CERIAS

The Center for Education and Research in Information Assurance and Security

HexSafe: Efficient Memory Safety For C

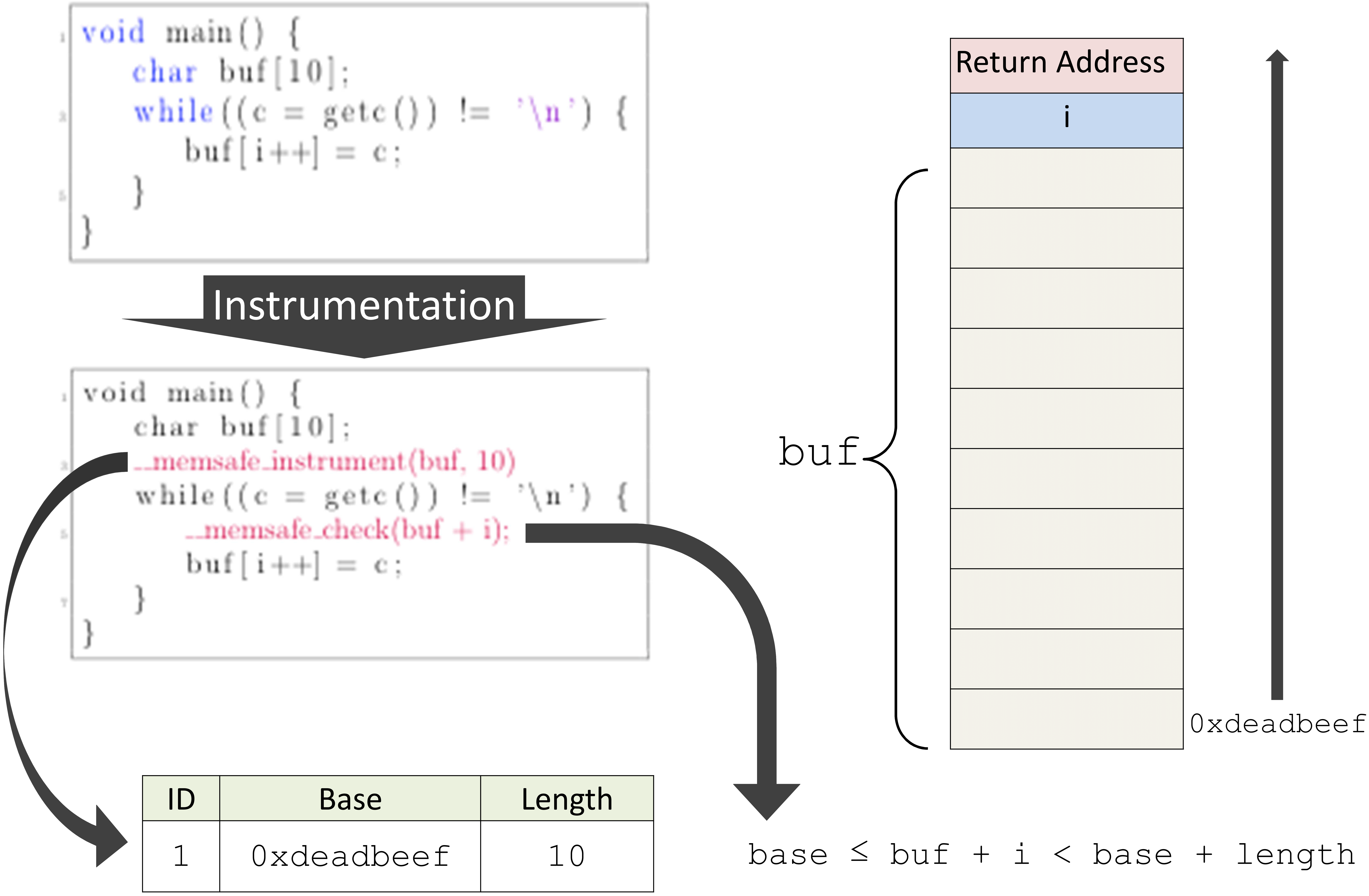
Nathan Burow, Derrick McKee, Mathias Payer

Problem Statement

- MLOC of C/C++ in critical systems
- C/C++ have no security checks
- Constant stream of exploits:
  - Heartbleed
  - Data breaches
  - APT
- Underlying problem: Programmers don't enforce ***Memory Safety***

Our Approach

- Use LLVM to insert missing security checks
  - Call our runtime to validate bounds
- New hybrid metadata approach that leverages 64 bit architectures:
  - 48 of 64 bits used for virtual addresses
  - Store an ID in the unused 16 bits
  - ID is index into our metadata table
- Advantages:
  - Faster metadata look up
  - IDs propagate naturally with pointers



```
void main() {
    char buf[10];
    while((c = getc()) != '\n') {
        buf[i++] = c;
    }
}
```

```
void main() {
    char buf[10];
    __memsafe_instrument(buf, 10)
    while((c = getc()) != '\n') {
        __memsafe_check(buf + i);
        buf[i++] = c;
    }
}
```

ID	Base	Length
1	0xdeadbeef	10

$$\text{base} \leq \text{buf} + \text{i} < \text{base} + \text{length}$$