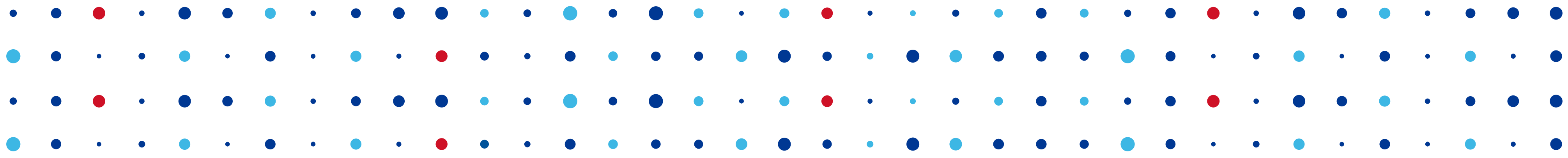
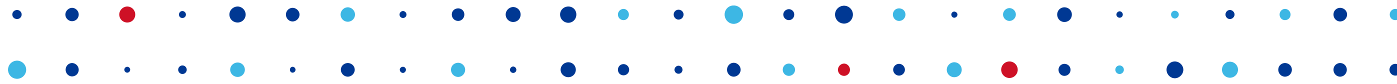


Statická a dynamická analýza kódu

Zkušenosti z vývoje Knot DNS

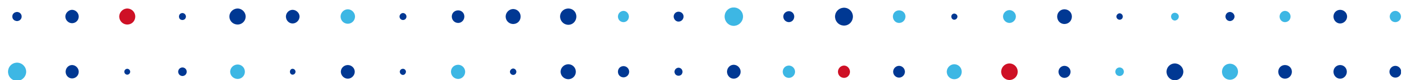
Jan Včelák • jan.vcelak@nic.cz • 13. 11. 2015



CZ.NIC | SPRÁVCE DOMÉNY CZ

Obsah prezentace

- Statická analýza
  - Coverity
  - clang-analyzer
  - cppcheck
- Dynamická analýza
  - Valgrind
  - Sanitizery v Clang



CZ.NIC | SPRÁVCE DOMÉNY CZ

Statická analýza

- Na základě zdrojových kódu
- Obrovský stavový prostor
- Symbolické vyhodnocování
- Rozsah a citlivost analýzy
  a) Flow-sensitive
  b) Path-sensitive
  c) Context-sensitive

- Coverity
  - a) + b) + c) v rámci projektu
  - asistence překladače
- clang-analyzer
  - a) + b) + c) v rámci souboru
  - asistence překladače
- cppcheck
  - a) + jednoduché c) v rámci souboru
  - nevyžaduje překladač

Statická analýza – příklad

```c
void leak_when_five(int x)
{
    char *mem = malloc(x);
    if (x < 5) { free(mem); }
    if (x > 5) { free(mem); }
}

void leak_sum_three(int x, int y)
{
    int sum = x + y;
    if (sum == 3) {
        char *byte = malloc(1);
        *byte = 1;
    }
}
```

```c
int main(void)
{
    leak_when_five(5);
    leak_sum_three(2, 3);
    return 0;
}
```

CZ.NIC | SPRÁVCE DOMÉNY CZ

Statická analýza – příklad, cppcheck

```
void leak_when_five(int x)
{
    char *mem = malloc(x);
    if (x < 5) { free(mem); }
    if (x > 5) { free(mem); }
}

void leak_sum_three(int x, int y)
{
    int sum = x + y;
    if (sum == 3) {
        char *byte = malloc(1);
        *byte = 1;
    }
}
```

```
int main(void)
{
    leak_when_five(5);
    leak_sum_three(2, 3);
    return 0;
}
```

Memory pointed to by 'mem' is freed twice.

Memory leak: byte

cz.nic | SPRÁVCE DOMÉNY CZ

Statická analýza – příklad, clang-analyzer

```c
void leak_when_five(int x)
{
    char *mem = malloc(x);
    if (x < 5) { free(mem); }
    if (x > 5) { free(mem); }
}

void leak_sum_three(int x, int y)
{
    int sum = x + y;
    if (sum == 3) {
        char *byte = malloc(1);
        *byte = 1;
    }
}
```

```c
int main(void)
{
    leak_when_five(5);
    leak_sum_three(2, 3);
    return 0;
}
```

Potential leak of memory pointed to by 'mem'

Dynamická analýza

- Analyzuje běžící kód
- Zpravidla jednoúčelové zaměření (paměť, vlákna, datové toky)
- Spuštění musí dosáhnout výskytu chyby, aby byla nalezena

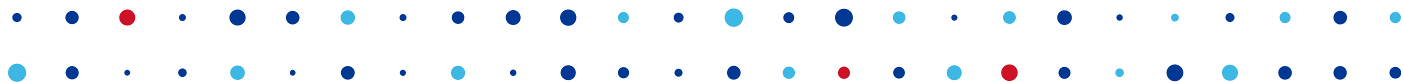
- Valgrind
  - **Memcheck**, Cachegrind, Callgrind, Massif, Hellgrind, DRD
- Clang
  - **Address Sanitizer**, **Memory Sanitizer**, **Leak Sanitizer**, Thread Sanitizer, Dataflow Sanitizer, Control Flow Integrity

Dynamická analýza – Valgrind

- Nevyžaduje speciální překlad
- Syntetický procesor, pouze jedno vlákno
- Alokace: nahrazuje malloc() a free(), přidává red-zones
- Globální proměnné, zásobník: experimentální SGCheck
- 10–50 × doba běhu
  12–18 × nároky na paměť

Dynamická analýza – Address Sanitizer

- Vyžaduje speciální překlad, -fsanitize=address
- Linkovaný kód je analyzován omezeně
- Leak Sanitizer (součástí)
  Memory Sanitizer (nelze použít současně)
- 1,5–3 × doba běhu (s optimalizacemi)
  2–4 × nároky na alokovanou paměť (někdy 1–20 ×)
  3 × paměť na zásobníku

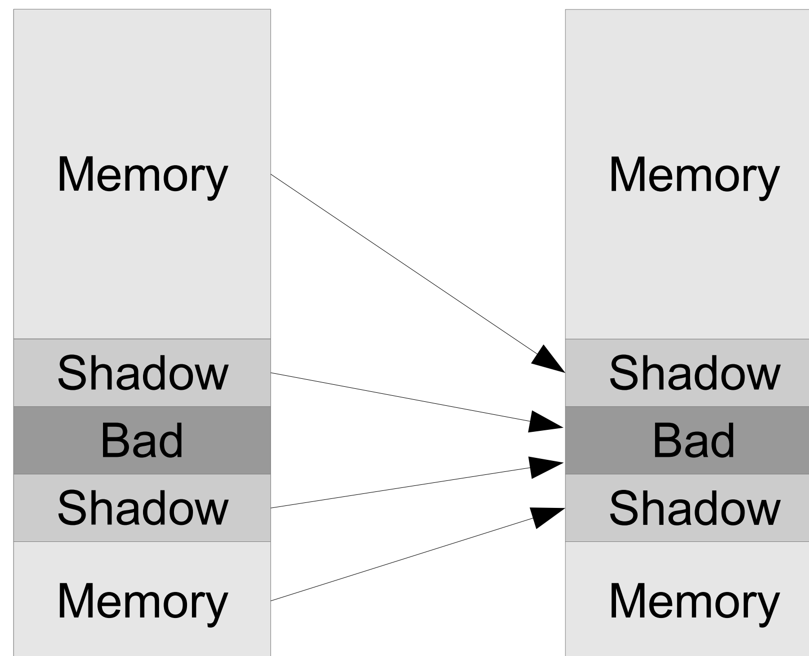


CZ.NIC | SPRÁVCE DOMÉNY CZ

Dynamická analýza – Address Sanitizer

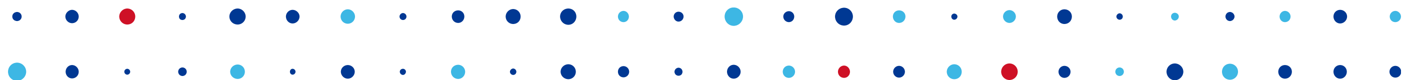
- Každý přístup do paměti je kontrolován
- Rezervuje osminu adresového prostoru pro stínovou paměť
- Přístup zarovnaný na 8 bajtů:

```
shadow = (addr >> 3) + offset;
if (*shadow) {
    report_and_crash(addr);
}
```

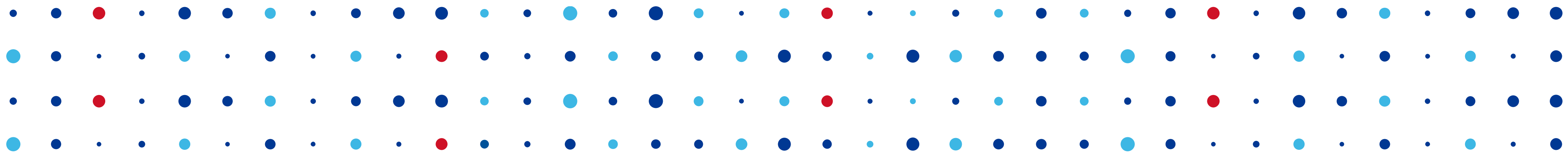


Shrnutí

- Statická i dynamická analýza má své opodstatnění
- Přístupy je vhodné kombinovat
- Existují kvalitní open-source nástroje



cz.nic | SPRÁVCE DOMÉNY CZ



Děkuji za pozornost

Jan Včelák • jan.vcelak@nic.cz