

HARDENING PROGRAMS WITH GENETIC ALGORITHMS

1. ATTACK SIMULATION (OR ARMS RACE)

A more difficult but interesting approach might be to model an evolutionary “arms race” between a program (e.g. the `lighttpd` web server) and an exploit (e.g. the `hoagie` attack). One such scenario could be represented as follows:

- (1) `Hoagie` is written by some hacker, and a pre-patch `lighttpd` is vulnerable.
- (2) The genetic programming automatic repair technique fixes `lighttpd` such that `hoagie` fails.
- (3) A slightly modified version (more on this later) of the repair technique evolves `hoagie` into a new exploit to which `lighttpd` is once more vulnerable.
- (4) The repair technique fixes `lighttpd`.
- (5) And so on...

Notably, steps 1 and 2 have already been demonstrated, and the difficulty here lies predominantly with step 3, and possibly step 4. Part of the problem is that only one data point would be available to the genetic algorithm’s fitness function in evolving `hoagie`. Basically, this is “does the exploit succeed?” or something to that effect. So as things stand, the method would be conducting a random search. We would likely need to create a specially crafted fitness function for each tested exploit.

In creating a new fitness function, we might check several “obvious things” — although such things would be unique to each exploit. In `hoagie`’s case, we might check:

- (1) Does `hoagie` connect to the server?
- (2) Does it make a request?
- (3) And so on...

More generally, it may be useful to collect a larger dataset regarding the runtime properties of a running exploit. Information collected from `pin` or something similar might be a reasonable input to the fitness function.

The larger and more dangerous problem is that the existing technique (with or without a special fitness function) is not at all capable of evolving program exploits in this way. Previous “fixes” that we generated were relatively simple in nature. This concern has been at least partially ameliorated, however, as an attack-repair cycle has been demonstrated for a toy program and exploit pair.