

# Decker: Attack Surface Reduction via On-Demand Code Mapping

Chris Porter, Sharjeel Khan, Santosh Pande

*ASPLOS 2023*



Lightning talk

**CREATING THE NEXT<sup>®</sup>**

## Reducing code surface as a defense

- ❑ Modern applications are bloated with unused functionality
- ❑ Novel code reuse attacks leverage this bloated code
- ❑ Recent code reduction techniques are typically
  - ❑ Sound but conservative (too much attack surface)
  - ❑ Aggressive but unsound (can lead to crashes)
- ❑ In this work, we build a prototype called Decker that takes a constructive approach to attack surface reduction

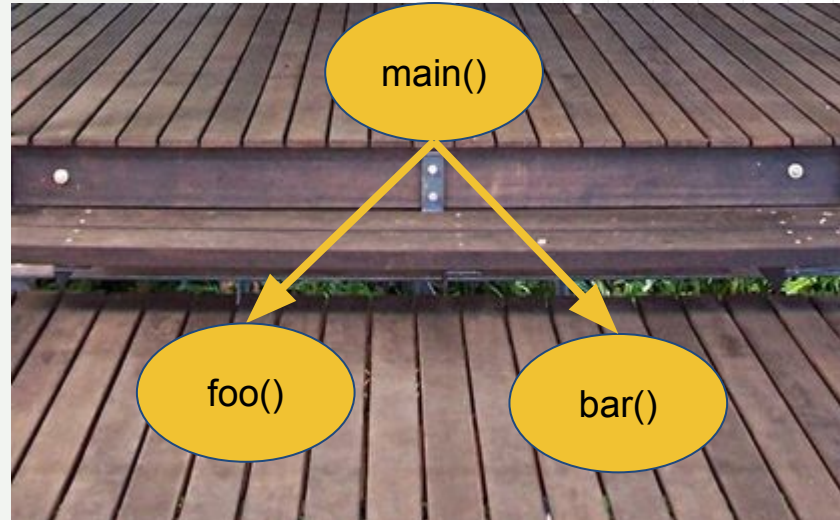
# Example



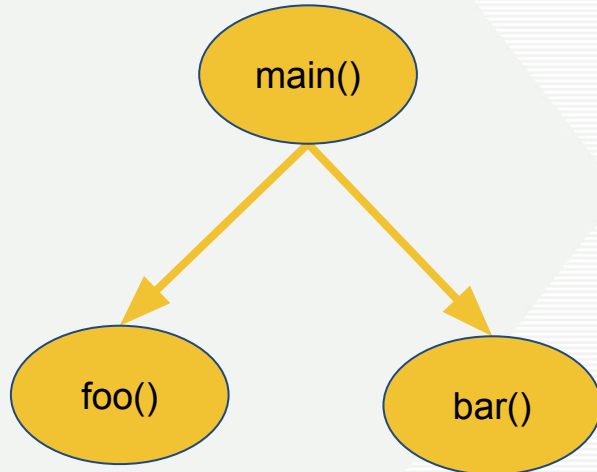
A deck

Image source: Wikipedia (deck)

# Example



# Example

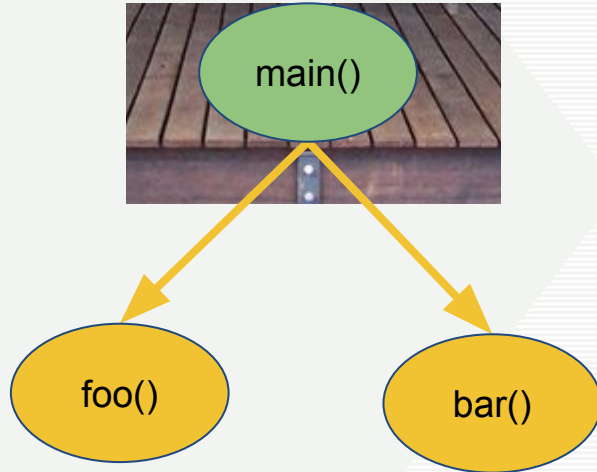


```
main()  
...  
if(x)  
    foo()  
    bar()
```

```
foo()  
...
```

```
bar()  
...
```

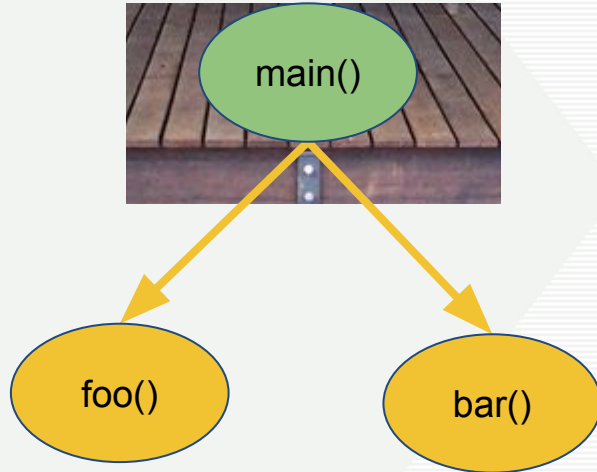
# Example



Program  
counter

```
main()  
...  
if (x)  
    foo()  
    bar()  
  
foo()  
...  
  
bar()  
...
```

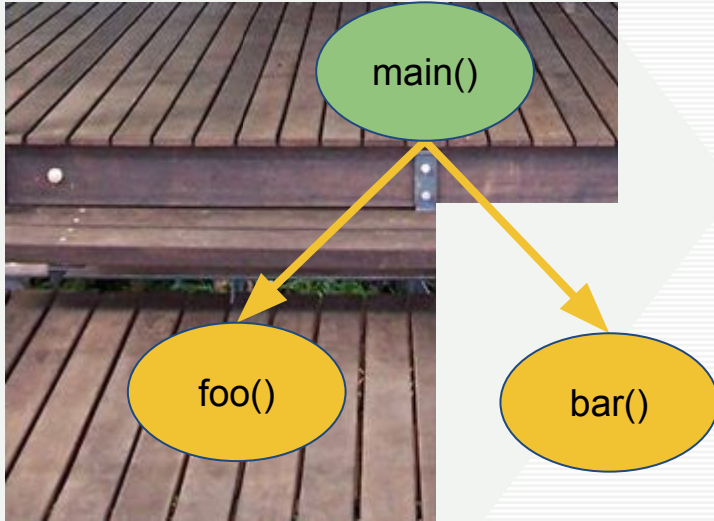
# Example



```
main ()  
  ...  
  if (x)  
    → foo ()  
  bar ()  
  
foo ()  
  ...  
  
bar ()  
  ...
```

A code snippet in a monospace font, enclosed in a black box. A blue arrow points from the `foo ()` line in the `main ()` block to the `foo ()` block below. The code shows a `main ()` function with an `if (x)` statement that calls `foo ()`, followed by `bar ()`. Below this, the `foo ()` and `bar ()` functions are shown with ellipses indicating their internal code.

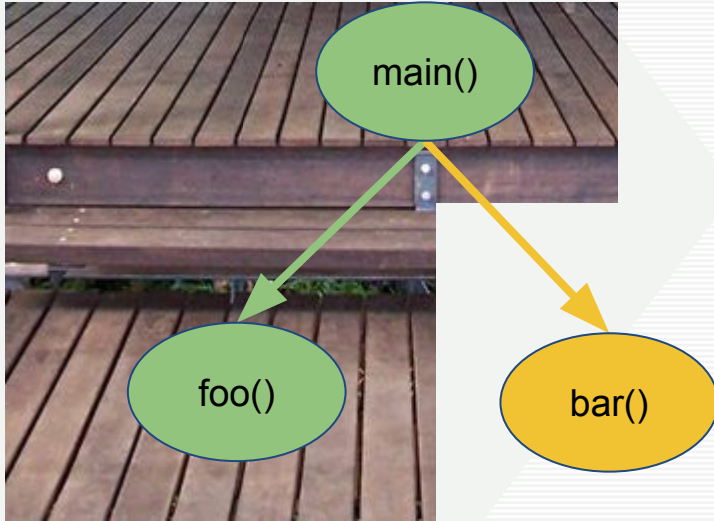
# Example



```
main ()  
  ...  
  if (x)  
    → foo ()  
  bar ()  
  
foo ()  
  ...  
  
bar ()  
  ...
```



# Example

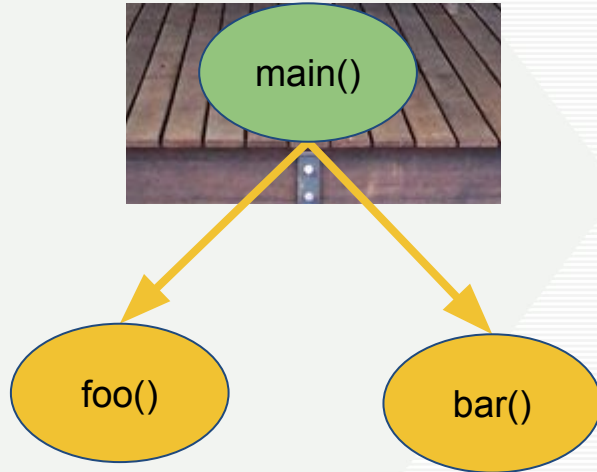


```
main ()  
...  
if (x)  
    foo ()  
    bar ()
```

**foo ()**  
...

**bar ()**  
...

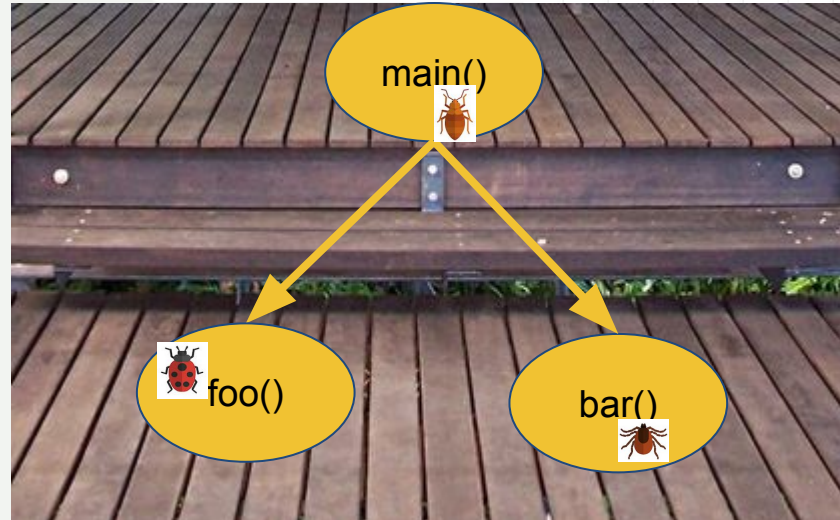
# Example



```
main ()  
  ...  
  if (x)  
    → foo ()  
    bar ()  
  
foo ()  
  ...  
  
bar ()  
  ...
```

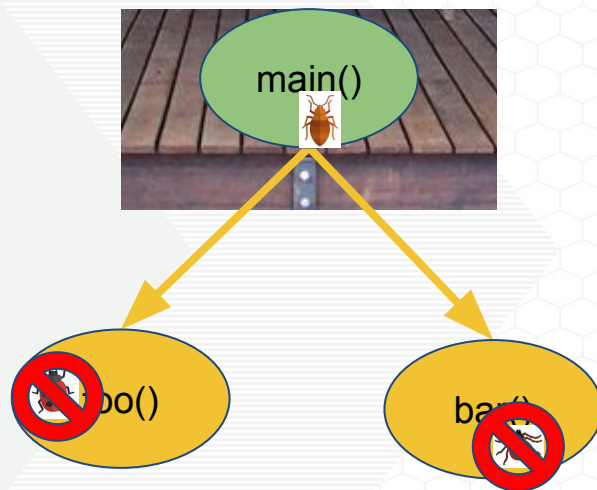
A code snippet in a monospaced font, enclosed in a black box. The code shows a `main()` function that calls `foo()` and `bar()`. A blue arrow points from the `foo()` call in the code to the `foo()` node in the call graph diagram to the left. Below `main()` and `bar()` are three dots indicating further code.

# Example



Code reuse attack components:  

# Example



Code reuse attack components:    

# Thank you!

- ❑ Please come to our talk for more details!
- ❑ Static compiler and dynamic runtime techniques
- ❑ Technique is **sound**
- ❑ nginx case study:
  - ❑ Windows and Linux support
  - ❑ Breaks real-world attacks
  - ❑ Large attack surface reduction (~80%)
  - ❑ Performant (~2% overhead)