CS 453/698: Software and Systems Security

Module: An In-depth Study of Memory Errors

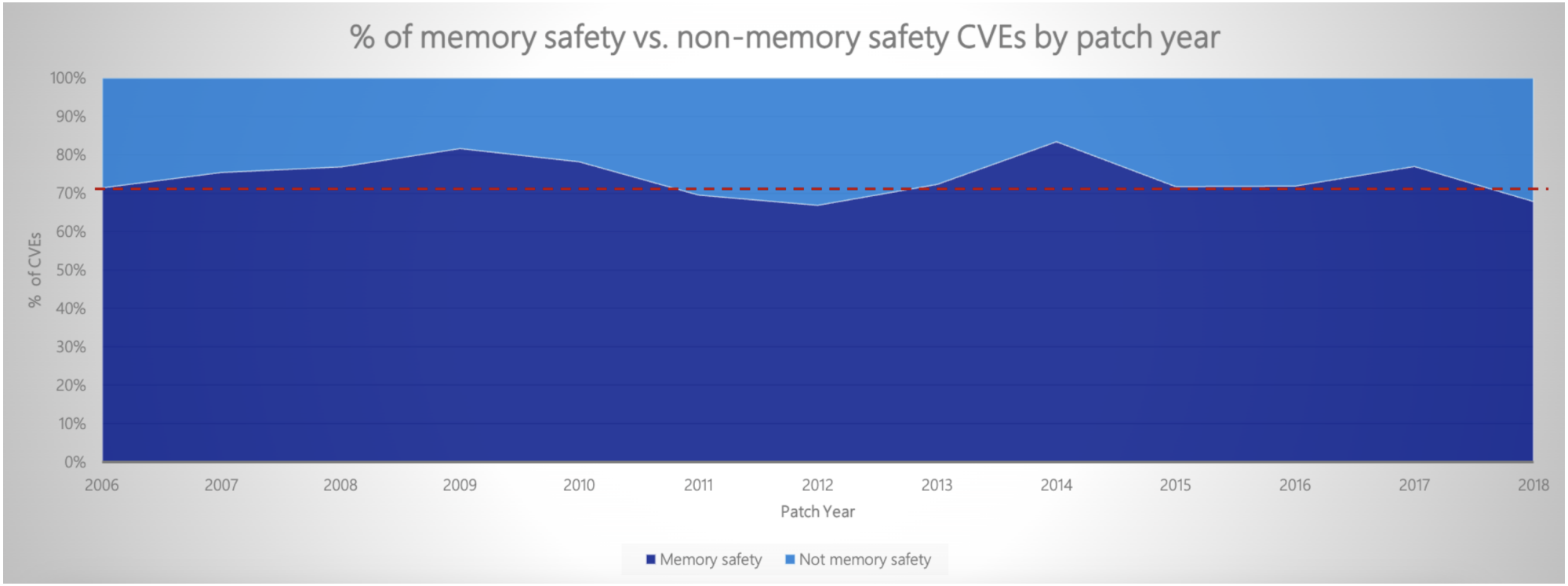
Lecture: (casual discussion) memory-safe practices

Meng Xu *(University of Waterloo)*

Spring 2025

Outline

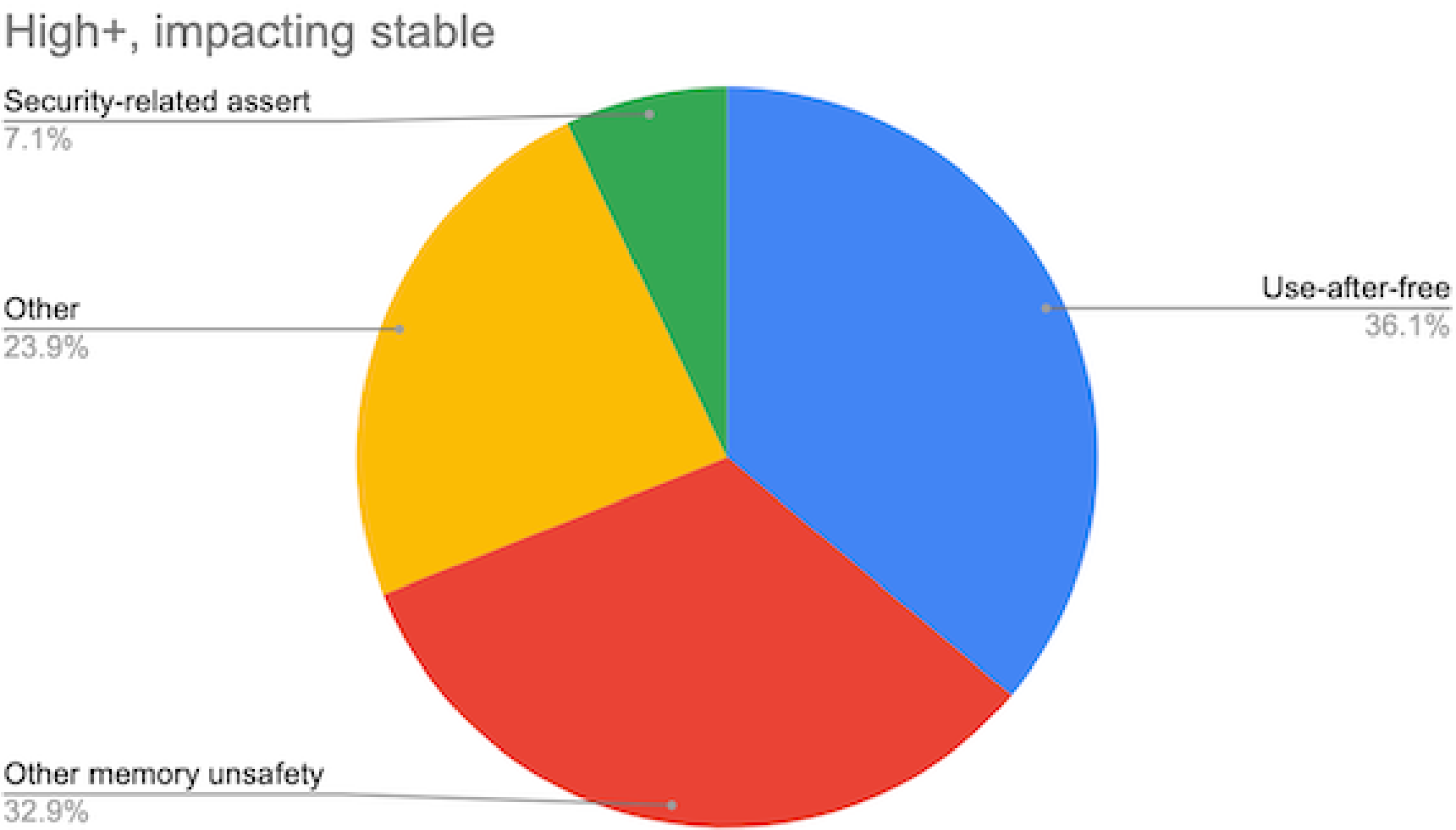
Memory errors are prevalent



Source: BlackHat IL 2019 talk by Matt Miller from Microsoft

Around 70% of all the vulnerabilities in Microsoft products addressed through a security update each year (2006 - 2018) are memory safety issues

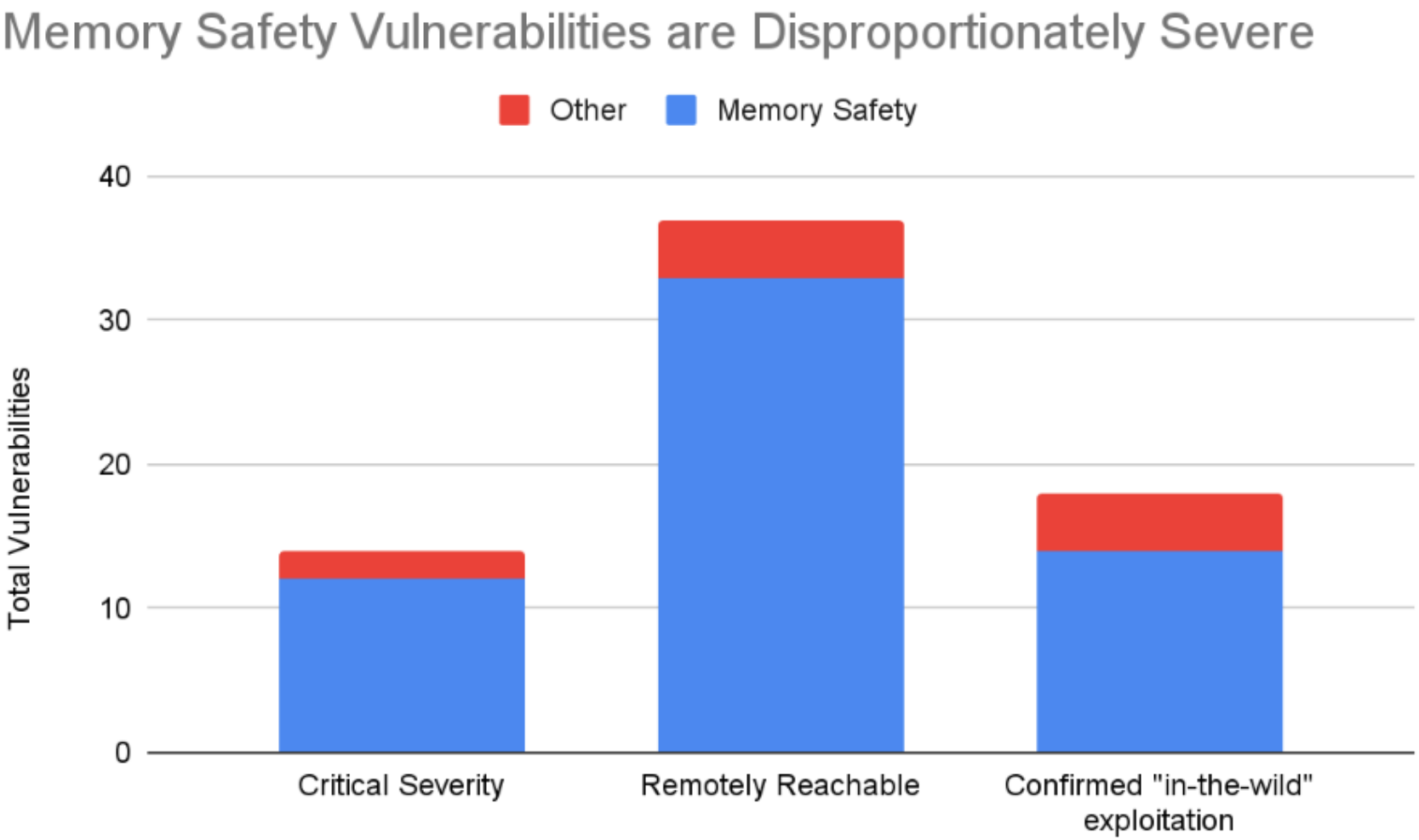
Memory errors are prevalent



Source: Chromium Memory Safety Report from Google.

Analysis based on 912 high or critical severity security bugs in Chromium reported in 2015 - 2020

Memory errors are prevalent



Source: Blog post Memory Safe Languages in Android 13 from Google.

Memory safety vulnerabilities disproportionately represent Android's most severe vulnerabilities

Statistics can be misleading...

This is a personal note: one explanation why we have a disproportionately high number of memory errors reported amongst all security vulnerabilities is that — we know memory errors too well.

Statistics can be misleading...

This is a personal note: one explanation why we have a disproportionately high number of memory errors reported amongst all security vulnerabilities is that — we know memory errors too well.

- Memory errors have universally accepted definitions (e.g., why the website is named Stack Overflow?)
  - Once you find a memory error, you do not need to diligently argue that this is a bug and not a feature

Statistics can be misleading...

This is a personal note: one explanation why we have a disproportionately high number of memory errors reported amongst all security vulnerabilities is that — we know memory errors too well.

- Memory errors have universally accepted definitions (e.g., why the website is named Stack Overflow?)
  - Once you find a memory error, you do not need to diligently argue that this is a bug and not a feature
- Memory errors often lead to a set of known consequences that are generally considered severe (e.g., data leak or denial-of-service)
  - Once you find a memory error, you do not need to construct a working exploit to justify it

Statistics can be misleading...

This is a personal note: one explanation why we have a disproportionately high number of memory errors reported amongst all security vulnerabilities is that — we know memory errors too well.

- Memory errors have universally accepted definitions (e.g., why the website is named Stack Overflow?)
  - Once you find a memory error, you do not need to diligently argue that this is a bug and not a feature
- Memory errors often lead to a set of known consequences that are generally considered severe (e.g., data leak or denial-of-service)
  - Once you find a memory error, you do not need to construct a working exploit to justify it
- Finding memory errors typically do not require program-specific domain knowledge (the bug is rooted in C/C++ language semantics instead of program logic)
  - If you have a technique that can find memory errors in one codebase, you can scale it up to millions of codebases developed in C/C++.

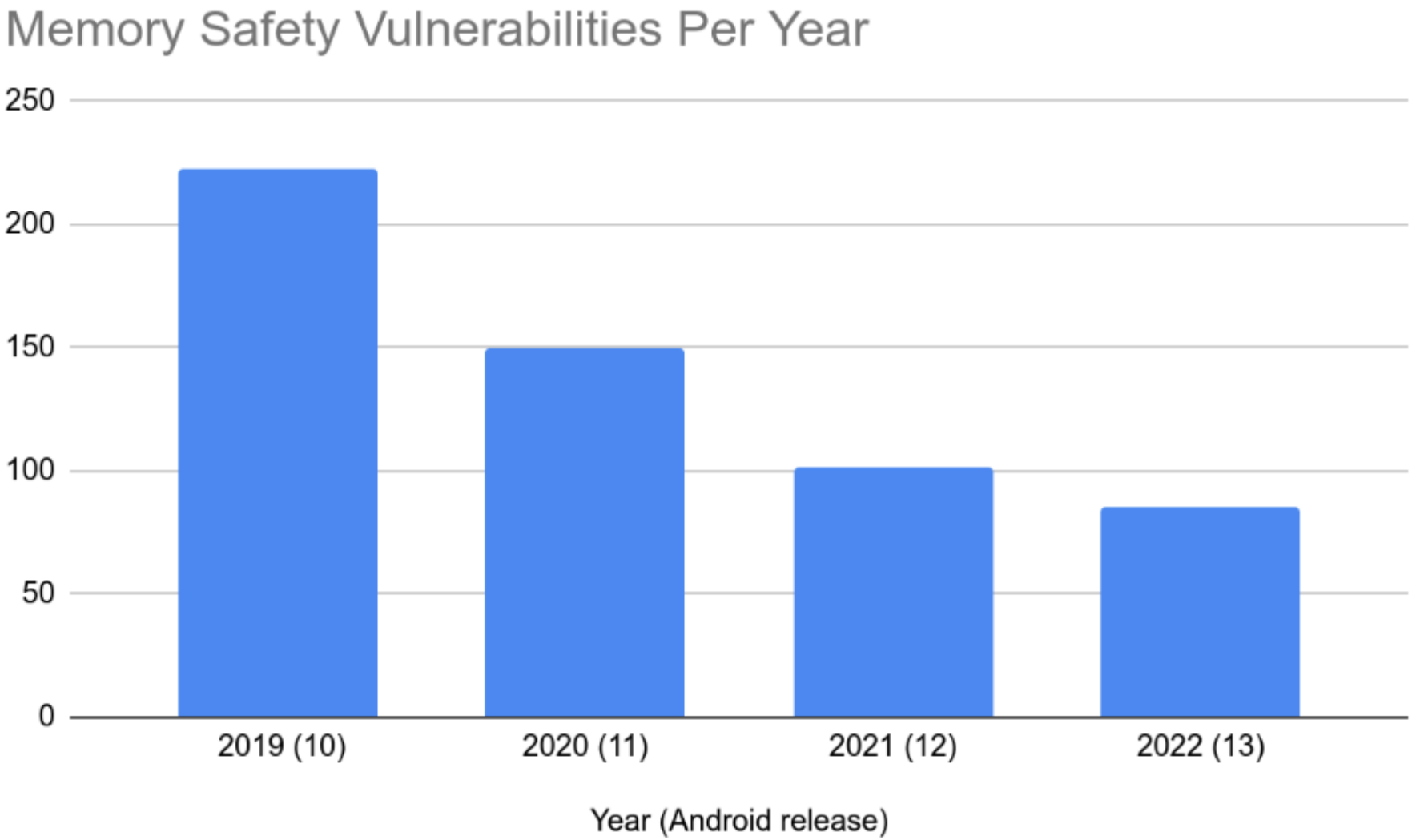
Statistics can be misleading...

This is a personal note: one explanation why we have a disproportionately high number of memory errors reported amongst all security vulnerabilities is that — we know memory errors too well.

- Memory errors have universally accepted definitions (e.g., why the website is named Stack Overflow?)
  - Once you find a memory error, you do not need to diligently argue that this is a bug and not a feature
- Memory errors often lead to a set of known consequences that are generally considered severe (e.g., data leak or denial-of-service)
  - Once you find a memory error, you do not need to construct a working exploit to justify it
- Finding memory errors typically do not require program-specific domain knowledge (the bug is rooted in C/C++ language semantics instead of program logic)
  - If you have a technique that can find memory errors in one codebase, you can scale it up to millions of codebases developed in C/C++.

In fact, very few types of vulnerabilities meet these requirements.

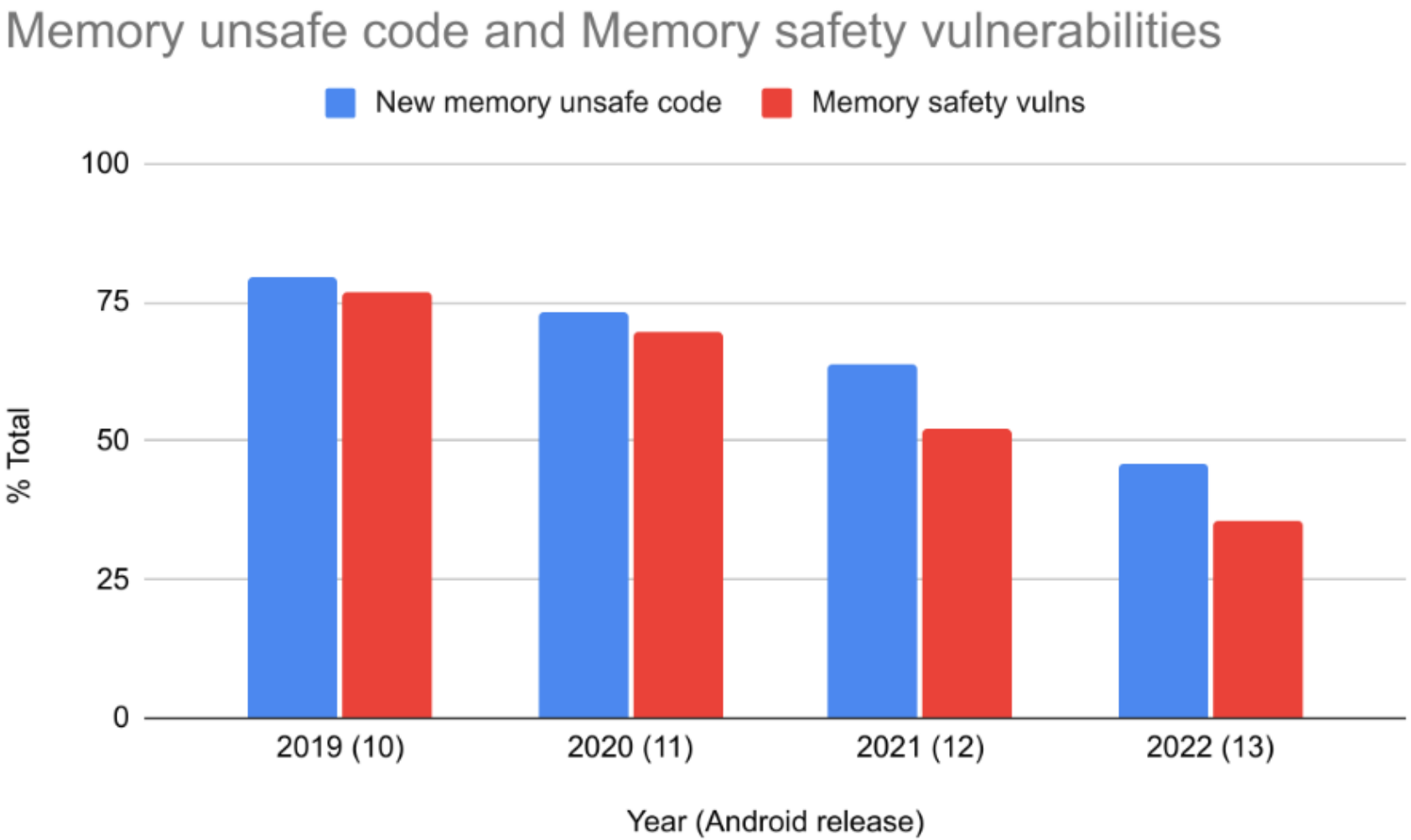
Gradual adoption of memory-safe languages



Source: Blog post Memory Safe Languages in Android 13 from Google.

Number of memory safety vulnerabilities starts to decrease with the adoption of memory-safe languages

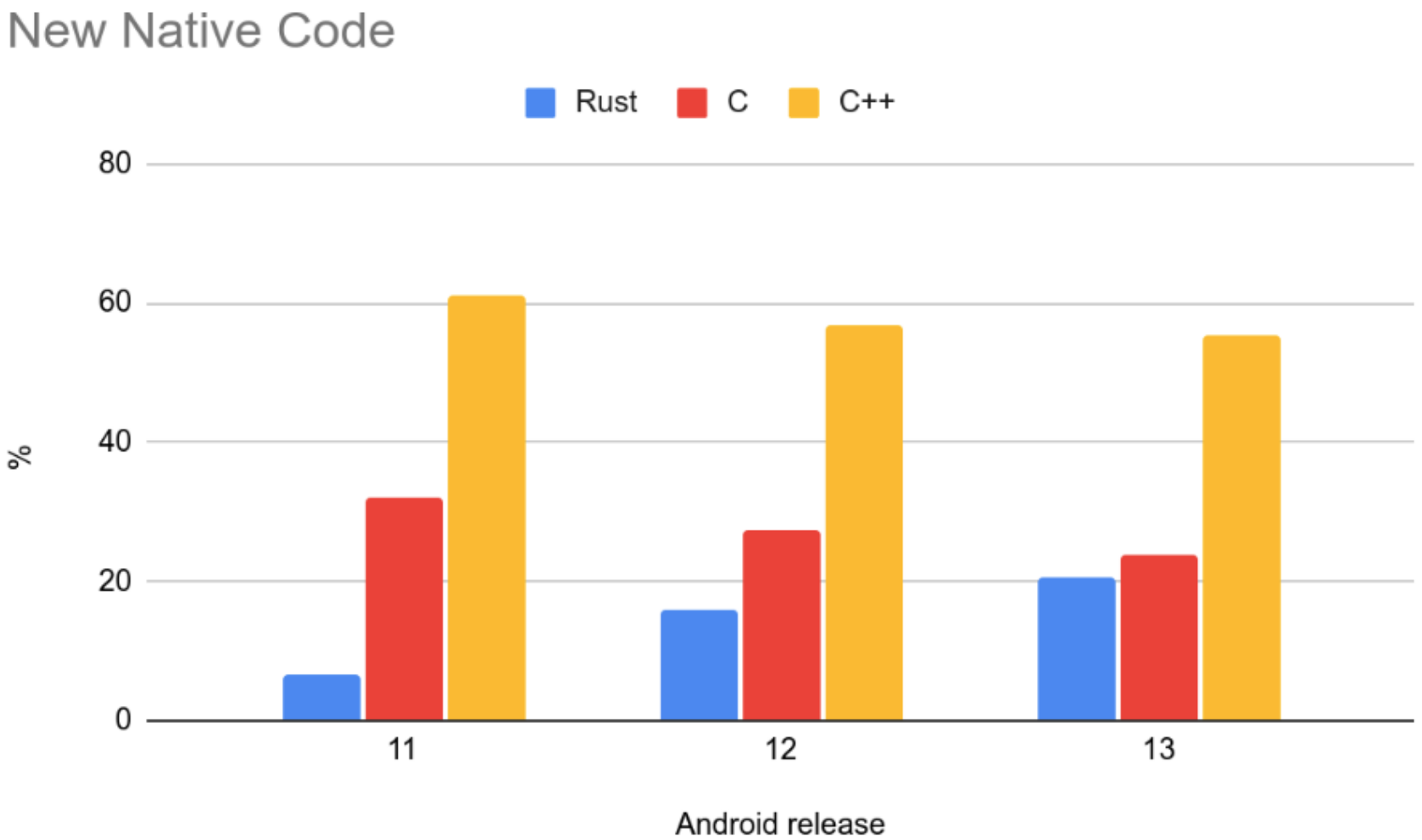
Gradual adoption of memory-safe languages



Source: Blog post Memory Safe Languages in Android 13 from Google.

Number of memory safety vulnerabilities correlates to the portion of unsafe code

Gradual adoption of memory-safe languages



Source: Blog post Memory Safe Languages in Android 13 from Google.

Rust on the rise in Android native implementations

Looking into the future

White House Press Release: Future Software Should Be Memory Safe on February 26, 2024.

ONCD Technical Report: Back to the Building Blocks: A Path Toward Secure and Measurable Software published in February 2024.

Outline

(Potentially incomplete) list of memory-safe languages

Based on technical report "The case for memory-safe roadmaps" from NSA:

- C#
- Go
- Java
- Python
- Rust
- Swift

Java/Python

Q: How is spatial safety guaranteed?

Java/Python

Q: How is spatial safety guaranteed?

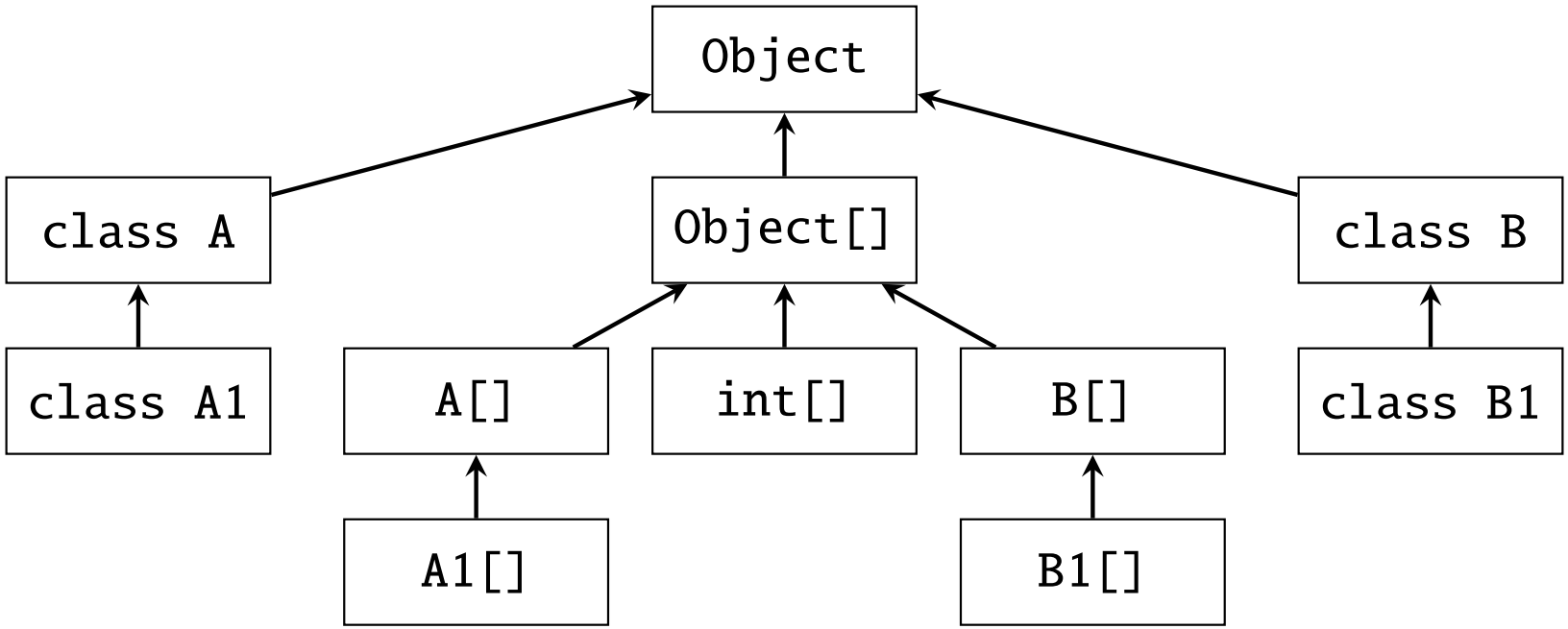
An example of the famous `ArrayIndexOutOfBoundsException`

```
1 String[] names = { "tom", "bob", "harry" };
2 for (int i = 0; i <= names.length; i++) {
3     System.out.println(names[i]);
4 }
```

The pattern looks similar to what happens in C when you have an out-of-bound memory access, but it is not a memory error in Java — **Why?**

How does Java VM track bounds

The key answer is: Java does not allow arbitrary casting.



- Upward cast is always allowed.
- Downward cast may be allowed.
- Re-interpret cast is never allowed.

Java/Python

Q: How is temporal safety guaranteed?

Java/Python

Q: How is temporal safety guaranteed?

A: Garbage collection

- Automatically managed by the Java VM
- Identifies which objects are still in use (referenced) and which are not in use (unreferenced)
- Triggered upon certain conditions, such as
  - setting a reference to `null`
  - re-assigning a new object to a reference

(Safe) Rust

Q: How is spatial safety guaranteed?

(Safe) Rust

Q: How is spatial safety guaranteed?

A: Bounds check

(Safe) Rust

Q: How is spatial safety guaranteed?

A: Bounds check

Q: How is temporal safety guaranteed?

(Safe) Rust

Q: How is spatial safety guaranteed?

A: Bounds check

Q: How is temporal safety guaranteed?

A: Several ways, including

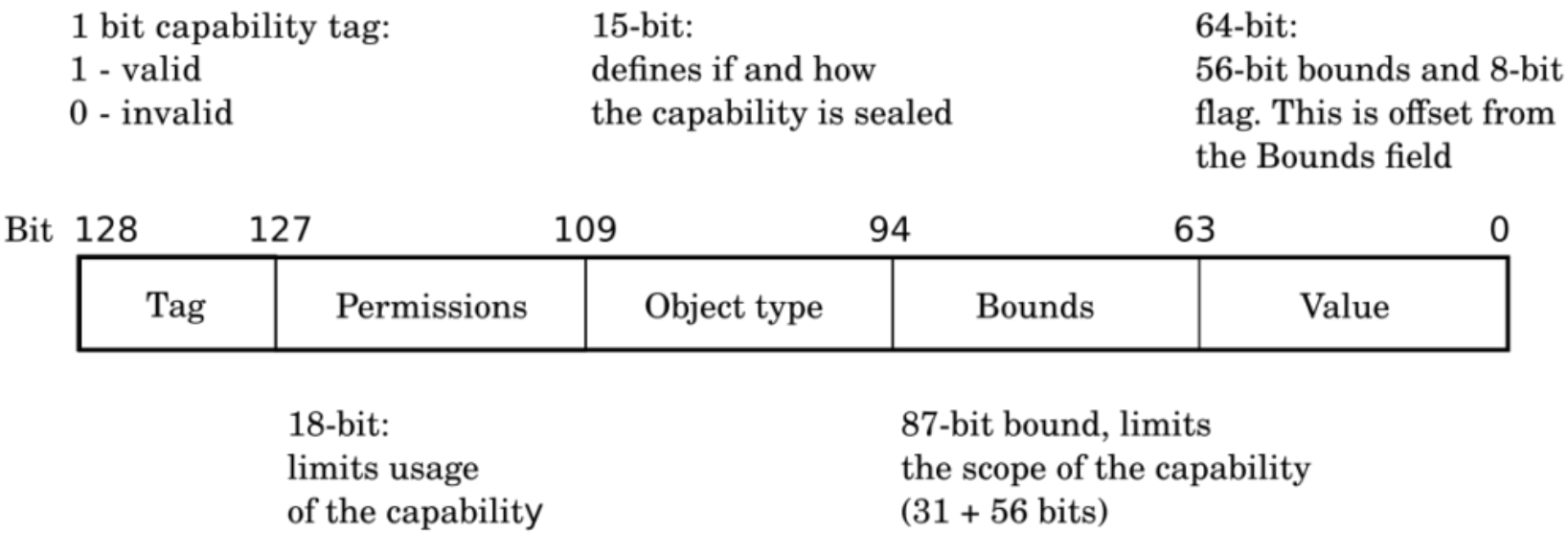
- Linear ownership transfer
- Lifetime annotation
- Reference counting

Outline

Re-defining pointers

A pointer is not only an N-bit value representing a memory address, rather, it is a capability granting certain permissions to access a restrictive range in the memory address space.

CHERI memory capability



A “pointer”, or rather, a memory capability, in the view of the CHERI Morello architecture (source of image: Pawel Zalewski’s blog post).

CHERI basic idea

```
#include <stdio.h>
int x=1;
int secret_key = 4091;
int main() {
  int *p = &x;
  p = p+1;
  int y = *p;
  printf("%d\n",y);
}
```

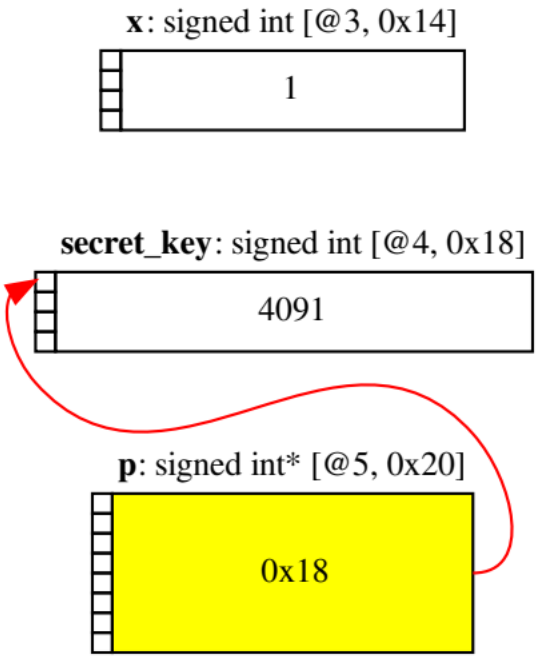
CHERI basic idea

```
#include <stdio.h>
int x=1;
int secret_key = 4091;
int main() {
  int *p = &x;
  p = p+1;
  int y = *p;
  printf("%d\n",y);
}
```

Q: What will happen?

CHERI basic idea

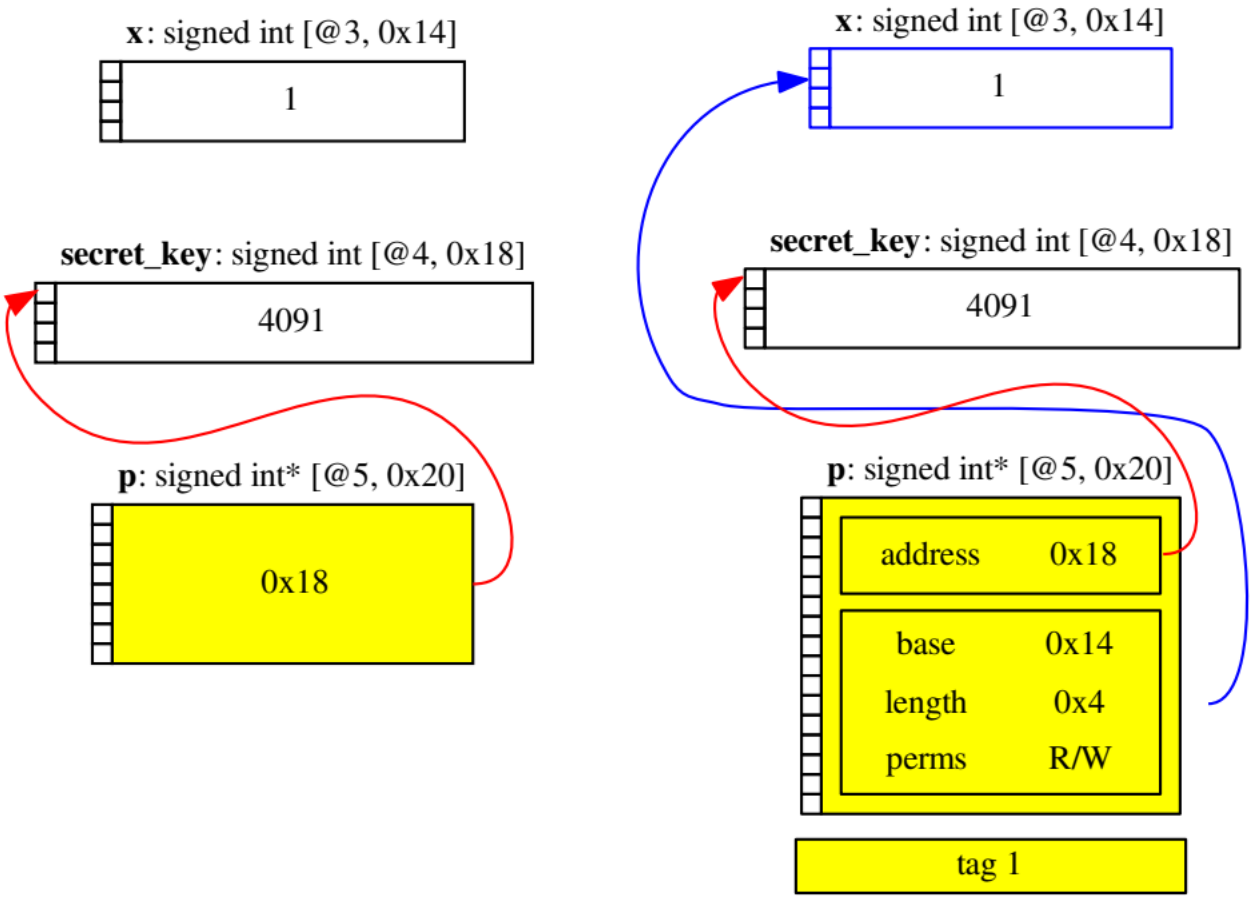
```
#include <stdio.h>
int x=1;
int secret_key = 4091;
int main() {
  int *p = &x;
  p = p+1;
  int y = *p;
  printf("%d\n",y);
}
```



Q: What will happen?

CHERI basic idea

```
#include <stdio.h>
int x=1;
int secret_key = 4091;
int main() {
  int *p = &x;
  p = p+1;
  int y = *p;
  printf("%d\n",y);
}
```



Q: What will happen?

CHERI software stack

Completely re-vamped software stack:

- Compilers: *custom-made* Clang/LLVM
- Operating systems: *hand-tuned* FreeBSD, FreeRTOS
- Applications: *ported* WebKit, OpenSSH, and PostgreSQL

⟨ End ⟩