Tabula Rasa: Starting Safe Stays Safe

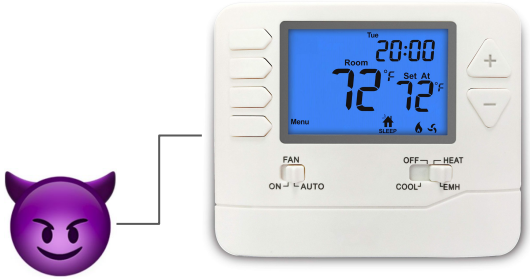
Tyler Potyondy, Samir Rashid, Leon Schuermann, Anthony Tarbinian, Pat Pannuto

PRINCETON UNIVERSITY

UNIVERSITY OF CALIFORNIA SAN DIEGO

Modern, networked embedded devices face a massively increased attack surface area!

(non-networked microcontroller)



Attacker requires physical access to compromise device.

(controllable over the internet)

Internet connectivity means attacker can compromise from anywhere!

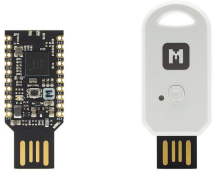
Applications using embedded OSes are incredibly diverse!

Embedded OSes must:

- Span a diverse set of target platforms
- Provide features while being aggressively modular to fit on resource constrained devices (avoid unused features)

(smart watch)

(insulin pump)



(security key)

Applications using embedded OSes are incredibly diverse!

Embedded OSes must:

- Span a diverse set of target platforms
- Provide features while being aggressively modular to fit on resource constrained devices (avoid unused features)

(smart watch)

(insulin pump)

Embedded OSes are often, by design, *highly* configurable (including opt-in security features)

(security key)

Do applications built upon embedded operating systems enable configurable, opt-in security features?

Our Study

To investigate this, we look at how embedded applications built using embedded OSes use configurable security features.

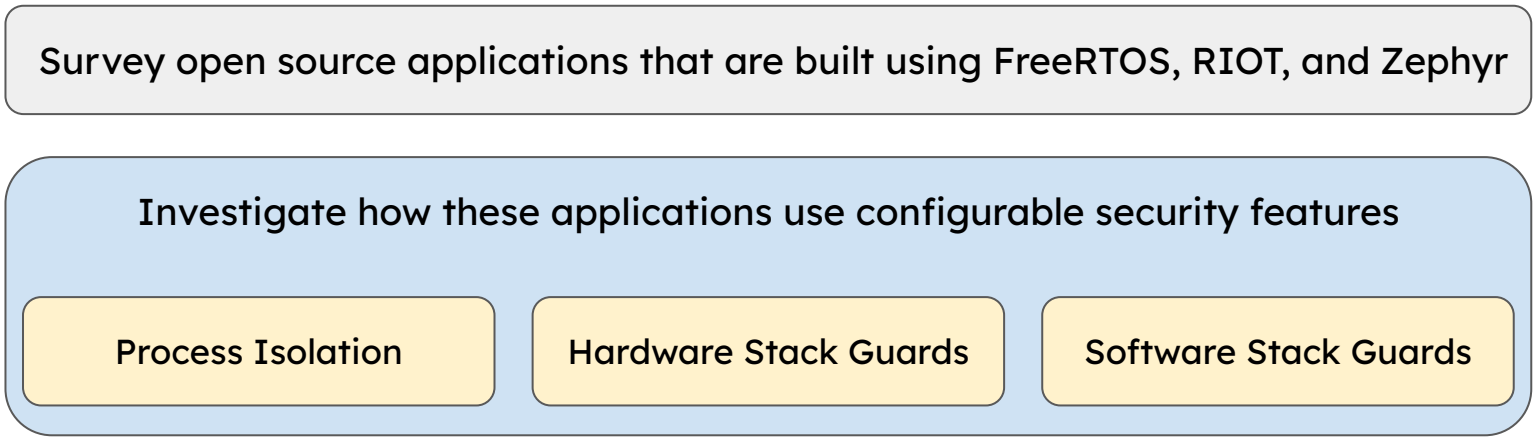
Our Study

To investigate this, we look at how embedded applications built using embedded OSes use configurable security features.

Survey open source applications that are built using FreeRTOS, RIOT, and Zephyr

Our Study

To investigate this, we look at how embedded applications built using embedded OSes use configurable security features.



Default/Available OS Security Feature Usage

	Hardware Stack Guard	Kernel/Process Isolation	Process/Process Isolation	MPU Support
FreeRTOS	○	○	○	○
RIOT	●	×	×	●
Zephyr	○	○	○	○

× No support ○ Supported, default-off ● Supported, default-on

Table 1: Survey of features available across embedded OSes.

Default/Available OS Security Feature Usage

	Hardware Stack Guard	Kernel/Process Isolation	Process/Process Isolation	MPU Support
FreeRTOS	○	○	○	○
RIOT	●	×	×	●
Zephyr	○	○	○	○

× No support ○ Supported, default-off ● Supported, default-on

Table 1: Survey of features available across embedded OSes.

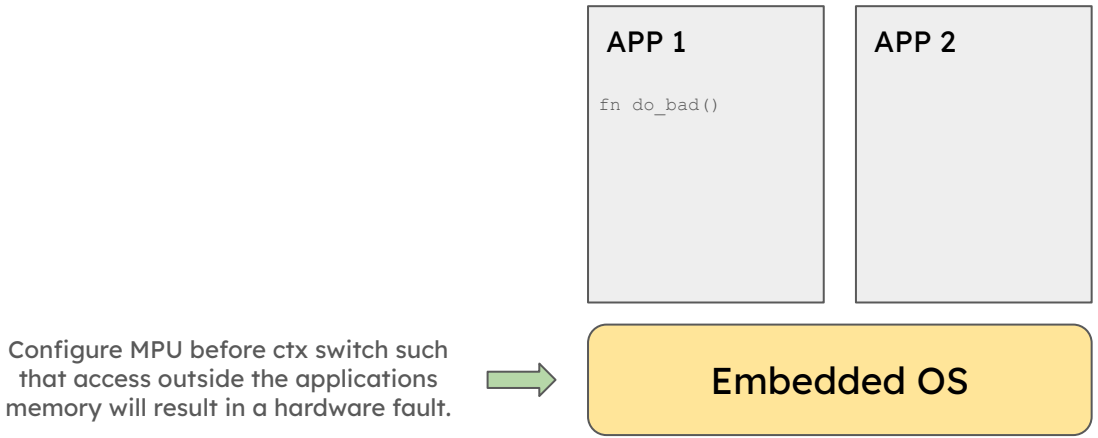
Default/Available OS Security Feature Usage

	Hardware Stack Guard	Kernel/Process Isolation	Process/Process Isolation	MPU Support
FreeRTOS	○	○	○	○
RIOT	●	×	×	●
Zephyr	○	○	○	○

× No support ○ Supported, default-off ● Supported, default-on

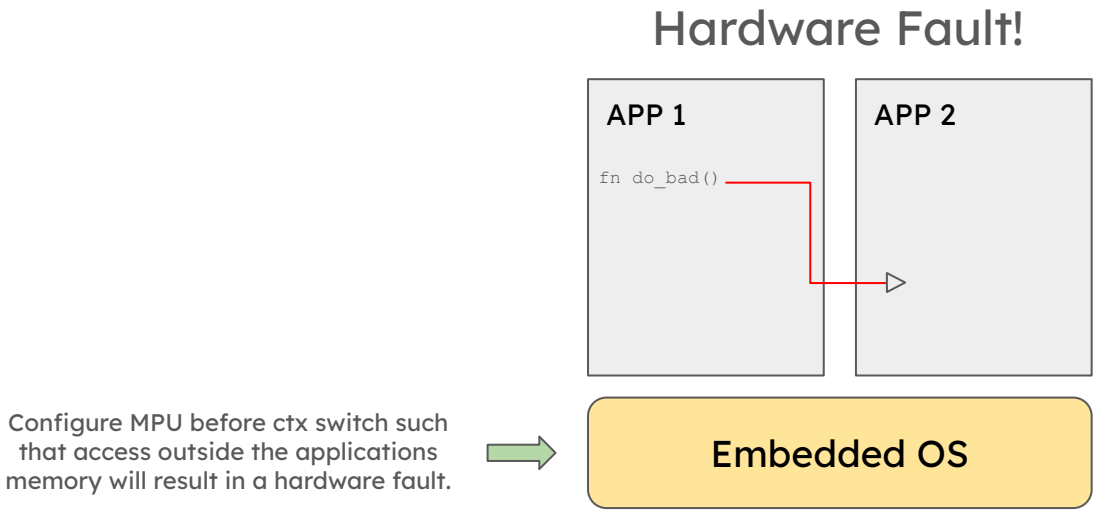
Table 1: Survey of features available across embedded OSes.

How does process isolation make embedded systems secure?



Isolate applications / tasks so that misbehavior (buggy or malicious) of one task does not compromise other tasks or the system as a whole.

How does process isolation make embedded systems secure?



Isolate applications / tasks so that misbehavior (buggy or malicious) of one task does not compromise other tasks or the system as a whole.

How do stack guards make devices more secure?

```
char buf [MAX_PACKET_LEN];

// recv packet
pkt = radio.receive();

// Copy packet based on len field.
memcopy(buf, pkt.data, pkt.len)
```

Detect and guard system against stack overflows.

How do stack guards make devices more secure?

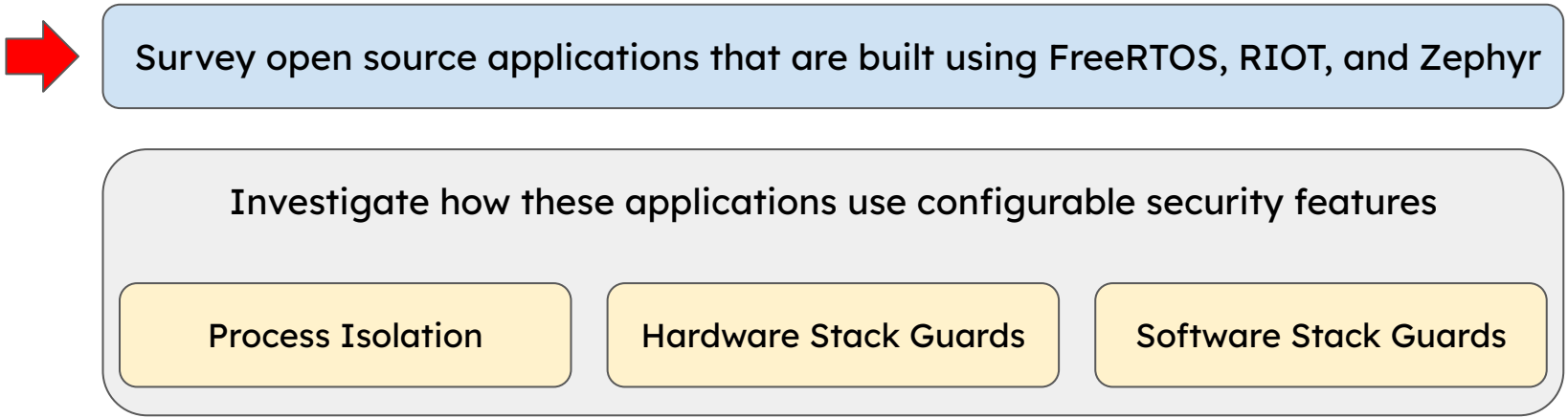
```
char buf [MAX_PACKET_LEN];

// recv packet
pkt = radio.receive();

// Copy packet based on len field.
memcopy(buf, pkt.data, pkt.len)
```

Unchecked and potentially malformed

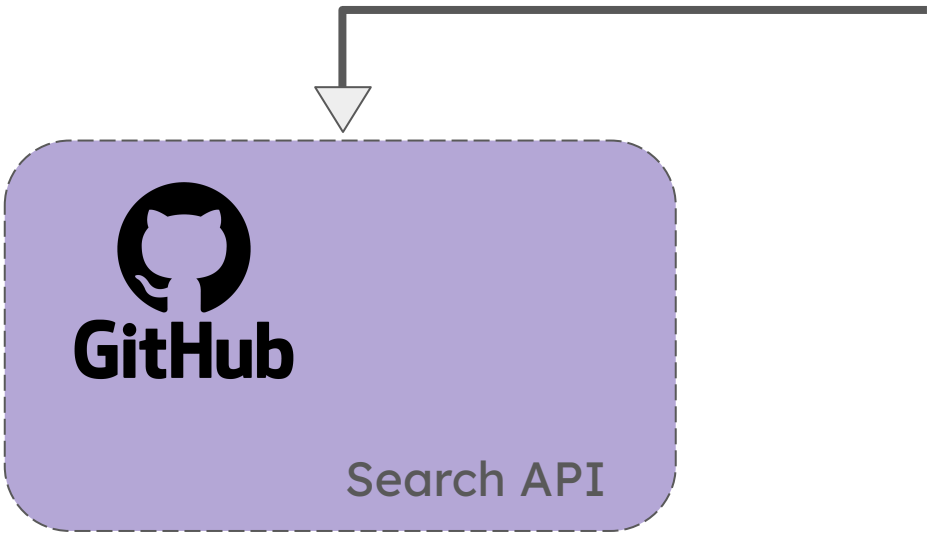
Detect and guard system against stack overflows.

Our Study

To investigate this, we look at how embedded applications built using embedded OSes use configurable security features.



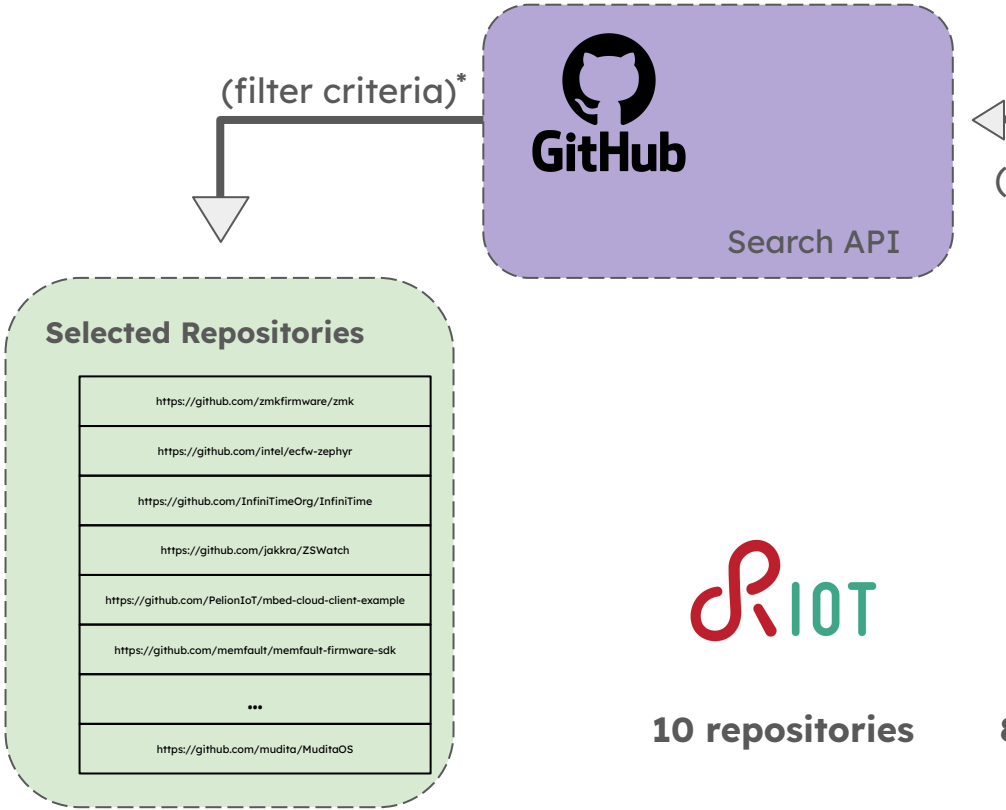
Repository Survey



Embedded OS	GitHub Query Terms
FreeRTOS	`freertos`
	`filename:FreeRTOSConfig.h`
Zephyr	`zephyr`
	`filename:west.yml`
RIOT	`RIOTBASE ?=`

Utilize Github search API to find open source applications built using each of the selected embedded OSes.

Repository Survey



Embedded OS	GitHub Query Terms
FreeRTOS	`freertos`
	`filename:FreeRTOSConfig.h`
Zephyr	`zephyr`
	`filename:west.yml`
RIOT	`RIOTBASE ?=`

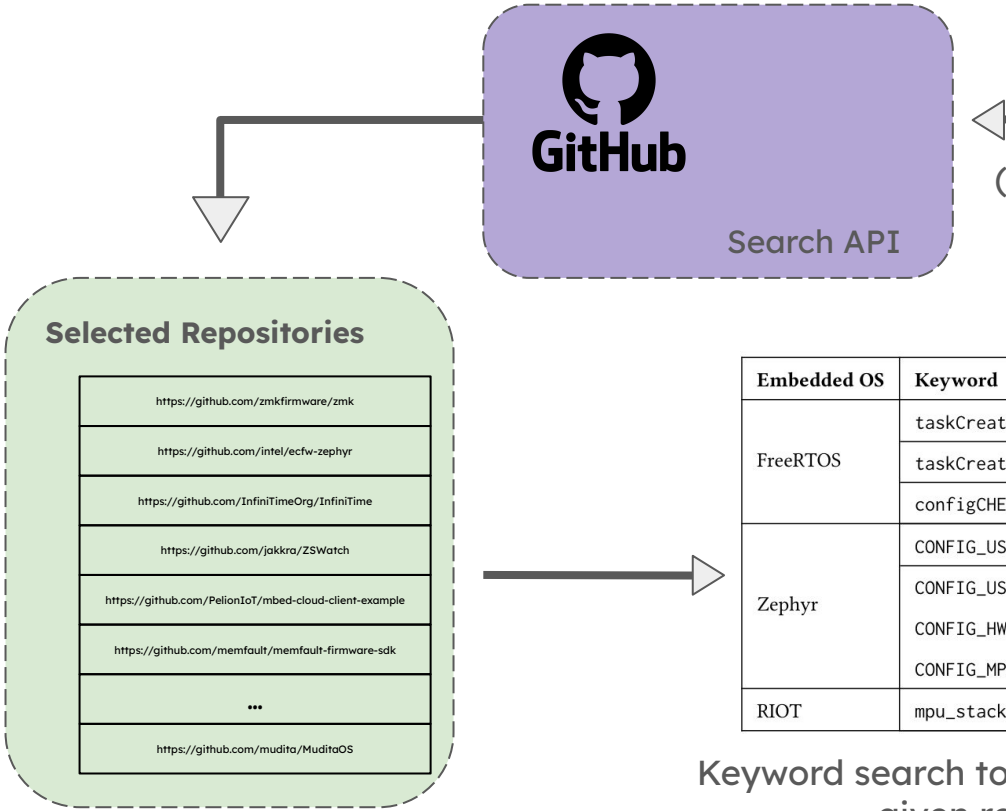
10 repositories

87 repositories

249 repositories

*Filter criteria elaborated further in paper.

Repository Survey



(search terms)

Embedded OS	GitHub Query Terms
FreeRTOS	`freertos`
	`filename:FreeRTOSConfig.h`
Zephyr	`zephyr`
	`filename:west.yml`
RIOT	`RIOTBASE ?=`

Embedded OS	Keyword	Feature
FreeRTOS	`taskCreateRestricted()`	Process Isolation
	`taskCreateRestricted()`	HW Stack Guard
	`configCHECK_FOR_STACK_OVERFLOW`	SW Stack Guard
Zephyr	`CONFIG_USERSPACE`	Process Isolation

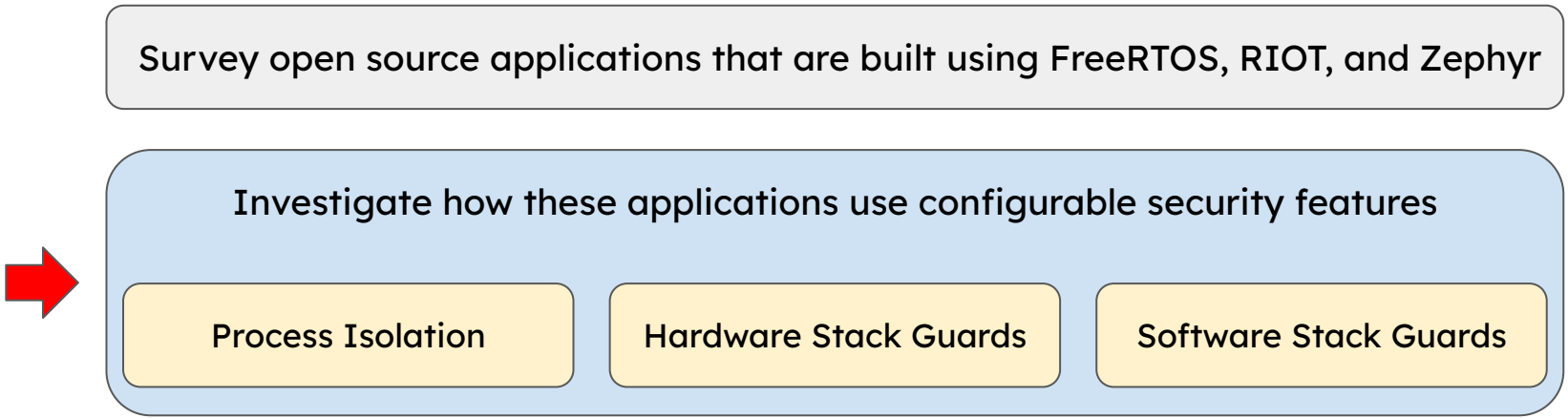
	`CONFIG_USERSPACE`
	`CONFIG_HW_STACK_PROTECTION`
	`CONFIG_MPU_STACK_GUARD`
		HW Stack Guard

RIOT	`mpu_stack_guard`	HW Stack Guard

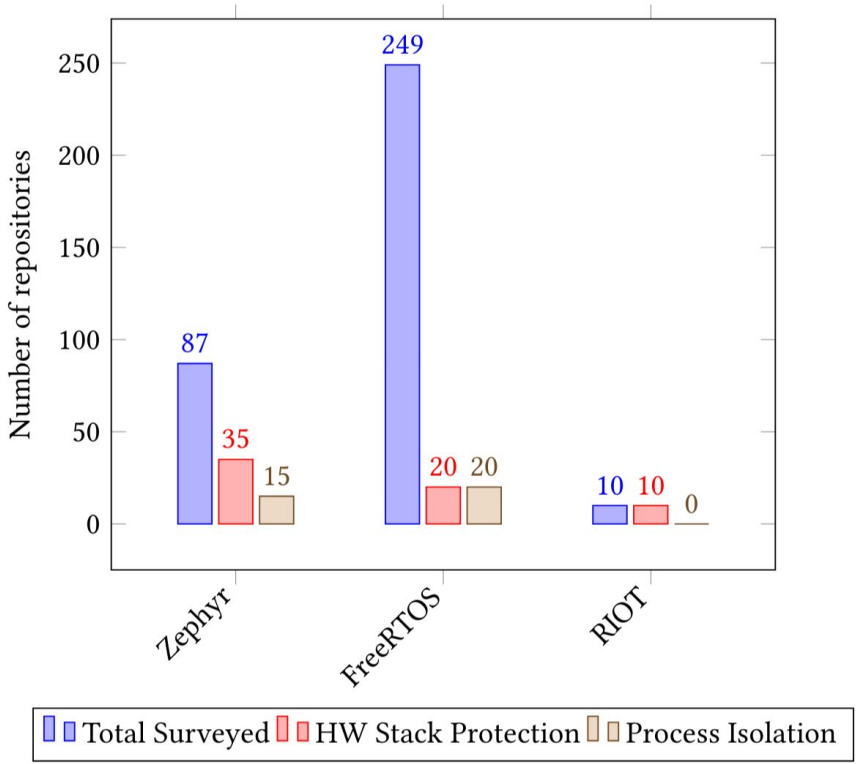
Keyword search to determine which features a given repository enables.

Our Study

To investigate this, we look at how embedded applications built using embedded OSes use configurable security features.



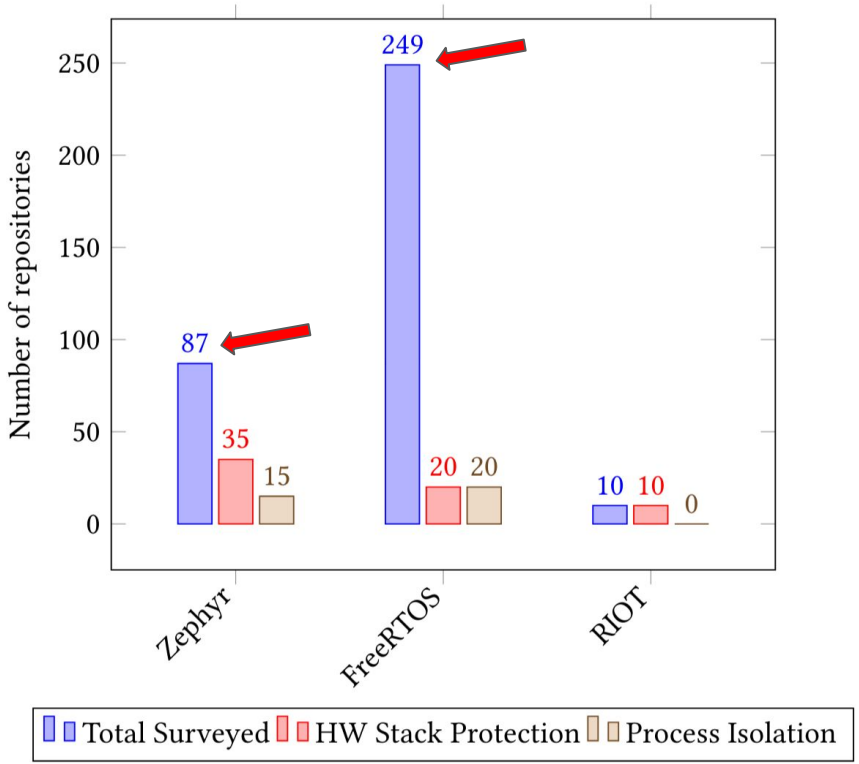
Repository Survey – Results



Repository Survey – Results

Result 1 – Zephyr / FreeRTOS MPU Usage

89% of surveyed FreeRTOS/Zephyr projects do not enable the full suite of opt-in, configurable MPU security features.



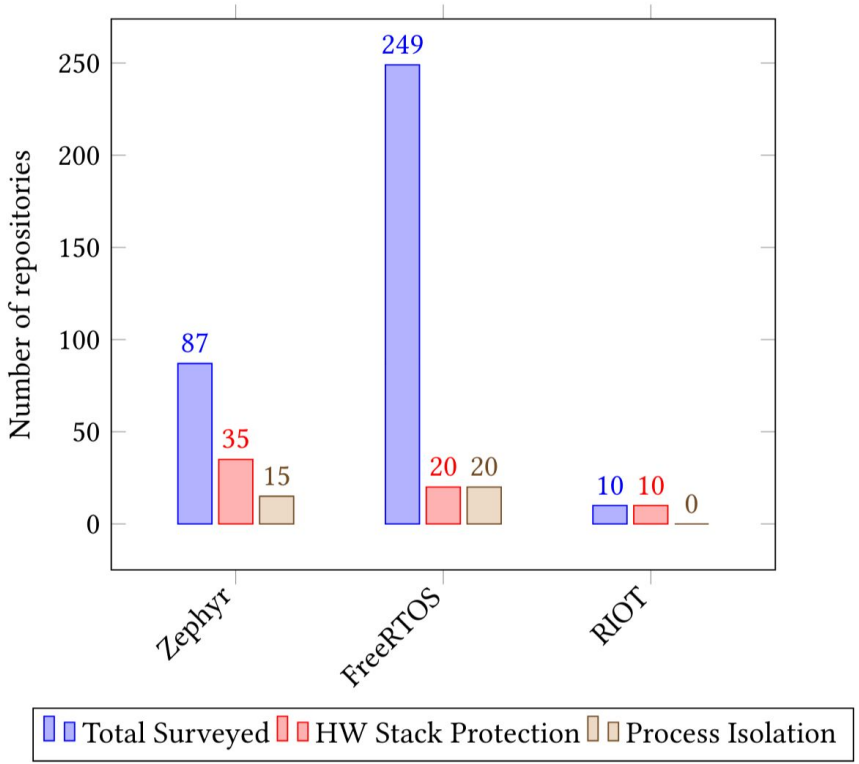
Repository Survey – Results

Result 1 – Zephyr / FreeRTOS MPU Usage

89% of surveyed FreeRTOS/Zephyr projects do not enable the full suite of opt-in, configurable MPU security features.

Result 2 – FreeRTOS SW vs HW StackGuards

FreeRTOS SW StackGuards used 6x more than HW StackGuards in surveyed projects.



*FreeRTOS SW stack guards implemented by compiler (no development overhead).

Repository Survey – Results

Result 1 – Zephyr / FreeRTOS MPU Usage

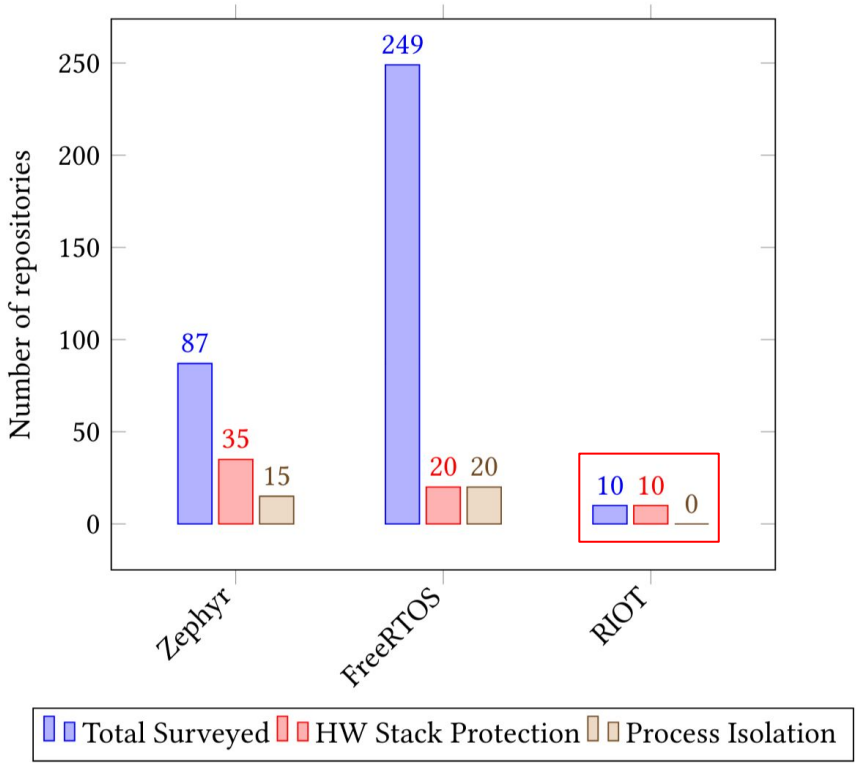
89% of surveyed FreeRTOS/Zephyr projects do not enable the full suite of opt-in, configurable MPU security features.

Result 2 – FreeRTOS SW vs HW StackGuards

FreeRTOS SW StackGuards used 6x more than HW StackGuards in surveyed projects.

Result 3 – RIOT Opt-Out Configuration

RIOT MPU features must explicitly be disabled; no surveyed RIOT project disables MPU security features.



Case Study (I) - Intel Embedded Controller (Zephyr)

- MCU in many x86 systems that is responsible for:
  - Power sequencing
  - Handling keyboard and mouse input
  - Thermal management
- Intel EC reference firmware is Zephyr-OS based
- Firmware organized into Zephyr tasks
- **Does not enable optional Zephyr MPU/isolation features**

Case Study (II) - InfiniTime (FreeRTOS)

- Firmware for open source PineTime smart watch.
- Features include:
  - NimBLE Bluetooth Low Energy library
  - Running multiple applications
  - Apple HealthKit integration
  - Over-the-air updating
- **Does not enable optional FreeRTOS MPU/isolation features.**

Case Study (I & II) - Takeaways

- Neither case studies enable optional MPU/isolation features.
- Both utilize shared global state between tasks.
- Both include safety critical tasks

Case Study (I & II) - Takeaways

- Neither case studies enable optional MPU/isolation features
- Both utilize shared global state between tasks
- Both include safety critical tasks

Cannot “just enable” isolation features – would require significant refactoring.

Majority of surveyed projects do not enable optional, off-by-default OS security features.

Performance overhead?

Hardware security features unavoidably increase a runtime and sometime space overhead (e.g. reconfigure / store MPU state).

*Zephyr documentation: “[memory protection] is optional as it leverages hardware features (such as Memory Protection Units or Memory Management Units), **and incurs in some overhead that smaller systems might not be able to afford.**”*

*FreeRTOS documentation: “If you need **every last drop of performance** out of your processor, then the overhead of using an MPU might be a deal breaker for you”*

Performance overhead?

But many IoT applications do not require **“every last drop”** of performance

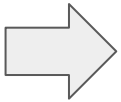
Performance overhead?

But many IoT applications do not require **“every last drop”** of performance

We suspect that the primary overhead impeding the usage of configurable security features is not performance but **developer overhead**.

Developer Overhead

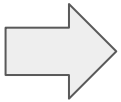
FreeRTOS SW StackGuards used 6x more than HW StackGuards in surveyed projects.



Features with little development overhead are widely used.

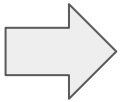
Developer Overhead

FreeRTOS SW StackGuards used 6x more than HW StackGuards in surveyed projects.



Features with little development overhead are widely used.

Case studies utilize shared state across tasks.



Enabling MPU isolation features would require significant refactoring.

Developer Overhead – FreeRTOS documentation

*“Use of an MPU necessarily makes application design **more complex**”*

*“You should also be wary that working with an MPU can be **difficult and, at times, frustrating**. It will take **more time to design your application** as you must consider MPU regions for each of your tasks. Mistakes in these regions, […] **can be confusing to debug.**”*

Modern, networked embedded devices face a massively increased attack surface area!

(local microcontroller control)

Attacker requires physical access to compromise device.

(controllable over the internet)

Internet connectivity means attacker can compromise from anywhere!

Modern embedded devices require increased security.

Modern, networked embedded devices face a massively increased attack surface area!

(local microcontroller control)

Attacker requires physical access to compromise device.

(controllable over the internet)

Internet connectivity means attacker can compromise from anywhere!

Modern embedded devices require increased security.

	Hardware Stack Guard	Kernel/Process Isolation	Process/Process Isolation	MPU Support
FreeRTOS	○	○	○	○
RIOT	●	✕	✕	●
Zephyr	○	○	○	○

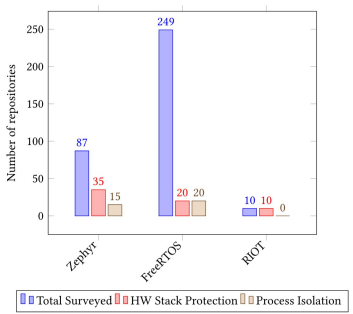
✕ No support ○ Supported, default-off ● Supported, default-on

Table 1: Survey of features available across embedded OSes.

Embedded OSes provide security primitives but they are often disabled by default.

Modern, networked embedded devices face a massively increased attack surface area!

Modern embedded devices require increased security.



Most applications do not enable OS security features.

	Hardware Stack Guard	Kernel/Process Isolation	Process/Process Isolation	MPU Support
FreeRTOS	○	○	○	○
RIOT	●	✕	✕	●
Zephyr	○	○	○	○

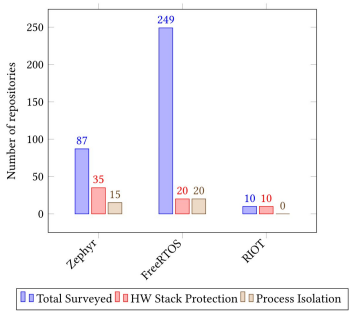
✕ No support ○ Supported, default-off ● Supported, default-on

Table 1: Survey of features available across embedded OSes.

Embedded OSes provide security primitives but they are often disabled by default.

Modern, networked embedded devices face a massively increased attack surface area!

Modern embedded devices require increased security.



Most applications do not enable OS security features.

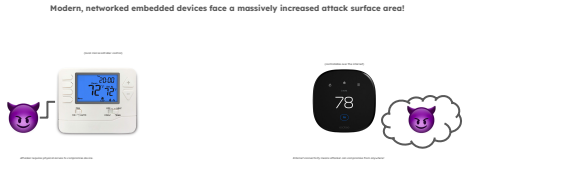
	Hardware Stack Guard	Kernel/Process Isolation	Process/Process Isolation	MPU Support
FreeRTOS	○	○	○	○
RIOT	●	✕	✕	●
Zephyr	○	○	○	○

✕ No support ○ Supported, default-off ● Supported, default-on

Table 1: Survey of features available across embedded OSes.

Embedded OSes provide security primitives but they are often disabled by default.

We suspect developer overhead is the primary reason for security feature under-utilization.



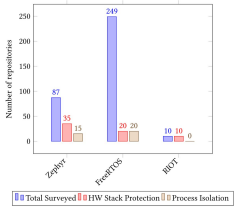
Modern embedded devices require increased security.

	Hardware Stack Guard	Kernel/Process Isolation	Process/Process Isolation	MPU Support
FreeRTOS	○	○	○	○
RIOT	●	×	×	●
Zephyr	○	○	○	○

× No support ○ Supported, default-off ● Supported, default-on

Table 1: Survey of features available across embedded OSes.

Embedded OSes provide security primitives but they are often disabled by default.



Most applications do not enable OS security features.

We suspect developer overhead is the primary reason for security feature under-utilization.

Future Directions

- Expand study
  - Additional embedded OSes.
  - Additional security features.
- Investigate further why security features are opt-in instead of opt-out.
  - Why do embedded OS designers often default to “less-secure”?
- **What would you like to see in such a paper?**