Techniky vyhýbania sa zneužitiu zraniteľností v MS Windows

Marián Novotný

ESET, spol. s r.o.

O čom bude prezentácia

- Niečo o mne
- Softvérové zraniteľnosti
- Príklady zraniteľností v jazyku C
- Motivácia vyhýbania sa zneužitiu zraniteľností
- Taktiky vyhýbania sa zneužitiu zraniteľností v MS WINDOWS
- Záverečné zhrnutie

Niečo o mne

- Momentálne
  - Dizajnér bezpečnostných sieťových algoritmov
    - Analýza, návrh a implementácia nových funkcionalít pre personálny firewall a systém detekcie útokov
    - Práca na produkte ESS pre Windows OS
  - Čiastočná spolupráca s FI MUNI
    - LABAK, záverečné práce
- V minulosti
  - Bezpečnostný konzultant
  - Výskum v oblasti bezpečnostných protokolov
    - PhD práca "Návrh a analýza bezpečnostných protokolov"

Softvérové zraniteľnosti

- Zapríčinené ľudskou chybou – programátor, designer, správca
- Rozlišujeme
    - Implementačné
    - návrhové
    - operačné
- Klasifikácia podľa Seven Pernicious Kingdoms
    - Validácia a reprezentácia užívateľského vstupu
    - Zneužitie API
    - Bezpečnostné vlastnosti
    - Stav aplikácie v určitom čase
    - Ošetrenie chybových stavov
    - Kvalita kódu
    - Zapúzdrenie
    - Prostredie
- Registrované v CVE – MITRE orgnizácia

Životný cyklus softvérovej zraniteľnosti

- Objavenie zraniteľnosti bezpečnostnými výskumníkmi
- Publikovanie zraniteľnosti - POC
- Vydanie záplaty
  - MS Patch Tuesday – druhý utorok v mesiaci
- Inštalovanie záplaty
- Mnoho červov z minulosti využívali POC kód výskumníkov
  - ILOVEYOU, Code Red, Code Red II, Nimda, Sadmind, Slammer, Blaster, Sobig.F, Agobot, Bagle, Nachi
- Súčasnosť
  - Výskumníci koordinujú publikovanie zraniteľnosti so sw spoločnosťami

Výpočet skóre zraniteľnosti

- Common Vulnerability Scoring System (CVSS)
  - Otvorený štandard
  - Výpočet na základe metrík ohodnotených expertom
  - Skóre 0-10, critical, major, minor
- V súčasnosti verzia 2, na verzii 3 sa pracuje
- Metriky
  - Základné metriky
    - Acces vector
    - Acces complexity
    - autentizácia
  - Časové metriky
    - Existuje funkčný POC?
    - Existuje záplata od vendora?

Implementačné zraniteľnosti

- Korupcia pamäte
  - pretečenie buffra – buffer overflow
  - Použitie pamäte po uvoľnení – use after free
- Formátovanie stringov

```
char* s;
//načítaj string s zo vstupu
printf(s);
```

  - %d %u %x %s %n
  - %n je vypnutý defaultne od Visual studio 2010

Príklady chýb v jazyku C

- Celočíselné pretečenie, podtečenie

```
Unsigned int a, b;
a = 0; a --; b=0xFFFFFFFF; b++;
```

- Celočíselné pretečenie a korupcia pamäte v OPENSSH 3.1

```
u_int nresp = packet_get_int();
if (nresp > 0) {
        response = xmalloc(nresp * sizeof(char*));
        for (size_t i = 0; i < nresp; i++){
        response[i] = packet_get_string(NULL);
        }
packet_check_eom();
```

Príklady chýb v jazyku C (2)

- Znamienkové celé čísla

```
int length = network_get_int(sockfd);
if(length < 0 || length + 1 >= MAXCHARS){
//bad length
}
```

  - length = 0x7FFFFFFF ?

```
struct dummyStruct {
   int firstField : 1;
   ...
};
```

- Používanie nebezpečných funkcii

```
char buffer[25];
printf("Enter Text : ");
gets(buffer);
```

Príklady chýb v jazyku C - Konverzie typov

- Explicitná konverzia
  - Problem pri rozširovaní, zúžovaní
- Priradenie
  - Pravý operand priradenia je skonvertovaný na l'avý typ operandu
- Volanie funkcie
  - Premenné zadané na vstup funkcie sú skonvertované podl'a prototypu
- Návratová hodnota funkcie
  - Premenná, ktorú vracia funkcia je skonvertovaná podl'a prototypu funkcie
- Vyhodnocovanie vyrazov
  - Komplikované pravidlá pri vyhodnocovaní výrazov rôznych typov

Zneužitie zraniteľnosti - exploitovanie

- Útočník má pod kontrolou vstupné dáta zraniteľného programu
- Možnosti
    - Spustenie ľubovolného kódu
    - Prepísanie dôležitých časti pamäte
    - Zhodenie programu, služby
- Ciele
    - šírenie malware
    - Kompromitácia počítača
    - Získanie privilégií
    - DOS útok

Pamäť procesu x86– zjednodušený pohľad

- Program má virtuálnu pamäť
  - Každý byte má adresu
  - Kernel, user – normálne (0x00000000-0x7FFFFFFF) boot.ini zmeniť
  - Skladá sa z oblastí volných, rezervovaných a komitnutých
  - Mapovanie virtuálnej na fyzickú MMU
- Oblasti virtuálnej pamäte
  - Image – dll, PE súbory
  - PEB, TEB
  - HEAP
    - Dynamicky alokovaná pamäť (malloc)
    - Nízke adresy -> vysoké adresy
  - STACK
    - Pamäť počas vykonávania funkcií
    - Vysoké adresy -> nízke adresy
    - Registre EBP, ESP
- Registre
  - ESP, EBP, EIP
  - EAX, EBX, ECX, EDX, ESI, EDI

Zaujimavé oblasti pamäta

- Stratégie alokácie pamäti v programe
  - Heap – malloc, free, new, delete
  - Stack
- Stack
  - Pri volaní funkcie sa nakopírujú argumenty funkcie
  - Uloží sa IP, SP
  - Nasledujú lokálne premenné
- HEAP
  - Dynamicky alokované dáta uložené v datových štruktúrach ako spájaný zoznam

Buffer overflow – zjednodušený príklad

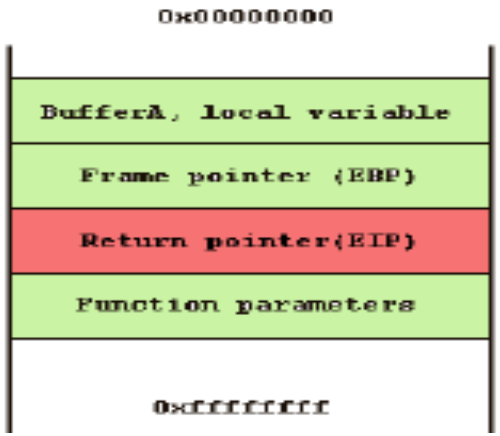
```
#include <stdio.h>

int stacktest(char *buf)
{
    unsigned char bufferA[40];

    strcpy(bufferA, buf); // Stack Overflow happens here
    return 0;
}

int main(int argc, char *argv[])
{
    if(argc != 2)
    {
     printf("\nStack-based buffer overflow test");
     printf("\nUsage: stackbo.exe string\n\n");
     exit(1);
    }

    stacktest(argv[1]);
    return 0;
}
```



Motivácia pre vývoj techník na vyhýbanie sa exploitom

- Zraniteľnosť neznamená, že sa vytvorí funkčný exploit
- Náklady na vývoj sú determinované ziskom útočníka
- Zraniteľnosti MS Windows priniesli obrovske škody MS, organizáciám,
    - IIS Code Red (MS01-033)
    - SQL Slammer (MS02-039)
    - Blaster (MS03-026)
    - Sasser (MS04-011)
- Zreniteľnosti sú objavované aj automatizovane
    - fuzzing
- Techniky vyhýbania sa exploitom na úrovni OS sa snažia
    - Zvýšiť náklady na vývoj funkčného exploitu
    - Znížiť nasaditeľnosť exploitu
    - Zabrániť použitiu známych techník exploitovania
    - Ochrana pre aplikácie spúštané v MS WINDOWS nielen pre samotný OS

Data Execution Prevention (DEP)

- Zabraňuje spúšťaniu kódu z určitých oblastí pamäte
    - Stack, heap,…
- Procesor musí podporovať NX bit (No eXecute AMD) resp. Execute Disabled (XD Intel) + Physical Address Extension (PAE) mód
  - Linkovať pomoocou /NXCOMPAT vo visual studiu
  - Za runtime SetProcessDEPPolicy
  - Od windows xp sp 2
  - Techniky na obchádzanie DEP
    - ROP – return to libc - MS08-067 Conficker v xpsp2

Address Space Layout Randomization (ASLR)

- Exploity predpokladali presné adresy importovaných funkcií, knižníc
- ALSR randomizuje adresy regiónov vo virtuálnej pamäti (32 bit)
    - Stack 14 bitov
    - HEAP 5 bitov
    - Images 8 bitov
    - PEB, TEB štruktúry 4 bity
    - Win 8 randomizuje ďalšie funkcie a pridáva entropiu
- Je nastavený bit 0x40 v PE poli DLLCHARACTERISTICS
- Linkovať pomocou /DYNAMICBASE
- Vista +
- Podpora DEP na ochranu pred ROP
    - MS08-067 Conficker vs VISTA
    - neučinné, že existuje dll, ktorá nie je kompilovaná cez /DYNAMICBASE

GS

- Prvá verzia od MS visual studio C++ 2002
  - /GS prepínač
- Za dôležité lokálne premenné funkcie pridá prolog funkcie náhodné číslo
- Epilóg skontroluje pri návrate, či sa to číslo neprepísalo
- MS visual studio C++ 2005
  - Preusporiadanie lokálnych premenných a parametrov funkcie na stacku

Záverečné zhrnutie

- Ochrana na úrovni OS sťažuje vývoj a nasadzovanie exploitov
- Mení sa charakter a spôsoby realizácie útokov
- Množstvo features v OS, ktoré sme nespomenuli
    - SAFESEH/SEHOP
    - Heap ochrany
    - Sandbox – app container vo windows 8
- Zaujimavá oblasť výskumu, kde dobré nápady sú cenené
    - Microsoft BlueHat Prize
        - 200 000$ pre víťaza
        - Vasilis Pappas – kBouncer – anti ROP

Q&A

Ďakujem za pozornosť