Zadanie 4 (Bezpečné programovanie)

October 28, 2024

Cieľom zadania je oboznámiť sa so štandardnými zraniteľnosťami v jazyku C. K dispozícii budete mať jednoduchú aplikáciu, napísanú v jazyku C, ktorá obsahuje množstvo rôznych zraniteľností. Ukážkovú aplikáciu (v súbore `main.c`) môžeme skompilovať napr. nasledujúcim príkazom:

```
gcc main.c -o main
```

Úlohy:

1. Identifikujte zraniteľnosti v ukážkovej aplikácii. Na identifikáciu využite dva hlavné prístupy. Statickú a dynamickú analýzu. Statická analýza skúma zdrojové súbory, bez samotného spustenia aplikácie (nástroje ako napr. `cppcheck` [3], `coverity` [2], atď.) Dynamická analýza naopak spúšťa aplikáciu a testuje rôzne vstupy (napr. `AFL` [1], `LibFuzzer` [4], atď.). Vyberte si dva nástroje (jeden z každej kategórie), popíšte akým spôsobom fungujú, ich inštaláciu a spúšťanie testov. Pomocou vami vybraných nástrojov nájdite zraniteľnosti a porovnajte výstupy. Poznámka: v tomto bode je cieľom nájsť čo najviac zraniteľností.

2. Identifikované zraniteľnosti v krátkosti popíšte. Uveďte miesto kde sa nachádzajú v zdrojovom kóde, popíšte samotný princíp zraniteľnosti a taktiež v stručnosti uveďte aké následky môže mať daná zraniteľnosť (t.j. čo dokáže potenciálny útočník spôsobiť). Taktiež uveďte, akým spôsobom je možné danú zraniteľnosť opraviť (či už samotnou úpravou v zdrojovom kóde alebo mitigáciami v operačnom systéme). Poznámka: uvedeným spôsobom popíšte aspoň **5** zraniteľností.

3. **BONUS - 3b**. Pokúste sa exploitovať ľubovoľnú z vami objavených zraniteľností. Cieľom je napísať plne funkčný exploit, ktorý dokáže spustiť vami vybraný kód. Je povolené skompilovať zdrojový kód bez ochrán, prípadne možeme predpokladať vypnuté ochrany na úrovni OS (ASLR, DEP). **Upozornenie: body budú započítané iba po prezentácii exploitu pred cvičiacim; dokumentáciu nie je nutné písať.**

Deadline zadania je **12.11.2024** o **13:37**. Zadanie sa odovzdáva do **akademického informačného systému** (AIS) do zvoleného miesta odovzdania. Odovzdáva sa dokumentácia vo formáte **PDF** (!!!). Odovzdáva **jeden** člen z tímu.

Literatúra

[1] Afl. `https://github.com/google/AFL`.

[2] Coverity scan - static analysis. `https://scan.coverity.com/`.

[3] Cppcheck - a tool for static c/c++ code analysis. `https://cppcheck.sourceforge.io/`.

[4] libfuzzer – a library for coverage-guided fuzz testing. `https://llvm.org/docs/LibFuzzer.html`.