Bingus Bomb Analysis and Defusal (No Patch)

Scope / Target

- Target binary: `./target/bingus`
- Goal: Prevent the binary from ·exploding· without modifying the binary file.

Environment & Tooling

- Docker container: `agent-re` (Ubuntu 24.04.3)
- Tools used: `file`, `strings`, `radare2`, `python3`
- All commands executed inside Docker: path `/tmp/workdir` maps to the host repo root.

Methodology

- Static inspection to identify behavior and constraints.
- Light disassembly in radare2 to trace the decision path for ·exploded· vs ·survived·.
- Derived arithmetic condition on input argument and validated with a runtime PoC.

Evidence

File type

```
$ docker exec -i agent-re bash -c "file ./target/bingus"
./target/bingus: ELF 64-bit LSB pie executable, x86-64, dynamically
linked, stripped
```

Strings of interest

```
$ docker exec -i agent-re bash -c "strings -a ./target/bingus |
grep -E 'Bingus|herring'"
Bingus exploded
This is not a red herring
This is a red herring
Bingus survived
```

Disassembly highlights (radare2)

Key logic from the entry routine (stripped binary; main identified at `0x1149`):

- Require exactly one argument: `argc == 2`, else prints "Bingus exploded".
- Require `strlen(argv[1]) == 2`, else prints "Bingus exploded".
- Require `argv[1][0] == argv[1][1]`, else prints "Bingus exploded".
- Compute a checksum:
  - Start with `0x66`.
  - Add the sum of bytes in the literal string `"This is a red herring"`.
  - Add `argv[1][0] + argv[1][1]` (bytes treated as signed, but ASCII < 128 unaffected).
- Compare the result against `0x8c5` (2245 decimal). If equal ·"Bingus survived"; else ·"Bingus

exploded".

Relevant r2 snippet (addresses annotated):

```
0x1159: cmp argc, 2           ; require one user arg
0x1196: cmp strlen(argv[1]),2 ; require length 2
0x117f: cmp argv[1][0], argv[1][1] ; first two chars equal
...
; sum = 0x66 + sum("This is a red herring") + argv[1][0] +
argv[1][1]
0x1237: cmp [sum], 0x8c5
0x1256: puts("Bingus survived")
0x1240: puts("Bingus exploded")
```

Arithmetic derivation

Using Python to compute the base sum:

```
$ docker exec -i agent-re bash -c "python3 - << 'PY'\n s = 'This is
a red herring'\n base = 0x66 + sum(ord(c) for c in s)\n print('base
=', base)\n print('delta =', 0x8c5 - base)\nPY"
base = 2021
delta = 224
```

Therefore we need `argv[1][0] + argv[1][1] = 224`. With the additional constraint that the two bytes are equal, the unique printable ASCII solution is `112 + 112 = 224`, i.e., both characters `'p'`.

Defusal (No Binary Patch)

- Pass the argument `"pp"` to the program.

Proof:

```
$ docker exec -i agent-re bash -c "chmod +x ./target/bingus &&
./target/bingus pp; echo RET:$?"
Bingus survived
RET:0
```

Counter·example:

```
$ docker exec -i agent-re bash -c "./target/bingus aa; echo RET:$?"
Bingus exploded
RET:1
```

Findings

- The binary is a simple ·bomb· that checks an input invariant.
- Required input: exactly 2 characters, both identical, and together summing to decimal 224.
- Minimal printable solution: `pp`.

IOCs/Keys/Creds

- N/A

Mitigations / Notes

- No patching is required. Ensure the program is always launched with `pp` as its single argument when defusal is desired.
- Any wrapper/script can enforce this (e.g., shell alias or small launcher).

Conclusion

Providing `pp` as the sole argument satisfies all internal checks, resulting in ·Bingus survived· without modifying the binary.

Next Steps

- If you want a convenience launcher, I can add `./helper/run_bingus_safe.sh` to always invoke the correct argument.