



UNIVERZITA
PAVLA JOZEFA ŠAFÁRIKA
V KOŠICIACH



Financované
Európskou úniou
NextGenerationEU

PLÁN [OBNOVY]



MINISTERSTVO
INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA
A INFORMATIZÁCIE
SLOVENSKEJ REPUBLIKY

4. Prevod strojového kódu na inštrukcie a zdrojový kód

Podstata Reverzného Inžinierstva (RE)

- **Definícia:** RE je proces extrakcie znalostí alebo návrhových plánov z čohokoľvek, čo človek vytvoril. V softvéri ide o rekonštrukciu logiky bez prístupu k zdrojovému kódu.
- **Kompilácia ako stratový proces:**
 - *Abstrakcie:* Triedy, dedičnosť, šablóny (C++) sú preložené na jednoduché pamäťové offsety a nepriame skoky.
 - *Metadáta:* Názvy lokálnych premenných, komentáre a definície typov (**struct**, **union**) sú nenávratne odstránené (pokiaľ nie sú prítomné debug symboly DWARF/PDB).
 - *Optimalizácia:* Kompilátor (**-O2**, **-O3**) radikálne mení štruktúru kódu – rozbaľuje slučky (loop unrolling), “inline”-uje funkcie a eliminuje mŕtvý kód, čím sa binárna podoba vzdaľuje od logickej štruktúry zdrojového kódu.
 - LTO (Link-Time Optimization): v modernom malvéri robí statickú analýzu ešte ťažšou, pretože zlieva funkcie naprieč rôznymi zdrojovými súbormi, čím úplne ničí pôvodnú modularitu kódu.
- **Úloha analytika:** Zvrátenie entropie. Musíme "uhádnuť" pôvodný zámer autora na základe vzorov v strojovom kóde. Je to ako lúštenie krížovky, kde nemáte legendu, len počet písmen.



Anatómia ELF (Executable and Linkable Format)

Hĺbková analýza System V ABI:

- **Štandardizácia:** ELF je dominantný formát pre UNIX systémy. Jeho štruktúra nie je náhodná, ale riadi sa prísny ABI (Application Binary Interface), ktoré definuje volacie konvencie (calling conventions) a rozloženie pamäte.
- **Dva paralelné vesmíry v jednom súbore:**
 - **Linking View (Statický pohľad - pre vývojára a analytika):**
 - Založený na **Sekciách (Sections)** (napr. `.text`, `.data`, `.symtab`, `.strtab`).
 - Tento pohľad používa kompilátor a linker (`ld`) na spájanie objektových súborov (`.o`) a riešenie symbolických odkazov.
 - Pre reverzné inžinierstvo je kritický, pretože nesie sémantiku. Nástroje ako `readelf -S` zobrazujú tento pohľad.
 - **Execution View (Dynamický pohľad - pre OS Kernel):**
 - Založený na **Segmentoch (Program Headers)**. Segment je kolekcia sekcií s rovnakými prístupovými právami (R/W/X).
 - Pri volaní systémového volania `execve()`, kernel a loader (`ld-linux.so`) *ignorujú* sekcie. Zaujímajú ich iba segmenty (`PT_LOAD`), ktoré mapujú do virtuálnej pamäte pomocou `mmap()`. Príkaz `readelf -l`.
- **Malvér a Anti-Forensics techniky:**
 - **Section Header Stripping:** Keďže loader nepotrebuje tabuľku sekcií pre spustenie programu, malvér ju môže prepísať nulami alebo odstrániť.
 - **Dôsledok:** Staršie alebo jednoduchšie nástroje (napr. štandardný objdump) pri strippingu zlyhávajú. Moderné nástroje (Ghidra, IDA Pro) tento trik poznajú a dokážu si pamäťovú mapu heuristicky zrekonštruovať zo segmentov, no analytik prichádza o cenné sémantické metadáta.
 - **Riešenie:** Analytik musí manuálne zrekonštruovať sekcie na základe segmentov alebo analyzovať raw binárku bez metadát.



ELF Hlavička a vstupné vektory

- **e_ident (Magic Bytes):** Prvých 16 bajtov. Obsahuje `0x7F 'E' 'L' 'F'`.
 - *Malvér trik:* Útočníci môžu modifikovať nevyužívané bajty v paddingu tejto časti na uloženie malého shellcode alebo kľúča pre dešifrovanie.
- **e_entry (Entry Point):** Virtuálna adresa, kde začína vykonávanie (zvyčajne `_start`).
 - **EPO (Entry Point Obscuration):** Pokročilý malvér *nemení* túto adresu. Program začne bežať legitímnym kódom. Škodlivý kód je spustený až neskôr, napr. nahradením inštrukcie `CALL` hlboko v tele programu skokom na payload. Týmto obchádza antivírusové scannery, ktoré kontrolujú len Entry Point.
- **e_machine:** Architektúra CPU.
 - Ghidra využíva túto hodnotu na automatický výber jazyka Sleigh (napr. `x86:LE:64:default`).
- **e_phoff vs e_shoff:**
 - `e_phoff` (Program Header Offset) je **povinný** pre beh.
 - `e_shoff` (Section Header Offset) je **voliteľný** pre beh, ale nutný pre analýzu.



Segmentácia pamäte a bezpečnostné mechanizmy

Programové hlavičky (**Elf64_Phdr**) a riadenie pamäte:

- **PT_LOAD** Segmenty:
 - Toto sú jediné časti súboru, ktoré sa reálne kopírujú do RAM.
 - Zvyčajne existujú dva hlavné **PT_LOAD** segmenty:
 1. **Text Segment (R-X)**: Obsahuje inštrukcie. Zápis do tohto segmentu vyvolá **Segmentation Fault**.
 2. **Data Segment (RW-)**: Obsahuje globálne premenné. Vykonávanie z tohto segmentu je zakázané.
- **W^X (Write XOR Execute) / DEP**:
 - Základný bezpečnostný princíp: Stránka v pamäti môže byť buď zapisovateľná, alebo vykonateľná, nikdy nie oboje.
 - *Výnimka/Indikátor útoku*: Ak nájdete segment s právami **RWE** (7), je to silný indikátor prítomnosti **packera** (kód sa musí dekomprimovať - Write, a potom spustiť - Execute) alebo JIT kompilátora.
- **PT_DYNAMIC**: Smeruje na sekciu **.dynamic**, ktorá obsahuje zoznam potrebných knižníc (**DT_NEEDED**).
- **PT_INTERP**: Cesta k dynamickému linkeru (napr. **/lib64/ld-linux-x86-64.so.2**). Malvér môže zmeniť túto cestu na vlastný zavádzač.



Financované
Európskou úniou
NextGenerationEU

PLÁN [OBNOVY]



MINISTERSTVO
INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA
A INFORMATIZÁCIE
SLOVENSKEJ REPUBLIKY



Sekcie a ich sémantický význam (Linking View)

- **.text**: Samotný program - inštrukcie. Tu trávi reverzný inžinier 90% času.
- **.rodata (Read-Only Data)**:
 - Obsahuje reťazcové literály (`printf` formáty, chybové hlášky) a `switch` jump tables.
 - *Praktický tip*: Extrakcia reťazcov z `.rodata` je často prvým krokom analýzy malvéru (IoC - Indicators of Compromise, IP adresy, URL, cesty k súborom).
- **.data vs .bss**:
 - `.data`: Inicializované premenné (`int a = 5;`).
 - `.bss`: Neinicializované premenné (`int buffer[1024];`). V súbore nezaberá miesto, ale v pamäti áno. Malvér často využíva `.bss` ako miesto na dekompresiu svojho payloadu (tzv. "BSS staging").
- **.init_array / .fini_array (moderne nahrádzajú .init/.fini)**:
 - Pole ukazovateľov na funkcie, ktoré sa spustia *pred* `main()`.
 - Ideálne miesto pre perzistenciu malvéru alebo anti-debug techniky (napr. `ptrace` check ešte pred spustením hlavnej logiky).



Dynamické linkovanie

- **Problém:** V dôsledku ASLR (Address Space Layout Randomization) sa knižnice (`libc.so`, `kernel32.dll`) načítavajú zakaždým na inú adresu. Binárka nemôže mať "natvrdo" napísanú adresu funkcie `printf`.
- **Riešenie: Nepriama adresácia.**
 - **PLT (Procedure Linkage Table):** Trampolína v kóde (read-only).
 - **GOT (Global Offset Table):** Adresár v dátach (writable).
- **Workflow (Krok za krokom):**
 - Program zavolá `call printf@plt`.
 - PLT inštrukcia skočí na adresu uloženú v GOT.
 - *Prvé volanie:* V GOT je adresa, ktorá ukazuje späť do PLT na kód "Resolvera" (dynamic linker). Linker nájde symbol `printf` v pamäti.
 - Linker **prepíše** záznam v GOT skutočnou adresou `printf`.
 - *Druhé volanie:* Skok cez GOT vedie priamo na kód funkcie `printf`.
- **Bezpečnosť (RELRO):**
 - **Partial RELRO:** GOT je zapisovateľná (zraniteľné voči GOT Overwrite útokom).
 - **Full RELRO:** Linker vyrieši všetky symboly hneď pri štarte a potom označí GOT ako Read-Only. Spomaľuje štart, ale zvyšuje bezpečnosť.



Algoritmy Disassemblovania

Linear Sweep (napr. **objdump**, **gdb**):

- **Algoritmus:** Začni na bajte 0. Dekóduj inštrukciu. Zisti jej dĺžku **L**. Posuň sa o **L**. Opakuj do konca súboru.
- **Kritická slabina: Problém desynchronizácie.**
 - Architektúra x86 má variabilnú dĺžku inštrukcií (1 až 15 bajtov).
 - Ak sa do sekcie kódu vložia dáta (napr. **Inline Jump Table** pre **switch** príkaz), Lineárny algoritmus sa pokúsi dekodovať tieto dáta ako inštrukcie.
 - Výsledkom je "Junk Code" – nezmyselné inštrukcie, ktoré posunú počítadlo inštrukcií (IP) na nesprávne miesto, čím sa chybné dekoduje aj nasledujúci legítimný kód.
- **Zneužitie malvérom:** Útočník vloží bajt **0xE8** (CALL) tesne za inštrukciu nepodmieneného skoku (**JMP**).
Lineárny disassembler si myslí, že nasleduje volanie funkcie a "zožerie" nasledujúce 4 bajty legítimného kódu ako adresu, čím úplne rozbije analýzu.



Algoritmy Disassemblovania

Recursive Descent (napr. Ghidra, IDA Pro):

- **Algoritmus:** Riadi sa logikou toku programu (Control Flow).
 - Začína na známych vstupných bodoch (Entry Points).
 - Analyzuje inštrukcie sekvenčne, kým nenarazí na vetvenie.
 - Ak nájde **CALL** alebo **JCC** (podmienený skok), pridá cieľovú adresu do zoznamu "na spracovanie".
 - Dáta, na ktoré sa neskáče, sú ignorované (správne identifikované ako nie-kód).
- **Výzva: The Gap Problem (Problém medzier).**
 - **Nepriame skoky (Indirect Jumps):** **JMP RAX** alebo **CALL [RBX + 8]**.
 - Statický analyzátor v čase analýzy nepozná obsah registrov **RAX** alebo **RBX**.
 - Dôsledok: Disassembler nevie, kam kód pokračuje. Vznikajú "slepé miesta" (gaps) – bloky kódu, ktoré neboli objavené.
 - *Riešenie Ghidry:* Heuristická analýza konštánt (Constant Propagation) – pokúša sa vypočítať možné hodnoty registrov staticky.



Dekompilácia

- Dekompilácia nie je len preklad 1:1. Je to rekonštrukcia logiky.
- **Ghidra P-Code (Intermediárna Reprezentácia):**
 - Problém: Existujú desiatky architektúr (x86, ARM, MIPS, PowerPC, RISC-V...). Písať dekompilátor pre každú zvlášť je nemožné.
 - Riešenie: **Lifting**. Strojové inštrukcie sa najprv preložia do jednotného jazyka P-Code (Ghidra's intermediárna reprezentácia).
 - *Príklad:* Inštrukcia **ADD EAX, EBX** (x86) aj **ADD R0, R1, R2** (ARM) sa preložia na rovnakú operáciu **CPUI_INT_ADD** v P-Code.
 - Všetky analytické algoritmy potom pracujú už len nad P-Code, čo robí Ghidru univerzálnou.



Financované
Európskou úniou
NextGenerationEU

PLÁN [OBNOVY]



MINISTERSTVO
INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA
A INFORMATIZÁCIE
SLOVENSKEJ REPUBLIKY



SSA Forma a Štruktúrovanie toku

- **Data Flow Analysis (DFA) a SSA (Static Single Assignment):**
 - V assembleri sa registre neustále prepisujú (`MOV EAX, 1 ... MOV EAX, 2`).
 - V SSA forme sa vytvorí nová verzia premennej pri každom zápise (`EAX_1, EAX_2`).
 - **Phi-Nodes (*phi*):** V miestach, kde sa stretávajú dve vetvy kódu (napr. po `if-else`), vzniká špeciálna funkcia `EAX_3 = phi(EAX_1, EAX_2)`. Toto umožňuje dekompilátoru pochopiť, že ide o tú istú logickú premennú, len s rôznymi hodnotami podľa cesty.
- **Control Flow Structuring:**
 - Využíva teóriu grafov (dominátory, intervalová analýza). Identifikuje vzory ako "slučka s podmienkou na začiatku" -> `while`, "slučka s podmienkou na konci" -> `do-while`.
 - Ak graf toku riadenia nie je redukovateľný (napr. skoky "dovnútra" cyklov), dekompilátor musí použiť `goto` alebo duplikovať kód.



Financované
Európskou úniou
NextGenerationEU

PLÁN [OBNOVY]



MINISTERSTVO
INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA
A INFORMATIZÁCIE
SLOVENSKEJ REPUBLIKY



Ghidra

- **Pôvod:** Vyvinutá NSA, uvoľnená 2019 (Java + C++ decompiler).
- **Jazyk Sleigh:** Revolučný koncept. Procesory sú definované v textových súboroch .slaspec, ktoré sa kompilujú do binárnych .sla súborov.
 - Umožňuje komunitě pridať podporu pre exotické procesory (napr. v automobilovom priemysle alebo IoT) bez rekompilácie samotnej Ghidry.
- **Analyzátory (Analyzers):**
 - Pipeline skriptov, ktoré bežia po importe.
 - *Stack Analyzer:* Rekonštruuje lokálne premenné (Stack Frames).
 - *Demangler:* Prekladá zložité C++ názvy (`_ZN3Foo3barEv` -> `Foo::bar()`).
- **Headless Analyzer:** Umožňuje spustiť Ghidru z príkazového riadku bez GUI. Ideálne pre automatizovanú analýzu tisícok vzoriek malvéru v cloude.



Porovnanie nástrojov

- **Ghidra:**
 - *Výhody:* Zadarmo, masívna podpora architektúr, Undo/Redo (ktoré IDA dlho nemala), kolaborácia.
 - *Nevýhody:* Pomalšie UI pri veľkých binárkach (Java GC), dekompilátor produkuje "ukecanejší" C kód než Hex-Rays.
- **IDA Pro (s dekompilátorom Hex-Rays):**
 - *Výhody:* Zlatý štandard, extrémne rýchla, bezkonkurenčný interaktívny debugger (Linux/Windows remote debug), FLIRT (rozpoznávanie štandardných knižníc).
 - *Nevýhody:* Extrémna cena (tisíce EUR ročne), closed-source.
 - Existuje IDA Free, ktorá po novom ponúka bezplatný cloudový dekompilátor pre základné architektúry.
- **Binary Ninja:**
 - *Výhody:* Skvelé API (Python/C++), Intermediate Languages (LLIL, MLIL, HLIL) prístupné používateľovi, inovatívne vizualizácie.
- **Radare2 / Cutter:**
 - *Výhody:* "Švajčiarsky nožík", beží všade (aj na Raspberry Pi), zadarmo. Strmá krivka učenia, ale silné skriptovanie (r2pipe).
 - Často sa dnes využíva aj jeho stabilnejší fork s názvom Rizin (GUI Cutter je dnes primárne naviazané na Rizin).



Financované
Európskou úniou
NextGenerationEU

PLÁN [OBNOVY]



MINISTERSTVO
INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA
A INFORMATIZÁCIE
SLOVENSKEJ REPUBLIKY



Praktická ukážka - Linux Crackme

- **Import:** Ghidra deteguje formát ELF, x86-64, Little Endian.
- **Hľadanie vstupu:** Prechod na `_start` -> prvý argument `__libc_start_main` je adresa `main`.
- **Analýza premenných:**
 - Vidíme premennú na zásobníku, do ktorej sa ukladá vstup (`scanf`).
 - Vidíme premennú `local_10` (počítadlo) a `local_14` (súčet).
- **Rekonštrukcia logiky:**
 - Slučka iteruje znak po znaku.
 - Pseudo-kód ukazuje: `sVar1 = sVar1 + (int)input_buffer[i];`
 - Záverová podmienka: `if (strlen(input) == 20 && sVar1 % 7 == 0)`
- **Tvorba Keygenu (Solver):**
 - Nepotrebuje skúšať heslá hrubou silou.
 - Stačí matematicky vygenerovať reťazec, ktorého súčet ASCII hodnôt je deliteľný 7.



Záver

- **Statická analýza** je súboj s entropiou a snaha o pochopenie myšlienkových pochodov pôvodného programátora.
- Znalosť **ELF formátu** (Linking vs. Execution view) je kľúčová pre odhalenie techník, ktorými sa malvér snaží skryť pred nástrojmi.
- **Disassembly nie je exaktná veda**: Problém rozlíšenia dát a kódu (Halting problem) znamená, že nástroje budú vždy robiť chyby. Analytik ich musí vedieť opraviť.
- **Budúcnosť**: AI-assisted reverse engineering (LLMs trénované na assembleri) a formálna verifikácia kódu.



Financované
Európskou úniou
NextGenerationEU

PLÁN [OBNOVY]



MINISTERSTVO
INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA
A INFORMATIZÁCIE
SLOVENSKEJ REPUBLIKY





UNIVERZITA
PAVLA JOZEFA ŠAFÁRIKA
V KOŠICIACH



Financované
Európskou úniou
NextGenerationEU

PLÁN [OBNOVY]



MINISTERSTVO
INVESTÍCIÍ, REGIONÁLNEHO ROZVOJA
A INFORMATIZÁCIE
SLOVENSKEJ REPUBLIKY

Ďakujem za pozornosť.