Bezpečnosť informačných a komunikačných systémov

Katedra elektroniky a multimediálnych telekomunikácií

Referát č.1
Podstata útoku s využitím pretečenia vyrovnávacej pamäte a možných proti opatreniach na úrovni GCC prekladača a tiež na úrovni operačného systému

Martin Chlebovec **29. Nov. 2020**

Vyrovnávacia pamäť (buffer)

Vyhradená časť pamäte určená [1] na dočasné uchovanie dát pred spracovaním. Do vyrovnávacej pamäte sa zhromažďujú dáta, ktoré sa kopírujú, alebo sú určené pre iné - výstupné zariadenia. Buffer tak dokáže efektívne vyrovnávať rozdiely v rýchlosti spracovania a prijímania dát.

Z hľadiska implementácie buffra ho môžeme rozdeliť na:

- Hardvérový
- Softvérový

Hardvérový buffer je najčastejšie tvorený samostatným čipom a môžeme na neho naraziť na počítačových zberniciach, vo vnútri výstupných a vstupno-výstupných zariadení, ale aj v mikrokontrolérovej technike.
Softvérový buffer je tvorený programovo a využíva pridelenie (alokovanie) istej časti operačnej pamäte RAM, ktorá je vyhradená práve pre tento účel. Tento typ vyrovnávacej pamäte sa používa aj v jednoduchých používateľských programoch, napríklad v jazyku C.

Buffer v jazyku C je tvorený [2]:

- **Textovým segmentom** - Obsahuje spustiteľný kód bez možnosti zmeny (len na čítanie)
- **Dátovým segmentom** - Ukladá statické/globálne premenné, ktoré inicializuje programátor
- **BSS segmentom** - Ukladá neinicializované statické/globálne premenné. Operačný systém BSS segment vypĺňa nulami, takže všetky neinicializované premenné sú inicializované nulami.
- **Heap segmentom** - Poskytuje dostupný priestor pre dynamické pridelenie pamäte
- **Zásobníkovým (stack) segmentom** - ukladanie lokálnych premenných definovaných vo vnútri funkcií, ich volania a ďalšie údaje, napr. argumenty, návratová adresa.

Pretečenie vyrovnávacej pamäte (buffra)

Pretečením pamäte sa rozumie stav [3], kedy program zapíše dáta mimo vyhradeného miesta, ktorým vyrovnávacia pamäť disponuje. Je to spôsobené nesprávnym návrhom programu. Pretečenie môže nastať veľkým počtom vstupných dát, ktoré pamäť objemovo prekročia (neošetrený vstup), prípadne pozmenia susedné dáta v štruktúre. Pretečenie môže byť ale spôsobené aj cieleným útokom, ktorý sa označuje metódou rozbíjania stacku.

Vo výsledku môže program pokračovať v behu s chybnými dátami, alebo môže dôjsť k pádu programu. Pád programu je najčastejšie spôsobený nekonečnou rekurziou, ktorá vyčerpá buffer. Z pohľadu segmentov buffra hrozí pretečenie v segmente Heap a Stacku.

Pretečenie Heap

Heap obsahuje dynamické programové dáta. Typ útoku na Heap spočíva v prepise vnútorných štruktúr, napríklad ukazovateľa spojkového zoznamu. To prepíše aj programové dynamické pridelenie pamäte (malloc, calloc). Tým sa umožní kopírovanie dát do programových dát z inej časti pamäte, susednej Heap štruktúry.

Pretečenie Stacku

Stack obsahuje lokálne premenné vo vnútri funkcií, ich volanie a ďalšie dôležité údaje. Princíp útoku na Stack spočíva vo vložení spustiteľného programu (shell-kódu) do poškodeného stacku a to zmenou návratovej adresy na pamäť mimo stacku. Tým môže útočník získať plnú kontrolu nad procesom a tak aj k počítaču, na ktorom daný proces beží. Existujú aj ďalšie spôsoby, napríklad prepísanie lokálnej premennej, čím je možné zmeniť jej veľkosť a zasiahne tak do susedných pamäťových blokov štruktúry. Typickým príkladom v jazyku C je použitie funkcie strcpy(), ktorá skopíruje istú veľkosť, avšak nekontroluje hranice stacku.

Ochrana pred pretečením vyrovnávacej pamäte na úrovni OS a programátora

Jednou z účinných metód ochrany pred pretečením vyrovnávacej pamäte je KTP - Kontrola toku programu, čo znamená monitoring a porovnanie behu programu voči referencii.

- WDP (Watchdog processor) [4]
    - monitoruje zbernice medzi CPU a pamäťou
    - overuje, či je spustený správny blok programu a ich poradie
    - musí disponovať referenčnými informáciami o programe
    - detekcia chýb:
        - chybná / nesprávna inštrukcia
        - zlý skok v programe

Účinne je možné ochrániť program pred pretečením buffra aj pri programovaní a fáze návrhu samotného programu. Kontrolou zdrojového kódu s efektívnym kódom využívajúcim triedy a funkcie pre bezpečnú prácu s pamäťou je možné eliminovať prepis susedných pamäťových blokov, ktoré by sa za normálnych okolností prepísali pri forme útoku.

Ekvivalenty pôvodných funkcií (obsahujú v názve n) pre C a C++ pre bezpečnú prácu s pamäťou [5]:

- strcpy → strncpy → strlcpy
- strlen → strnlen
- strcmp → strncmp
- strcat → strncat
- strdup → strndup
- sprintf → snprintf
- wcscpy → wcsncpy
- wcslen → wcsnlen

Ďalšou možnosťou je využitie automatizovaných nástrojov, napríklad Kanárik, ktorý sa vloží za definované premenné. V prípade pretečenia pamäte je prepísaný aj Kanárik, program sa ukončí. Kanárik môže byť statický, prípadne náhodný a XOR, ktoré výrazný zvyšujú zložitosť prípadného pokusu o útok, nakoľko útočník nevie vypočítať, kde sa prípadný Kanárik nachádza.

Ochrana pred pretečením vyrovnávacej pamäte na úrovni kompilátora GCC

Prekladač GCC [6] umožňuje prostredníctvom prepínača *-fno-stack-protector* pridať do skompilovaného programu časť kódu, ktorá umožňuje detegovať pretečenie buffra. Kontroluje predovšetkým funkcie pre alokáciu pamäte a funkcie s vyrovnávacou pamäťou viac ako 8 bajtov. Pri spustení funkcie sa program prepínača inicializuje a na konci funkcie sa vyhodnotí.

V prípade detekcie pretečenia sa program ukončí a vypíše sa chybová hláška. Prípadne prepínačom *-fstack-protector-all* je možné kontrolovať všetky funkcie v programe. Prekladač GCC obsahuje aj možnosť overenia pretečenia stacku prepínačom -fstack-check.

GCC obsahuje aj integráciu Debuggera GDB, ktorý je možné využiť aj pre Debugging časti kódu a odhaliť tak možné chyby, pretečenia Heap (neuvoľnená pamäť) pri využívaní c funkcií calloc(), malloc() a iné.

Zoznam použitých literárnych zdrojov.

[1]. Data buffer [online]. Wikipedia - The Free Encyclopedia [cit. 2020-08-06]. Dostupné z: https://en.wikipedia.org/wiki/Data_buffer

[2]. Memory Layout of C Programs [online]. Geeks for Geeks [cit. 2019-01-30]. Dostupné z: https://www.geeksforgeeks.org/memory-layout-of-c-program/

[3]. Heap overflow and Stack overflow in C [online]. Tutorials Point [cit. 2019-04-03]. Dostupné z: https://www.tutorialspoint.com/heap-overflow-and-stack-overflow-in-c

[4]. Spoľahlivosť a kontrola toku programu, Maroš Ďuríček, Vnorené systémy 2014, s. 16-18

[5]. The many ways to copy a string in C [online]. Martyn Afford [cit. 2019-09-30]. Dostupné z: https://www.mafford.com/text/the-many-ways-to-copy-a-string-in-c/

[6]. GCC online documentation [online]. GNU Project - Free Software Foundation (FSF) [cit. 2020-07-23]. Dostupné z: https://gcc.gnu.org/onlinedocs/