Motivation

- Functions that take variable number of arguments are not checked statically, e.g.,
  int printf(const char*format,...);
- Vulnerabilities arise from differences between how the caller passes the variadic arguments and how the callee uses them.
- HexVASAN analyzes variadic functions and enforces runtime integrity checks.

Attacks Model

- In a format string attack, an attacker controls the first argument and changes how many parameters are used and interpreted.
- HexVASAN prevents such attacks.

HexVASAN Architecture

```c
int add(int n, ...) {
    va_list list;
    va_start(list, n);
    for (int i=0; i < n; i++) {
        total = total + va_arg(list, int);
    }
    va_end(list);
    return total;
}
int main(int argc, const char *argv[]) {
    int a, b, c, result;
    float d;
    result = add(4, a, b, c, d);
    return 0;
}
```

Result

```
Num. of arguments
1st argument
2nd argument
3rd argument
4th argument
Caller Side (main)
Callee Side (func add)
Runtime stack
```

Conclusion

- Real software such as Firefox, Chromium uses variadic functions both directly and indirectly, leading to a potential attack surface.
- HexVASAN successfully tracks if there is any type mismatch and prevents exploits.
- Incurs 0.45% and 1% overhead for SPEC CPU2006 and Firefox, respectively.