3rd national RISC-V student contest 2022-2023

Sponsored by Thales, the GDR SOC² and the CNFM

Guidelines to report results

Prerequisites

The following steps need to be successful before analyzing the attack results:

- The kit available at https://github.com/thalesgroup/cva6-softcore-contest is running in your Linux environment.
- You can launch all targets of the Makefile as described in the **`README.md`** file.
- The RIPE application executes on the ZYBO Z7-20 board.

In addition, before working on your modifications, you need to ensure that:

- You are using the same tool versions as in the **`README.md`** file.
- You get the same metrics with the reference design as the organization team (see below in **Reference**).

Before reporting results for the contest, you need to ensure that:

- You fulfill all constraints described in the **`Annonce RISC-V contest 2022-2023 v1.pdf`** file

Analyzing the RIPE results

In this year's competition, the goal is to protect the Zephyr OS running on the CV32A6-based processor from executing a corrupted payload.

The corrupted payload is simply printing the string "Executing attack… success" by different means. Ten scenarios are implemented, here are their RIPE parameters:

test nb	technique	inject param	code_ptr	location	function
1	direct	no nop	ret	stack	memcpy
2	direct	no nop	funcptrstackvar	stack	memcpy
3	indirect	no nop	funcptrstackvar	stack	memcpy
4	direct	data	var_leak	heap	sprintf

5	direct	ret to libc	ret	stack	memcpy
6	indirect	ret to libc	funcptrheapvar	heap	memcpy
7	indirect	ret to libc	structfuncptrheap	heap	homebrew
8	indirect	ret to libc	longjumpbufheap	heap	memcpy
9	direct	rop	ret	stack	memcpy
10	direct	rop	structfuncptrheap	heap	memcpy

For each attack you get one point if the print function is not executed during a RIPE execution in the Zephyr environment on the CV32A6 on the ZYBO Z7-20 board. You are not allowed to use dishonest techniques to block the RIPE feedbacks, for instance by blocking the print function.

In addition, the contestants' solution must be able to execute the test program **perf_baseline**. The generation of this project is done through West, the Zephyr's build tool, in the same way as the RIPE project. The generation gives the memory usage in its output:

Memory region	Used Size	Region Size	%age Used
RAM:	61936 B	1 GB	0.01%
IDT_LIST:	0 GB	2 KB	0.00%

As with the RIPE attack, the test program must be run on the CV32A6 on the ZYBO Z7-20 board. After its execution, the test program outputs the number of cycles spent for its execution. This number of cycles will be used to decide between tied teams.

Output log of perf_baseline:

```
Begining of execution with depth 12, call number 50, seed value 63728127.000000
SUCCESS: computed value 868200.000000 - duration: 25.300611 sec 632515274 cycles
```

Reference

The reference project is the one you can find in https://github.com/thalesgroup/cva6-softcore-contest.

Before getting further in the contest, you have to check that it provides the same results on your Linux machine than on the organizers' machine:

- All ten attacks generate the "Executing attack… success" print.
- Used size of the test program perf_baseline is 61936 bytes.
- The test program perf_baseline runs successfully on your board.
- Execution time as reported at the end of the execution of perf_baseline with default parameters is around 25.3 seconds.

If the results with the reference project are not the same as above, you first need to check that you are using the same versions of the tools. If you still have a minor difference after this, get in touch with the organizers.

Reporting the results

In both the 6-page report and in your recorded video, you'll clearly report the results as:

- **Number** and **list** of countered RIPE attacks on the CV32A6 and Zephyr on the ZYBO Z7-20.
- Time and **number of cycles** spent in perf_baseline before and after your modifications.

In the report, you'll also have to report:

- If hardware modification are present, the FPGA resources (LUT, registers, BRAM) and the frequency obtained.

- Used size of the test program perf_baseline.

The jury will double check in their environment the results of the teams who claim the best results in their reports. This also ensures that results are compared based on the same tool versions. So you'll publish a modified GitHub repository, as a fork of https://github.com/thalesgroup/cva6-softcore-contest. You'll create a **`report`** folder where you will upload:

- Your 6-page report written as a scientific paper
- Zephyr build log
- Serial communication log of the execution of the ten tests and the test program.

Remember that your design shall run on the FPGA board and fulfill all constraints listed in **`Annonce RISC-V contest 2022-2023 v1.pdf.`**

During your recorded video, consider showing the execution of your solution on the Zybo board or executions on your machine.

A few remarks

You can use the QEMU emulator to quickly evaluate your software modifications. Be aware that the attack behaviour can be different between emulation and the CV32A6 FPGA. The reported results must be obtained on the FPGA board.

The bitstream image uses only the BRAM, no DDR controller is implemented. You are allowed to use the cva6_fpga_ddr Makefile target if your software requires more than the available BRAM memory. Be aware that this target has not been tested with the Zephyr tools on our side.

We suggest that you keep your GitHub repository private until the end of the contest (May 12th, 2023 at 23:59 CEST). Short after this date, you will make it public. We'll look at the file timestamps to ensure you completed the contest on time.

Your presentation video shall last at most 10 minutes. All team members are encouraged to participate. You can transmit your video to the organizers until May 19th, 2023. The winners' video will be broadcast at the prize ceremony.

If you intend to modify the compiler, please pay close attention to the Security analysis and its impact on contest criteria in the Annex. It will give you the spirit of the rule around software modification. If you have any doubt about your compiler modifications, don't hesitate to contact the organization to clarify the rules.

Finally, do not hesitate to submit your results, even if you think they are limited. The contest difficulty is the same for all teams.

Annex

This annex was prepared as an answer to a team's question about the modification of the compiler.

Security analysis

Threat definition

The attacker is reasonably skilled in RISC-V assembly, and has enough time to **analyze the structure and behavior of the target application** and all its software stack (including OS), typically by acquiring a platform by purchase or theft.

Attack surface

Although Zephyr does not support live package update, neither did we provide separated binaries, a **vulnerable application developed by a third-party** is the more likely attack vector, and the one we want to defend against. The vulnerability typically consists in a buffer susceptible to overflow, but a variety of other vulnerabilities may be present. In the other hand, we do not consider updates on operating system or firmware to be in the scope of the contest.

Attack vector

The attacker has no physical access to the device, and is unable to change its image to an arbitrary one. He/she can however exploit a vulnerable update, or native **vulnerable (yet legitimate) application running on the target**. The exploitation is typically performed using remote connection.

While exploiting the fallback mechanism may indeed be a solution to load said vulnerable application, it falls out of the scope of the contest since we do not consider how the application was introduced in the system.

Support assets targeted by the attacker

The attacker targets the platform's **software integrity**. More precisely, he/she wants to **hijack its control flow** in order to (1) run malicious code and/or (2) extract a secret resident in memory (e.g. encryption key, password…).

Business assets targeted by the attacker

Using the hijacked control flow of the application, the attacker can target a variety of business assets and cause considerable damage and prejudice, whose nature depends on the operational context. Typical examples are (1) immediate or delayed remote control on the platform through the installation of a persistent backdoor, theft of authentication secrets or registration of rogue devices or servers and (2) exfiltration of business secrets (blueprints, patents description, commercial secrets…). **Defense against such high-level exploitation is not in the scope of the contest**.

Impact on contest criteria

Compiler modifications

As above-mentioned, we want to address third-party applications, which we know to be the most common source of vulnerabilities. Making assumptions on their compilation chain configuration is unrealistic, as specialized application may rely on specific compilers or linkers, or even include assembly. Furthermore, the platform supplier must be responsible of the platform security – which means that relying on the third-party developing process is a bad strategy.

In the other hand, since we allowed modification on operating system, it is possible to modify the compiler in order to perform these changes. In the attack scenario, the applications would be compiled separately from the OS, and as such could use a different compiler. In the contest, this is not the case, so changes in the compiler will impact the target application (in our case, HOPE-RIPE).

In order to take account of the attack scenario and risk analysis, we allow modification on the compiler by the contestant, at the condition that these changes do not affect the application (i.e. user mode) behavior (i.e. CFG, memory offsets, etc.). The behavior of the system calls (i.e. kernel mode) can indeed be affected by said changes. We recommend to implement such changes in assembly rather than using a compiler, if possible, as to ensure they do not affect the application.

Version

These guidelines can undergo some evolutions based upon the teams' feedback. Please check their final version before submitting your results.

Version	Date	Comment
V1	2023-01-13	Initial version