MI-PB-12

Význam a použití aplikačního binárního rozhraní v reverzním inženýrství.

Softwarové inženýrství: studium a použití inženýrských přístupů při návrhu, vývoji a údržbě SW

Reverzní inženýrství: proces analýzy předmětného systému, abychom mohli vytvořit reprezentaci tohoto systému na vyšší úrovni abstrakce
Využití: zjištění algoritmů, metod, souborových formátů, protokolů, nalezení zranitelností, návrh spolupracujícího produktu, rozhodnutí o škodlivosti produktu

Získání zdrojového kódu ze spustitelného programu:

- **Disassemblování:** transformace spustitelného kódu ze strojového jazyka do assembleru cílového procesoru
- **Dekompilace:** transformace spustitelného kódu do vyššího programovacího jazyka

Statická analýza: na neběžícím spustitelném kódu s cílem prostudovat, zdokumentovat chování programu. Většinou pomocí disassembleru

Dynamická analýza: na běžícím kódu s cílem prostudovat a zdokumentovat chování programu. Většinou pomocí debuggeru

Aplikační binární rozhraní (ABI)

Též **volací konvence.**

Dokument popisující, jak by se měl binární kód na cílové platformě chovat, aby byl kompatibilní s binárním kódem jiných tvůrců a s operačním systémem.

Zahrnuje:

- způsob předávání parametrů funkcím
- zarovnání zásobníku
- použití registrů CPU (které lze změnit, které musí být zachovány)

- jak jsou měněna jména symbolů
- vytváření struktur/tříd v paměti
- volání virtuálních funkcí C++
- indentifikace typové informace za běhu (RTTI)
- zpracování výjimek

Zarovnání dat

Přístup k nezarovnaným datům pomalejší, na některých CPU nemožný

Různé kompilátory -- možné problémy se zarovnáním ⇒ řízení zaronání (`.align num_bytes` , `#pragma pack(num_bytes)`)

Každý datový typ jiné zarovnání

Zarovnání zásobníku:

- Historicky, 32b systémy zarovnávají na 4 B, 64b systémy na 16 B

Name Mangling

Přetěžování funkcí/operátorů/metod ⇒ **nejednoznačný název symbolu**

Přidání další informace do názvu symbolu = **name mangling** (dekorace jmen)

V C++ zahrnuje:

- jméno symbolu
- `const` , `volatile` vlastnosti
- `public` / `protected` / `private`
- pokud jde o jméno funkce, tak argumenty, někdy i návratový typ
- skalární/vektorový typ

Pro analytika obsahuje velké množství **užitečných informací**

Použití registrů

Některé registry:

- Lze měnit -- volající může jejich hodnotu zničit
- Slouží jako ukazatel na zásobník/rámec zásobníku
- Předávají parametry funkcím
- Nesou návratovou hodnotu funkce

Volací konvence

Určují:

- Jak jsou funkce volány
- Kdo a kam umisťuje parametry
- Jak se vrací výsledek
- Kdo odstraňuje použité parametry (volající/volaný)

Implicitní v 32bit C: `__cdecl` : Parametry na zásobníku, pořadí parametrů C, uklízí volající
64bit Linux: Parametry v `rdi, rsi, rdx, rcd, r8, r9` , pořadí C, uklízí volaný

Konvence **vždy součástí deklarace** funkce/metody:

```
bool WINAPI MessageBeep(uint uType);
```

je ve skutečnosti

```
bool __stdcall MessageBeep(uint uType);
```

Vynechání/změna způsobuje vážné chyby.

Třídy a struktury

`struct = class` , rozdíl jenom ve výchozí ochraně prvků

Všechna pole ve struktuře zarovnána podle pravidel ABI (použití vycpávacích bytů mezi nimi)

Určení velikosti struktury:

- **na zásobníku:**

    - Typicky alokována instrukcí `sub esp, __LOCAL_SIZE` , kde velikost je součástí výrazu `__LOCAL_SIZE`
- **na haldě:**
    - Alokace pomocí `new` / `malloc` /... -- velikost jako argument
- **odvození z instrukcí,** které s ní pracují
    - Iterace přes pole struktur

```
for(i=0; i<N; ++i)
    pStructArray[i].f_Field = 0;
```

V assembleru je proměnná `i` zvětšena o velikost jedné struktury.

    - Kopírování struktury -- velikost musí být známá

Dědičnost: instance rodičovské třídy uloženy v paměti těsně před aktuální třídou

Polymorfismus:

- Virtuální metody ⇒ **tabulka virtuálních metod (VMT)**
- **Vícenásobná dědičnost:** více VMT
- VMT v paměti jako **první položka třídy**, za ní data této třídy, za nimi druhá VMT atd.
- **Abstraktní třída** s nějakými definovanými metodami: taky má VMT, neimplementované virtuální metody jsou ve VMT nahrazeny adresou funkce `_purecall`
    - `_purecall` : volá handler purecall (pokud existuje), potom ukončí program
- **VMT v reverzním inženýrství:**
    - VMT globální pro všechny instance daného typu ⇒ ukazatel na VMT pomůže určit typ neznámého objektu
    - Určení vícenásobné dědičnosti: nalezení ukazatelů na VMT v paměti objektu
    - Zkoumání ukazatelů v každé VMT, idektifikace kódu patřícího danému objektu

Identifikace typové informace za běhu

Operátory `typeid` (určení typu polymorfního objektu), `dynamic_cast` (přetypování na podtřídu/nadtřídu) -- **využití RTTI (Run Time Type Identification)**

Třída `type_info` :

- Návratový typ `typeid`
- Obsahuje `char*` se jménem objektu

Operátor `dynamic_cast`:

- Typicky zpětné přetypování ze základní třídy směrem k odvozené
- **Přetypování na `void*`:** volání funkce `__RTTICastToVoid`
  - Funkce si vyzvedne ukazatel na `RTTICompleteObjectLocator` z pole těsně před VMT
    - Struktura `RTTICompleteObjectLocator`:
      - Obsahuje ukazatel na `TypeDescriptor`
    - Struktura `TypeDescriptor`:
      - pointer na VMT
      - manglované jméno typu
- **Přetypování na ne-void:**
  - Volání `RTDynamicCast`
  - Každá VMT[-1] obsahuje ukazatel na `RTTICompleteObjectLocator`, který obsahuje jméno typu

Odhalení celé hierarchie tříd:

- `RTTICompleteObjectLocator` obsahuje ukazatel na `RTTIClassHierarchyDescriptor`
- `Class Hierarchy Descriptor` obsahuje ukazatel na `RTTIBaseClassArray`
- `Base Class Array` obsahuje pole ukazatelů na `RTTIBaseClassDescriptor`
- `Base Class Descriptor` obsahuje `Type Descriptor` třídy (ten obsahuje její jméno)

Strukturovaná obsluha výjimek

- Speficická pro Windows
- Výjimka na systémové úrovni (dereference null pointeru) $\Rightarrow$ volán handler výjimky definovaný ve struktuře uložené na zásobníku
- Využití: útočník přepíše pole `handler` ve struktuře `EXCEPTION_REGISTRATION`
  - Typicky přepíše na instrukce `pop-pop-ret` (odstraní návratovou adresu a `ExceptionRecord*`), `ret` skočí na začátek `EXCEPTION_REGISTRATION`, kde je pole `prev`, které útočník naplnil prvními 4 byty kódu exploitu (vyžaduje spustitelný zásobník)