrieved Jan.2019

[44] Return-to-libc Exploit, (Feb.11), https://medium.com/@nikhilh20/return-to-libc-exploit-aa3fe6fb0d69, retrieved Mar.2019

[45] Bypassing DEP with ROP (32-bit), (Dec.2017), https://bytesoverbombs.io/bypassing-dep-with-rop-32-bit-39884e8a2c4a, retrieved Mar.2019

[46] Rogowski, R., Morton, M., Li, F., Monrose, F., Snow, K.Z. and Polychronakis, M., 2017, April. Revisiting browser security in the modern era: New data-only attacks and defenses. In 2017 IEEE European Symposium on Security and Privacy (EuroS&P) (pp. 366-381). IEEE.

[47] Liu, J. and Xu, C., 2018. Pwning microsoft edge browser: From memory safety vulnerability to remote code execution.

[48] Format String Exploitation-Tutorial, https://www.exploit-db.com/docs/english/28476-linux-format-string-exploitation.pdf, retrieved Apr.2019

[49] Ryan "elfmaster" O'Neill, Learning Linux Binary Analysis, Packt, 2016

[50] How to hijack the Global Offset Table with pointers for root shells, (Apr.2006), https://www.exploit-db.com/papers/13203, retrieved Apr.2019

[51] Smashing the Stack, (Apr.2014), http://phrack.org/issues/49/14.html, retrieved Oct.2018

[52] Ryan "elfmaster" O'Neill, Learning Linux Binary Analysis, Packt, 2016

[53] Reis, C., Barth, A. and Pizano, C., 2009. Browser Security: Lessons from Google Chrome: Google Chrome developers focused on three key problems to shield the browser from attacks. Queue, 7(5), pp.3-8.

[54] Russinovich, M.E., Solomon, D.A. and Ionescu, A., 2012. Windows internals, part 2. Pearson Education.

[55] Pietrek, M., 1993. Windows Internals: The Implementation of the Windows Operating Environment. Addison-Wesley Longman Publishing Co., Inc..

[56] Yosifovich, P., Solomon, D.A. and Ionescu, A., 2017. Windows Internals, Part 1: System architecture, processes, threads, memory management, and more. Microsoft Press.

[57] Windows Access Control For pentesters — part 1, March 2021, https://medium.com/@jasemalsadi/intro-to-windows-access-control-part-1-ff4c20b918e7, retrieved April 2021

[58] Russinovich, M. and Solomon, D.A., 2009. Windows internals: including Windows server 2008 and Windows Vista. Microsoft press.

[59] Provecho, E.F., 2017. Testing User Account Control (UAC) on Windows 10.

[60] Parent Process vs. Creator Process, January 2021, https://scorpiosoftware.net/2021/01/10/parent-process-vs-creator-process/, retrieved January 2022

[61] Brown, K., 2000. Programming Windows Security. Upper Saddle River, NJ: Addison-Wesley.

[62] Govindavajhala, S. and Appel, A.W., 2006. Windows access control demystified. Princeton university.

[63] Access Token, July 2021, https://learn.microsoft.com/en-us/windows/win32/secauthz/access-tokens, retrieved July 2021

[64] Wu, J., Arrott, A. and Osorio, F.C.C., 2014, October. Protection against remote code execution exploits of popular applications in Windows. In 2014 9th International Conference on Malicious and Unwanted Software: The Americas (MALWARE) (pp. 26-31). IEEE.

[65] Eagle, C., 2011. The IDA Pro Book, 2nd Edition. No Starch Press.

[66] Wei, T., Wang, T., Duan, L. and Luo, J., 2010, October. Secure dynamic code generation against spraying. In Proceedings of the 17th ACM conference on Computer and communications security, pp. 738-740.

[67] Calbucci, M., 2000. Windows 2000 Security Descriptors. Dr. Dobb's Journal: Software Tools for the Professional Programmer, 25(11), pp.57-61.

[68] Ferguson, J., 2008. Reverse engineering code with IDA Pro. Syngress.

[69] Probert, D.B., Windows Kernel Internals Windows Service Processes. Microsoft Corporation-http://www. iu-tokyo. ac. jp/edu/training/ss/lecture/newdocuments/Lecrures/11WindowsServices/WindowsServices. ppt.

[70] Assi, M.J., Fahad, A.A. and Al-Sarray, B., 2022. Root Cause Analysis and Improvement In Windows System Based on Windows Performance Toolkit WPT. Iraqi Journal of Science, pp.5046-5057.

[71] Li, X., Wen, Y., Huang, M.H. and Liu, Q., 2011, December. An overview of bootkit attacking approaches. In 2011 Seventh International Conference on Mobile Ad-hoc and Sensor Networks pp. 428-431. IEEE.

[72] Probert, D., 2010. Windows Kernel Architecture Internals.

[73] Willems, C., 2011. Internals of Windows memory management (not only) for malware analysis. None.

[74] Gadaleta, F., Strackx, R., Nikiforakis, N., Piessens, F. and Joosen, W., 2014. On the effectiveness of virtualization-based security. arXiv preprint arXiv:1405.6058.

[75] Deogirikar, J. and Vidhate, A., 2017, February. Security attacks in IoT: A survey. In 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC) (pp. 32-37). IEEE.

[76] Isolated User Mode (IUM) Processes - Trustlets, July 2021, https://learn.microsoft.com/en-us/windows/win32/procthread/isolated-user-mode--ium--processes#trustlets, retrieved July 2021

[77] Lukacs, S., Lutas, A.V., Lutas, D.H. and Sebestyen, G., 2014, May. Hardware virtualization based security solution for embedded systems. In 2014 IEEE International Conference on Automation, Quality and Testing, Robotics (pp. 1-6). IEEE.

[78] Xiang, C., Bhagoji, A.N., Sehwag, V. and Mittal, P., 2021, August. PatchGuard: A Provably Robust Defense against Adversarial Patches via Small Receptive Fields and Masking. In USENIX Security Symposium (pp. 2237-2254).

[79] Pearce, L., 2018. Windows Internals and Malware Behavior: Malware Analysis Day 3 (No. LA-UR-18-25467). Los Alamos National Lab. (LANL), Los Alamos, NM (United States).

[80] Custer, H., 1992. Inside windows NT. Microcomputer Applications.

[81] Windows vs Linux, https://echelonlinux.org/, retrieved December 2021

[82] Cerrudo, C., 2005. Hacking windows internals. Blackhat Europe, Amsterdam, Netherlands.

[83] Sahel Alouneh, Mazen Kharbutli, Rana AlQurem, Stack Memory Buffer Overflow Protection Based on Duplication and Randomization, The 4th International Conference on Emerging Ubiquitous Systems and Pervasive Networks, Procedia Computer Science 21, 2013, pages 250 – 256

[84] Moser, A., Kruegel, C. and Kirda, E., 2007, May. Exploring multiple execution paths for malware analysis. In 2007 IEEE Symposium on Security and Privacy (SP'07) (pp. 231-245). IEEE.

[85] Gandotra, E., Bansal, D. and Sofat, S., 2014. Malware analysis and classification: A survey. Journal of Information Security, 2014.

[86] Sikorski, M. and Honig, A., 2012. Practical malware analysis: the hands-on guide to dissecting malicious software. No Starch Press.

[87] Monnappa, K.A., 2018. Learning Malware Analysis: Explore the concepts, tools, and techniques to analyze and investigate Windows malware. Packt Publishing Ltd.

[88] Yong Wong, M., Landen, M., Antonakakis, M., Blough, D.M., Redmiles, E.M. and Ahamad, M., 2021, November. An inside look into the practice of malware analysis. In Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (pp. 3053-3069).

[89] Quist, D.A. and Liebrock, L.M., 2009, October. Visualizing compiled executables for malware analysis. In 2009 6th International Workshop on Visualization for Cyber Security (pp. 27-32). IEEE.

[90] Mohaisen, A., Alrawi, O. and Mohaisen, M., 2015. AMAL: high-fidelity, behavior-based automated malware analysis and classification. computers & security, 52, pp.251-266.

[91] Liu, W., Ren, P., Liu, K. and Duan, H.X., 2011, September. Behavior-based malware analysis and detection. In 2011 first international workshop on complexity and data mining (pp. 39-42). IEEE.

[92] Bayer, U., Kirda, E. and Kruegel, C., 2010, March. Improving the efficiency of dynamic malware analysis. In Proceedings of the 2010 ACM Symposium on Applied Computing (pp. 1871-1878).

[93] Prayudi, Y. and Riadi, I., 2015. Implementation of malware analysis using static and dynamic analysis method. International Journal of Computer Applications, 117(6).

[92] Eagle, C., 2004. Attacking Obfuscated Code with IDA Pro. Black Hat.

[94] Or-Meir, O., Nissim, N., Elovici, Y. and Rokach, L., 2019. Dynamic malware analysis in the modern era—A state of the art survey. ACM Computing Surveys (CSUR), 52(5), pp.1-48.

[95] Eagle, C. and Nance, K., 2020. The Ghidra Book: The Definitive Guide. no starch press.

[96] Megira, S., Pangesti, A.R. and Wibowo, F.W., 2018, December. Malware analysis and detection using reverse engineering technique. In Journal of Physics: Conference Series (Vol. 1140, No. 1, p. 012042). IOP Publishing.

[97] Bermejo Higuera, J., Abad Aramburu, C., Bermejo Higuera, J.R., Sicilia Urban, M.A. and Sicilia Montalvo, J.A., 2020. Systematic approach to malware analysis (SAMA). Applied Sciences, 10(4), p.1360.

[98] Sun, H.M., Lin, Y.H. and Wu, M.F., 2006. API monitoring system for defeating worms and exploits in MS-Windows system. In Information Security and Privacy: 11th Australasian Conference, ACISP 2006, Melbourne, Australia, July 3-5, 2006. Proceedings 11 (pp. 159-170). Springer Berlin Heidelberg.

[99] Ucci, D., Aniello, L. and Baldoni, R., 2019. Survey of machine learning techniques for malware analysis. Computers & Security, 81, pp.123-147.

[100] Lengyel, T.K., Maresca, S., Payne, B.D., Webster, G.D., Vogl, S. and Kiayias, A., 2014, December. Scalability, fidelity and stealth in the DRAKVUF dynamic malware analysis system. In Proceedings of the 30th annual computer security applications conference (pp. 386-395).

[101] Ferreira, D.T., 2023. Automatic binary patching for flaws repairing using static rewriting and reverse dataflow analysis (Doctoral dissertation).

[102] Vasilescu, M., Gheorghe, L. and Tapus, N., 2014, September. Practical malware analysis based on sandboxing. In 2014 RoEduNet Conference 13th Edition: Networking in Education and Research Joint Event RENAM 8th Conference (pp. 1-6). IEEE.

[103] Sabanal, P.V. and Yason, M.V., 2007. Reversing C++. Black Hat DC.

[104] Egele, M., Scholte, T., Kirda, E. and Kruegel, C., 2008. A survey on automated dynamic malware-analysis techniques and tools. ACM computing surveys (CSUR), 44(2), pp.1-42.

[105] Ferenc, R., Beszédes, Á., Tarkiainen, M. and Gyimóthy, T., 2002, October. Columbus-reverse engineering tool and schema for C++. In International Conference on Software Maintenance, 2002. Proceedings. (pp. 172-181). IEEE.

[106] Bruce Dang , Practical Reverse Engineering: x86, x64, ARM, Windows Kernel, Reversing Tools, and Obfuscation, Wiley Publishing, 2014

[107] Reverse-Tips, June 2022, https://ppppz.net/2022/06/12/Reverse-Tips/, retrieved June 2022.

[108] Eldad Eilam, Reversing: Secrets of Reverse Engineering, Wiley Publishing, 2005

[109] DRAKVUF Black-box Binary Analysis System, https://drakvuf.com/, retrieved Feb. 2019.

[110] CVE-2020-1135, November 2019, https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-1135, retrieved December 2019.

[111] Sutton, M., Greene, A. and Amini, P., 2007. Fuzzing: brute force vulnerability discovery. Pearson Education.

[115] Madjid Kara, Olfa Lamouchi, Amar Ramdane-Cherif, Software quality assessment algorithm based on fuzzy logic, International Journal of ubiquitous systems and pervasive networks (JUSPN), volume 8, issue1, 2017, pages 01-09

[113] Cha, S.K., Woo, M. and Brumley, D., 2015, May. Program-adaptive mutational fuzzing. In 2015 IEEE Symposium on Security and Privacy (pp. 725-741). IEEE.

[114] Chen, C., Cui, B., Ma, J., Wu, R., Guo, J. and Liu, W., 2018. A systematic review of fuzzing techniques. Computers & Security, 75, pp.118-137.

[115] Böhme, M., Pham, V.T. and Roychoudhury, A., 2016, October. Coverage-based greybox fuzzing as markov chain. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (pp. 1032-1043).

[116] Böhme, M., Pham, V.T., Nguyen, M.D. and Roychoudhury, A., 2017, October. Directed greybox fuzzing. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (pp. 2329-2344).

[117] Pham, V.T., Böhme, M., Santosa, A.E., Căciulescu, A.R. and Roychoudhury, A., 2019. Smart greybox fuzzing. IEEE Transactions on Software Engineering, 47(9), pp.1980-1997.

[118] Zeller, A., Gopinath, R., Böhme, M., Fraser, G. and Holler, C., 2019. The fuzzing book.

[119] Serebryany, K., 2016, November. Continuous fuzzing with libfuzzer and addresssanitizer. In 2016 IEEE Cybersecurity Development (SecDev) (pp. 157-157). IEEE.

[120] Lee, B., Song, C., Jang, Y., Wang, T., Kim, T., Lu, L. and Lee, W., 2015, February. Preventing Use-after-free with Dangling Pointers Nullification. In NDSS.

[121] Österlund, S., Razavi, K., Bos, H. and Giuffrida, C., 2020, August. Parmesan: Sanitizer-guided greybox fuzzing. In Proceedings of the 29th USENIX Conference on Security Symposium (pp. 2289-2306).

[122] Zhu, X. and Böhme, M., 2021, November. Regression greybox fuzzing. In Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (pp. 2169-2182).

[123] Herrera, A., Gunadi, H., Magrath, S., Norrish, M., Payer, M. and Hosking, A.L., 2021, July. Seed selection for successful fuzzing. In Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis (pp. 230-243).

[124] Devi, D., Pathak, D. and Nandi, S., 2010. Vulnerabilities in Web Browsers. Indian Institute of Technology., Guwahati., India.

[125] Barth, A., Jackson, C., Reis, C. and TGC Team, 2008. The security architecture of the chromium browser. In Technical report. Stanford University.

[126] Analysis of a Chrome Zero Day: CVE-2019-5786, March 2020, https://www.mcafee.com/blogs/other-blogs/mcafee-labs/analysis-of-a-chrome-zero-day-cve-2019-5786/, retrieved March 2019

[127] Sotirov, A. and Dowd, M., 2008. Bypassing browser memory protections. Proceedings of BlackHat, 2008.

[128] Sayar, I., Bartel, A., Bodden, E. and Le Traon, Y., 2023. An in-depth study of java deserialization remote-code execution exploits and vulnerabilities. ACM Transactions on Software Engineering and Methodology, 32(1), pp.1-45.

[129] CVE-2019-5786 Chrome 72.0.3626.119 stable FileReader UaF exploit for Windows 7 x86, Mar. 2019, https://github.com/exodusintel/CVE-2019-5786, retrieved June 2021

[130] Manhas, S. and Taterh, S., 2018. A Comparative Analysis of Various Vulnerabilities Occur in Google Chrome. In Soft Computing: Theories and Applications: Proceedings of SoCTA 2016, Volume 1 (pp. 51-59). Springer Singapore.