

Sigurnost računalnih sustava

Sigurnost programske podrške – ranjivosti i napadi

doc. dr. sc. Ante Đerek

doc. dr. sc. Stjepan Groš

izv. prof. dr. sc. Miljenko Mikuc

izv. prof. dr. sc. Marin Vuković

Obavezna literatura

- Computer Security and the Internet: Tools and Jewels from Malware to Bitcoin, Second Edition by Paul C. van Oorschot. Springer, 2021.
 - Chapter 6: Software Security - Exploits and Privilege Escalation
 - Usvojiti cijelo poglavlje!

Motivacija: prvi program koji ste vidjeli na FER-u

```
#include <stdio.h>

int main(void) {
    int n, rez;
    scanf("%d", &n);

    // izracunaj apsolutnu vrijednost
    if (n < 0) {
        rez = -1 * n;
    } else {
        rez = n;
    }

    printf("Ulaz: %d Rezultat: %d", n, rez);
    return 0;
}
```

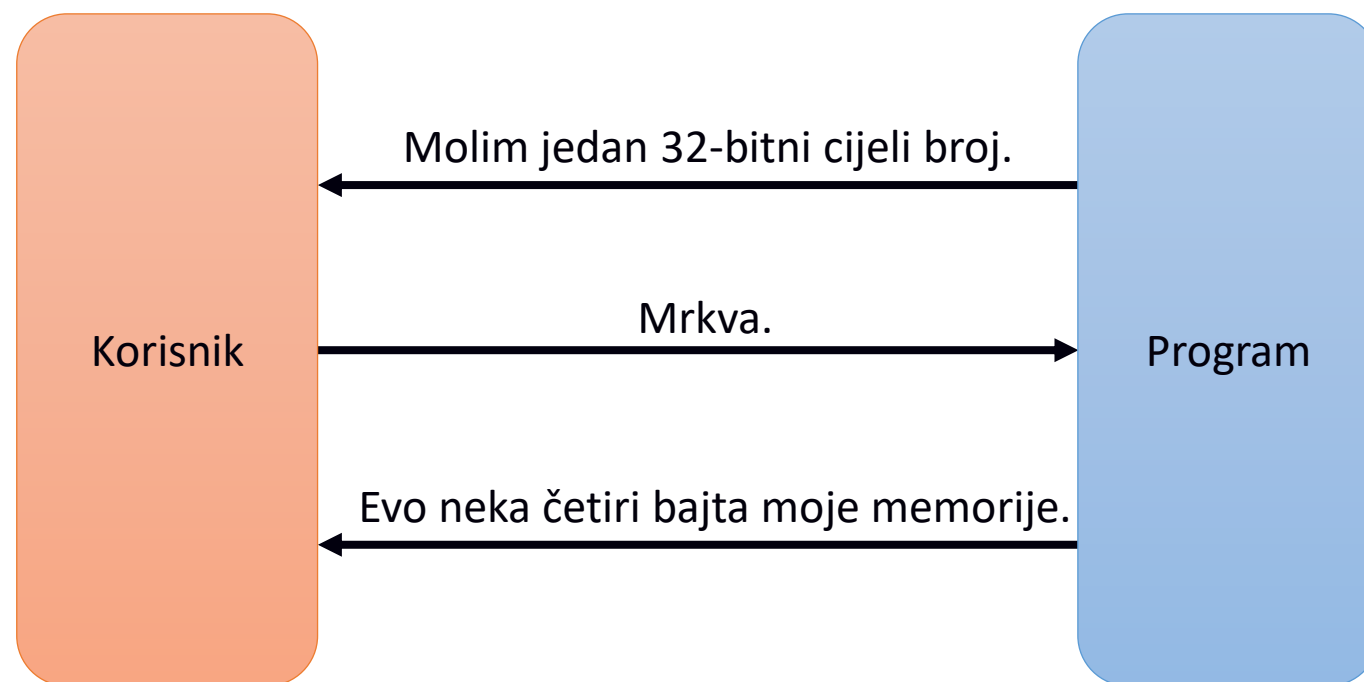
datoteka prog1.c

Razuman ulaz

```
$ ./a.out
-12
Ulaz: -12 Rezultat: 12
```

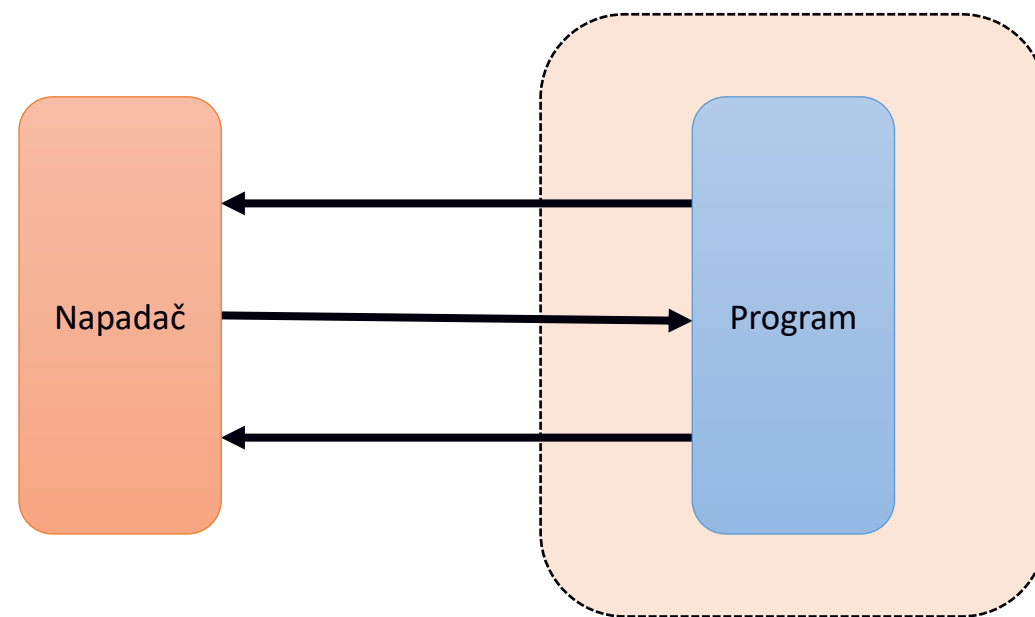
Nerazuman ulaz

```
$ ./a.out
Mrkva
Ulaz: -34521342 Rezultat: 34521342
```



Pretpostavke

- Program se izvršava u privilegiranom okružju u odnosu na napadača.
 - Program je na jednom računalu, napadač na drugom.
 - Program ima administratorske ovlasti, napadač je obični korisnik.
 - Program je hipervizor, napadač ima kontrolu nad virtualnom mašinom.
 - ...
- Ciljevi napadača mogu biti:
 - Dohvaćanje povjerljivih informacija.
 - Preuzimanje kontrole odnosno *eskalacija privilegija*.
 - Prekid rada sustava (*Denial of service*).
 - ...



Vrijednost neinicijaliziranih lokalnih varijabli?

```
int check_password(char *s) {  
    char pwd[21] = "super_tajna_lozinka!";  
    return strcmp(pwd, s);  
}  
  
int my_main(void) {  
    int n, rez;  
    scanf("%d", &n);  
    // izracunaj apsolutnu vrijednost  
    if (n < 0) {  
        rez = -1 * n;  
    } else {  
        rez = n;  
    }  
    printf("Ulaz: %d Rezultat: %d", n, rez);  
    return 0;  
}  
  
int main(int argc, char *argv[]) {  
    check_password(argv[1]);  
    my_main();  
}
```

```
$ ./a.out blabla
```

```
Mrkva
```

```
Ulaz: 560032622 Rezultat: 560032622
```

```
$ python -c "print hex(560032622)[2:].decode('hex')"
```

```
!akn
```

Defenzivno programiranje

- „Obično” programiranje:
 - *Program u razumnim okolnostima, za razumne ulaze treba davati razumne izlaze.*
- Defenzivno programiranje:
 - *Čak i u potpuno nerazumnim okolnostima i za potpuno nerazumne ulaze, program se ne smije ponašati nerazumno.*
- Potreban je konzervativan (paranoičan) pristup prilikom:
 - ... interakcije s korisnikom i svim dijelovima sustava izvan direktne kontrole programera.
 - ... formiranja pretpostavki o okolnostima u kojima će se programski kod koristiti.
- Nije nužan:
 - ... elegantan oporavak od svih mogućih pogrešaka.

Naši ciljevi u ovom predavanju

- Potrebno je razumjeti napade kako bi shvatili rizike i oblikovali obrane.
- Primjeri čestih i opasnih ranjivosti programske podrške!
 - Provjera ispravnosti ulaza (*input validation*)
 - Napadi prelijevanjem međuspremnik (buffer overflow attacks)
 - Problem uzrokovani istovremenim izvršavanjem (*race conditions*)

Rank	ID	Name	Score	KEV Count (CVEs)	Rank Change vs. 2021
1	CWE-787	Out-of-bounds Write	64.20	62	0
2	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	45.97	2	0
3	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	22.11	7	+3 ▲
4	CWE-20	Improper Input Validation	20.63	20	0
5	CWE-125	Out-of-bounds Read	17.67	1	-2 ▼
6	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	17.53	32	-1 ▼
7	CWE-416	Use After Free	15.50	28	0
8	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	14.08	19	0
9	CWE-352	Cross-Site Request Forgery (CSRF)	11.53	1	0
10	CWE-434	Unrestricted Upload of File with Dangerous Type	9.56	6	0
11	CWE-476	NULL Pointer Dereference	7.15	0	+4 ▲
12	CWE-502	Deserialization of Untrusted Data	6.68	7	+1 ▲
13	CWE-190	Integer Overflow or Wraparound	6.53	2	-1 ▼
14	CWE-287	Improper Authentication	6.35	4	0
15	CWE-798	Use of Hard-coded Credentials	5.66	0	+1 ▲
16	CWE-862	Missing Authorization	5.53	1	+2 ▲
17	CWE-77	Improper Neutralization of Special Elements used in a Command ('Command Injection')	5.42	5	+8 ▲
18	CWE-306	Missing Authentication for Critical Function	5.15	6	-7 ▼
19	CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer	4.85	6	-2 ▼
20	CWE-276	Incorrect Default Permissions	4.84	0	-1 ▼
21	CWE-918	Server-Side Request Forgery (SSRF)	4.27	8	+3 ▲
22	CWE-362	Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	3.57	6	+11 ▲
23	CWE-400	Uncontrolled Resource Consumption	3.56	2	+4 ▲
24	CWE-611	Improper Restriction of XML External Entity Reference	3.38	0	-1 ▼
25	CWE-94	Improper Control of Generation of Code ('Code Injection')	3.32	4	+3 ▲

Izvor: 2022 CWE Top 25 Most Dangerous Software Weaknesses

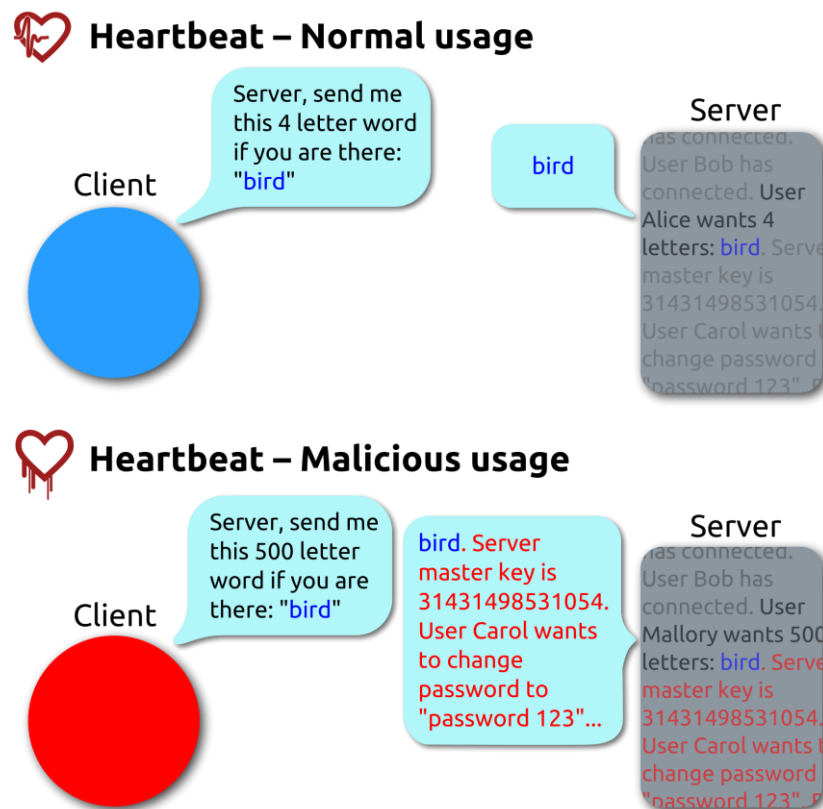
Sigurnost programske podrške – ranjivosti i napadi

Provjera ispravnosti ulaza

Provjera ispravnosti ulaza

- Provjera sintakse ulaza
 - Je li ulaz ispravne veličine?
 - Primjer ranjivosti: Prelijevanje međuspremnika
 - Je li ulaz ispravno formatiran?
 - Primjer ranjivosti: Razni *injection* napadi
- Provjera semantike (značenja) ulaza
 - Ima li ulaz smisla u kontekstu programa?
 - Primjer ranjivosti: *Heartbleed*

Primjer: *Heartbleed* ranjivost (CVE-2014-0160)



- Ranjivost u OpenSSL kriptografskoj biblioteci.
- Moguće doći do sadržaja memorije web poslužitelja koji koristi TLS protokol.
 - Zaporke, kriptografski ključevi, kolačići, ...
- Jedna od posljedica: <https://www.coreinfrastructure.org/>

Primjer: *Heartbleed* ranjivost – izvorni kod

```
26 ssl/d1_both.c
@@ -1459,26 +1459,36 @@ dtls1_process_heartbeat(SSL *s)
1459     unsigned int payload;
1460     unsigned int padding = 16; /* Use minimum padding */
1461
1462 -     /* Read type and payload length first */
1463 -     hbtype = *p++;
1464 -     n2s(p, payload);
1465 -     pl = p;
1466 -
1467     if (s->msg_callback)
1468         s->msg_callback(0, s->version, TLS1_RT_HEARTBEAT,
1469             &s->s3->rrec.data[0], s->s3->rrec.length,
1470             s, s->msg_callback_arg);
1471
1472 +     /* Read type and payload length first */
1473 +     if (1 + 2 + 16 > s->s3->rrec.length)
1474 +         return 0; /* silently discard */
1475 +     hbtype = *p++;
1476 +     n2s(p, payload);
1477 +     if (1 + 2 + payload + 16 > s->s3->rrec.length)
```

Izvor: github.com/openssl/openssl/commit/96db9023b881d7cd9f379b0c154650d6c108e9a3

Primjer: *Command injection*

```
$userName = $_POST["user"];  
$command = 'ls -l /home/' . $userName;  
system($command);
```

Izvor: <https://cwe.mitre.org/data/definitions/78.html>

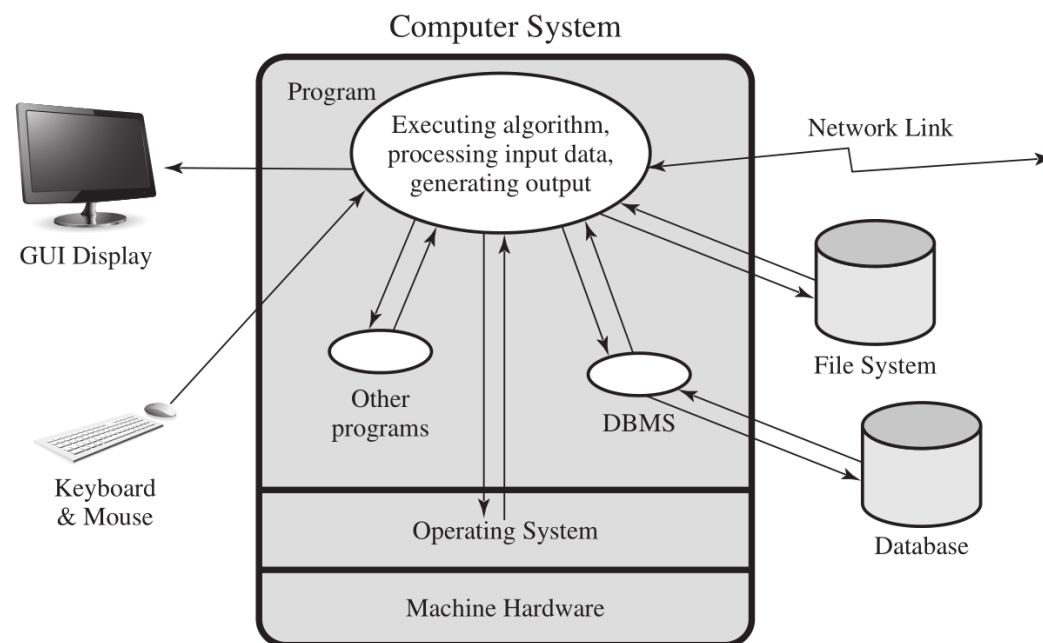
- Napadač pošalje `";rm -rf /"` kao parameter `user`.
- php skripta izvrši `"ls -l /home/;rm -rf /"`

Izazov: parsiranje je teško 😊

- Primjer: Gradimo web servis za prevođenje, kako zabraniti *localhost* URI?
 - `http://localhost`
 - `http://0.0.0.0`
 - `http://127.0.0.1`
 - `http://[::]`
 - `http://[0000::1`
 - `http://[0:0:0:0:0:ffff:127.0.0.1]`
 - `http://2130706433`
 - `http://017700000001`
 - `http://0x7f000001/`
 - `http://www.google.com@127.0.0.1`
- Sintakse “jednostavnih” objekata (ime datoteke, putanja, email adresa, IP adresa, URI, ...) su dosta komplicirane.
- Treba voditi računa o različitim jezicima, kodiranjima znakova.
- *Blacklist vs whitelist* pristup?
- Regularni izrazi?
- Preporuka: sve pretvarati u kanonski oblik pa onda provjeravati ispravnost?

Izazov: što je točno ulaz u program?

- Potrebno je odlučiti što su točno ulazi u vaš program i kojim komponentama (ne)možete vjerovati.
- Primjer: Ako pišete Python program, koji koristi Python biblioteku, koja koristi prevedenu C biblioteku, koja čita /etc/localtime prilikom inicijalizacije, je li ta datoteka ulaz u vaš program i može li joj se uvijek vjerovati?



Izvor: Stallings, Brown,
Computer security:
principles and practice

Sigurnost programske podrške – ranjivosti i napadi

Neispravno konkurentno korišćenje zajedničkih resursa (*race condition*)

Konkurentno korištenje zajedničkih resursa

- U okolini u kojoj se istovremeno izvršava više komada koda (dretve, procesi, servisi, itd.), točan redoslijed operacija je nepredvidiv, a može imati posljedice na ukupne rezultate izvršavanja.
- Neka poželjna svojstva:
 - Isključivost (mutual exclusion): U svakom trenutku samo jedan komad koda pristupa kritičnom resursu.
 - Atomičnost (atomicity): Kritični dio koda se ne može izvršiti djelomično, ili se izvrši sve ili ništa.

Race conditions – jednostavan primjer

Što se događa ako se u ovom trenutku paralelno započne i završi druga transakcija?

```
$transfer_amount = GetTransferAmount();
$balance = GetBalanceFromDatabase();

if ($transfer_amount < 0) {
    FatalError("Bad Transfer Amount");
}
$newbalance = $balance - $transfer_amount;
if (($balance - $transfer_amount) < 0) {
    FatalError("Insufficient Funds");
}
SendNewBalanceToDatabase($newbalance);
NotifyUser("Transfer of $transfer_amount succeeded.");
NotifyUser("New balance: $newbalance");
```

Izvor: <https://cwe.mitre.org/data/definitions/362.html>

- Rješenje ovog problema: koristiti transakcije na bazi koje osiguravaju atomarnost.

Race conditions – naša definicija

- Neočekivano ponašanje programa uzrokovano konkurentnim izvršavanjem.
- Teška detekcija i popravljjanje:
 - Konurentnost je programerima neuintitivna
 - Bugovi se rijetko pojavljuju i teško ih je reproducirati
- Čest izvor sigurnosnih problema
 - Napadač također možda ima pristup dijeljenom resursu!

Race conditions – TOCTOU

- Moguće je da se okolnosti promjene između sigurnosne provjere određenog resursa i korištenja tog resursa (*Time-of-check to time-of-use*)!

Napadač u ovom trenutku pobriše datoteku i zamijeni je simboličkom vezom na datoteku kojoj ne bi trebao imati pristup.

```
function readFile($filename){
    $user = getCurrentUser();

    //resolve file if its a symbolic link
    if(is_link($filename)){
        $filename = readlink($filename);
    }

    if(fileowner($filename) == $user){
        echo file_get_contents($realFile);
        return;
    }
    else {
        echo 'Access denied';
        return false;
    }
}
```

- Rješenje ovog problema: raditi sa opisnicima (*file descriptors*) umjesto imenama datoteka.

Sigurnost programske podrške – ranjivosti i napadi

Preljev međuspremnik (*buffer overflow*)

Buffer overflow napadi – crtice iz povijesti

- Morris crv (1988)
 - *buffer overflow* u *fingerd* servisu je jedan od načina širenja
- Smashing the Stack for Fun and Profit (1996)
 - <http://phrack.org/issues/49/14.html>
- Code Red crv (2001)
 - *buffer overflow* u Microsoft IIS 5.0, oko 359000 zaraženih računala
- The Slammer crv (2003)
 - *buffer overflow* u Microsoft SQL Server 2000, deseci tisuća zaraženih računala u prvih 10 minuta
- Iphone ranjivosti (2010-te)
 - Steaks4uce, SHatter (*heap overflow, use-after-free*)

- Konverzijska specifikacija %s omogućuje čitanje niza znakova do pojave prve praznine, oznake novog retka ili tabulatora

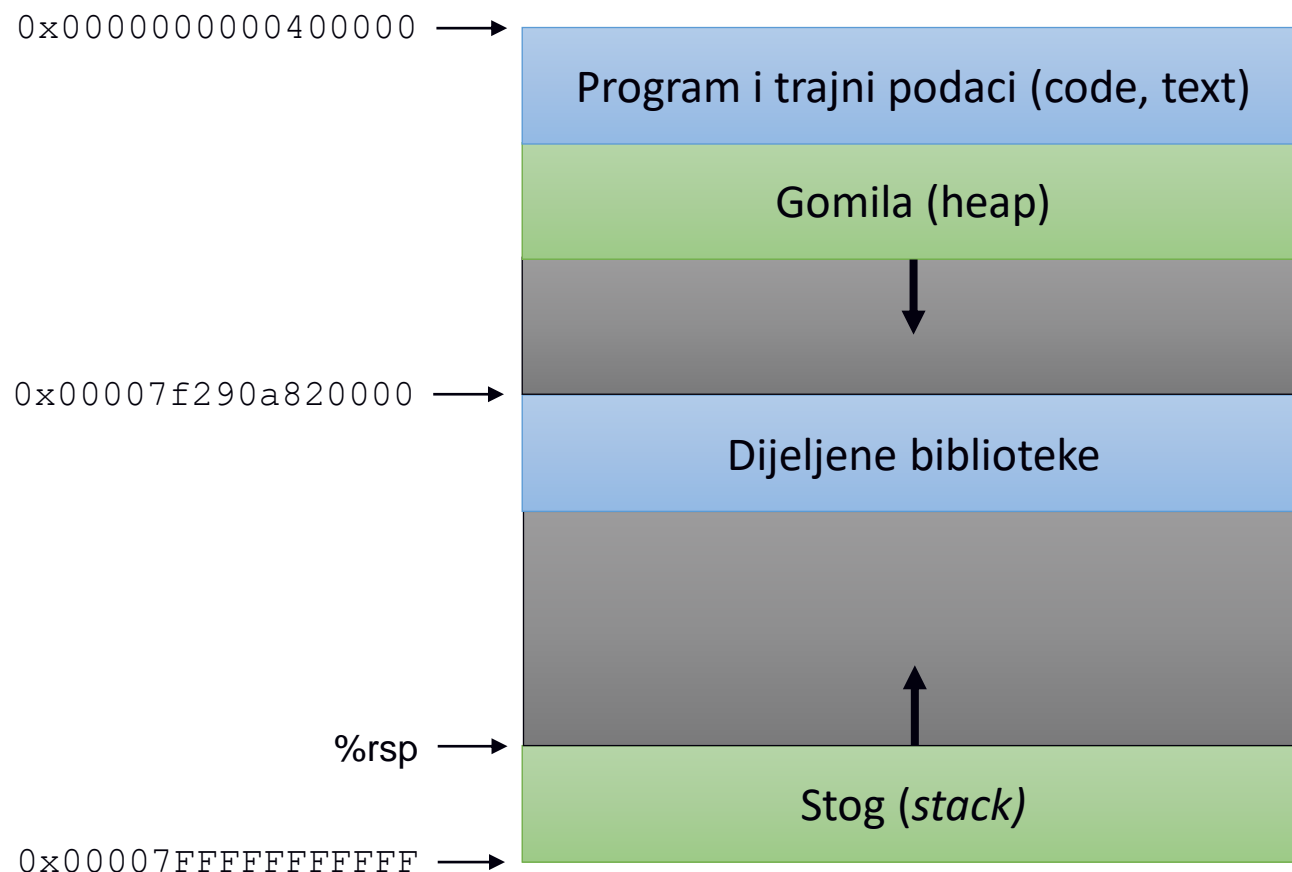
```
char ime[10 + 1], prezime[10 + 1];  
scanf("%s %s", ime, prezime);  
printf("%s, %s", prezime, ime);
```

```
Anamarija Horvat Novak↵  
Horvat, Anamarija
```

- Čitanje konverzijskom specifikacijom %s na kraj niza ugrađuje '\0'
- Čitanje niza znakova pomoću funkcije scanf i konverzijske specifikacije %s je potencijalno opasno
 - što će se dogoditi ako korisnik upiše predugo ime ili prezime (u odnosu na duljinu niza znakova navedenu u definiciji)

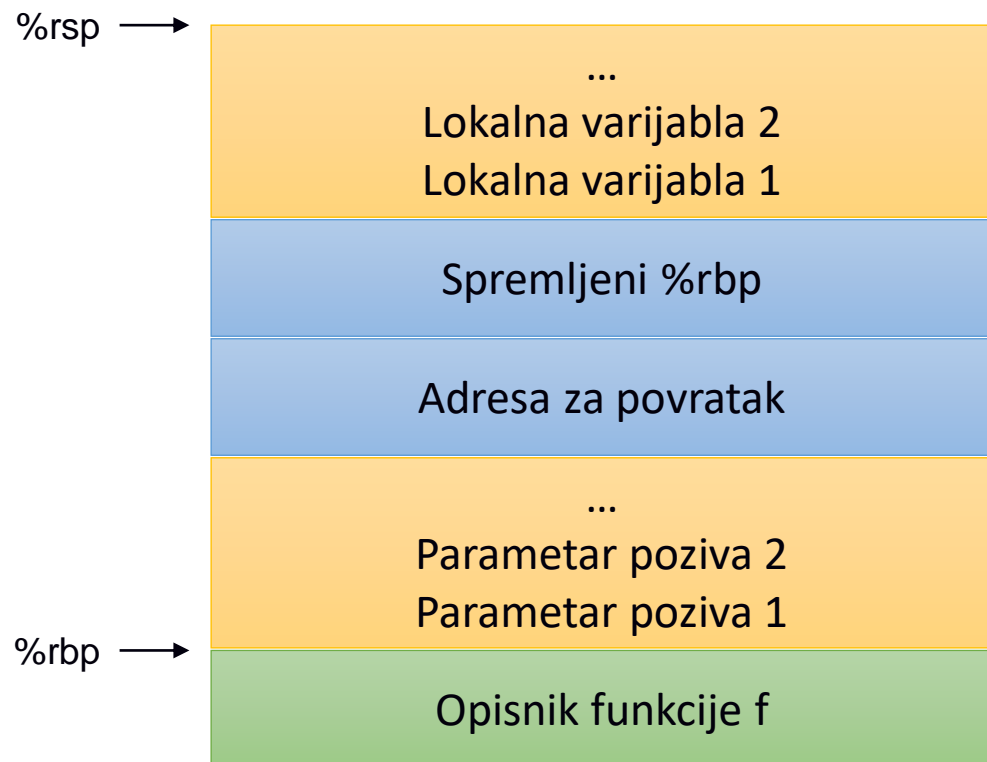
```
Anamarija Horvat-Novak↵
```


Memorijski prostor tipičnog procesa (Linux x86-64, pojednostavljeno)



- Na stog se spremaju opisnici funkcija (*call frame*) prilikom poziva
- Stog raste prema gore (prema manjim adresama)
- Za dinamički alociranu memoriju se koristi gomila (mislimo i na područja alocirana kroz `mmap`)

Stog i pozivanje funkcija (cdecl, pojednostavljeno)



Funkcija f zove funkciju g

1. f stavi vrijednosti parametara na stog
2. instrukcija `call` spremi adresu trenutne instrukcije (`%rip`) na stog da bi znali gdje nastaviti s izvođenjem kad g završi
3. g spremi bazu starog opisnika (`%rbp`) na stog i napravi mjesto na stogu za novi opisnik (smanjivanjem `%rsp-a`)
4. g se izvršava i koristi opisnik za lokalne varijable
5. g „izbriše” svoj opisnik (prilagođavanjem `%rsp-a` i čitanjem `%rbp-a`)
6. instrukcija `ret` pročita adresu sa vrha stoga i nastavi izvođenje od te adrese.

Stog i pozivanje funkcija (x86-64)

Argument Register Overview

Argument Type	Registers
Integer/Pointer Arguments 1-6	RDI, RSI, RDX, RCX, R8, R9
Floating Point Arguments 1-8	XMM0 - XMM7
Excess Arguments	Stack
Static chain pointer	R10

Izvor: System V AMD64 ABI, Wikipedia

Parameter type	fifth and higher	fourth	third	second	leftmost
floating-point	stack	XMM3	XMM2	XMM1	XMM0
integer	stack	R9	R8	RDX	RCX
Aggregates (8, 16, 32, or 64 bits) and <code>__m64</code>	stack	R9	R8	RDX	RCX
Other aggregates, as pointers	stack	R9	R8	RDX	RCX
<code>__m128</code> , as a pointer	stack	R9	R8	RDX	RCX

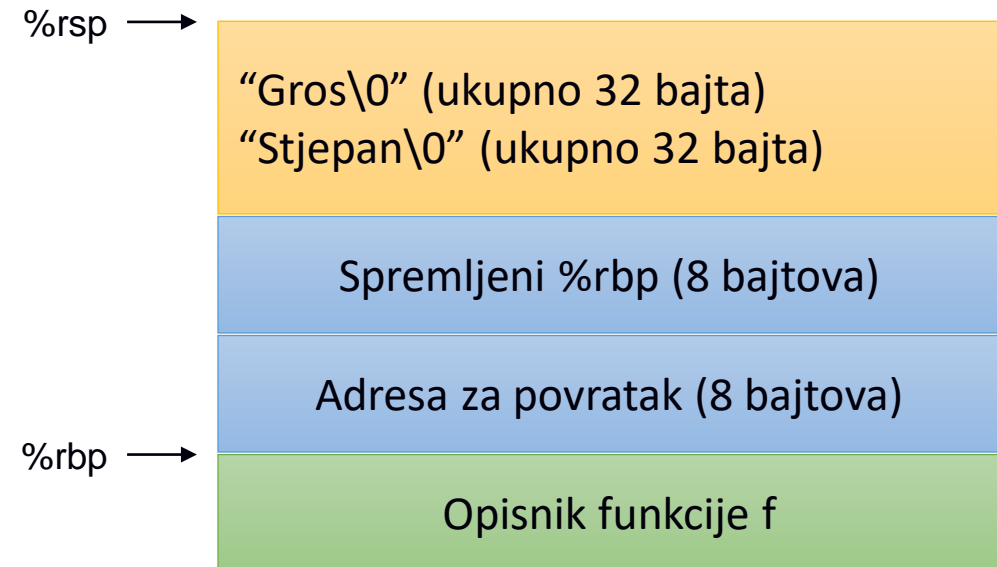
Izvor: x64 calling convention, Microsoft

Normalno izvršavanje

```
#include <stdio.h>

int main() {
    char ime[31 + 1], prezime[31 + 1];
    scanf("%s %s", ime, prezime);
    printf("%s, %s", prezime, ime);
    return 0;
}
```

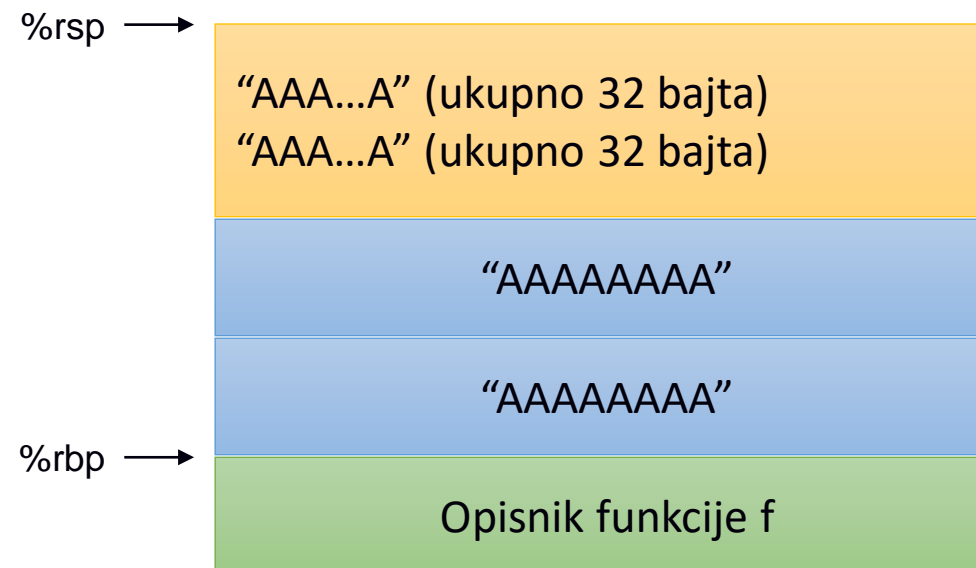
```
$ echo "Stjepan Gros" | ./a.out
Stjepan Gros
```



Preljevanje međuspremnika

```
$ echo "Stjepan A<80 puta>" | ./a.out  
Segmentation fault (core dumped)
```

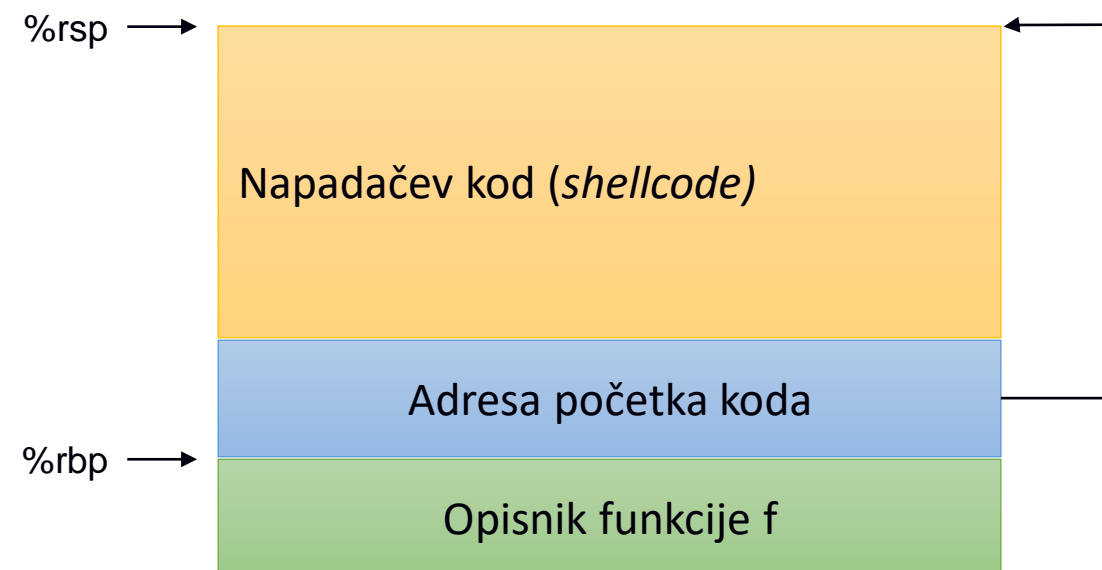
- Prepisali smo adresu za povratak na stogu!
- Nakon instrukcije `ret` program je pokušao nastaviti izvršavanje na adresi koja odgovara nizu znakova „AAAAAAAA” (0x4141414141414141)!



Preljevanje međuspremnika – iskorištavanje ranjivosti

```
$ echo "Stjepan <kod>..<adresa>" | ./a.out  
...root password changed...
```

- Napadač kao ulaz daje strojni kod koji želi da se izvrši te njegovu očekivanu adresu
- Ulaz je pažljivo namješten tako da adresa točno prepíše adresu za povratak na spremljenu na stogu.
- Nakon instrukcije `ret` izvršava se kod napadača!



Shellcode

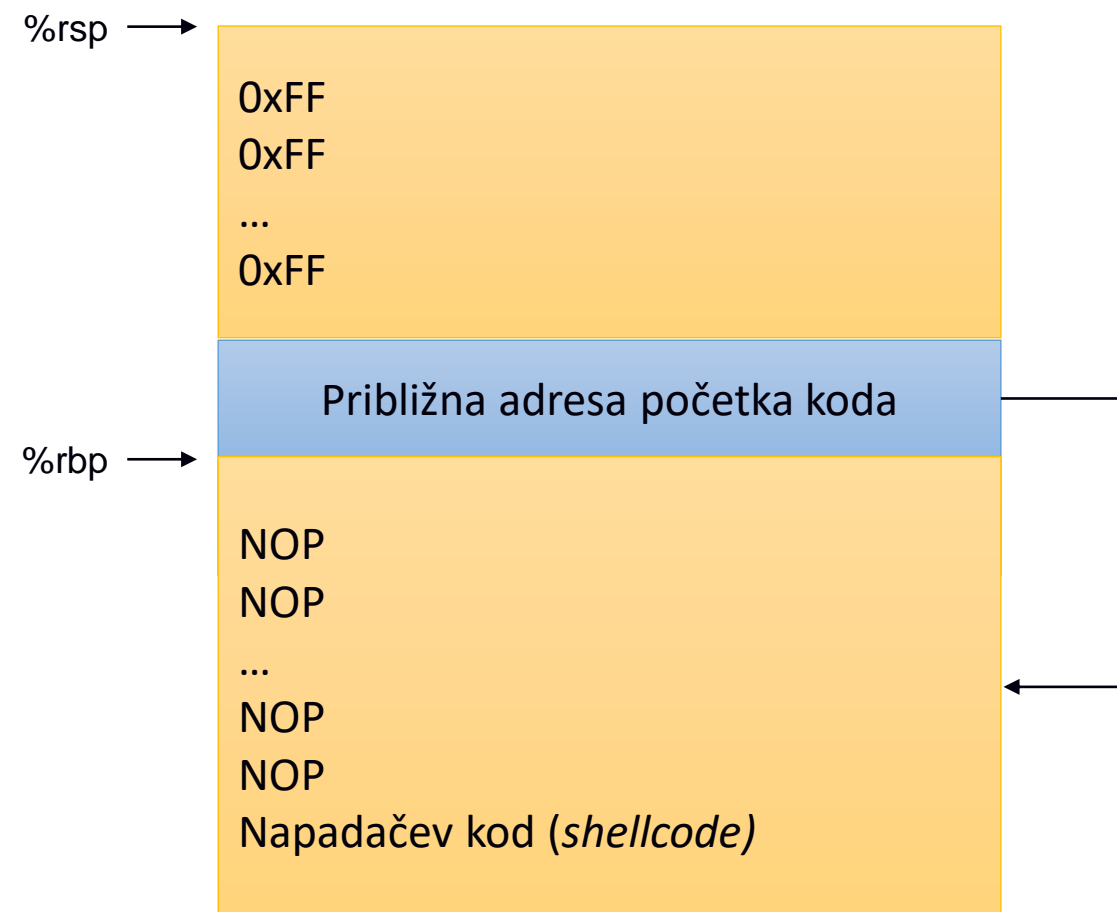
- Najčešće kratak strojni kod koji omogućuje ili olakšava napadaču postizanje cilja.
- Postoji puno lako dostupnih primjera za razne ciljeve i okoline.

- [Linux/x86-64 - Dynamic null-free reverse TCP shell - 65 bytes](#) by Philippe Dugre
- [Linux/x86-64 - execveat\("/bin//sh"\) - 29 bytes](#) by ZadYree, vaelio and DaShrooms
- [Linux/x86-64 - Add map in /etc/hosts file - 110 bytes](#) by Osanda Malith Jayathissa
- [Linux/x86-64 - Connect Back Shellcode - 139 bytes](#) by MadMouse
- [Linux/x86-64 - access\(\) Egghunter - 49 bytes](#) by Doreth.Z10
- [Linux/x86-64 - Shutdown - 64 bytes](#) by Keyman
- [Linux/x86-64 - Read password - 105 bytes](#) by Keyman
- [Linux/x86-64 - Password Protected Reverse Shell - 136 bytes](#) by Keyman
- [Linux/x86-64 - Password Protected Bind Shell - 147 bytes](#) by Keyman
- [Linux/x86-64 - Add root - Polymorphic - 273 bytes](#) by Keyman
- [Linux/x86-64 - Bind TCP stager with egghunter - 157 bytes](#) by Christophe G
- [Linux/x86-64 - Add user and password with open,write,close - 358 bytes](#) by Christophe G
- [Linux/x86-64 - Add user and password with echo cmd - 273 bytes](#) by Christophe G
- [Linux/x86-64 - Read /etc/passwd - 82 bytes](#) by Mr.Un1k0d3r
- [Linux/x86-64 - shutdown -h now - 65 bytes](#) by Osanda Malith Jayathissa
- [Linux/x86-64 - TCP Bind 4444 with password - 173 bytes](#) by Christophe G

Izvor: shell-storm.org

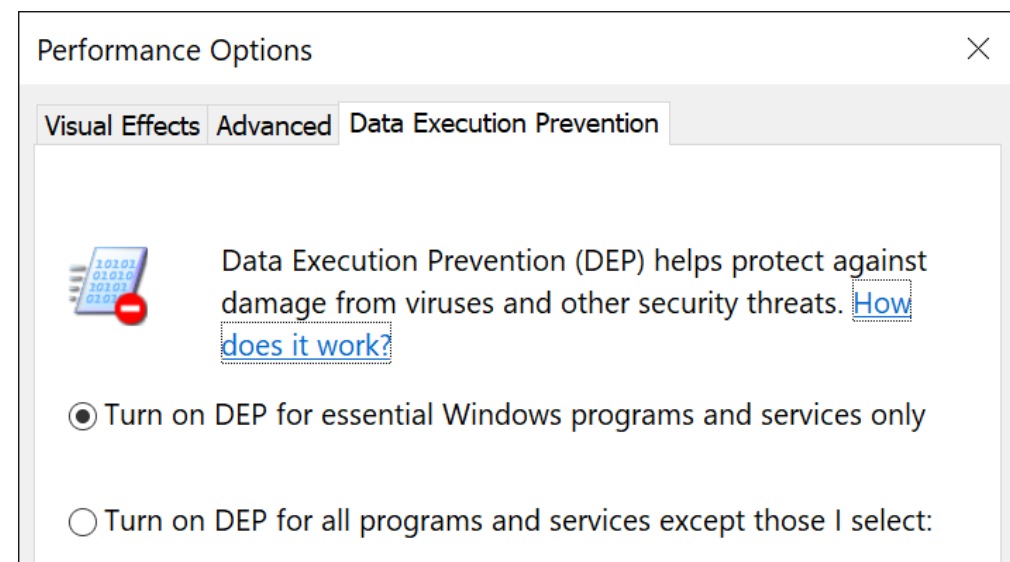
Preljevanje međuspremnika – kako pogoditi adrese?

- Napadač na temelju strojnog koda programa može znati točno kako izgleda opisnik.
- Međutim, napadač ne može skroz pouzdano predvidjeti točnu adresu vrha stoga.
- Rješenje: prije samog koda staviti dugačak niz `nop` instrukcija (ili drugih instrukcija koje ništa ne rade) – sada je dovoljno da napadač pogodi *približnu* adresu koda.
- Niz `nop` instrukcija se naziva NOP sanjke (*NOP sled*)



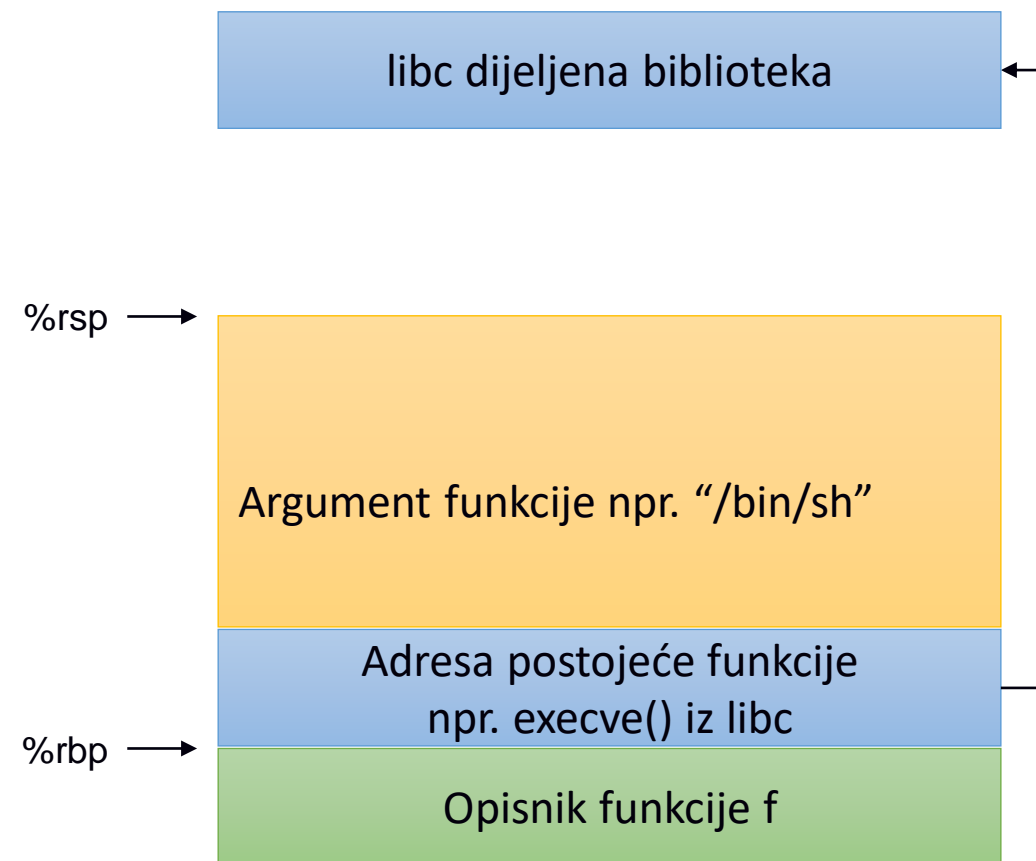
Obrana: *write xor execute*

- Razlikujemo podatke i programe: procesoru kažemo da neke memorijske stranice sadrže podatke te da se *ne smiju* izvršavati (NX bit).
- Općenito, ako se u memorijsku stranica može pisati (*writable*) onda se nikad ne može izvršavati kao kod (*executable*) – i obratno
- Većina modernih sustava (Windows, Linux, IOS) ima uključenu ovu obranu.



Zaobiliženje NX bita – *return-to-libc* napad

- Napadač prepisuje stog tako da se poziva postojeća funkcija
- Dodatno, napadač na prepisani stog stavlja i argumente za tu funkciju po njegovoj želji (pojednostavljeno jer će kod npr. x86-64 arhitekture prvih par argumenata funkcije biti u registrima).



Return Oriented Programming

- *Gadget* je kratki i jednostavni komad strojnog koda u memoriji programa koji završava instrukcijom `ret`.

Dump of assembler code for function main:

```
0x0000000000401775 <+0>:      endbr64
0x0000000000401779 <+4>:      push    rbp
0x000000000040177a <+5>:      mov     rbp, rsp
0x000000000040177d <+8>:      lea     rax, [rip+0x96880]
0x0000000000401784 <+15>:     mov     rdi, rax
0x0000000000401787 <+18>:     call   0x40c140 <puts>
0x000000000040178c <+23>:     mov     eax, 0x0
0x0000000000401791 <+28>:     pop     rbp
0x0000000000401792 <+29>:     ret
```

End of assembler dump.

0x000000000040178c:
mov eax, 0; pop rbp; ret;

0x0000000000401791:
pop rbp; ret;

Return Oriented Programming

- Napadač prepisuje stog tako da umjesto adrese za povratak stavlja *niz adresa* gadgeta zajedno s eventualnim parametrima koje ti gadgeti konzumiraju sa stoga.
- Ako je napadaču na raspolaganju dovoljno gadgeta može postići (skoro) proizvoljnu funkcionalnost.

%rsp →

0x000000000040178c (adresa gadgeta 1)
0x00007FFFFFFFE000 (gadget 1 ovo mice sa stoga)
0x00000000004146a6 (adresa gadgeta 2)
...

0x000000000040178c:
mov eax, 0; pop rbp; ret;

0x0000000000401791:
pop rbp; ret;

0x00000000004146a6:
syscall; ret;

```
rop += rebase_0(0x000000000000c108) # 0x000000000040c108: pop r13; ret;
rop += '//bin/sh'
rop += rebase_0(0x0000000000001870) # 0x0000000000401870: pop rbx; ret;
rop += rebase_0(0x000000000000c50e0)
rop += rebase_0(0x00000000000084a02) # 0x0000000000484a02: mov qword ptr [rbx], r13; pop rbx; pop rbp; pop r12; pop r13; ret;
rop += p(0xdeadbeefdeadbeef)
rop += p(0xdeadbeefdeadbeef)
rop += p(0xdeadbeefdeadbeef)
rop += p(0xdeadbeefdeadbeef)
rop += rebase_0(0x000000000000c108) # 0x000000000040c108: pop r13; ret;
rop += p(0x0000000000000000)
rop += rebase_0(0x0000000000001870) # 0x0000000000401870: pop rbx; ret;
rop += rebase_0(0x000000000000c50e8)
rop += rebase_0(0x00000000000084a02) # 0x0000000000484a02: mov qword ptr [rbx], r13; pop rbx; pop rbp; pop r12; pop r13; ret;
rop += p(0xdeadbeefdeadbeef)
rop += p(0xdeadbeefdeadbeef)
rop += p(0xdeadbeefdeadbeef)
rop += p(0xdeadbeefdeadbeef)
rop += rebase_0(0x0000000000001e3f) # 0x0000000000401e3f: pop rdi; ret;
rop += rebase_0(0x000000000000c50e0)
rop += rebase_0(0x00000000000009e6e) # 0x0000000000409e6e: pop rsi; ret;
rop += rebase_0(0x000000000000c50e8)
rop += rebase_0(0x00000000000086787) # 0x0000000000486787: pop rdx; add al, 0; ret;
rop += rebase_0(0x000000000000c50e8)
rop += rebase_0(0x00000000000072d35) # 0x0000000000472d35: pop rax; ret;
rop += p(0x000000000000003b)
rop += rebase_0(0x000000000000146a6) # 0x00000000004146a6: syscall; ret;
```

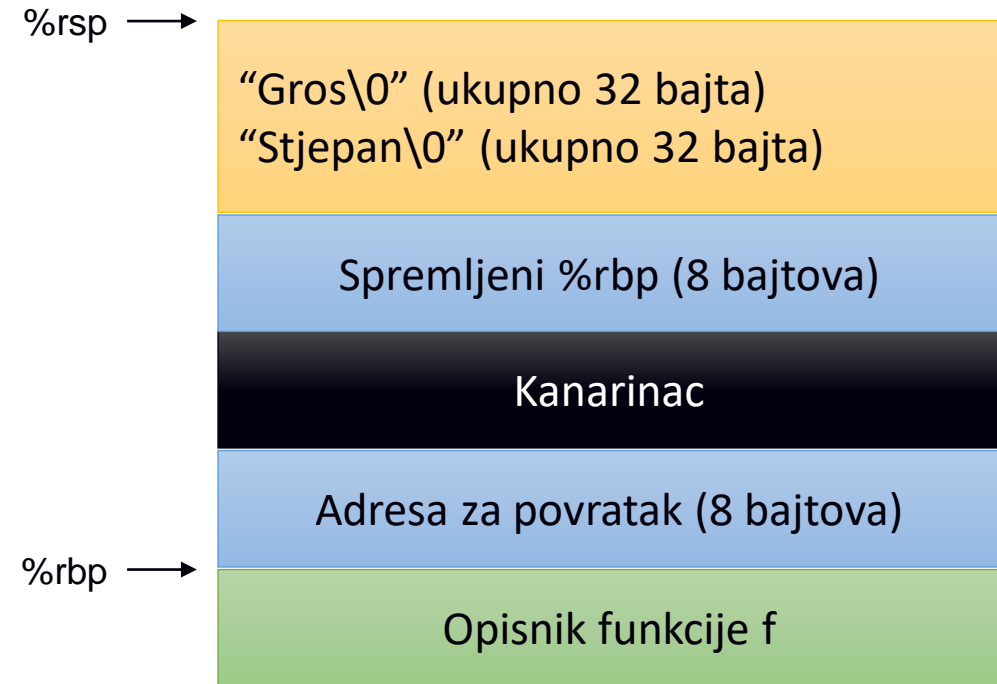
Obrana: randomizacija memorijskog prostora

- *Address space layout randomization* (ASLR)
- Adrese memorijskih stranica su (djelomično) slučajno odabrane.
- Većina modernih sustava (Windows, Linux, IOS) ima uključenu neku varijantu ove obrane.
- U 32-bitnim sustavima obrana nije efikasna zbog malo moguće entropije.
- Kako zaobići ovu obranu?

```
650:  ./a.out
000055b6d43fc000      4K r-x-- a.out
000055b6d45fc000      4K r---- a.out
000055b6d45fd000      4K rw--- a.out
000055b6d4f70000     132K rw--- [ anon ]
00007fd34e3b1000     104K r-x-- libpthread-2.27.so
00007fd34e3cb000    2044K ----- libpthread-2.27.so
00007fd34e5ca000       4K r---- libpthread-2.27.so
00007fd34e5cb000       4K rw--- libpthread-2.27.so
00007fd34e5cc000      16K rw--- [ anon ]
00007fd34e5d0000      12K r-x-- libdl-2.27.so
00007fd34e5d3000    2044K ----- libdl-2.27.so
00007fd34e7d2000       4K r---- libdl-2.27.so
00007fd34e7d3000       4K rw--- libdl-2.27.so
00007fd34e7d4000    1948K r-x-- libc-2.27.so
00007fd34e9bb000    2048K ----- libc-2.27.so
00007fd34ebbb000      16K r---- libc-2.27.so
00007fd34ebbf000       8K rw--- libc-2.27.so
00007fd34ebc1000      16K rw--- [ anon ]
00007fd34ebc5000      24K r-x-- libgtk3-nocsd.so.0
00007fd34ebcb000    2044K ----- libgtk3-nocsd.so.0
00007fd34edca000       4K r---- libgtk3-nocsd.so.0
00007fd34edcb000       4K rw--- libgtk3-nocsd.so.0
00007fd34edcc000     164K r-x-- ld-2.27.so
00007fd34efd1000      16K rw--- [ anon ]
00007fd34eff5000       4K r---- ld-2.27.so
00007fd34eff6000       4K rw--- ld-2.27.so
00007fd34eff7000       4K rw--- [ anon ]
00007ffee0ee1000     132K rw--- [ stack ]
00007ffee0f17000      12K r---- [ anon ]
00007ffee0f1a000       8K r-x-- [ anon ]
ffffffffffff600000     4K r-x-- [ anon ]
total                10840K
```

Obrana: kanarinci na stogu (*stack canaries*)

- Prevoditelj generira kod koji
 - Prilikom pozivanja funkcije stavlja na stog određenu vrijednost nepoznatu napadaču.
 - Prije završetka funkcije provjerava je li točno ta vrijednost još uvijek na stogu.

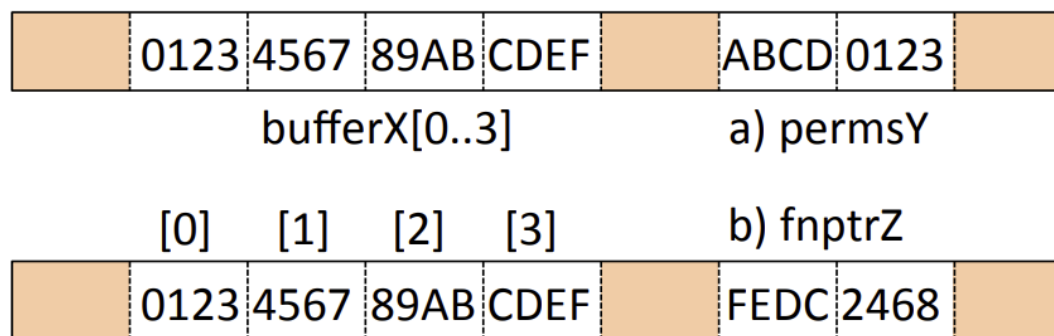


Sigurnost programske podrške – ranjivosti i napadi

Ranjivosti zasnovane na gomili

Preljev međuspremnika u gomili (*heap overflow*)

- Preljev međuspremnika u gomili prepisuje:
 - Susjedne podake na gomili.
 - Metapodatke alokatora.
- Posljedice ovise o detaljima ranjivog programa te alokatora
 - U mnogim slučajevima moguće izvršavanje proizvoljnog koda.



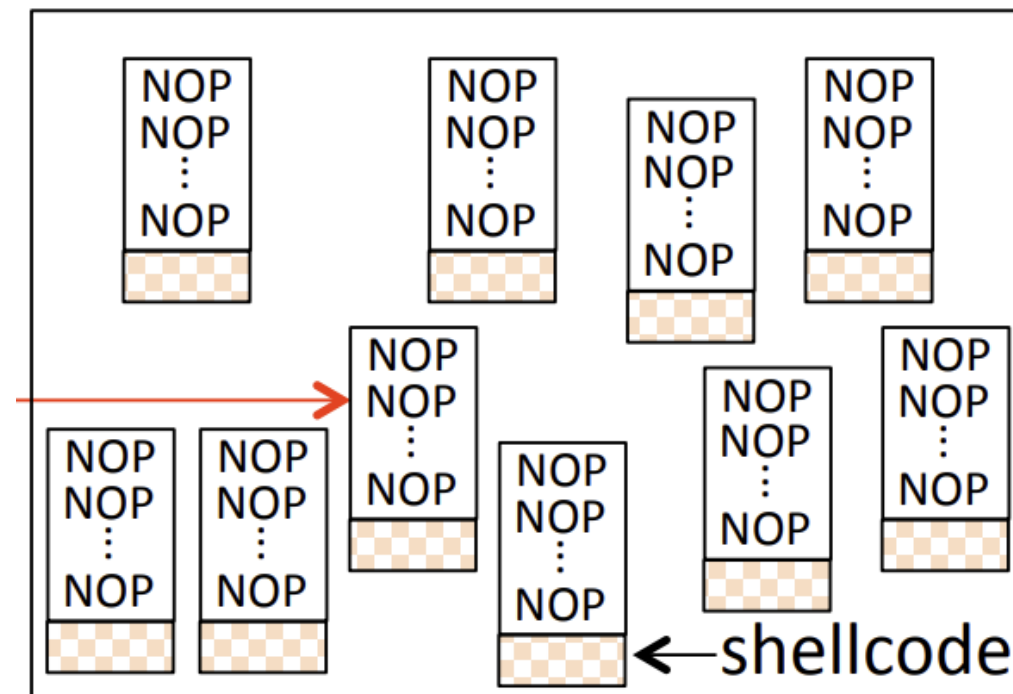
Izvor: Computer Security and the Internet: Tools and
Jewels from Malware to Bitcoin, Paul C. van Oorschot

Korištenje dealocirane memorije (use-after-free)

- Pretpostavka: program je oslobodio memoriju (*free*) ali još uvijek barata s njom (*use*), čak štoviše napadač može čitati/pisati putem “starog” pokazivača.
 - Kada ista memorija ponovno bude alocirana, napadač ima pristup (čitanje/pisanje) putem „starog” pokazivača.
 - Mnogi drugi napadi...

Heap spraying tehnika

- **Pretpostavke:**
 - Stog ima NX bit, ali je moguće izvršavati kod na gomili.
 - Napadač može dodavati proizvoljne podatke na gomilu.
 - Napadač je pronašao ranjivost pomoću koje može nastaviti izvršavanje na proizvoljnoj adresi.
- **Problem:**
 - Napadač ne može jednostavno odrediti specifičnu adresu na gomili na kojoj se nalaze njegovi podaci.
- **Napad:**
 - U prvom koraku napadač na gomilu doda puno blokova zloćudnog koda zajedno s NOP sanjkama.
 - U drugom koraku skoči na proizvoljnu adresu unutar gomile.



Izvor: Computer Security and the Internet: Tools and Jewels from Malware to Bitcoin, Paul C. van Oorschot

Ostale povezane ranjivosti

- Preljev cijelih brojeva (*integer overflow*)
 - Zaobilaženje provjera
- Ranjivosti kod formatiranja (*string format vulnerabilities*)
 - Curenje informacija, pisanje proizvoljnih podataka u memoriju
- ...

Kako spriječiti *buffer overflow* napade?

- Uključiti do sada nabrojane zaštite
- Ne koristiti nesigurne libc funkcije
 - scanf, gets, strcpy, strcmp, strncat, ...
- Alati za analizu koda (Coverity)
- Pisati programe u tipiziranom „memorijski sigurnom” programskom jeziku (npr. Rust, Go za sistemsku programsku podršku, Java za aplikacije).

Hvala!