RADICALLY OPEN SECURITY

Code Audit Report

Servo

V 1.1
Amsterdam, December 18th, 2024
Public

Document Properties

Client	Servo
Title	Code Audit Report
Targets	Code paths for CSS floats, tables, and fonts Unsafe blocks in Stylo
Version	1.1
Pentesters	Harry Pantazis, Morgan Hill
Authors	Harry Pantazis, Morgan Hill, Marcus Bointon
Reviewed by	Marcus Bointon
Approved by	Melanie Rieback

Version control

Version	Date	Author	Description
0.1	October 2nd, 2024	Harry Pantazis, Morgan Hill	Initial draft
0.2	October 13th, 2024	Marcus Bointon	Review
0.3	November 21st, 2024	Marcus Bointon	Further review
1.0	December 18th, 2024	Marcus Bointon	1.0
1.1	December 18th, 2024	Morgan Hill	Mark public

Contact

For more information about this document and its contents please contact Radically Open Security B.V.

Name	Melanie Rieback
Address	Science Park 608 1098 XH Amsterdam The Netherlands
Phone	+31 (0)20 2621 255
Email	info@radicallyopensecurity.com

Table of Contents

1	**Executive Summary**	**4**
1.1	Introduction	4
1.2	Scope of work	4
1.3	Project objectives	4
1.4	Timeline	4
1.5	Results In A Nutshell	4
1.6	Summary of Findings	5
1.6.1	Findings by Threat Level	5
1.6.2	Findings by Type	6
1.7	Summary of Recommendations	6
2	**Methodology**	**7**
2.1	Planning	7
2.2	Risk Classification	7
3	**Findings**	**9**
3.1	CLN-009 — Vulnerable dependency time	9
3.2	CLN-004 — Servo TableBuilder arithmetic underflow to out-of-bound access attempt	10
3.3	CLN-002 — Servo TableBuilder slots length arithmetic underflow	11
3.4	CLN-007 — Servo table layout casting usize -> i32	12
4	**Non-Findings**	**14**
4.1	NF-001 — Servo table len(offsets) array may be < 1, resulting in out-of-bounds array access	14
4.2	NF-006 — Servo table construct forgets box_slot	14
4.3	NF-010 — jemalloc workaround for running Servo with Miri	15
4.4	NF-011 — Font tests can't be run with Miri due to FFI	16
4.5	NF-015 — Servo ARC undefined behavior	16
4.6	NF-017 — Fuzzing Servo fonts component	24
5	**Future Work**	**29**
6	**Conclusion**	**30**
Appendix 1	**Testing team**	**32**

1 Executive Summary

1.1 Introduction

Between June 3, 2024 and October 3, 2024, Radically Open Security B.V. carried out a penetration test for Servo.

This report contains our findings as well as detailed explanations of exactly how ROS performed the penetration test.

1.2 Scope of work

The scope of the penetration test was limited to the following targets:

- Code paths for CSS floats, tables, and fonts
- Unsafe blocks in Stylo

The scoped services are broken down as follows:

- Code reading: 1.5 days
- Fuzzing: 2.5 days
- Reporting: 1 days
- **Total effort: 5 days**

1.3 Project objectives

ROS will perform an audit of Servo's float, table, and font CSS code paths with Servo developers in order to assess the security of these aspects of the web engine. To do so ROS will access source code and guide Servo in attempting to find vulnerabilities, exploiting any such found to try and gain further access and elevated privileges.

1.4 Timeline

The security audit took place between June 3, 2024 and October 3, 2024.

1.5 Results In A Nutshell

During this crystal-box penetration test we found 1 Moderate, 1 Low and 2 N/A-severity issues.

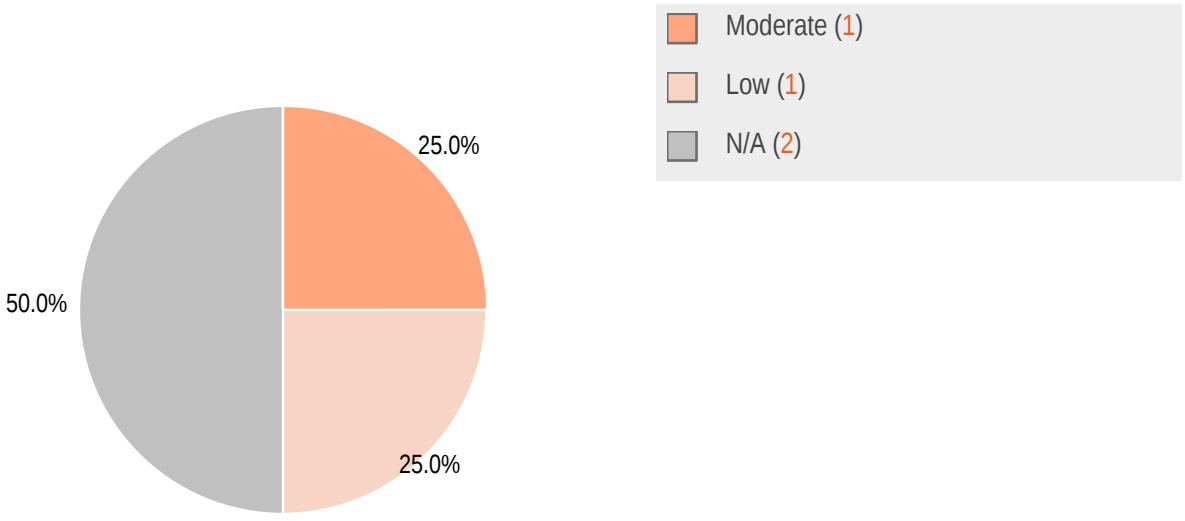
This audit found 1 moderate issue: an out-of-date dependency on the time crate CLN-009 (page 9). There was also a low severity arithmetic underflow CLN-004 (page 10) as well as an inaccessible and therefore N/A severity

underflow CLN-002 (page 11). There was also some potentially overflowing casting CLN-007 (page 12). The rest of this report describes avenues that where pursued but ultimately didn't result in findings within the time constraints of this audit.

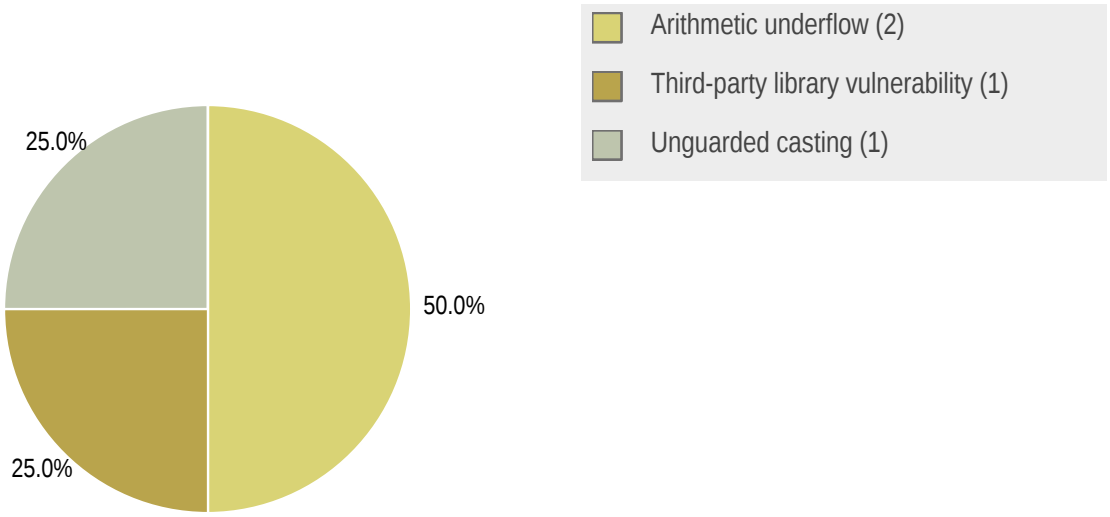
1.6 Summary of Findings

ID	Type	Description	Threat level
CLN-009	Third-Party Library Vulnerability	Servo uses an outdated version of the time crate that is vulnerable to a known issue.	Moderate
CLN-004	Arithmetic underflow	By calling methods on a TableBuilder object in a specific order, an integer underflow can be triggered, followed by an attempted out-of-bounds access, which is caught by Rust, resulting in a panic.	Low
CLN-002	Arithmetic underflow	An arithmetic underflow condition is currently impossible to trigger, but may become accessible as its surrounding logic evolves.	N/A
CLN-007	Unguarded casting	Casting from an unsigned platform pointer-width integer into a fixed-width signed integer may result in erroneous layouts.	N/A

1.6.1 Findings by Threat Level



1.6.2 Findings by Type



1.7 Summary of Recommendations

ID	Type	Recommendation
CLN-009	Third-Party Library Vulnerability	• Upgrade the `time` crate to the latest version.
CLN-004	Arithmetic underflow	• Use a checked subtraction and ensure the index is in range before accessing it.
CLN-002	Arithmetic underflow	• Use a checked subtraction to handle the possible underflow condition as a defense-in-depth measure.
CLN-007	Unguarded casting	• If this is not intended/expected behavior, validate that `n` is not bigger than `i32::MAX`.

2 Methodology

2.1 Planning

Our general approach during code audits is as follows:

1. **Static Code Analysis**
   We performed static analysis on the Rust code with `cargo geiger` and `cargo audit`. We performed dynamic testing by running existing unit tests with miri, an experimental Rust interpreter that can be used to detect various types of bugs (often in unsafe Rust).
2. **Code Analysis and Fuzzing**
   Manual code review and static code analysis were accompanied by the development of fuzzing targets and developing entirely new targets. We used `cargo fuzz` and designed our targets to conform to the libFuzzer interface, which can be used with a variety of fuzzers including AFL++.
   While reading the code we would occasionally spot interesting code paths and develop small stubs to exercise them. These stubs were then used with stepwise debuggers to understand how the code operates at runtime in greater depth.

2.2 Risk Classification

Throughout the report, vulnerabilities or risks are labeled and categorized according to the Penetration Testing Execution Standard (PTES). For more information, see: http://www.pentest-standard.org/index.php/Reporting

These categories are:

- **Extreme**
  Extreme risk of security controls being compromised with the possibility of catastrophic financial/reputational losses occurring as a result.
- **High**
  High risk of security controls being compromised with the potential for significant financial/reputational losses occurring as a result.
- **Elevated**
  Elevated risk of security controls being compromised with the potential for material financial/reputational losses occurring as a result.
- **Moderate**
  Moderate risk of security controls being compromised with the potential for limited financial/reputational losses occurring as a result.

- **Low**

  Low risk of security controls being compromised with measurable negative impacts as a result.

3 Findings

We have identified the following issues:

3.1 CLN-009 — Vulnerable dependency time

Vulnerability ID: CLN-009

Vulnerability type: Third-Party Library Vulnerability

Threat level: Moderate

Description:

Servo uses an outdated version of the `time` crate that is vulnerable to a known issue.

Technical description:

Servo's `time` crate dependency is hard-coded to version `0.1.45`, which is vulnerable to RUSTSEC-2020-0071.

```
Crate:     time
Version:   0.1.45
Title:     Potential segfault in the time crate
Date:      2020-11-18
ID:        RUSTSEC-2020-0071
URL:       https://rustsec.org/advisories/RUSTSEC-2020-0071
Severity:  6.2 (medium)
Solution:  Upgrade to >=0.2.23
```

The affected functions set environment variables without synchronization. On Unix-like operating systems, this can crash in multithreaded programs. Programs may segfault due to dereferencing a dangling pointer if an environment variable is read in a different thread than the affected functions.

Non-Unix targets, Windows, macOS, wasm are unaffected.

Impact:

Memory misuse leading to a segmentation fault, which can be used as a primitive for further exploitation.

Recommendation:

- Upgrade the `time` crate to the latest version.

3.2 CLN-004 — Servo TableBuilder arithmetic underflow to out-of-bound access attempt

Vulnerability ID: CLN-004

Vulnerability type: Arithmetic underflow

Threat level: Low

Description:

By calling methods on a `TableBuilder` object in a specific order, an integer underflow can be triggered, followed by an attempted out-of-bounds access, which is caught by Rust, resulting in a panic.

Technical description:

When the slots vector is empty, an underflow occurs here: https://github.com/servo/servo/blob/7eac599aa1d6bcf8858c51d90763373f0dd5f289/components/layout_2020/table/construct.rs#L470

```
fn current_y(&self) -> usize {
    self.table.slots.len() - 1
}
```

The underflow wraps in release builds. The result of the underflow is then used to index the slots vector: https://github.com/servo/servo/blob/7eac599aa1d6bcf8858c51d90763373f0dd5f289/components/layout_2020/table/construct.rs#L474

```
fn current_x(&self) -> usize {
    self.table.slots[self.current_y()].len()
}
```

The following test reproduces this underflow and subsequent attempted out-of-bounds access.

```
mod test {
    #[test]
    fn table_slot_underflow() {
        let mut table_builder = super::TableBuilder::new(
            style::properties::ComputedValues::initial_values().to_arc(),
            style::properties::ComputedValues::initial_values().to_arc(),
            super::BaseFragmentInfo::anonymous(),
        );
```

```
        table_builder.add_cell(super::TableSlotCell::mock_for_testing(42, 23, 25));
    }
}
```

Release build behavior:

```
---- table::test::table_slot_underflow stdout ----
thread 'table::test::table_slot_underflow' panicked at components/layout_2020/table/
construct.rs:474:25:
index out of bounds: the len is 0 but the index is 18446744073709551615
```

Impact:

A consumer of the `layout_2020` crate could unwittingly trigger a panic when building a table. As this is safe Rust, the out-of-bounds access is limited to denial of service.

Recommendation:

One solution is to use a checked subtraction and ensure the index is in range before accessing it:

```
fn current_y(&self) -> usize {
    self.table.slots.len().checked_sub(1).unwrap_or(0)
}

fn current_x(&self) -> usize {
    self.table
        .slots
        .get(self.current_y())
        .map_or(0, |l| l.len())
}
```

It may be preferable to propagate an error rather than default values, as applying the above fix then hits an unreachable code path: https://github.com/servo/servo/blob/7eac599aa1d6bcf8858c51d90763373f0dd5f289/components/layout_2020/table/construct.rs#L148

3.3 CLN-002 — Servo TableBuilder slots length arithmetic underflow

Vulnerability ID: CLN-002

Vulnerability type: Arithmetic underflow

Threat level: N/A

Description:

An arithmetic underflow condition is currently impossible to trigger, but may become accessible as its surrounding logic evolves.

Technical description:

```
let coords_for_slot_above =
    TableSlotCoordinates::new(target_coords.x, self.slots.len() - 2);
```

Due to the use of an unsigned integer, if the length of `slots` were to be less than 2, then an arithmetic underflow would occur. This arithmetic underflow would wrap to `usize::MAX` or `usize::MAX -1` resulting in an extreme y coordinate.

The line in question can be found here: https://github.com/servo/servo/blob/7eac599aa1d6bcf8858c51d90763373f0dd5f289/components/layout_2020/table/construct.rs#L189

This condition is not accessible from the public interface of the module/crate.

Impact:

Currently this condition is unreachable, however if it became reachable, the consequence would be a cascading extreme value that may trigger other bugs or unexpected behaviors.

Recommendation:

- Use a checked subtraction to handle the possible underflow condition as a defense-in-depth measure.

3.4 CLN-007 — Servo table layout casting usize -> i32

Vulnerability ID: CLN-007

Vulnerability type: Unguarded casting

Threat level: N/A

Description:

Casting from an unsigned platform pointer-width integer into a fixed-width signed integer may result in erroneous layouts.

Technical description:

On 64 bit targets, `usize` is equivalent to `u64`. The code casts this to an `i32`, which means the most significant bits will be truncated, with the most significant of the remaining 32 bits used as a sign. https://github.com/servo/servo/blob/7eac599aa1d6bcf8858c51d90763373f0dd5f289/components/layout_2020/table/layout.rs#L489

The callers do not sanitize values of `n` that are greater than `i32::MAX`, making it possible to craft a table with negative `baseline_border_spacing`.

Impact:

Unlikely to be security relevant; most likely just visual anomaly.

Recommendation:

- If this is not intended/expected behavior, validate that `n` is not bigger than `i32::MAX`.

4 Non-Findings

In this section we list some of the things that were tried but turned out to be dead ends.

4.1 NF-001 — Servo table len(offsets) array may be < 1, resulting in out-of-bounds array access

There is a hardcoded access to index 0 here: https://github.com/servo/servo/blob/7eac599aa1d6bcf8858c51d90763373f0dd5f289/components/layout_2020/table/mod.rs#L163

However, in practice this will never attempt out-of-bounds access because `TableSlot::Spanned` variants are always created with `TableSlot::new_spanned()`, which ensures the length of the vector is always at least 1.

https://github.com/servo/servo/blob/7eac599aa1d6bcf8858c51d90763373f0dd5f289/components/layout_2020/table/mod.rs#L263

TableSlot offsets can only be inserted into the vector and cannot be removed, leaving no real-world situation where a spanned table slot would contain a vector of 0 elements. https://github.com/servo/servo/blob/7eac599aa1d6bcf8858c51d90763373f0dd5f289/components/layout_2020/table/construct.rs#L213

The types do allow the construction of a spanned table slot with 0 offsets which would result in a panic here, though this is a matter of developer preference rather than a security issue. The documentation clearly explains that one or more offsets are expected.

https://github.com/servo/servo/blob/7eac599aa1d6bcf8858c51d90763373f0dd5f289/components/layout_2020/table/mod.rs#L236

4.2 NF-006 — Servo table construct forgets box_slot

Box slots are forgotten, resulting in them being leaked without `Drop` being called here: https://github.com/servo/servo/blob/7eac599aa1d6bcf8858c51d90763373f0dd5f289/components/layout_2020/table/construct.rs#L768.

The developers informed us that the intention is to avoid an assertion in the `Drop` implementation. The assertion in question is here: https://github.com/servo/servo/blob/7eac599aa1d6bcf8858c51d90763373f0dd5f289/components/layout_2020/dom.rs#L93.

The developers suggest that it has been done this way with a view to implementing incremental layouts in future.

There is no clear security relevance to this behavior; Leaking in itself is safe.

4.3 NF-010 — jemalloc workaround for running Servo with Miri

The use of jemalloc blocks running Servo with Miri. The following output is generated because Miri does not implement a stub for jemalloc:

```
error: unsupported operation: can't call foreign function `_rjem_malloc` on OS `linux`
   --> /home/test/.cargo/registry/src/index.crates.io-6f17d22bba15001f/jemallocator-0.5.4/src/
lib.rs:105:13
    |
105 |             ffi::malloc(layout.size())
    |             ^^^^^^^^^^^^^^^^^^^^^^^^^^ can't call foreign function `_rjem_malloc` on OS
 `linux`
    |
    = help: if this is a basic API commonly used on this target, please report an issue with Miri
    = help: however, note that Miri does not aim to support every FFI function out there; for
 instance, we will not support APIs for things such as GUIs, scripting languages, or databases
    = note: BACKTRACE:
    = note: inside `<jemallocator::Jemalloc as std::alloc::GlobalAlloc>::alloc` at /home/
test/.cargo/registry/src/index.crates.io-6f17d22bba15001f/jemallocator-0.5.4/src/lib.rs:105:13:
 105:39
note: inside `servo_allocator::_::__rust_alloc`
   --> /home/test/servo/components/allocator/lib.rs:7:1
    |
7   | #[global_allocator]
    | ^^^^^^^^^^^^^^^^^^
    = note: inside `std::alloc::alloc` at /home/test/.rustup/toolchains/nightly-aarch64-unknown-
linux-gnu/lib/rustlib/src/rust/library/alloc/src/alloc.rs:100:9: 100:52
    = note: inside `std::alloc::Global::alloc_impl` at /home/test/.rustup/toolchains/nightly-
aarch64-unknown-linux-gnu/lib/rustlib/src/rust/library/alloc/src/alloc.rs:183:73: 183:86
    = note: inside `<std::alloc::Global as std::alloc::Allocator>::allocate` at /home/test/.rustup/
toolchains/nightly-aarch64-unknown-linux-gnu/lib/rustlib/src/rust/library/alloc/src/alloc.rs:243:9:
 243:39
    = note: inside closure at /home/test/.rustup/toolchains/nightly-aarch64-unknown-linux-gnu/lib/
rustlib/src/rust/library/alloc/src/sync.rs:532:26: 532:49
    = note: inside
 `std::sync::Arc::<std::mem::MaybeUninit<std::thread::Inner>>::allocate_for_layout::
    <{closure@std::sync::Arc<std::thread::Inner>::new_uninit::{closure#0}}, fn(*mut u8) -> *mut
 alloc::sync::ArcInner<std::mem::MaybeUninit<std::thread::Inner>> {std::ptr::mut_ptr::<impl
 *mut u8>::cast::<alloc::sync::ArcInner<std::mem::MaybeUninit<std::thread::Inner>>>}>` at /home/
test/.rustup/toolchains/nightly-aarch64-unknown-linux-gnu/lib/rustlib/src/rust/library/alloc/src/
sync.rs:1892:19: 1892:35
    = note: inside `std::sync::Arc::<std::thread::Inner>::new_uninit` at /home/test/.rustup/
toolchains/nightly-aarch64-unknown-linux-gnu/lib/rustlib/src/rust/library/alloc/src/sync.rs:530:27:
 534:14
    = note: inside `std::thread::Thread::new_inner` at /home/test/.rustup/toolchains/nightly-
aarch64-unknown-linux-gnu/lib/rustlib/src/rust/library/std/src/thread/mod.rs:1344:27: 1344:53
    = note: inside `std::thread::Thread::new_main` at /home/test/.rustup/toolchains/nightly-aarch64-
unknown-linux-gnu/lib/rustlib/src/rust/library/std/src/thread/mod.rs:1332:18: 1332:51
    = note: inside `std::rt::init` at /home/test/.rustup/toolchains/nightly-aarch64-unknown-linux-
gnu/lib/rustlib/src/rust/library/std/src/rt.rs:98:18: 98:36
    = note: inside closure at /home/test/.rustup/toolchains/nightly-aarch64-unknown-linux-gnu/lib/
rustlib/src/rust/library/std/src/rt.rs:140:42: 140:67
    = note: inside `std::panicking::r#try::do_call::<{closure@std::rt::lang_start_internal::
{closure#1}}, ()>` at /home/test/.rustup/toolchains/nightly-aarch64-unknown-linux-gnu/lib/rustlib/
src/rust/library/std/src/panicking.rs:557:40: 557:43
    = note: inside `std::panicking::r#try::<(), {closure@std::rt::lang_start_internal::
{closure#1}}>` at /home/test/.rustup/toolchains/nightly-aarch64-unknown-linux-gnu/lib/rustlib/src/
rust/library/std/src/panicking.rs:521:19: 521:88
```

```
    = note: inside `std::panic::catch_unwind::<{closure@std::rt::lang_start_internal::{closure#1}},
 ()>` at /home/test/.rustup/toolchains/nightly-aarch64-unknown-linux-gnu/lib/rustlib/src/rust/
library/std/src/panic.rs:350:14: 350:33
    = note: inside `std::rt::lang_start_internal` at /home/test/.rustup/toolchains/nightly-aarch64-
unknown-linux-gnu/lib/rustlib/src/rust/library/std/src/rt.rs:140:5: 140:70
    = note: inside `std::rt::lang_start::<()>` at /home/test/.rustup/toolchains/nightly-aarch64-
unknown-linux-gnu/lib/rustlib/src/rust/library/std/src/rt.rs:161:17: 166:6
    = note: this error originates in the attribute macro `global_allocator` (in Nightly builds, run
 with -Z macro-backtrace for more info)
```

It is however possible to work around this by instructing Servo to build against the system allocator. The Servo alloc component has a `use-system-allocator` feature switch, which Miri does implement, so runs can go further by enabling this feature.

4.4 NF-011 — Font tests can't be run with Miri due to FFI

One of the limitations of Miri is that it only implements a subset of common foreign functions. Consequently, in cases such as this, it is not possible to run code that depends on FFI.

```
test test_font_fallback ... error: unsupported operation: can't call foreign function
 `FT_New_Library` on OS `linux`
   --> /home/test/servo/components/fonts/platform/freetype/library_handle.rs:103:30
    |
103 |                  let result = FT_New_Library(freetype_memory, &mut freetype_library);
    |                               ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ can't call
 foreign function `FT_New_Library` on OS `linux`
```

4.5 NF-015 — Servo ARC undefined behavior

Stylo implements its own atomic reference counter (ARC) types in the Servo ARC. These ARC types differ from Rust's `std::sync::Arc` in that it is designed to receive ARC pointers from C++ which translates pointers to `T` rather than `Arc<T>`. To work around this, Servo ARC conjures a temporary ARC in order to track references within Rust. The means the reference counter in Rust and the reference counter in C++ are not synchronized. However, as long as C++ increments the reference counter when transmitting the pointer to Rust and doesn't decrement this reference, the FFI boundary should be sound. The Rust side uses `core::mem:forget(arc.clone())` appropriately to ensure the reference count is incremented in the required locations.

Within the time available for this audit I am reasonably satisfied with the soundness of this implementation. Miri does highlight what it believes to be undefined behavior, but I think it is a false positive. I have included the output here for reference.

Miri `tables::test_empty_table` Tree borrows output:

```
test tables::test_empty_table ... error: Undefined Behavior: write access through <186864> at
 alloc65988[0x0] is forbidden
   --> /Users/pcwizz/.cargo/git/checkouts/stylo-d5871ad1ba1940d3/141446a/servo_arc/lib.rs:426:28
    |
426 |             let old_size = self.inner().count.fetch_add(1, Relaxed);
```

```
    |                           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ write access through
 <186864> at alloc65988[0x0] is forbidden
    |
    = help: this indicates a potential bug in the program: it performed an invalid operation, but
 the Tree Borrows rules it violated are still experimental
    = help: the accessed tag <186864> is a child of the conflicting tag <186856>
    = help: the conflicting tag <186856> has state Frozen which forbids this child write access
help: the accessed tag <186864> was created here
   --> /Users/pcwizz/work/ROS/nlnet-off-ngie-servo/servo/components/layout_2020/table/
construct.rs:251:13
    |
251 |             ComputedValues::initial_values().to_arc(),
    |             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
help: the conflicting tag <186856> was created here, in the initial state Frozen
   --> /Users/pcwizz/work/ROS/nlnet-off-ngie-servo/servo/components/layout_2020/table/
construct.rs:251:13
    |
251 |             ComputedValues::initial_values().to_arc(),
    |             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    = note: BACKTRACE (of the first span) on thread `tables::test_empty_table`:
    = note: inside `<servo_arc::Arc<style::properties::generated::ComputedValues> as
 std::clone::Clone>::clone` at /Users/pcwizz/.cargo/git/checkouts/stylo-d5871ad1ba1940d3/141446a/
servo_arc/lib.rs:426:28: 426:68
    = note: inside
 `servo_arc::Arc::<style::properties::generated::ComputedValues>::from_raw_addrefed` at /Users/
pcwizz/.cargo/git/checkouts/stylo-d5871ad1ba1940d3/141446a/servo_arc/lib.rs:275:21: 275:32
    = note: inside `style::properties::generated::ComputedValues::to_arc` at /Users/pcwizz/work/ROS/
nlnet-off-ngie-servo/servo/target/miri/aarch64-apple-darwin/debug/build/style-3ab73a6644f7c785/out/
properties.rs:68696:18: 68696:46
note: inside `layout_2020::table::TableBuilder::new_for_tests`
   --> /Users/pcwizz/work/ROS/nlnet-off-ngie-servo/servo/components/layout_2020/table/
construct.rs:251:13
    |
251 |             ComputedValues::initial_values().to_arc(),
    |             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
note: inside `tables::test_empty_table`
   --> components/layout_2020/tests/tables.rs:44:29
    |
44  |         let table_builder = TableBuilder::new_for_tests();
    |                             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^
note: inside closure
   --> components/layout_2020/tests/tables.rs:43:26
    |
42  |     #[test]
    |     ------- in this procedural macro expansion
43  |     fn test_empty_table() {
    |                          ^
    = note: this error originates in the attribute macro `test` (in Nightly builds, run with -Z
 macro-backtrace for more info)
```

Miri `tables::test_empty_table` output Stacked borrows:

```
test tables::test_empty_table ... error: Undefined Behavior: trying to retag from <194575> for
 SharedReadWrite permission at alloc66502[0x0], but that tag does not exist in the borrow stack for
 this location
   --> /Users/pcwizz/.cargo/git/checkouts/stylo-d5871ad1ba1940d3/141446a/servo_arc/lib.rs:339:18
    |
339 |         unsafe { &*self.ptr() }
    |                  ^^^^^^^^^^^^
    |                  |
```

```
    |                  trying to retag from <194575> for SharedReadWrite permission at
 alloc66502[0x0], but that tag does not exist in the borrow stack for this location
    |                  this error occurs as part of retag at alloc66502[0x0..0xd8]
    |
    = help: this indicates a potential bug in the program: it performed an invalid operation, but
 the Stacked Borrows rules it violated are still experimental
    = help: see https://github.com/rust-lang/unsafe-code-guidelines/blob/master/wip/stacked-
borrows.md for further information
help: <194575> was created by a SharedReadOnly retag at offsets [0x10..0xd8]
   --> /Users/pcwizz/work/ROS/nlnet-off-ngie-servo/servo/components/layout_2020/table/
construct.rs:251:13
    |
251 |             ComputedValues::initial_values().to_arc(),
    |             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    = note: BACKTRACE (of the first span) on thread `tables::test_empty_table`:
    = note: inside `servo_arc::Arc::<style::properties::generated::ComputedValues>::inner` at /
Users/pcwizz/.cargo/git/checkouts/stylo-d5871ad1ba1940d3/141446a/servo_arc/lib.rs:339:18: 339:30
    = note: inside `<servo_arc::Arc<style::properties::generated::ComputedValues> as
 std::clone::Clone>::clone` at /Users/pcwizz/.cargo/git/checkouts/stylo-d5871ad1ba1940d3/141446a/
servo_arc/lib.rs:414:12: 414:24
    = note: inside
 `servo_arc::Arc::<style::properties::generated::ComputedValues>::from_raw_addrefed` at /Users/
pcwizz/.cargo/git/checkouts/stylo-d5871ad1ba1940d3/141446a/servo_arc/lib.rs:275:21: 275:32
    = note: inside `style::properties::generated::ComputedValues::to_arc` at /Users/pcwizz/work/ROS/
nlnet-off-ngie-servo/servo/target/miri/aarch64-apple-darwin/debug/build/style-3ab73a6644f7c785/out/
properties.rs:68696:18: 68696:46
note: inside `layout_2020::table::TableBuilder::new_for_tests`
   --> /Users/pcwizz/work/ROS/nlnet-off-ngie-servo/servo/components/layout_2020/table/
construct.rs:251:13
    |
251 |             ComputedValues::initial_values().to_arc(),
    |             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
note: inside `tables::test_empty_table`
   --> components/layout_2020/tests/tables.rs:44:29
    |
44  |         let table_builder = TableBuilder::new_for_tests();
    |                             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
note: inside closure
   --> components/layout_2020/tests/tables.rs:43:26
    |
42  |     #[test]
    |     ------- in this procedural macro expansion
43  |     fn test_empty_table() {
    |                          ^
    = note: this error originates in the attribute macro `test` (in Nightly builds, run with -Z
 macro-backtrace for more info)
```

Miri `parser::tests::parent_selector` Tree borrows:

```
test parser::tests::parent_selector ... error: Undefined Behavior: write access through <7089406> at
 alloc2584450[0x0] is forbidden
    --> /home/test/.rustup/toolchains/nightly-aarch64-unknown-linux-gnu/lib/rustlib/src/rust/
library/core/src/sync/atomic.rs:3365:24
     |
3365 |             Release => intrinsics::atomic_xsub_release(dst, val),
     |                        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ write access through
 <7089406> at alloc2584450[0x0] is forbidden
     |
     = help: this indicates a potential bug in the program: it performed an invalid operation, but
 the Tree Borrows rules it violated are still experimental
```

```
     = help: the accessed tag <7089406> is a child of the conflicting tag <7089384>
     = help: the conflicting tag <7089384> has state Frozen which forbids this child write access
help: the accessed tag <7089406> was created here
    --> /home/test/stylo/servo_arc/lib.rs:1133:39
     |
1133 |                 let _ = Arc::from_raw(&*x);
     |                                       ^^^
help: the conflicting tag <7089384> was created here, in the initial state Frozen
    --> /home/test/stylo/servo_arc/lib.rs:1070:55
     |
1070 |             let borrow = unsafe { ArcBorrow::from_ref(&(*ptr).data) };
     |                                                       ^^^^^^^^^^^^
     = note: BACKTRACE (of the first span) on thread `parser::tests::`:
     = note: inside `std::sync::atomic::atomic_sub::<usize>` at /home/test/.rustup/toolchains/
nightly-aarch64-unknown-linux-gnu/lib/rustlib/src/rust/library/core/src/sync/atomic.rs:3365:24:
 3365:65
     = note: inside `std::sync::atomic::AtomicUsize::fetch_sub` at /home/test/.rustup/toolchains/
nightly-aarch64-unknown-linux-gnu/lib/rustlib/src/rust/library/core/src/sync/atomic.rs:2703:26:
 2703:62
note: inside `<servo_arc::Arc<servo_arc::HeaderSlice<builder::SpecificityAndFlags,
 parser::Component<parser::tests::DummySelectorImpl>>> as std::ops::Drop>::drop`
    --> /home/test/stylo/servo_arc/lib.rs:536:12
     |
536  |         if self.inner().count.fetch_sub(1, Release) != 1 {
     |            ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
     = note: inside
 `std::ptr::drop_in_place::<servo_arc::Arc<servo_arc::HeaderSlice<builder::SpecificityAndFlags,
 parser::Component<parser::tests::DummySelectorImpl>>>> -
 shim(Some(servo_arc::Arc<servo_arc::HeaderSlice<builder::SpecificityAndFlags,
 parser::Component<parser::tests::DummySelectorImpl>>>))` at /home/test/.rustup/toolchains/nightly-
aarch64-unknown-linux-gnu/lib/rustlib/src/rust/library/core/src/ptr/mod.rs:542:1: 542:56
note: inside `<servo_arc::ArcUnion<servo_arc::HeaderSlice<builder::SpecificityAndFlags,
 parser::Component<parser::tests::DummySelectorImpl>>, servo_arc::HeaderSlice<(),
 parser::Selector<parser::tests::DummySelectorImpl>>> as std::ops::Drop>::drop`
    --> /home/test/stylo/servo_arc/lib.rs:1133:43
     |
1133 |                 let _ = Arc::from_raw(&*x);
     |                                           ^
     = note: inside
 `std::ptr::drop_in_place::<servo_arc::ArcUnion<servo_arc::HeaderSlice<builder::SpecificityAndFlags,
 parser::Component<parser::tests::DummySelectorImpl>>, servo_arc::HeaderSlice<(),
 parser::Selector<parser::tests::DummySelectorImpl>>>> -
 shim(Some(servo_arc::ArcUnion<servo_arc::HeaderSlice<builder::SpecificityAndFlags,
 parser::Component<parser::tests::DummySelectorImpl>>, servo_arc::HeaderSlice<(),
 parser::Selector<parser::tests::DummySelectorImpl>>>))` at /home/test/.rustup/toolchains/nightly-
aarch64-unknown-linux-gnu/lib/rustlib/src/rust/library/core/src/ptr/mod.rs:542:1: 542:56
     = note: inside
 `std::ptr::drop_in_place::<parser::SelectorList<parser::tests::DummySelectorImpl>> -
 shim(Some(parser::SelectorList<parser::tests::DummySelectorImpl>))` at /home/test/.rustup/
toolchains/nightly-aarch64-unknown-linux-gnu/lib/rustlib/src/rust/library/core/src/ptr/mod.rs:542:1:
 542:56
     = note: inside `std::ptr::drop_in_place::
     <std::result::Result<parser::SelectorList<parser::tests::DummySelectorImpl>,
 cssparser::ParseError<'_, parser::SelectorParseErrorKind<'_>>>> -
 shim(Some(std::result::Result<parser::SelectorList<parser::tests::DummySelectorImpl>,
 cssparser::ParseError<'_, parser::SelectorParseErrorKind<'_>>>))` at /home/test/.rustup/toolchains/
nightly-aarch64-unknown-linux-gnu/lib/rustlib/src/rust/library/core/src/ptr/mod.rs:542:1: 542:56
note: inside `parser::tests::parent_selector`
    --> selectors/parser.rs:4448:39
     |
4448 |         assert!(parse("foo &").is_ok());
```

```
     |                                       ^
note: inside closure
    --> selectors/parser.rs:4447:25
     |
4446 |     #[test]
     |     ------- in this procedural macro expansion
4447 |     fn parent_selector() {
     |                         ^
```

Miri `parser::tests::parent_selector` Stacked borrows:

```
test parser::tests::parent_selector ... error: Undefined Behavior: attempting a write access using
 <7269319> at alloc2584008[0x18], but that tag does not exist in the borrow stack for this location
    --> /home/test/.rustup/toolchains/nightly-aarch64-unknown-linux-gnu/lib/rustlib/src/rust/
library/core/src/ptr/mod.rs:1534:9
     |
1534 |         intrinsics::write_via_move(dst, src)
     |         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
     |         |
     |         attempting a write access using <7269319> at alloc2584008[0x18], but that tag does
 not exist in the borrow stack for this location
     |         this error occurs as part of an access at alloc2584008[0x18..0x50]
     |
     = help: this indicates a potential bug in the program: it performed an invalid operation, but
 the Stacked Borrows rules it violated are still experimental
     = help: see https://github.com/rust-lang/unsafe-code-guidelines/blob/master/wip/stacked-
borrows.md for further information
help: <7269319> would have been created here, but this is a zero-size retag ([0x18..0x18]) so the
 tag in question does not exist anywhere
    --> /home/test/stylo/servo_arc/lib.rs:808:35
     |
808  |                 let mut current = std::ptr::addr_of_mut!(p.as_mut().data.data) as *mut T;
     |                                   ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
     = note: BACKTRACE (of the first span) on thread `parser::tests::`:
     = note: inside `std::ptr::write::<parser::Component<parser::tests::DummySelectorImpl>>` at /
home/test/.rustup/toolchains/nightly-aarch64-unknown-linux-gnu/lib/rustlib/src/rust/library/core/
src/ptr/mod.rs:1534:9: 1534:45
note: inside `servo_arc::Arc::<servo_arc::HeaderSlice<builder::SpecificityAndFlags,
 parser::Component
 <parser::tests::DummySelectorImpl>>>::from_header_and_iter_alloc::<{closure@servo_arc::Arc
 <servo_arc::HeaderSlice<builder::SpecificityAndFlags,
 parser::Component<parser::tests::DummySelectorImpl>>>::from_header_and_iter_with_size
 <builder::ExactChain<std::iter::Rev<smallvec::Drain<'_,
 [parser::Component<parser::tests::DummySelectorImpl>; 32]>>,
 std::iter::Cloned<std::slice::Iter<'_, parser::Component<parser::tests::DummySelectorImpl>>>>::
{closure#0}}, builder::ExactChain<std::iter::Rev<smallvec::Drain<'_,
 [parser::Component<parser::tests::DummySelectorImpl>; 32]>>, std::iter::Cloned<std::slice::Iter<'_,
 parser::Component<parser::tests::DummySelectorImpl>>>>`
    --> /home/test/stylo/servo_arc/lib.rs:810:21
     |
810  | /                     ptr::write(
811  | |                         current,
812  | |                         items
813  | |                             .next()
814  | |                             .expect("ExactSizeIterator over-reported length"),
815  | |                     );
     | |_____________________^
note: inside `servo_arc::Arc::<servo_arc::HeaderSlice<builder::SpecificityAndFlags,
 parser::Component<parser::tests::DummySelectorImpl>>>::
 from_header_and_iter_with_size::<builder::ExactChain<std::iter::Rev<smallvec::Drain<'_,
```

```
 [parser::Component<parser::tests::DummySelectorImpl>; 32]>>, std::iter::Cloned<std::slice::Iter<'_,
 parser::Component<parser::tests::DummySelectorImpl>>>>>`
    --> /home/test/stylo/servo_arc/lib.rs:859:9
     |
859  | /         Arc::from_header_and_iter_alloc(
860  | |             |layout| unsafe { alloc::alloc(layout) },
861  | |             header,
862  | |             items,
863  | |             num_items,
864  | |             /* is_static = */ false,
865  | |         )
     | |_________^
note: inside `servo_arc::Arc::<servo_arc::HeaderSlice<builder::SpecificityAndFlags,
 parser::Component<parser::tests::DummySelectorImpl>>>::
 from_header_and_iter::<builder::ExactChain<std::iter::Rev<smallvec::Drain<'_,
 [parser::Component<parser::tests::DummySelectorImpl>; 32]>>, std::iter::Cloned<std::slice::Iter<'_,
 parser::Component<parser::tests::DummySelectorImpl>>>>>`
    --> /home/test/stylo/servo_arc/lib.rs:876:9
     |
876  |         Self::from_header_and_iter_with_size(header, items, len)
     |         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
note: inside
 `builder::SelectorBuilder::<parser::tests::DummySelectorImpl>::build_with_specificity_and_flags`
    --> selectors/builder.rs:124:9
     |
124  |         Arc::from_header_and_iter(spec, ExactChain(self.components.drain(..).rev(),
 implicit_selector.iter().cloned()))
     |         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
 ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
note: inside `builder::SelectorBuilder::<parser::tests::DummySelectorImpl>::build`
    --> selectors/builder.rs:89:9
     |
89   |         self.build_with_specificity_and_flags(sf, parse_relative)
     |         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
note: inside `parser::parse_selector::<'_, parser::tests::DummyParser,
 parser::tests::DummySelectorImpl>`
    --> selectors/parser.rs:2740:24
     |
2740 |     return Ok(Selector(builder.build(parse_relative)));
     |                        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
note: inside closure
    --> selectors/parser.rs:514:36
     |
514  |                 let mut selector = parse_selector(parser, input, state, parse_relative);
     |                                    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
     = note: inside `cssparser::Parser::<'_, '_>::parse_entirely::<{closure@selectors/
parser.rs:512:71: 512:78}, parser::Selector<parser::tests::DummySelectorImpl>,
 parser::SelectorParseErrorKind<'_>>` at /home/test/.cargo/registry/src/
index.crates.io-6f17d22bba15001f/cssparser-0.34.0/src/parser.rs:695:22: 695:33
     = note: inside `cssparser::parser::parse_until_before::<'_, '_, {closure@selectors/
parser.rs:512:71: 512:78}, parser::Selector<parser::tests::DummySelectorImpl>,
 parser::SelectorParseErrorKind<'_>>` at /home/test/.cargo/registry/src/
index.crates.io-6f17d22bba15001f/cssparser-0.34.0/src/parser.rs:1065:18: 1065:56
     = note: inside `cssparser::Parser::<'_, '_>::parse_until_before::<{closure@selectors/
parser.rs:512:71: 512:78}, parser::Selector<parser::tests::DummySelectorImpl>,
 parser::SelectorParseErrorKind<'_>>` at /home/test/.cargo/registry/src/
index.crates.io-6f17d22bba15001f/cssparser-0.34.0/src/parser.rs:803:9: 803:86
note: inside `parser::SelectorList::<parser::tests::DummySelectorImpl>::parse_with_state::<'_,
 parser::tests::DummyParser>`
    --> selectors/parser.rs:512:28
     |
```

```
512  |               let selector = input.parse_until_before(Delimiter::Comma, |input| {
     |  ________________________________^
513  | |                 let start = input.position();
514  | |                 let mut selector = parse_selector(parser, input, state, parse_relative);
515  | |                 if forgiving && (selector.is_err() || input.expect_exhausted().is_err()) {
...    |
519  | |                 selector
520  | |             })?;
     | |_____________^
note: inside `parser::SelectorList::<parser::tests::DummySelectorImpl>::parse::<'_,
 parser::tests::DummyParser>`
    --> selectors/parser.rs:472:9
     |
472  | /         Self::parse_with_state(
473  | |             parser,
474  | |             input,
475  | |             SelectorParsingState::empty(),
476  | |             ForgivingParsing::No,
477  | |             parse_relative,
478  | |         )
     | |_________^
note: inside `parser::tests::parse_ns_relative_expected`
    --> selectors/parser.rs:3933:22
     |
3933 |           let result = SelectorList::parse(
     |  ________________________^
3934 | |             parser,
3935 | |             &mut CssParser::new(&mut parser_input),
3936 | |             parse_relative,
3937 | |         );
     | |_________^
note: inside `parser::tests::parse_ns_relative`
    --> selectors/parser.rs:3915:9
     |
3915 |         parse_ns_relative_expected(input, parser, parse_relative, None)
     |         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
note: inside `parser::tests::parse_relative`
    --> selectors/parser.rs:3885:9
     |
3885 |         parse_ns_relative(input, &DummyParser::default(), parse_relative)
     |         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
note: inside `parser::tests::parse`
    --> selectors/parser.rs:3878:9
     |
3878 |         parse_relative(input, ParseRelative::No)
     |         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
note: inside `parser::tests::parent_selector`
    --> selectors/parser.rs:4448:17
     |
4448 |         assert!(parse("foo &").is_ok());
     |                 ^^^^^^^^^^^^^^
note: inside closure
    --> selectors/parser.rs:4447:25
     |
4446 |     #[test]
     |     ------- in this procedural macro expansion
4447 |     fn parent_selector() {
     |
```

Miri `tests::slices_and_thin` Stacked borrows:

```
test tests::slices_and_thin ... error: Undefined Behavior: attempting a write access using <208681>
 at alloc70149[0x20], but that tag does not exist in the borrow stack for this location
    --> /Users/pcwizz/.rustup/toolchains/nightly-aarch64-apple-darwin/lib/rustlib/src/rust/library/
core/src/ptr/mod.rs:1534:9
     |
1534 |         intrinsics::write_via_move(dst, src)
     |         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
     |         |
     |         attempting a write access using <208681> at alloc70149[0x20], but that tag does not
 exist in the borrow stack for this location
     |         this error occurs as part of an access at alloc70149[0x20..0x24]
     |
     = help: this indicates a potential bug in the program: it performed an invalid operation, but
 the Stacked Borrows rules it violated are still experimental
     = help: see https://github.com/rust-lang/unsafe-code-guidelines/blob/master/wip/stacked-
borrows.md for further information
help: <208681> would have been created here, but this is a zero-size retag ([0x20..0x20]) so the tag
 in question does not exist anywhere
    --> servo_arc/lib.rs:808:35
     |
808  |                 let mut current = std::ptr::addr_of_mut!(p.as_mut().data.data) as *mut T;
     |                                   ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
     = note: BACKTRACE (of the first span) on thread `tests::slices_and_thin`:
     = note: inside `std::ptr::write::<i32>` at /Users/pcwizz/.rustup/toolchains/nightly-aarch64-
apple-darwin/lib/rustlib/src/rust/library/core/src/ptr/mod.rs:1534:9: 1534:45
note: inside `Arc::<HeaderSlice<tests::Canary,
 i32>>::from_header_and_iter_alloc::<{closure@servo_arc/lib.rs:860:13: 860:21},
 std::vec::IntoIter<i32>>`
    --> servo_arc/lib.rs:810:21
     |
810  | /                     ptr::write(
811  | |                         current,
812  | |                         items
813  | |                             .next()
814  | |                             .expect("ExactSizeIterator over-reported length"),
815  | |                     );
     | |_____________________^
note: inside `Arc::<HeaderSlice<tests::Canary,
 i32>>::from_header_and_iter_with_size::<std::vec::IntoIter<i32>>`
    --> servo_arc/lib.rs:859:9
     |
859  | /         Arc::from_header_and_iter_alloc(
860  | |             |layout| unsafe { alloc::alloc(layout) },
861  | |             header,
862  | |             items,
863  | |             num_items,
864  | |             /* is_static = */ false,
865  | |         )
     | |_________^
note: inside `Arc::<HeaderSlice<tests::Canary,
 i32>>::from_header_and_iter::<std::vec::IntoIter<i32>>`
    --> servo_arc/lib.rs:876:9
     |
876  |         Self::from_header_and_iter_with_size(header, items, len)
     |         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
note: inside `tests::slices_and_thin`
    --> servo_arc/lib.rs:1196:21
     |
1196 |             let x = Arc::from_header_and_iter(c, v.into_iter());
     |                     ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
note: inside closure
```

```
    --> servo_arc/lib.rs:1191:25
     |
1190 |     #[test]
     |     ------- in this procedural macro expansion
1191 |     fn slices_and_thin() {
     |
```

4.6 NF-017 — Fuzzing Servo fonts component

We created three distinct fuzzing targets to test the robustness of the font handling logic within Servo's fonts component. Each target was designed to explore different aspects of font processing in Servo. For this we used cargo-fuzz.

The three fuzz targets are:

1. FontTemplate::new_for_remote_web_font()
2. PlatformFont::new_from_data()
3. fallback_font_families()

The targets focus on `FontTemplate::new_for_remote_web_font()`, `PlatformFont::new_from_data()`, and `fallback_font_families()` to test remote font template creation, platform font data loading, and the fallback font mechanism, respectively. By fuzzing `FontTemplate::new_for_remote_web_font()`, we ensured that malformed font URLs and data are handled gracefully. The `PlatformFont::new_from_data()` fuzzer helped validate the function's ability to deal with corrupted or non-standard font data without causing crashes or unexpected behavior. Finally, fuzzing `fallback_font_families()` tested Servo's resilience in selecting appropriate fallback fonts for various Unicode characters.

FontTemplate::new_for_remote_web_font()

- Fuzzing harness implementation:

```
#![no_main]
use libfuzzer_sys::fuzz_target;
use servo_url::ServoUrl;
use std::sync::Arc;
use style::stylesheets::DocumentStyleSheet;
use fonts::*;

fuzz_target!(|data: (String, Vec<u8>)| {
    // Unpack the datas
    let url_string = data.0;
    let font_data = Arc::new(data.1);
    let stylesheet_flag = Option::None;

    // Create a basic CSSFontFaceDescriptors
    // TODO: Arbitrary it
    let css_font_template_descriptors = CSSFontFaceDescriptors {
        family_name: LowercaseString::new("hello"),
        weight: None,
        style: None,
```

```
        stretch: None,
        unicode_range: None,
    };

    // Try to create a ServoUrl from the string
    if let Ok(url) = ServoUrl::parse(&url_string) {
        // Fuzz the new_for_remote_web_font function
        let _ = FontTemplate::new_for_remote_web_font(url, font_data,
 &css_font_template_descriptors, stylesheet_flag);
    }
});
```

- Example log:

```
INFO: Running with entropic power schedule (0xFF, 100).
INFO: Seed: 2176315550
INFO: Loaded 1 modules   (951707 inline 8-bit counters): 951707 [0x55738e7ffd50, 0x55738e8e82eb),
INFO: Loaded 1 PC tables (951707 PCs): 951707 [0x55738e8e82f0,0x55738f76dca0),
INFO:        0 files found in /home/kali/servo-src/components/fonts/fuzz/corpus/
fuzz_new_for_remote_web_font
INFO: -max_len is not provided; libFuzzer will not generate inputs larger than 4096 bytes
INFO: A corpus is not provided, starting from an empty corpus
<a href="#f2-servo-tablebuilder-slots-length-arithmetic-underflow"/>      INITED cov: 94 ft: 95
 corp: 1/1b exec/s: 0 rss: 134Mb
        NEW_FUNC[1/1]: 0x55738e3d75c0  (/home/kali/servo-src/target/x86_64-unknown-linux-gnu/
release/fuzz_new_for_remote_web_font+0x51355c0) (BuildId: b438683415607283)
<a href="#f4-servo-tablebuilder-arithmetic-underflow-to-out-of-bound-access-attempt"/>      NEW
 cov: 106 ft: 113 corp: 2/3b lim: 4 exec/s: 0 rss: 134Mb L: 2/2 MS: 2 ChangeByte-CrossOver-
#5      NEW    cov: 121 ft: 134 corp: 3/6b lim: 4 exec/s: 0 rss: 134Mb L: 3/3 MS: 1 CrossOver-
#12     NEW    cov: 122 ft: 136 corp: 4/8b lim: 4 exec/s: 0 rss: 134Mb L: 2/3 MS: 2 EraseBytes-
ChangeByte-
#14     NEW    cov: 123 ft: 137 corp: 5/11b lim: 4 exec/s: 0 rss: 134Mb L: 3/3 MS: 2 ChangeByte-
ChangeBit-
        NEW_FUNC[1/1]: 0x55738dee3cc0  (/home/kali/servo-src/target/x86_64-unknown-linux-gnu/
release/fuzz_new_for_remote_web_font+0x4c41cc0) (BuildId: b438683415607283)

<SNIP>

#829071049      REDUCE cov: 3427 ft: 17148 corp: 3698/1229Kb lim: 4096 exec/s: 1715 rss: 857Mb L:
 1236/4076 MS: 3 PersAutoDict-ShuffleBytes-EraseBytes- DE: "c\000"-
#829102166      REDUCE cov: 3427 ft: 17148 corp: 3698/1229Kb lim: 4096 exec/s: 1715 rss: 857Mb L:
 50/4076 MS: 2 ShuffleBytes-EraseBytes-
#829532399      REDUCE cov: 3427 ft: 17148 corp: 3698/1229Kb lim: 4096 exec/s: 1714 rss: 857Mb L:
 208/4076 MS: 3 ChangeASCIIInt-ChangeByte-EraseBytes-
#829597791      REDUCE cov: 3427 ft: 17148 corp: 3698/1229Kb lim: 4096 exec/s: 1714 rss: 857Mb L:
 11/4076 MS: 2 EraseBytes-ChangeBit-
#829629808      REDUCE cov: 3427 ft: 17148 corp: 3698/1229Kb lim: 4096 exec/s: 1714 rss: 857Mb L:
 15/4076 MS: 2 EraseBytes-ChangeBit-
#829684354      REDUCE cov: 3427 ft: 17148 corp: 3698/1229Kb lim: 4096 exec/s: 1714 rss: 857Mb L:
 1285/4076 MS: 1 EraseBytes-
#829972386      REDUCE cov: 3427 ft: 17148 corp: 3698/1229Kb lim: 4096 exec/s: 1713 rss: 857Mb L:
 317/4076 MS: 2 ChangeBinInt-EraseBytes-
#830070963      REDUCE cov: 3427 ft: 17148 corp: 3698/1229Kb lim: 4096 exec/s: 1713 rss: 857Mb L:
 15/4076 MS: 2 ChangeByte-EraseBytes-
#830112284      REDUCE cov: 3427 ft: 17148 corp: 3698/1229Kb lim: 4096 exec/s: 1713 rss: 857Mb L:
 327/4076 MS: 1 EraseBytes-
```

```
#830136885      REDUCE cov: 3427 ft: 17148 corp: 3698/1229Kb lim: 4096 exec/s: 1713 rss: 857Mb L:
 122/4076 MS: 1 EraseBytes-
```

This function was tested extensively, and no crashes or memory corruption issues were discovered.

PlatformFont::new_from_data()

- Fuzzing harness implementation:

```
#![no_main]
use libfuzzer_sys::fuzz_target;
use fonts::*;
use std::sync::Arc;
use fonts::platform::font::PlatformFont;
//use fonts::FontIdentifier;
use servo_url::ServoUrl;


fuzz_target!(|data: Vec<u8> | {
    let path = "/tmp/hello.ttf";
    let font_data = Arc::new(data);
    let face_index = 0;

    match ServoUrl::from_file_path(path.clone()) {
        Ok(url) => {
            let identifier = FontIdentifier::Web(url);
            //match PlatformFont::new_from_data(identifier, font_data, face_index, None) {
            //    Ok(_) => {}, // Handle success
            //    Err(e) => println!("Failed to create font: {:?}", e),
            //}
            let _ = PlatformFont::new_from_data(identifier, font_data, face_index, None);
        },
        Err(e) => println!("Invalid URL: {:?}", e),
    }
});
```

- Example log:

```
INFO: Running with entropic power schedule (0xFF, 100).
INFO: Seed: 3399497878
INFO: Loaded 1 modules   (943055 inline 8-bit counters): 943055 [0x55ac8a5da1f0, 0x55ac8a6c05bf),
INFO: Loaded 1 PC tables (943055 PCs): 943055 [0x55ac8a6c05c0,0x55ac8b5242b0),
INFO:      125 files found in /home/kali/servo-src/components/fonts/fuzz/corpus/fuzz_new_from_data
INFO: -max_len is not provided; libFuzzer will not generate inputs larger than 4096 bytes
INFO: seed corpus: files: 125 min: 1b max: 513b total: 8892b rss: 135Mb
#126    INITED cov: 144 ft: 264 corp: 30/1739b exec/s: 0 rss: 135Mb
#16384  pulse  cov: 144 ft: 264 corp: 30/1739b lim: 675 exec/s: 5461 rss: 135Mb
#32768  pulse  cov: 144 ft: 264 corp: 30/1739b lim: 837 exec/s: 4681 rss: 135Mb
#65536  pulse  cov: 144 ft: 264 corp: 30/1739b lim: 1156 exec/s: 4681 rss: 135Mb
#125889 REDUCE cov: 144 ft: 264 corp: 30/1737b lim: 1756 exec/s: 4496 rss: 135Mb L: 259/513 MS: 3
 ChangeByte-CopyPart-EraseBytes-
#131072 pulse  cov: 144 ft: 264 corp: 30/1737b lim: 1806 exec/s: 4519 rss: 135Mb
#262144 pulse  cov: 144 ft: 264 corp: 30/1737b lim: 3112 exec/s: 4599 rss: 144Mb
#524288 pulse  cov: 144 ft: 264 corp: 30/1737b lim: 4096 exec/s: 4599 rss: 186Mb

<SNIP>
```

```
#2147483648     pulse  cov: 144 ft: 276 corp: 31/3786b lim: 4096 exec/s: 4791 rss: 583Mb
```

Fuzzing this target revealed that the system generally handled corrupted font data safely, rejecting malformed inputs without crashing. However, there were instances where memory consumption increased significantly when processing large, malformed font files.

fallback_font_families()

- Fuzzing harness implementation:

```
#![no_main]
use libfuzzer_sys::fuzz_target;
use fonts::{FallbackFontSelectionOptions, fallback_font_families};

fuzz_target!(|data: (char, Option<char>)| {
    let (character, next_character) = data;
    let _ = fallback_font_families(FallbackFontSelectionOptions::new(character, next_character));
});
```

- Example log:

```
INFO: Loaded 1 modules   (942681 inline 8-bit counters): 942681 [0x55cd2f33c370, 0x55cd2f4225c9),
INFO: Loaded 1 PC tables (942681 PCs): 942681 [0x55cd2f4225d0,0x55cd30284b60),
INFO:      467 files found in /home/kali/servo-src/components/fonts/fuzz/corpus/
fuzz_fallback_font_selection
INFO: -max_len is not provided; libFuzzer will not generate inputs larger than 4096 bytes
INFO: seed corpus: files: 467 min: 5b max: 9b total: 2599b rss: 134Mb
#468    INITED cov: 931 ft: 978 corp: 463/2579b exec/s: 0 rss: 134Mb
#16384  pulse  cov: 931 ft: 978 corp: 463/2579b lim: 165 exec/s: 8192 rss: 134Mb
#32768  pulse  cov: 931 ft: 978 corp: 463/2579b lim: 328 exec/s: 8192 rss: 134Mb
#65536  pulse  cov: 931 ft: 978 corp: 463/2579b lim: 656 exec/s: 9362 rss: 134Mb
#131072 pulse  cov: 931 ft: 978 corp: 463/2579b lim: 1305 exec/s: 8192 rss: 135Mb
#262144 pulse  cov: 931 ft: 978 corp: 463/2579b lim: 2605 exec/s: 7943 rss: 159Mb
#1048576        pulse  cov: 931 ft: 978 corp: 463/2579b lim: 4096 exec/s: 6853 rss: 308Mb

<SNIP>

#4294967296   pulse  cov: 931 ft: 978 corp: 463/2561b lim: 4096 exec/s: 6198 rss: 540Mb
```

This function exhibited robust performance during testing with a wide variety of Unicode characters. The fallback mechanism consistently selected appropriate fonts, with no observable crashes or performance degradation.

Summary

The fuzzing efforts on Servo's `fonts` module, targeting `FontTemplate::new_for_remote_web_font()`, `PlatformFont::new_from_data()`, and `fallback_font_families()`, yielded insightful results. Each harness explored distinct parts of the font-handling pipeline, providing valuable insights into the system's behavior when exposed to malformed inputs.

In conclusion, the fuzzing results indicate that Servo's font-handling code is largely resilient against malformed inputs. As a future best practice, fuzzing should be implemented by developers across components in a similar fashion to the rust-cssparser implementation.

5 Future Work

- **Servo fonts component fuzzing enhancements**

  Servo, as a project, inherits the security properties of Rust, and thus lacks a need for platform-specific code in comparison to its codebase's size. One area where platform specific behavior is required lies in the `fonts` component, where different platforms require different fonts.

  While we performed fuzzing on the `fonts` component with Linux as a target in non-finding NF-017 (page 24), future fuzzing campaigns should target the different platforms supported by Servo (macOS, Windows, Android). Additionally, the fuzzing campaigns performed used fonts found under `tests` path of the component in the codebase as the input corpus, which are `CSSTest` and `dejavu-fonts-ttf-2.37`. While these are perfectly good candidates for testing the functionality of the component as it interacts with other components, they do not thoroughly test the security posture of the fonts-handling code.

  In particular, TTFs (TrueType fonts) are binary files that contain glyph data, outlines, and **instructions** for rendering text. Mishandling these can lead to security issues such as memory corruption or even code execution. A custom font fuzzing interface should be used as the input corpus or even as the fuzzing campaign orchestrator. One such example project is BrokenType by Google's Project Zero.

- **Servo ARC soundness study**

  Servo ARC generated a few interesting undefined behavior warnings when running with Miri. In this audit we evaluated these warnings manually, by looking at the code, as we were limited on time/scope to go much further. As Servo ARC is a core component of Servo we believe it may justify more attention. We would suggest scoping a project to validate the implementation further, including inspecting the built objects to compare our understanding of Servo ARC with what is actually running. It would also be interesting to validate the C++ side at the same time to ensure the FFI boundary is clean.

- **Retest of findings**

  When mitigations for the vulnerabilities described in this report have been deployed, a repeat test should be performed to ensure that they are effective and have not introduced other security problems.

- **Regular security assessments**

  Security is an ongoing process and not a product, so we advise undertaking regular security assessments and penetration tests, ideally prior to every major release or every quarter.

6 Conclusion

We discovered 1 Moderate, 1 Low and 2 N/A-severity issues during this penetration test.

This audit had a very narrow focus on the font rendering and layout components of Servo and Stylo. This targeted scope was required to keep to a reasonable time budget for a first audit in such a large code base. This engagement has produced future work items that could help guide scoping subsequent audits focused on other parts of the Servo ecosystem.

Dependency management is a persistent problem in software development. Thankfully tools such as `cargo audit`, or more generically GitHub's Dependabot, exist to assist developers in tracking the requirement for updates. We found one outdated dependency during our audit in CLN-009 (page 9).

Arithmetic on computers is deceptively difficult. The seemingly innocuous statement `y=x-1` leads us as security auditors to wonder if `x` is unsigned, perhaps we can make `y` an extreme value to our advantage somewhere else in the code. Rust's safety thwarts attempts to turn this into out-of-bounds memory access as we might in C/C++, leaving us in the domain of application logic bugs. In this audit we discovered two such arithmetic underflows in CLN-002 (page 11) and CLN-004 (page 10).

Casting between fixed-width and platform-width or signed and unsigned integers is a source of many historical vulnerabilities. In Rust the behavior is defined and therefore safe, in the sense that we know what will happen. However, we don't know whether this defined behavior is desirable behavior for this application. We noted one instance highlighting this question in CLN-007 (page 12).

Servo lives in a split modality between Rust and C++, with the Stylo component being consumed by both Gecko and Servo. This Firefox lineage leads to some rather interesting accommodations such as the Servo ARC as a solution to passing atomic reference-counted objects between the C++ and Rust contexts. Within the scope of this audit did not discover any concrete problems, though Miri did suggest some avenues for further investigation. We suggest that this would be an excellent target for a future audit to ensure that Rust's memory safety is not being tainted.

It was a pleasure to look at even a small part of such an important project. We had clear and productive communication with the developers, which made this engagement flow nicely. We are also thankful to NLNET for funding this project through NGIE. We look forward to seeing the Servo project develop further to protect a pluralistic web rendering ecosystem.

We recommend fixing all of the issues found and then performing a retest in order to ensure that mitigations are effective and that no new vulnerabilities have been introduced.

Finally, we want to emphasize that security is a process – this penetration test is just a one-time snapshot. Security posture must be continuously evaluated and improved. Regular audits and ongoing improvements are essential in order to maintain control of your corporate information security. We hope that this pentest report (and the detailed explanations of our findings) will contribute meaningfully towards that end.

Please don't hesitate to let us know if you have any further questions, or need further clarification on anything in this report.

Appendix 1 Testing team

Harry Pantazis	Harry is a Junior Penetration Tester at ROS with a background in DevOps and system administration as well as OSCP certified. He has experience with code auditing, binary analysis, fuzzing and web application testing, with a focus on delivering precise and actionable insights.
Morgan Hill	Morgan is a seasoned security consultant with a background in IoT and DevOps. He currently specialises in Rust and AVoIP.
Melanie Rieback	Melanie Rieback is a former Asst. Prof. of Computer Science from the VU, who is also the co-founder/CEO of Radically Open Security.