Bezpečnost ICT 1 (BPC-IC1)

Semestrální projekt předmětu

Akademický rok	**2022/2023**

Číslo skupiny	**LL14**
Název projektu	**Kalkulačka**
Členové skupiny	**1) Beránek Karel** **2) Český Michal**

Popis aplikace/Cíle projektu (Abstrakt)

Aplikace je kalkulačka, která se dá využívat jak pro jednoduchém matematické operace, tak pro osvojení si základní exploataci aplikací. V aplikaci jsou naimplementovány celkem 4 různé typy zranitelností. Cílem exploatace je najít všechny tyto typy zranitelností. Pokud útočník danou chybu najde a správně exploitje tak se mu v terminálu zobrazí hlášení, které zobrazuje nalezenou chybu a úspěšnou exploitaci.

Implementované zranitelnosti

1) Intiger overflow, 2) Stack overflow, 3) Heap overflow, 4) String format

1) Potřebný HW a SW

Aplikace a samotný exploit je napsán v jazyce C#. Pro vytváření aplikace a exploitu bylo využíváno vývojového prostředí MS Visual Studio community edition. Nicméně pro úpravu samostatných kódů a následné kompilaci lze využívat další vývojových prostředí jako je třeba VS Code. Samotná aplikace a exploit lze spustit také pomocí souboru .exe který, je umístěn v /bin/Debug složce. Nicméně pokud se provádějí nějaké změny v exploitu, je nutné tento program opětovně zkompilovat.

2) Implementační postup

Aplikace je kalkulačka, která se dá využívat jak pro jednoduché matematické operace, tak pro osvojení si základní exploataci aplikací. Kalkulačka disponuje celkem 6 matematickými operacemi. Aplikace nejdříve uživateli zobrazí základní informace a návod, jak ji používat. Poté vyzve uživatele o zadání prvního čísla dále o zadání operátoru a následně pro druhé číslo. Podle toho, jaký byl zadán operátor tak se využije funkce pro výpočet výsledku. Spočítaný výsledek je následně uživateli zobrazen v terminálu.
Aplikace obsahuje celkem 4 bezpečnostní chyby. Jako první je chyba Intiger overflow, druhá chyba je Stack overflow, třetí chyba je Heap overflow a čtvrtá chyba je String format.

Intiger overflow je naimplementován na operaci sčítání, lépe řečeno na proměnné, která zobrazuje výsledek. Dvě čísla, které uživatel zadal a vložil jej pomocí exploitu do aplikace, jsou uloženy do proměnných typu string, které se následně přetypovávají na typ intiger. To znamená že pokud se do jedné z proměnných nebo celkový součet dvou čísel přesáhne hranici 2 147 483 647 nastane chyba Intiger overflow. Tomuto problému by se dalo snadno vyhrnou, a to tak že by proměnná do které se vkládá výsledek byla typu double. V normálním případě by tato chyba nezpůsobila pád aplikace ale špatný výsledek sčítání. V tomto projektu je tato chyba ošetřena takovým způsobem, že útočníka informuje o nalezení hledané chyby.
Chyba Stack overflow neboli chyba přetečení nastane, když program vytvoří moc vnořených úrovní, kterými je volána rekurzivní funkce, kde pak dochází k přetečení zásobníku. Tuto chybu obsahuje funkce pro výpočet faktoriálu.

```
if (a.Equals(0))
{
      return 1;
}
Else
{
      return a * factorialFunction(a - 1);
}
```

V táto funkci dochází k takzvané rekurzi, což je opětovné volání funkce. Při vkládání malých čísel aplikace bude pracovat správně. Pokud uživatel bude chtít vypočítat faktoriál většího čísla tak funkce bude volána vícekrát za sebou a bude možné že se zásobník naplní a přeteče. Ošetřit tuto chybu lze po přidání

podmínky pro ukončení rekurze. Lepší možností je se rekurzy úplně vyhnout a místo toho využívat třeba cyklus for. Stack overflow je velmi nežádoucí chyba, protože v běžných případech dochází k opětovnému pádu aplikacích.
Další zranitelnost, v podobě Heap overflow, obsahuje aplikace ve funkci pro výpočet sumy prvního vloženého čísla.

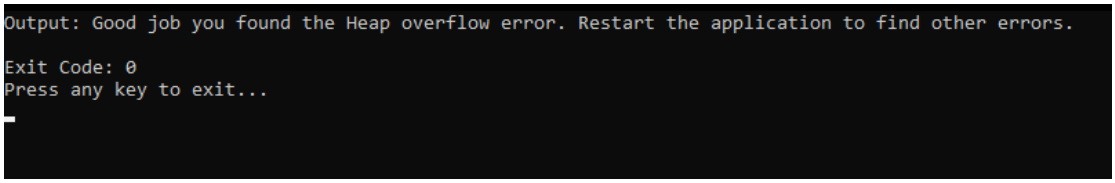
```
double[] arr = new double[(int)a + 1];

        for (double i = a; i >= 0; i--)
        {
        arr[(int)i] = i;
        sum += arr[(int)i];
        }

return sum;
```

Chyba Heap overflow je podobná chybě Stack overflow s tím rozdíle že se jedná o přetečení v oblasti heap. To znamená že program alokuje dynamicky paměť, ale není zde zajištěna detekce, zda má program dostatek místa pro nově alokovanou paměť. V tomto případě je využíváno pole array. Velikost pole záleží na prvním vloženém čísle. Pokud uživatel vloží číslo pět, tak program vytvoří pole o velikosti pět. To znamená že celková velikost pole se volí dynamicky podle požadavku uživatele. Pokud uživatel zvolí malé číslo neměl by nastat žádný problém. Nicméně pokud uživatel zvolí velké číslo, program alokuje dynamicky paměť, která bude moc malá pro námi požadovanou velikost pole. Tím pádem nastane přetečení v oblasti heap. Pro výpočet sumy se využívá cyklus for, který postupně sčítá hodnoty uložené v poli. Pokud bychom chtěli zabránit této chybě museli bychom přidat ošetření, které kontroluje velikost nově alokované paměti s maximálním možným místem v paměti. V tomto projektu při správném řešení exploatace této chyby, je vypsáno hlášení, které sděluje uživateli úspěšnou exploitaci. Pokud by toto řešení nebylo nijak ošetřeno výjimkou tak by docházelo k přepsání paměti, kterou by již využívala jiná část programu a došlo by k pádu aplikace.

Jako poslední naimplementovanou chybou je String format, která je v projektu naimplementována celkem na dvou místech. První případ ke je možné se s chybou setkat je při zadávání dvou čísel. Druhý případ se nachází u operace sčítání, který úzce souvisí s chybou Intiger overflow. Uživatel zadává čísla, která se ukládají do proměnných typu string a následně se jsou tyto proměnné převáděny na typ intiger nebo double. Pokud uživatel zadá místo čísla nějaký jiný znak, jako je třeba písmeno nebo tečka, dojde k vyvolání funkce tryParse, kde se testuje, jestli tuto proměnnou leze přetypovat na typ intiger nebo double. Pokud toto přetypování nebude možné tak je následně aplikace ukončena. Druhou možností, kde je možné se s chybou setkat je, když uživatel zadal jedno ze dvou čísel s desetinou čárkou a chce tyto dvě čísla sečíst. Při sčítání se pracuje s proměnnou typu intiger, který obsahuje pouze celá čísla, tudíž číslo s desetinou čárkou na něj není možné převést a dojde k vyvolání výjimky FormatException. První možnost lze ošetřit pouze pomocí vyvolání výjimky a vyzvání uživatele o

opětovné zadání čísla. U druhé možnosti to lze vyřešit změnou typu proměnných z intiger na double.

U exploatace bylo původně zamýšleno využití programovacího jazyku python s funkcemi subprocess, které měli být schopny otevřít aplikaci a vložit do ní hodnoty. Z důvodu že hodnoty, které program vkládal do kalkulačky nebylo možné přetypovat na požadovaný typ byl exploit napsán v jazyce C#. Exploit využívá funkce Process. Pro využívání této funkce je nutné poskytnout relativní cestu k aplikaci. To znamená k souboru .exe. Dále je nutné poskytnout první číslo, druhé číslo a požadovanou matematickou operaci které exploit bude vkládat do kalkulačky. Parametry pro výpočet jsou vloženy do aplikace pomocí pole typu string. Samotná aplikace rozdělí přijaté pole a jeho hodnoty uloží do svých proměnných. Exploit následně čeká na výstup kalkulačky která zobrazí výsledek do konzole. Toto exploit detekuje a výstup kalkulačky uloží do své proměnné output a zobrazí ji útočníkovi.
Výstup po úspěšné exploitaci může vypadat následovně.



3) Závěr

Snaha, v tomto projektu, byla vytvořit jednoduchou aplikaci (kalkulačku) která by sloužila pro osvojení si základů exploitace a hledání chyb v programech. V aplikaci byly naimplementovány celkem 4 zranitelnosti, a to Intiger overflow, Stack overflow, Heap overflow a String format. Tyto chyby můžou v aplikaci měnit výsledek nebo zapříčinit pád celé aplikaci. Z těchto důvodů byly chyby ošetřeny. Pokud útočník napíše správný exploit tak získá hlášení o úspěšné nalezení chyby v kódu. Dále byl vytvořen exploit, který právě těchto naimplementovaných chyb využívá. Aplikace a exploit byly napsány v jazyce C# z důvodu lepší komunikace mezi samostatnými programy.

4) Literatura

[1] CTFS & STUFF. In: *CTFS & STUFF* [online]. [cit. 2023-04-23]. Dostupné z: https://blog.lamarranet.com/

[2] IRONHACKERS. In: *IRONHACKERS* [online]. © 2023 IRONHACKERS [cit. 2023-04-23]. Dostupné z: https://ironhackers.es/en/tutoriales/introduccion-al-exploiting-parte-1-stack-0-2-protostar/