Developing a Test Suite for Transient-Execution Attacks on RISC-V and CHERI-RISC-V

Franz A. Fuchs*
franz.fuchs@cl.cam.ac.uk
University of Cambridge
Cambridge, UK

Jonathan Woodruff
jonathan.woodruff@cl.cam.ac.uk
University of Cambridge
Cambridge, UK

Simon W. Moore
simon.moore@cl.cam.ac.uk
University of Cambridge
Cambridge, UK

Peter G. Neumann
peter.neumann@sri.com
SRI International
Menlo Park, CA, USA

Robert N. M. Watson
robert.watson@cl.cam.ac.uk
University of Cambridge
Cambridge, UK

ABSTRACT

In this work, we present a flexible and extensible bare-metal test suite containing replications of all major transient-execution attacks in RISC-V, which are demonstrated on RiscyOO, an open-source out-of-order super-scalar RISC-V processor. While we characterize these attacks both in FPGA hardware and in simulation, its simplicity is particularly suited to verification in simulation during processor development. As an example, we evaluate a version of RiscyOO with CHERI security extensions. CHERI is an Instruction-Set Architecture (ISA) security extension that provides fine-grained memory protection and compartmentalization; it provides an interesting target for transient-execution attacks, which are used to violate memory safety to gain privileges and leak secrets. Our results give clear evidence that sophisticated RISC-V implementations can be vulnerable to the same transient-execution attacks as mainstream architectures, but give hope that open-source implementations as well as open-source verification tools can help discover and eliminate vulnerabilities during development. We further show that the CHERI-RISC-V instruction set does not imply defenses against transient execution attacks, but that a careful implementation and aggressive pointer bounding can mitigate some forms of attack. Our extensible test suite lays a common foundation for future research on transient-execution attacks and their mitigations on RISC-V systems, which should pave the way to successful and secure high-performance, open-source RISC-V processors.

KEYWORDS

security, transient-execution attacks, spectre, meltdown, CHERI

*Parts of the research were conducted when affiliated with KTH Royal Institute of Technology, Stockholm, Sweden.

CARRV 2021, June 17, 2021, Virtual Workshop

ACM Reference Format:
Franz A. Fuchs, Jonathan Woodruff, Simon W. Moore, Peter G. Neumann, and Robert N. M. Watson. 2021. Developing a Test Suite for Transient-Execution Attacks on RISC-V and CHERI-RISC-V. In *Fifth Workshop on Computer Architecture Research with RISC-V, June 17, 2021, Virtual Workshop.* ACM, New York, NY, USA, 7 pages.

1 INTRODUCTION

Due to its modular nature, RISC-V describes not only one Instruction-Set Architecture (ISA), but an entire set of ISAs [17, 18]. Processor designers must implement the base instruction set, but can choose which extensions they include or whether they do not implement any further instructions. Furthermore, implementors can choose to implement their own instructions customized for their applications' needs. This property makes RISC-V a highly flexible ISA that has received more and more attention in academia [3, 16].

The detection of transient-execution attacks caused an earthquake in the computer architecture and hardware security community. Following the initial attacks [10, 13], new transient-execution attacks have been detected on a regular basis [4]. While some have asserted that RISC-V was safe from transient-execution attacks due to the current prevelance of in-order implementations, more sophisticated RISC-V processors are already capable of transient-execution side-channel attacks [15]. Gonzalez et al. [6] proved the feasibility of Spectre-PHT and Spectre-BTB on the open-source BOOM processor and provided a preliminary mitigation mechanism protecting the L1 cache. Furthermore, Le et al. [12] reproduced Spectre-PHT on BOOM and retrieved the same results. In order to advance the research on RISC-V in this field, we developed a test suite comprising all major transient-execution attacks. To our knowledge, this test suite is the first to reproduce the full set of major transient-execution attacks on a RISC-V processor.

RISC-V is becoming a hotbed of *architectural* innovation in the instruction set, and in recent years has also been extended for hardware security research, e.g., the MI6 architecture [3]. The Spectre and Meltdown family of vulnerabilities are *microarchitectural* side effects of deeply speculative execution engines. Nevertheless the architectural vocabulary of the instruction set can sometimes facilitate a microarchitecture that is resistant to transient-execution side-channel attacks. We will examine to what degree the CHERI-RISC-V ISA extension can enable safety in speculative, out-of-order execution.

In this paper, we are making the following contributions:

- Reproducing all major transient-execution attacks on RISC-V
- Building a suite to test in simulation and on a FPGA while designing RTL[1]
- Demonstrating the feasibility of transient-execution attacks on a CHERI-RISC-V microprocessor
- Evaluating the threat the various attacks mean to CHERI systems in general

The paper is structured as follows: Section 2 introduces RISC-V and the transient-execution attacks reproduced by our test suite, as well as describing the abstract CHERI model and the concrete implementation we are using for our experiments. Section 3 presents CHERI's security model and the threat model. Section 4 introduces our test suite, which evaluates the feasibility of all major transient-execution attacks on a processor implementing both RISC-V and CHERI-RISC-V. Section 5 discusses our results and their implications for future CHERI implementations. Section 6 provides our conclusions.

2 BACKGROUND

2.1 RISC-V

RISC-V is an open-source, royalty-free ISA, which follows the Reduced Instruction Set Computing (RISC) principle, and RISC-V is a typical *load-store architecture* [2]. RISC-V offers three privilege levels: machine level, supervisor level, and user level [18]. The main advantage of RISC-V is its flexibility. Not only does it specify many optional extensions, but also the concrete implementation can decide whether the register width shall be 32 bits, 64 bits, or 128 bits. Furthermore, machine mode is the only privilege level that is prescribed for an implementation. That gives hardware designers a large degree of freedom to customize the implementation to the application's needs. RISC-V's application area ranges from small embedded systems to large-scale, high-performance servers.

2.2 Transient-Execution Attacks

In 2018, the detection of Spectre [10] and Meltdown [13] marked the beginning of the era of transient-execution attacks. This class of attacks relies on speculative execution, which all modern processors use to improve performance. The two main mechanisms of speculative execution are out-of-order execution and branch prediction. In both cases, a microprocessor will execute instructions where it cannot be sure whether they will actually be needed. In case the processor speculates correctly, this improves performance. Otherwise, the processor needs to roll back the speculatively executed instructions and start executing the correct instruction stream. Instructions that were erroneously executed are referred to as transiently executed instructions. Their result is microarchitecturally available as long as the misspeculation is not yet detected by the processor. The goal of transient-execution attacks is to trick the processor into speculatively executing an instruction stream that accesses a secret, typically in a different protection domain. Due to the execution eventually being rolled back, the secret has to be recovered via a side channel (e.g., a cache timing side channel), which is our focus here.

[1]https://github.com/CTSRD-CHERI/Test-Suite-Transient-Execution

Caches are heavily involved in transient-execution attacks for two reasons. First, transient-execution attacks rely on speculatively executed instructions and the condition that misspeculation remains undetected by the processor for as many cycles as possible. Increasing the load latency by evicting cache lines is essential to the success of many attacks demonstrated in literature, as well as to our example reproductions. Second, caches are used to encode microarchitectural state. Depending on whether a cache line of a virtual address is present or not, the load times will differ substantially. Most transient-execution attacks demonstrated in academia use the *FLUSH+RELOAD* [22] technique to encode a secret value. This involves flushing an entire cache before the attack takes place. During the attack, a load occurs to a virtual address depending on a secret. After the attack, the attacker probes the access times of all cache lines. The cache line encoding the secret value will lead to a cache hit and therefore to shorter access times. By comparing the access times of all cache lines, an attacker can leak a secret. Figure 1 shows the results of probing 16 L1 data cache lines in an example attack. The cache line at index 10 is accessed significantly faster, 23 cycles rather than more than 66 cycles, allowing the attacker to infer that this cache line has been accessed by the victim.

Neither RISC-V ISA nor the baseline RiscyOO implementation offer a dedicated `flush` instruction [17–19]. Therefore, attackers need to implement flush functions themselves. We implemented a software flush function as a loop of continuous memory accesses that will bring new lines into the caches, and eventually evict the target lines. In order to make our flush function performant, we used the parameters in Table 1. In our experiments, the cache state was predictable, due to the fact that our attack was running in a baremetal framework and was therefore the only code running on RiscyOO at any time. For real-world attacks, other code might operate on the processor and impact the cache state – or the cache state before the attack runs is unknown. Real-world attacks will therefore require a probabilistic approach but would nevertheless be successful.

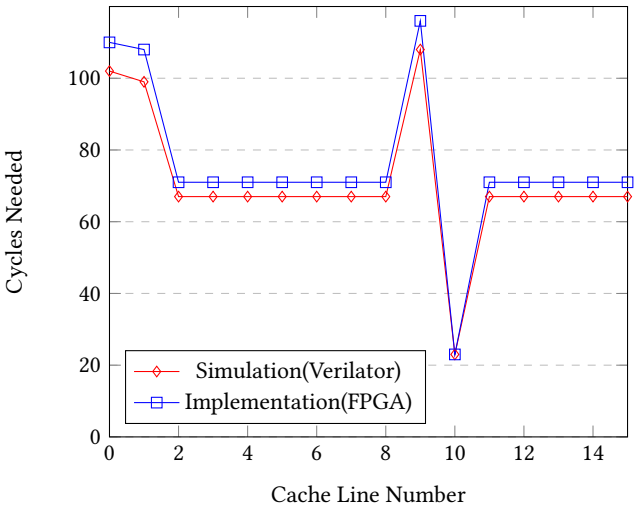

Figure 1: Results of probing the L1 cache after an attack has been conducted.

Transient-execution attacks are divided into Spectre-style and Meltdown-style attacks [4]; Spectre-style attacks follow control flow or data flow misprediction, whereas Meltdown-style attacks follow a faulting instruction. Following Arm's whitepaper [1], we identified six major transient-execution attacks applicable on RISC architectures: Spectre-PHT, Spectre-BTB, Spectre-RSB, Spectre-STL, Meltdown-US, and Meltdown-GP.

Spectre-PHT [10] targets direct branches and seeks to achieve the misprediction of the decision whether a particular branch will be microarchitecturally predicted to be *taken* or *not-taken*. An exemplary attack is depicted in Listing 1. In this case, the attacker trains the direct branch resulting from the `if` statement such that the processor always speculatively executes the body of the `if` statement. This way, an attacker can provide any value for `i` and speculatively access `array0` out-of-bounds in order to obtain a secret value. This secret value is encoded by performing a memory access to `array1`.

The goal of Spectre-BTB [10] attacks is to mispredict the target of a branch. Past branch targets are stored in the Branch Target Buffer (BTB) in order to enhance performance when the branch is executed again. However, many microarchitectures allow BTB entries to alias and therefore an attacker can speculatively steer victim code to gadgets.

Most processors have a dedicated microarchitectural structure to predict return addresses – the Return Stack Buffer (RSB). Similarly to Spectre-BTB, Spectre-RSB [11, 14] seeks to inject a malicious return address into the RSB and thus steer the predicted control-flow to attacker-chosen gadgets.

Spectre-STL (Store-To-Load) [7] is an attack targeting memory disambiguation. In order to improve performance, modern microprocessors want to execute loads as early as possible to hide possible latencies. To maintain correctness, loads must not be executed before dependent stores. However, microprocessors speculate on the store-to-load dependencies, to enhance performance. Spectre-STL exploits store-to-load misspeculation and achieves reading of stale values from the memory subsystem.

The goal of Meltdown-US (User-Supervisor) [13] attacks is to speculatively read from a supervisor-only page with user privileges. In order to enhance performance, some microprocessors speculatively access a page before checking the permissions. This means that there exists a window in which an attacker can access a page without holding the necessary privileges. Meltdown-GP (General Protection) [1] seeks to read a privileged register from user space. Similarly, some processors allow access to these register speculatively before checking the permissions, which would enable an attack.

2.3 CHERI Protection Model

Capability Hardware Enhanced RISC Instructions (CHERI) is a security extension of conventional ISAs that adds *capabilities* – unforgeable and bounded pointers. Figure 2 shows a 128-bit wide capability that is used for 64-bit architectures. A capability contains the address of the pointer and metadata including permissions and bounds information. Furthermore, a capability has a hidden validity tag (the 129th bit), which is atomically stored with the capability in both registers and memory. A capability authorizes access to a region of memory and no memory access is possible without a valid capability. Furthermore, all capability operations are monotonic and therefore cannot increase the privileges a capability grants. Capabilities do not protect just data, but also code.

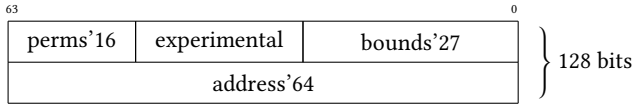

Figure 2: Bit representation of a 128-bit capability.

CHERI-RISC-V is the mapping of the abstract CHERI model to RISC-V . As RISC-V, CHERI-RISC-V is an ISA design space, which means that it leaves freedom for future extensions and concrete implementations. CHERI RiscyOO, which is based on MIT's *RiscyOO* [23], is a 64-bit out-of-order superscalar processor implementing CHERI-RISC-V with 128-bit-wide capabilities. The extension from RiscyOO to CHERI RiscyOO is depicted in Figure 3. The RiscyOO basis is highly configurable. For our experiments, we chose $n = 2$, which leads to two ALU pipelines and one FPU pipeline. Independently of which value we chose for n, RiscyOO is always configured with one memory pipeline. This makes CHERI RiscyOO a 2-superscalar processor. A subset of the remaining parameters important to our attack set-up is listed in Table 1. We used the same parameters in simulation and in synthesis.

Parameter	Value
L1 I/D Cache Size	32 KiB
L2 Cache Size	1 MiB
Out-of-order Window	64
Memory Queues Size	38

Table 1: Overview of RiscyOO's configuration for our experiments.

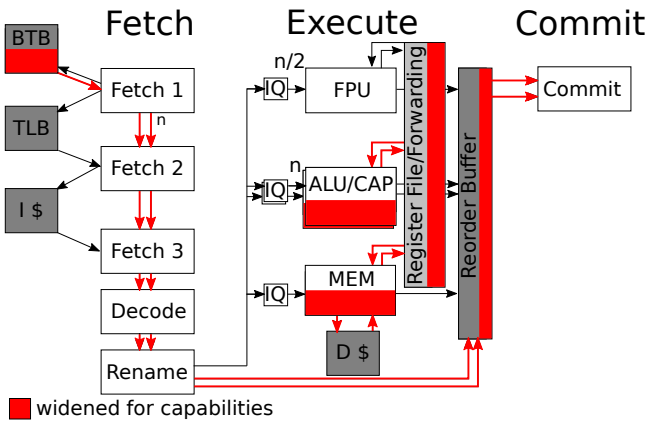

Figure 3: The CHERI RiscyOO pipeline.

3 SECURITY AND THREAT MODEL

The ultimate target of an attacker is to leak a secret. A secret is a value that the attacker has not sufficient permissions to access to, e.g., a value stored on a supervisor-only page, but the attacker has only user privileges. The goal of every test in our test suite is to get access to resources without having the necessary permissions and subsequently obtain a secret. CHERI systems are expected to respect the security model of the baseline ISA, which for our described experiments is RISC-V. Attackers should still be constrained to the RISC-V privilege level and the address space in which they are operating. Attackers should not be able to execute code in a more-privileged level, nor should they have access to another address space unless explicitly granted. The threat model used in this work allows an attacker to execute arbitrary instructions, but still any privilege escalation has to be prevented. Here, we follow previous work that argues that cache side channels themselves do not need to be prevented by hardware, but that software has to take appropriate measures [8]. Therefore, attackers are able to measure timing differences on CHERI-RISC-V systems and thus are able to conduct *FLUSH+RELOAD*. This follows previous work from Arm [1]. However, in our threat model it should not be possible for attackers to gain privileges nor to leak any secret – even if only through transient execution.

4 RESULTS

The essence of our test suite involves the transient-execution attacks in RISC-V assembly that can be run on the bare-metal processor without needing operating-system support. We extended our test suite by applying the general RISC-V attacks to use CHERI protection. The test suite was developed for two reasons. First, it should prove the feasibility of a set of microarchitectural attacks on a RISC-V processor and in its extension on a CHERI-RISC-V microarchitecture. Second, several attacks prove the absence of properties that might be important to certain implementations, e.g., CHERI systems must not allow access beyond existing capability bounds and permissions – even if only in transient execution. Our test suite allows running while developing hardware, and therefore can reveal undesired microarchitectural properties in early stages.

This section presents the results of reproducing the major transient-execution attacks on CHERI RiscyOO, which is both a RISC-V and a CHERI-RISC-V microarchitecture. An attack is successful if it can leak a secret value. For all attacks conducted, we used the *FLUSH+RELOAD* [22] cache timing side channel explained above and tested all attacks by simulating our design using *verilator*. Furthermore, we synthesized RiscyOO on a VCU-118 FPGA and conducted our experiments on that FPGA as well. Figure 1 depicts the cache-probing measurements of an exemplary attack (Spectre-RSB for CHERI-RISC-V). The results for all attacks that we could successfully conduct appear similar, and therefore the cache timing results are not presented for any of the attacks. In Figure 1, both the verilator and VCU-118 measurements show a significantly slower access time for one of the cache lines and thus the attacker can leak a secret. Furthermore, we can clearly observe that the simulation behaves similarly in memory delays compared to the VCU-118 delays, which makes our simulation results credible for future attack results and related experiments.

	RISC-V	CHERI-RISC-V
Spectre-PHT	*S*	*U*
Spectre-BTB	*S*	*S*
Spectre-RSB	*S*	*S*
Spectre-STL	*S*	*S*
Meltdown-US	*U*	n/a
Meltdown-US-CHERI	n/a	*U*
Meltdown-GP	*U*	n/a
Meltdown-GP-CHERI	n/a	*U*

Table 2: Overview of attempted transient-execution attacks on RISC-V and CHERI RiscyOO and whether they were successful(*S*), unsuccessful(*U*), or not applicable(n/a).

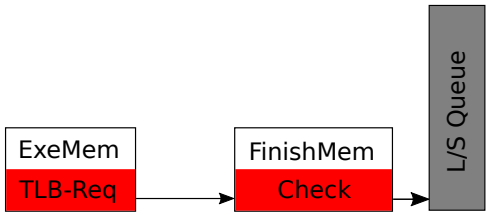

Figure 4: Last two stages of the RiscyOO memory execution pipeline.

A summary of the results of our experiments for both RISC-V and CHERI-RISC-V attacks on CHERI RiscyOO is depicted in Table 2. Compared to the transient-execution attacks described in Section 4, we added two Meltdown-style attacks that are CHERI adaptions of the respective two conventional Meltdown attacks. Like Meltdown-US, attackers attempt to access memory through Meltdown-US-CHERI which they were not granted privileges for, which means in CHERI that attackers do not have a valid capability to the sought region of memory. CHERI-RISC-V introduces an Access System Registers (ASR) bit, which specifies whether code is allowed to access Special Capability Registers (SCRs). These SCRs are capability-wide registers that are necessary for the correct operation of pure-capability code in CHERI-RISC-V. A Meltdown-GP-CHERI attack seeks to speculatively access an SCR and leak this register's value without having the necessary ASR privileges.

As depicted in Table 2, we could not successfully any of the four Meltdown-style attacks due to the fact that they are rendered impossible by (CHERI) RiscyOO's implementation. For data page accesses, the Page Table Entry (PTE) is received from the Translation Lookaside Buffer (TLB) in the final stage in the memory execution pipeline. This stage – depicted in Figure 4 – checks whether the page is supervisor-only or whether it can also be accessed from user space. In case the privileges are not sufficient, this pipeline stage will raise a page fault exception. This exception will be handled immediately and the load attempting to fetch a secret will never leave the core. Therefore, Meltdown-US is microarchitecturally prevented. For a similar reason, CHERI RiscyOO prevents Meltdown-US-CHERI attacks. In the penultimate stage of the memory execution pipeline, CHERI RiscyOO performs capability checks.

In case the capability pointer is out-of-bounds or the capability is invalid, an exception will be raised and the load will not leave the core, which prevents the attack. For both Meltdown-GP and Meltdown-GP-CHERI, the permissions for accessing system registers or SCRs are checked in the *Rename* stage, which is part of the in-order front-end of (CHERI) RiscyOO. In case the code does not have the necessary privilege level to access the sought register, the Rename stage code will detect this and mark the instruction as *executed* in the corresponding Reorder Buffer (ROB) entry. This precludes the instruction from ever being scheduled to an execution pipeline, which prevents a result from being executed. Therefore, neither Meltdown-GP nor Meltdown-GP-CHERI are possible on (CHERI) RiscyOO.

We successfully reproduced all four Spectre-style attacks on (CHERI) RiscyOO in both RISC-V and CHERI-RISC-V assembly with the exception of Spectre-PHT on CHERI-RISC-V. Even though the respective RISC-V and CHERI-RISC-V variants are similar and have the same goal, they have a different approach to reach their goal. A conventional RISC-V Spectre attack has to circumvent address space and privilege level boundaries whereas a CHERI-RISC-V Spectre attack focuses on gaining access to powerful capabilities. In CHERI, address spaces and privilege levels are existent, but they do not play a big role because capabilities are the central security mechanism. In our test suite, we provide both RISC-V and CHERI-RISC-V variants of all four Spectre attacks. For an attack, the respective variants share the similar underlying mechanisms. The CHERI-RISC-V variants can be seen as extensions to the RISC-V attacks because they also need to circumvent the capability security mechanism. For brevity reasons, we present only the CHERI-RISC-V variants of our test suite here.

The Spectre-PHT attack on CHERI-RISC-V poses a special case, because its success depends on the concrete capability configuration. Listing 1 shows the key lines of a Spectre-PHT attack where the variable `i` is under the attacker's control. Both `array0` and `array1` are represented as capabilities in CHERI. The goal of the attacker is to speculatively execute the code covered by the `if` statement for values with $i \geq size$. To reach the branch-direction misprediction, the Pattern History Table (PHT) has to be mistrained with previous calls to the victim code, such that it strongly predicts the control flow to the body of the `if` statement. CHERI systems can successfully mitigate this attack if the bounds of the capability guarding `array0`'s memory are configured so that it allows access only to `size` many elements and if the concrete implementation prevents out of capability bounds access in any case. Our CHERI compiler automatically establishes bounds information from memory allocations and thus preserves the programmer's intent. As explained for Meltdown-US-CHERI, it is not possible to transiently access memory out of capability bounds in CHERI RiscyOO. However, if `array0`'s capability grants access to more than `size` elements, a successful Spectre-PHT attack on CHERI RiscyOO is possible.

Listing 1: Spectre-PHT attack in C

```
if(i < size){
  int k = array0[i];
  int l = array1[k];
}
```

Furthermore, CHERI RiscyOO provides protection against flavors of Spectre-PHT attacks. In [9], the attacker's goal is to write to an arbitrary memory location instead of reading from it. A possible target would be overwriting a return address and therefore manipulating control-flow in speculation. However, CHERI clearly differentiates between whether a capability or data is being written to memory. A data write can never lead to valid capability being formed, because the tag bit will be stripped when writing to memory. Therefore, CHERI systems can mitigate this type of Spectre-PHT attack.

We reproduced multiple variants of Spectre-BTB in order to illuminate its threat to RISC-V and CHERI-RISC-V systems. All conducted experiments have in common that they want to speculatively jump from a privileged domain to an attacker-controlled gadget residing in a less privileged domain. This way, attackers can execute arbitrary code with privileges they previously did not have. In our attacks, we managed to execute attacker gadgets in S privilege mode. Furthermore, we were able to leak kernel capabilities to attacker gadgets. Spectre-BTB attacks are possible because of how RiscyOO's BTB is designed. It has 1024 entries, which store the address of the jump target. In CHERI RiscyOO, the BTB stores not just the target address, but the entire target capability; therefore, an entry comprises all privileges that come with that capability including the one bit validity tag. Due to collisions in the BTB caused by a hash function, it is possible for an attacker to substitute a BTB entry with an entry holding a valid capability to the attacker's domain. The first step is to insert an entry to the attacker gadget, which is achieved by executing a branch to that target. The next time the victim jump with the aliased BTB entry is executed, the control-flow will speculatively go to the attacker's gadget. For similar reasons, RiscyOO's RSB allows Spectre-RSB attacks. The RSB stores the entire target capability enabling speculation to powerful capabilities in CHERI RiscyOO. The attacker's goal is to achieve a mismatch between the software stack containing the correct return address and the RSB.

We could reproduce Spectre-STL both in RISC-V and in CHERI-RISC-V assembly on (CHERI) RiscyOO. Both attacks do not differ notably; however, the CHERI-RISC-V variant uses capabilities for memory accesses. Spectre-STL attacks rely on loads being speculatively executed before dependent stores, and therefore leaking a secret. CHERI does not put any constraints on store-to-load dependencies, nor on memory versioning. In RiscyOO, a load will be executed out-of-order as soon as all of its arguments are ready. If the store on which the load depends is delayed, e.g., it needs its address from memory, RiscyOO will speculatively execute a load before a dependent store, which enables Spectre-STL attacks in both RISC-V and CHERI-RISC-V on (CHERI) RiscyOO.

5 DISCUSSION

The test suite presented above is highly flexible and enables extensions not only for previously undetected attacks, but also for other ISAs that might offer different security properties, or might emphasize another concept. One large advantage of our test suite is that all tests run without an underlying operating system and enable us to detect properties of the microarchitecture rather than

effects by the operating system. An operating system often schedules multiple processes and introduces noise into the system, e.g., last-level cache lines are evicted by an unrelated process. Potentially, this shadows the possibility of an attack even though it is microarchitecturally possible. Our test suite is tailored for being used while designing RTL and allows developers to verify whether certain attacks are possible on the microarchitecture early on in the development process. Furthermore, the tests written in RISC-V and CHERI-RISC-V assembly facilitate comparison with compiler generated code. With our test suite, we lay the foundation to find out whether compilers generate code that is secure in transient execution. This is crucial in the testing phases of a target architecture of a compiler. The CHERI-RISC-V backend of the LLVM framework [5] currently in development can benefit from our test suite since we developed a clear understanding of what transient-execution attack patterns look like in CHERI-RISC-V, which will simplify detecting and mitigating them.

Our results in Section 4 clearly show that CHERI-RISC-V implementations can mitigate Spectre-PHT attacks. However, this is feasible only if the capability does not enable access to the secret. Otherwise, an attack is possible on CHERI-RISC-V systems as they do not constrain transient execution in another way. A secure program requires an as tight capability configuration as possible. This is not trivially possible for large programs, though. Furthermore, CHERI-128 capabilities use compressed bounds [21] in order to fit in the 128 bit format. This can lead to imprecise bounds and therefore the capability can enable access to more memory than needed by the data, which in turn opens the door to Spectre-PHT attacks. It is important to note that Spectre-PHT does not violate CHERI's security model as long as the attacker is able to access only memory that is granted by the authorizing capability. However, Spectre-PHT breaks the software's guarantee that memory can be guarded by an `if` statement. This is not a problem only for RISC-V microarchitectures, as many processors with different ISAs are vulnerable to Spectre-PHT attacks [4].

As described in the previous section, Spectre-BTB allows jumps to arbitrary targets, which are not limited to the current RISC-V privilege level. Furthermore, branch targets are not limited to the current compartment. This enables an attacker to jump from a powerful compartment, e.g., running in S privilege mode, to an attacker-chosen gadget in order to use S mode privileges or have access to a powerful capability. S mode privileges include the permission to access supervisor-only instructions and enables access to privileged registers, e.g., `satp` which is the register holding the root address of the page table tree. Spectre-BTB attacks clearly violate both the CHERI and the RISC-V security model as explained above. Due to the similarities in the attack styles, Spectre-RSB violates both security models in the same way. Future work has to find mitigation mechanisms for Spectre-BTB and Spectre-RSB and explain how these are secure, but still keeps the entire system performant. Similar to Spectre-PHT, Spectre-STL does not violate CHERI's security model. All memory accesses – even when they are executed transiently – are allowed by the respective capabilities. CHERI's security model does not include versions of memory, which is exploited by Spectre-STL attacks.

In order to enforce CHERI's security model, CHERI implementations need to ensure that an attacker never has access to more capabilities than architecturally present. As described above, Spectre-BTB and Spectre-RSB allow access to powerful capabilities. However, branch prediction is important to the performance of microarchitectures in general and this holds as well for CHERI implementations. Future hardware implementations must find ways to incorporate mitigation mechanisms in order to prevent unauthorized memory accesses and privilege escalation without degenerating the system to be non-performant. One effective mitigation mechanism not implemented yet is to use tags in the BTB and RSB, e.g., Compartment Identifiers (CIDs) [20]. CHERI was not designed to prevent transient-execution side-channel attacks, but it it is able to partially mitigate this attack class. Furthermore, CHERI's hardware-enforced capabilities enable precise reasoning about the privileges an attacker has.

6 CONCLUSIONS

We have presented an open-source test suite for the RISC-V architecture that reproduces all major transient-execution attacks. We have exercised this suite to characterize the RiscyOO out-of-order RISC-V implementation, finding that it was vulnerable to a majority of these attacks in both simulation and on FPGA. We demonstrated the flexibility of this suite by extending it to target CHERI-RISC-V, a security-focused ISA extension, and characterized the CHERI RiscyOO implementation. While the CHERI RiscyOO microarchitecture was not natively safer than its RiscyOO ancestor, default behaviour of the CHERI compiler was able to mitigate certain classes of attack by attaching bounds to pointers which were found to be enforced in transient execution. Nevertheless, most of our experiments were successful, demonstrating leakage of unintended values. Especially, we find Spectre-BTB and Spectre-RSB attacks to be a large threat to CHERI systems because they allow attackers to have access to powerful capabilities and execute arbitrary code gadgets in speculation. Still, even without consciously targeting transient-execution attacks, CHERI RiscyOO can mitigate one of the four Spectre attacks and offers a promising approach to support software-based mitigations. We expect this test suite to grow into a valuable tool for the development of new mitigation mechanisms across the RISC-V community for the development of new microarchitectures and architectural extensions.

ACKNOWLEDGMENTS

Parts of this work have been conducted during a Master's dissertation collaboratively supervised by members of the Department of Computer Science and Technology, University of Cambridge and members of the Department of Computer Science, KTH Royal Institute of Technology. This project was supported by the NCSC under the UK RISE Initiative. This work was supported in part by the Defense Advanced Research Projects Agency (DARPA) under contract HR0011-18-C-0016 ("ECATS"). The views, opinions, and/or findings contained in this report are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. We also acknowledge the EPSRC REMS Programme Grant (EP/K008528/1), Arm Limited, and Google, Inc.

REFERENCES

[1] Arm Limited. 2020. *Cache Speculation Side-channels.* Technical Report 2.5. 21 pages. https://developer.arm.com/support/arm-security-updates/speculative-processor-vulnerability

[2] Krste Asanović and David A. Patterson. 2014. *Instruction Sets Should Be Free: The Case For RISC-V.* Technical Report UCB/EECS-2014-146. University of California at Berkeley, Electrical Engineering and Computer Sciences.

[3] Thomas Bourgeat, Ilia Lebedev, Andrew Wright, Sizhuo Zhang, Arvind, and Srinivas Devadas. 2019. MI6: Secure Enclaves in a Speculative Out-of-Order Processor. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO '52)*. Association for Computing Machinery, New York, NY, USA, 42–56. https://doi.org/10.1145/3352460.3358310

[4] Claudio Canella, Jo Van Bulck, Michael Schwarz, Moritz Lipp, Benjamin Von Berg, Philipp Ortner, Frank Piessens, Dmitry Evtyushkin, and Daniel Gruss. 2019. A Systematic Evaluation of Transient Execution Attacks and Defenses. In *Proceedings of the 28th USENIX Conference on Security Symposium (SEC'19)*. USENIX Association, Santa Clara, CA, USA, 249–266.

[5] Brooks Davis, Robert N. M. Watson, Alexander Richardson, Peter G. Neumann, Simon W. Moore, John Baldwin, David Chisnall, Jessica Clarke, Nathaniel Wesley Filardo, Khilan Gudka, Alexandre Joannou, Ben Laurie, A. Theodore Markettos, J. Edward Maste, Alfredo Mazzinghi, Edward Tomasz Napierala, Robert M. Norton, Michael Roe, Peter Sewell, Stacey Son, and Jonathan Woodruff. 2019. *CheriABI: Enforcing valid pointer provenance and minimizing pointer privilege in the POSIX C run-time environment.* Technical Report UCAM-CL-TR-932. University of Cambridge, Computer Laboratory. https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-932.pdf

[6] Abraham Gonzalez, Ben Korpan, Jerry Zhao, Ed Younis, and Krste Asanović. 2019. Replicating and Mitigating Spectre Attacks on a Open Source RISC-V Microarchitecture. In *Third Workshop on Computer Architecture Research with RISC-V*.

[7] Jann Horn. 2018. speculative execution, variant 4: speculative store bypass. https://bugs.chromium.org/p/project-zero/issues/detail?id=1528. (Feb. 2018).

[8] David Kaplan, Jeremy Powell, and Tom Woller. 2020. *AMD SEV-SNP: Strengthening VM Isolationwith Integrity Protection and More.* Technical Report. Advanced Micro Devices Inc. https://www.amd.com/system/files/TechDocs/SEV-SNP-strengthening-vm-isolation-with-integrity-protection-and-more.pdf

[9] Vladimir Kiriansky and Carl Waldspurger. 2018. Speculative Buffer Overflows: Attacks and Defenses. (2018). arXiv:cs.CR/1807.03757

[10] Paul Kocher, Jann Horn, Anders Fogh, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. 2019. Spectre Attacks: Exploiting Speculative Execution. In *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*. IEEE, 1–19. https://doi.org/10.1109/SP.2019.00002

[11] Esmaeil Mohammadian Koruyeh, Khaled N. Khasawneh, Chengyu Song, and Nael B. Abu-Ghazaleh. 2018. Spectre Returns! Speculation Attacks using the Return Stack Buffer. In *12th USENIX Workshop on Offensive Technologies, WOOT 2018, Baltimore, MD, USA, August 13-14, 2018*, Christian Rossow and Yves Younan (Eds.). USENIX Association. https://www.usenix.org/conference/woot18/presentation/koruyeh

[12] Anh-Tien Le, Ba-Anh Dao, Kuniyasu Suzaki, and Cong-Kha Pham. 2020. Experiment on Replication of Side Channel Attack via Cache of RISC-V Berkeley Out-of-Order Machine (BOOM) Implemented on FPGA. In *Fourth Workshop on Computer Architecture Research with RISC-V (CARRV 2020)*.

[13] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Anders Fogh, Jann Horn, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. 2018. Meltdown: Reading Kernel Memory from User Space. In *27th USENIX Security Symposium (USENIX Security 18)*.

[14] Giorgi Maisuradze and Christian Rossow. 2018. Ret2spec: Speculative Execution Using Return Stack Buffers. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18)*. Association for Computing Machinery, New York, NY, USA, 2109–2122. https://doi.org/10.1145/3243734.3243761

[15] RISC-V Foundation. 2018. Tom's Hardware Article: RISC-V Foundation Trumpets Open-Source ISAs In Wake Of Meltdown, Spectre. https://riscv.org/news/2018/01/toms-hardware-article-risc-v-foundation-trumpets-open-source-isas-wake-meltdown-spectre/. (January 2018).

[16] Marno van der Maas and Simon W. Moore. 2020. Protecting Enclaves from Intra-Core Side-Channel Attacks through Physical Isolation. In *Proceedings of the 2nd Workshop on Cyber-Security Arms Race (CYSARM'20)*. Association for Computing Machinery, New York, NY, USA, 1–12.

[17] Editors Andrew Waterman and Krste Asanović. 2019. *The RISC-V Instruction Set Manual.* RISC-V Foundation. Volume I: User-Level ISA.

[18] Editors Andrew Waterman and Krste Asanović. 2019. *The RISC-V Instruction Set Manual.* RISC-V Foundation. Volume II: Privileged Architecture.

[19] Robert N. M. Watson, Peter G. Neumann, Jonathan Woodruff, Michael Roe, Hesham Almatary, Jonathan Anderson, John Baldwin, Graeme Barnes, David Chisnall, Jessica Clarke, Brooks Davis, Lee Eisen, Nathaniel Wesley Filardo, Richard Grisenthwaite, Alexandre Joannou, Ben Laurie, A. Theodore Markettos, Simon W. Moore, Steven J. Murdoch, Kyndylan Nienhuis, Robert Norton, Alexander Richardson, Peter Rugg, Peter Sewell, Stacey Son, and Hongyan Xia. 2020. *Capability Hardware Enhanced RISC Instructions: CHERI Instruction-Set Architecture (Version 8).* Technical Report UCAM-CL-TR-951. University of Cambridge, Computer Laboratory. https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-951.pdf

[20] Robert N. M. Watson, Jonathan Woodruff, Michael Roe, Simon W. Moore, and Peter G. Neumann. 2018. *Capability Hardware Enhanced RISC Instructions (CHERI): Notes on the Meltdown and Spectre Attacks.* Technical Report UCAM-CL-TR-916. University of Cambridge, Computer Laboratory. https://doi.org/10.48456/tr-916

[21] Jonathan Woodruff, Alexandre Joannou, Hongyan Xia, Anthony Fox, Robert M Norton, David Chisnall, Brooks Davis, Khilan Gudka, Nathaniel W Filardo, A Theodore Markettos, et al. 2019. Cheri Concentrate: Practical Compressed Capabilities. *IEEE Trans. Comput.* 68, 10 (2019), 1455–1469.

[22] Yuval Yarom and Katrina Falkner. 2014. FLUSH+RELOAD: A High Resolution, Low Noise, L3 Cache Side-Channel Attack. In *USENIX Security Symposium*. USENIX Association, San Diego, CA, USA, 719–732.

[23] Sizhuo Zhang, Andrew Wright, Thomas Bourgeat, and Arvind. 2018. Composable Building Blocks to Open up Processor Design. In *51st Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 2018, Fukuoka, Japan, October 20-24, 2018*. IEEE Computer Society, 68–81. https://doi.org/10.1109/MICRO.2018.00015