Topici speciale în logică și securitate I

Analiză variabilelor și adreselor de memorie

Paul Irofti

Master anul II, Sem. I, 2019-2020

Cum putem detecta *buffer overflow*

Tipul de defecte software ce sunt exploatate adesea iau forma:

```
char buf[10];
i = 0;
while (i < 20) {
    buf[i] = i;
    i = i + 1;
}
```

Cum se comportă *tool*-urile existente când întâlnesc o astfel de secvență?

Cum putem folosi analiza statică să găsim astfel de defecte?

Exemplu: rutină C

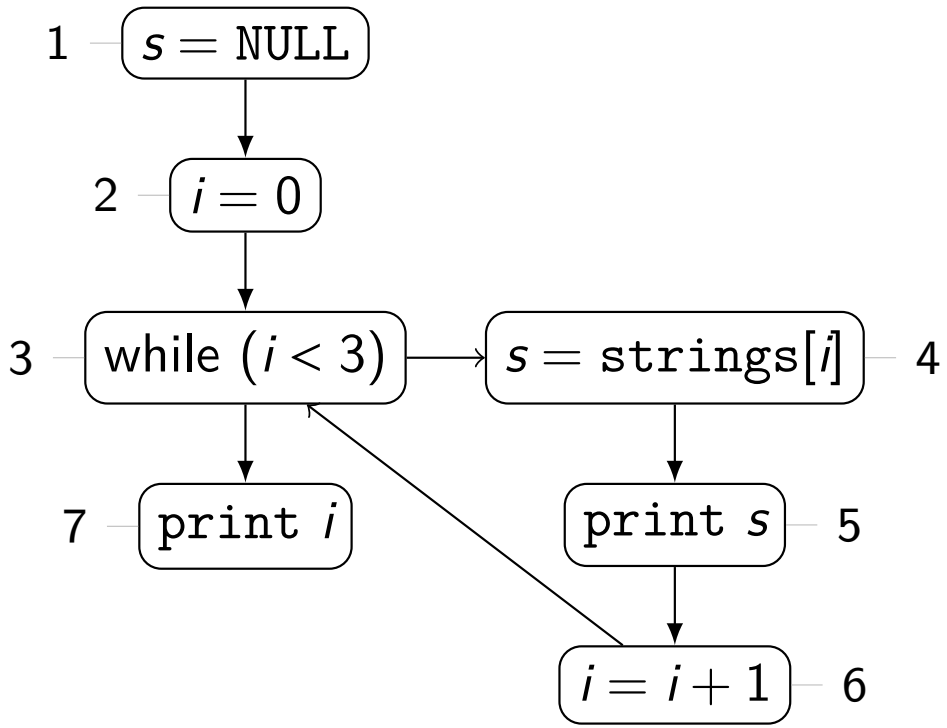
```
char *strings[] = { "Unu", "doi", "trei" };

char *s = NULL;
int i;

for (i = 0; i < 3; i++) {
    s = strings[i];
    printf("%s\n", s);
}

printf("%d\n", i);
```

Exemplu: Adaptare IMP și CFG



Convenții și notații

Notăm cu

- v_n^i – valorile posibile ale variabilei i în nodurile $n = \overline{1,6}$
- v_n^s – adresele de memorie către care s poate indica în nodurile $n = \overline{1,6}$

Descriem valorile lui i drept un interval și pe cele ale lui s drept un set abstract de adrese $\mathcal{A}$.

- `strings[i]` oferă adresa șirului i
- păstrăm această adresă drept $s_i \in \mathcal{A}$ pentru $i \in 1, 2, 3$
- notăm cu `NULL` sau $\emptyset$ adresa vidă, zero, neinițializată

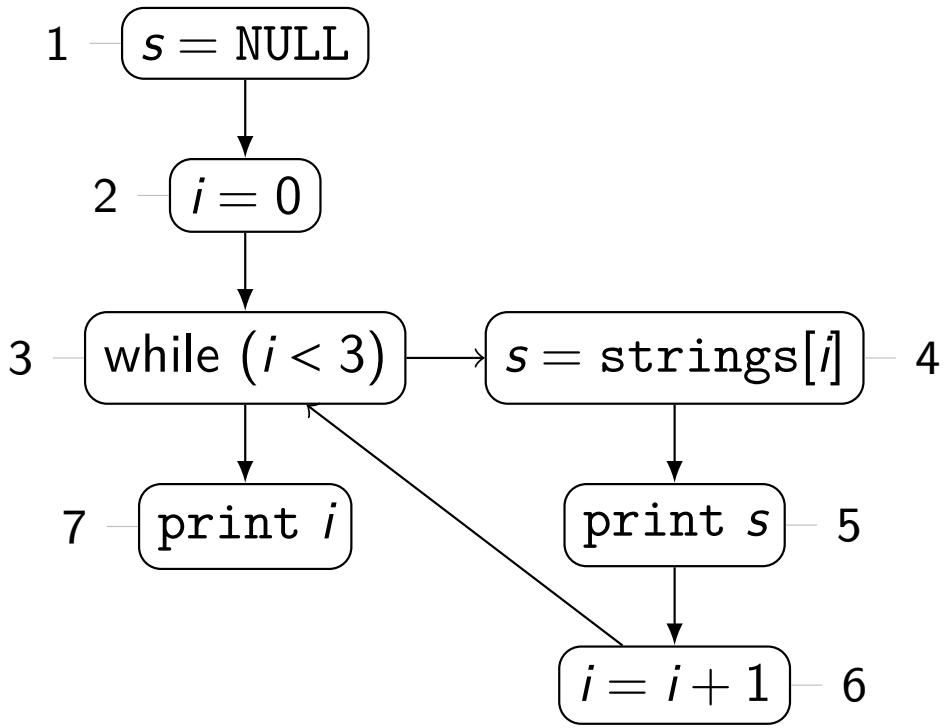
În sistemul de constrângeri vom avea nevoie de operatorul

$$[l_1, u_1] \overline{\Upsilon} [l_2, u_2] = [\min(l_1, l_2), \max(u_1, u_2)]$$

ce calculează intervalul minim ce conține cele două intervale date.

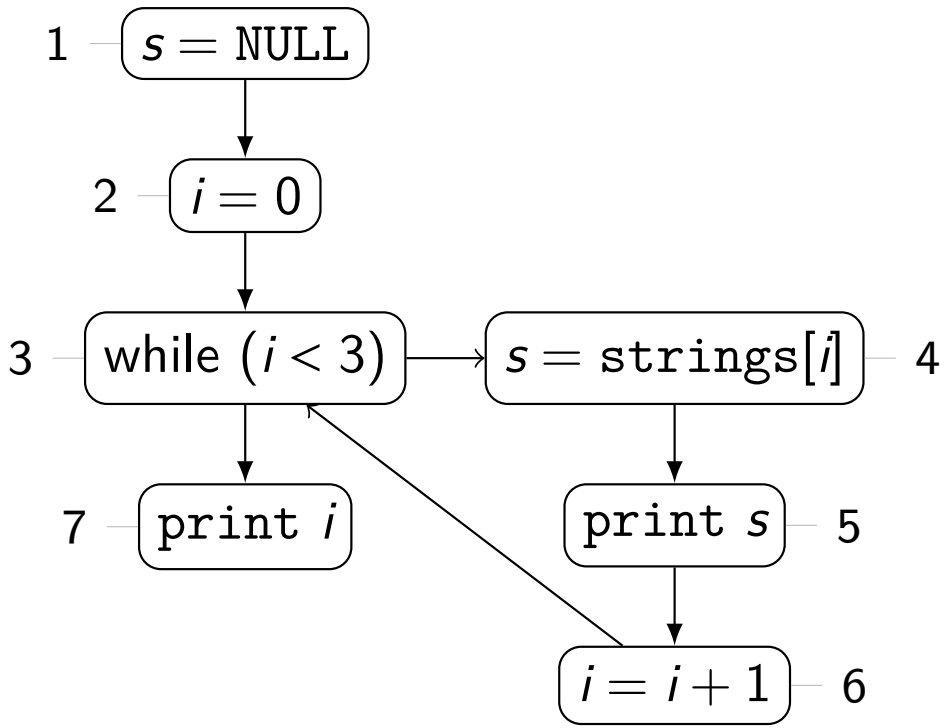
Exemplu: intervale

$$
\begin{aligned}
v_1^i &= [0,0] \\
v_2^i &= [0,0] \\
v_3^i &= v_2^i \,\overline{\Upsilon}\, v_6^i \\
v_4^i &= v_3^i \cap \left[-2^{31}, 2\right] \\
v_5^i &= v_4^i \cap \left[-2^{31}, 2\right] \\
v_6^i &= \left\{v + 1 \mid v \in v_5^i\right\} \\
v_7^i &= v_3^i \cap \left[3, 2^{31} - 1\right]
\end{aligned}
$$



Exemplu: adrese

$$
\begin{aligned}
v_1^s &= \emptyset \\
v_2^s &= v_1^s \\
v_3^s &= v_2^s \cup v_6^s \\
v_4^s &= \left\{ s_1 \mid 0 \in v_4^i \right\} \cup \\
&\quad \left\{ s_2 \mid 1 \in v_4^i \right\} \cup \\
&\quad \left\{ s_3 \mid 2 \in v_4^i \right\} \\
v_5^s &= v_4^s \\
v_6^s &= v_5^s \\
v_7^s &= v_3^s
\end{aligned}
$$



Exemplu: ecuațiile rezultate

Observați legătura dintre domeniul valorilor lui i reprezentate drept intervale și cel al `pointer`-ului s reprezentat ca seturi de adrese.

Intervale

$$
\begin{aligned}
v_1^i &= [0,0] \\
v_2^i &= [0,0] \\
v_3^i &= v_2^i \,\overline{\Upsilon}\, v_6^i \\
v_4^i &= v_3^i \cap \left[-2^{31}, 2\right] \\
v_5^i &= v_4^i \cap \left[-2^{31}, 2\right] \\
v_6^i &= \left\{v + 1 \mid v \in v_5^i\right\} \\
v_7^i &= v_3^i \cap \left[3, 2^{31} - 1\right]
\end{aligned}
$$

Adrese

$$
\begin{aligned}
v_1^s &= \emptyset \\
v_2^s &= v_1^s \\
v_3^s &= v_2^s \cup v_6^s \\
v_4^s &= \left\{s_1 \mid 0 \in v_4^i\right\} \cup \\
&\quad \left\{s_2 \mid 1 \in v_4^i\right\} \cup \\
&\quad \left\{s_3 \mid 2 \in v_4^i\right\} \\
v_5^s &= v_4^s \\
v_6^s &= v_5^s \\
v_7^s &= v_3^s
\end{aligned}
$$

Rezolvarea folosind teorema de punct fix

Soluția ecuațiilor de mai sus poate fi obținută cu teorema de punct fix:

- pornim de la starea inițială $\perp = (\emptyset, \ldots, \emptyset)$
- iterăm către vârful laticei cu $F^n(\perp) = F(F^{n-1}(\perp))$
- aici fiecare necunoscută $x_j \in \{x_1, \ldots, x_n\}$ reprezintă un tuplu alcătuit din intervalul x_j^i și setul adreselor x_j^s
- notăm starea inițială $x_j^i = \perp$ și $x_j^s = \emptyset$ astfel încât $x_j = \perp = \langle \perp, \emptyset \rangle$

Exercițiu: Determinarea celui mai mic punct fix:

	x_1^i	x_1^s	x_2^i	x_2^s	x_3^i	x_3^s	x_4^i	x_4^s	x_5^i	x_5^s	x_6^i	x_6^s	x_7^i	x_7^s
$\perp$	$\perp$	$\emptyset$	$\perp$	$\emptyset$	$\perp$	$\emptyset$	$\perp$	$\emptyset$	$\perp$	$\emptyset$	$\perp$	$\emptyset$	$\perp$	$\emptyset$
$F(\perp)$	$\perp$	{NULL}	$\perp$	$\emptyset$	$\perp$	$\emptyset$	$\perp$	$\emptyset$	$\perp$	$\emptyset$	$\perp$	$\emptyset$	$\perp$	$\emptyset$
$F^2(\perp)$	$\perp$	{NULL}	$[0, 0]$	{NULL}	$\perp$	$\emptyset$	$\perp$	$\emptyset$	$\perp$	$\emptyset$	$\perp$	$\emptyset$	$\perp$	$\emptyset$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Soluție: Determinarea celui mai mic punct fix

	x_1^i	x_1^s	x_2^i	x_2^s	x_3^i	x_3^s	x_4^i	x_4^s	x_5^i	x_5^s	x_6^i	x_6^s	x_7^i	x_7^s
$\bot$	$\bot$	$\emptyset$	$\bot$	$\emptyset$	$\bot$	$\emptyset$	$\bot$	$\emptyset$	$\bot$	$\emptyset$	$\bot$	$\emptyset$	$\bot$	$\emptyset$
1	$\bot$	{NULL}	$\bot$	$\emptyset$	$\bot$	$\emptyset$	$\bot$	$\emptyset$	$\bot$	$\emptyset$	$\bot$	$\emptyset$	$\bot$	$\emptyset$
2	$\bot$	{NULL}	$[0,0]$	{NULL}	$\bot$	$\emptyset$	$\bot$	$\emptyset$	$\bot$	$\emptyset$	$\bot$	$\emptyset$	$\bot$	$\emptyset$
3	$\bot$	{NULL}	$[0,0]$	{NULL}	$[0,0]$	{NULL}	$\bot$	$\emptyset$	$\bot$	$\emptyset$	$\bot$	$\emptyset$	$\bot$	$\emptyset$
4	$\bot$	{NULL}	$[0,0]$	{NULL}	$[0,0]$	{NULL}	$[0,0]$	$\{s_1\}$	$\bot$	$\emptyset$	$\bot$	$\emptyset$	$\bot$	{NULL}
5	$\bot$	{NULL}	$[0,0]$	{NULL}	$[0,0]$	{NULL}	$[0,0]$	$\{s_1\}$	$[0,0]$	$\{s_1\}$	$\bot$	$\emptyset$	$\bot$	{NULL}
6	$\bot$	{NULL}	$[0,0]$	{NULL}	$[0,0]$	{NULL}	$[0,0]$	$\{s_1\}$	$[0,0]$	$\{s_1\}$	$[1,1]$	$\{s_1\}$	$\bot$	{NULL}
7	$\bot$	{NULL}	$[0,0]$	{NULL}	$[0,1]$	{NULL, s_1}	$[0,0]$	$\{s_1\}$	$[0,0]$	$\{s_1\}$	$[1,1]$	$\{s_1\}$	$\bot$	{NULL}
8	$\bot$	{NULL}	$[0,0]$	{NULL}	$[0,1]$	{NULL, s_1}	$[0,1]$	$\underline{\{s_1, s_2\}}$	$[0,1]$	$\underline{\{s_1, s_2\}}$	$[1,1]$	$\underline{\{s_1, s_2\}}$	$\bot$	$\underline{\{\text{NULL}, s_1\}}$
9	$\bot$	{NULL}	$[0,0]$	{NULL}	$[0,1]$	{NULL, s_1}	$[0,1]$	$\{s_1, s_2\}$	$[0,1]$	$\{s_1, s_2\}$	$\underline{[1,2]}$	$\{s_1, s_2\}$	$\bot$	$\{\text{NULL}, s_1\}$
10	$\bot$	{NULL}	$[0,0]$	{NULL}	$\underline{[0,2]}$	$\underline{\{\text{NULL}, s_1, s_2\}}$	$[0,2]$	$\{s_1, s_2\}$	$[0,2]$	$\{s_1, s_2\}$	$[1,2]$	$\{s_1, s_2\}$	$\bot$	$\{\text{NULL}, s_1, s_2\}$
11	$\bot$	{NULL}	$[0,0]$	{NULL}	$[0,2]$	$\{\text{NULL}, s_1, s_2, s_3\}$	$[0,2]$	$\underline{\{s_1, s_2, s_3\}}$	$[0,2]$	$\{s_1, s_2, s_3\}$	$[1,2]$	$\{s_1, s_2, s_3\}$	$\bot$	$\{\text{NULL}, s_1, s_2, s_3\}$
12	$\bot$	{NULL}	$[0,0]$	{NULL}	$[0,2]$	$\{\text{NULL}, s_1, s_2, s_3\}$	$[0,2]$	$\{s_1, s_2, s_3\}$	$[0,2]$	$\{s_1, s_2, s_3\}$	$\underline{[1,3]}$	$\{s_1, s_2, s_3\}$	$\bot$	$\{\text{NULL}, s_1, s_2, s_3\}$
13	$\bot$	{NULL}	$[0,0]$	{NULL}	$\underline{[0,3]}$	$\{\text{NULL}, s_1, s_2, s_3\}$	$[0,2]$	$\{s_1, s_2, s_3\}$	$[0,2]$	$\{s_1, s_2, s_3\}$	$[1,3]$	$\{s_1, s_2, s_3\}$	$\underline{[3,3]}$	$\{\text{NULL}, s_1, s_2, s_3\}$

Observații

- execuția programului poate fi observată în semantica ecuaților
- o operație de *join* a două căi este regăsită drept o operație de reuniune în ecuații
- o operație de bifurcare (sau *meet*) a două căi este regăsită drept o operație de intersecție în ecuații
- simbolul $\perp$ indică stări ce nu pot fi atinse (încă); acest punct în program se mai numește *unreachable*
- punctele x_7^s de la iterațiile 4–12 nu au sens deoarece codul aferent este *unreachable* datorită x_7^i care este $\perp$ de la 4–12 $\Longrightarrow x_7^s = \perp$ pentru 4–12
- operațiile de *join* depind de domeniu: reuniune simplă de seturi pentru adrese sau operația complexă $\overline{\Upsilon}$ pentru intervale
- în exemplul anterior operațiile de *meet* sunt regăsite doar la intervale

Concluzie: utilizarea a două domenii simultan duce la noi observații și tipuri de analiză ce se bazează pe interacțiunea dintre cele două (*domain interaction*).

Latice

În general domeniile și operațiile aferente formează o latice.

Definiție: *Num* este domeniul numeric ce mărginește valorile posibile luate de o variabilă.

Definiție: *Pts* este domeniul indicatorilor către adrese (*points-to*) ce reprezintă zonele de memorie către care un *pointer* poate indica.

Teoremă: $(Num, \leq_N, \vee_N, \wedge_N)$ formează o latice.

- $\leq_N$ este operatorul de incluziune $\subset$
- $\vee_N = \overline{\Upsilon}$ este operația de *join* pentru intervale
- $\wedge_N$ este operația de *meet* pentru intervale

Teoremă: $(Pts, \leq_A, \vee_A, \wedge_A)$ formează o latice.

- cine este $\leq_A$?
- $\vee_A = \cup$ este operația de *join* pentru adrese
- cine este $\wedge_A$?

Domeniul abstract Pts

Definim $\mathcal{X}$ setul finit al variabilelor existente într-un program P și $\mathcal{A}$ setul finit al adreselor către care pot indica aceste variabile.

Atunci $Pts = \mathcal{X} \to \mathcal{P}(\mathcal{A})$ reprezintă funcția ce leagă fiecare variabilă $x \in \mathcal{X}$ de un subset de adrese $A(x) \in \mathcal{A}$.

Fie $A_1, A_2, A' \in Pts$.

Actualizare: $A \in Pts$ devine $A' = \{A \cup [x \to a] \mid a \in \mathcal{A}\}$ astfel încât $A'(x) = a$ și $A'(y) = A(y), \forall y \neq x$.

Ordine: $A_1 \leq_A A_2 \iff A_1(x) \subseteq A_2(x),\ \forall x \in \mathcal{X}$

Join: $A' = A_1 \vee_A A_2$ a.î. $A'(x) = A_1(x) \cup A_2(x),\ \forall x \in \mathcal{X}$.

Meet: Operația de *meet* poate fi privită ca o operație de actualizare cu ajutorul căreia filtrăm elementele din A.

Concluzie: $(Pts, \leq_A)$ reprezintă un CPO: pentru orice configurație de subseturi $B \in \mathcal{P}(Pts)$ există $A \in Pts$ astfel încât $A = \bigvee_A B \implies$ putem aplica iterații *Kleene*.

Exerciții

Ce putem spune despre `p`? Ce ne spune analiza statică?

```
int a, b, *p;
p = NULL;
if (rand())
    p = & a;
if (p)
    *p = 42;
```