Challenge 6 ("bloke2")

Description

> You've been so helpful lately, and that was very good work you did. Yes, I'm going to put it right here, on the refrigerator, very good job indeed. You're the perfect person to help me with another issue that came up.
>
> One of our lab researchers has mysteriously disappeared. He was working on the prototype for a hashing IP block that worked very much like, but not identically to, the common Blake2 hash family. Last we heard from him, he was working on the testbenches for the unit. One of his labmates swears she knew of a secret message that could be extracted with the testbenches, but she couldn't quite recall how to trigger it. Maybe you could help?

Details

> (...)
>
> You should be able to get to the answer by modifying testbenches alone, thoughthere are some helpful diagnostics inside some of the code files which you could uncomment if you want a look at what's going on inside. Brute-forcing won'treally help you here; some things have been changed from the true implementationof Blake2 to discourage brute-force attempts.
>
> (...)

Writeup

I will admit that while I was not expecting to see Verilog in a CTF, I was not caught entirely off-guard, thanks to a mandatory course (and in my opinion, one of the coolest courses) at my university, in which you're required to design a single-scalar RISC-V CPU and describe it using this very language.

The challenge turned out not to be very hard if you just looked in the right place, which neither I nor most people that I know or whose posts I have read have. Looking back, I think it's safe to say that it wasn't a very well designed challenge, since apparently, someone solved it using a single ChatGPT prompt, and as such, I will not be going into great depths in this writeup.

The several verilog files in the archive describe a hardware implementation of a modification of the Blake2 hash function, specifically its variants Blake2b and Blake2s, which differ basically only in their respective block sizes. Namely, the files `bloke2s.v` and `bloke2b.v` define specializations of a generic module in `bloke2.v`, which consists of a "data manager" (`data_mgr.v`) and a compression function f (`f_unit.v`), which in turn utilises a scheduler module (`f_sched.v`) and an inner function g (`g_unit.v`). From my observations, the g function, the number of rounds, initialization vectors, as

well as the `SIGMA` permutation and `R0,R1,R2,R3` rotation constants are identical between Blake and Bloke. I suspect the difference is in the construction of the compression function f, however, I did not confirm this and it turned out not to be relevant to the challenge at all.

In fact, none of the inner workings or properties of the hash function were relevant. The key to solving the riddle was hidden on line 53 of `data_mgr.v`:

```
localparam TEST_VAL =
512'h3c9cf0addf2e45ef548b011f736cc99144bdfee0d69df4090c8a39c520e18ec3bdc1277aad1706f756affca41
178dac066e4beb8ab7dd2d1402c4d624aaabe40;
```

This suspiciously looking "test value" of course contains the encrypted flag. Tracking down where this data gets read leads to line 67 of the same file:

```
h <= h_in ^ (TEST_VAL & {(W*16){tst}});
```

which depends on `tst`, which is a register (line 28, `reg tst;`) that is assigned the value of the `finish` input wire (line 40, `tst <= finish;`) on every `start` or `rst` signal. The `finish` input wire is set in `bloke2.v` on line 60 and transitively in either testbench on line 22. Effectively, the value of `tst` is determined by the value assigned to the `finish` register on line 59 of either testbench. Changing the value from `1'b0` to `1'b1` in `bloke2b.v`, leaving only the test case `hash_message("abc");` and running `make tests` produces the flag:

```
vvp  f_sched.test.outiverilog -g2012 -o bloke2b.test.out bloke2.v f_sched.v f_unit.v
g_over_2.v g.v g_unit.v data_mgr.v bloke2s.v bloke2b.v bloke2b_tb.vvvp
bloke2b.test.out706c656173655f73656e645f68656c705f695f616d5f74726170706564 5f696e5f615f6374665f
666c61675f666163746f727940666c6172652d6f6e2e636f6dReceived message:
please_send_help_i_am_trapped_in_a_ctf_flag_factory@flare-on.com
```

Revision #1
Created 23 December 2024 23:07:00 by Annatar
Updated 24 December 2024 00:09:30 by Annatar