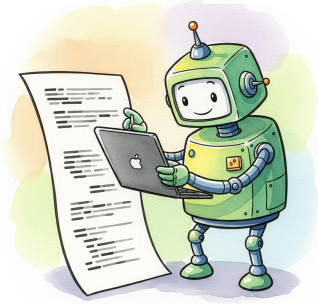
PatchAgent: A Practical Program Repair Agent Mimicking Human Expertise

Zheng Yu, Ziyi Guo, Yuhang Wu, Jiahao Yu, Meng Xu, Dongliang Mu, Yan Chen, Xinyu Xing

Northwestern University. University of Waterloo. Independent Researcher.



Problem Definition

Given a concrete input that triggers a vulnerability,

how to patch the program without breaking existing tests?

Agenda

- Motivation Example
- PatchAgent Design
- Evaluation and Results

A Global Buffer Overflow Bug

```
1  const M3OpInfo c_operations[] = { /* ... */ };
2  const M3OpInfo c_operationsFC[] = { /* ... */ };
3
4  static inline const M3OpInfo*
5  GetOpInfo(m3opcode_t opcode) {
6      switch (opcode >> 8) {
7      case 0x00:
8          return &c_operations[opcode];
9      case 0xFC:
10         return &c_operationsFC[opcode & 0xFF];
11     default:
12         return NULL;
13     }
14 }
15
16 M3Result
17 Compile_BlockStat(IM3Compilation o) {
18     m3opcode_t opcode;
19     Read_opcode (&opcode, &o);
20     IM3OpInfo opinfo = GetOpInfo(opcode);
21     _throwif(unknownOpcode, opinfo == NULL);
22     if (opinfo->compiler) { // global overflow
23         (*opinfo->compiler) (o, opcode)
24     } else {
25         Compile_Operator (o, opcode);
26     }
27 }
```

```
==35==ERROR: AddressSanitizer: global-buffer-overflow
READ of size 8 at 0x55bc18 thread T0
    #0 0x55969b in Compile_BlockStat /source/m3_compile.c:22
    #1 0x55c4c6 in Compile_Block /source/m3_compile.c:2277
    #2 0x55cbc3 in Compile_If /source/m3_compile.c:1648
    #3 0x5596ec in Compile_BlockStatement /source/m3_compile.c:2207
    #4 0x55ca29 in Parse_InitExpr /source/m3_parse.c:282
    ......
    #8 0x55d715 in LLVMFuzzerTestOneInput /app_fuzz/fuzzer.c:30
    #9 0x552e14 in fuzzer::Fuzzer::ExecuteCallback (BuildId:
f0fdeb36a)
    ......
    0x555bc18 is located 88 bytes after global variable c_operations

    SUMMARY: AddressSanitizer: global-buffer-overflow
```

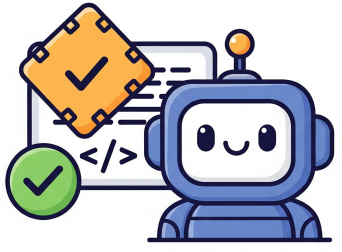
What ability do human/LLM have?

View Code

The viewcode API retrieves the code context by specifying file names and line numbers

Find Definition

The find_definition finds the definition location of symbols by specifying their names and reference locations



Validate

The validate API to check if the patch can mitigate the PoC and pass all functional tests

Human Expert

```
==35==ERROR: AddressSanitizer: global-buffer-overflow
READ of size 8 at 0x55bc18 thread T0
    #0 0x55969b in Compile_BlockStat /source/m3_compile.c:22
    #1 0x55c4c6 in Compile_Block /source/m3_compile.c:2277
    #2 0x55cbc3 in Compile_If /source/m3_compile.c:1648
    #3 0x5596ec in Compile_BlockStatement /source/m3_compile.c:2207
    #4 0x55ca29 in Parse_InitExpr /source/m3_parse.c:282
    ......
    #8 0x55d715 in LLVMFuzzerTestOneInput /app_fuzz/fuzzer.c:30
    #9 0x552e14 in fuzzer::Fuzzer::ExecuteCallback (BuildId: f0fdeb36a)
    ......
    0x555bc18 is located 88 bytes after global variable c_operations

    SUMMARY: AddressSanitizer: global-buffer-overflow
```

According to the address sanitizer report, a global overflow occurred at line 22 in the /source/m3_compile.c. Let me view lines 17 to 27 in /source/m3_compile.c.

Human Expert

```
==35==ERROR: AddressSanitizer: global-buffer-overflow
READ of size 8 at 0x55bc18 thread T0
    #0 0x55969b in Compile_BlockStat /source/m3_compile.c:22
    #1 0x55c4c6 in Compile_Block /source/m3_compile.c:2277
    #2 0x55cbc3 in Compile_If /source/m3_compile.c:1648
    #3 0x5596ec in Compile_BlockStatement /source/m3_compile.c:2207
    #4 0x55ca29 in Parse_InitExpr /source/m3_parse.c:282
    ......
    #8 0x55d715 in LLVMFuzzerTestOneInput /app_fuzz/fuzzer.c:30
    #9 0x552e14 in fuzzer::Fuzzer::ExecuteCallback (BuildId: f0fdeb36a)
    ......
    0x555bc18 is located 88 bytes after global variable c_operations

    SUMMARY: AddressSanitizer: global-buffer-overflow
```

According to the address sanitizer report, a global overflow occurred at line 22 in the /source/m3_compile.c. Let me view lines 17 to 27 in /source/m3_compile.c.

```
....
IM3OpInfo opinfo = GetOpInfo(opcode);
_throwif(unknownOpcode, opinfo == NULL);
if (opinfo->compiler) {
   (*opinfo->compiler) (o, opcode)
....
```

Human Expert

```
==35==ERROR: AddressSanitizer: global-buffer-overflow
READ of size 8 at 0x55bc18 thread T0
    #0 0x55969b in Compile_BlockStat /source/m3_compile.c:22
    #1 0x55c4c6 in Compile_Block /source/m3_compile.c:2277
    #2 0x55cbc3 in Compile_If /source/m3_compile.c:1648
    #3 0x5596ec in Compile_BlockStatement /source/m3_compile.c:2207
    #4 0x55ca29 in Parse_InitExpr /source/m3_parse.c:282
    ......
    #8 0x55d715 in LLVMFuzzerTestOneInput /app_fuzz/fuzzer.c:30
    #9 0x552e14 in fuzzer::Fuzzer::ExecuteCallback (BuildId: f0fdeb36a)
    ......
    0x555bc18 is located 88 bytes after global variable c_operations

    SUMMARY: AddressSanitizer: global-buffer-overflow
```

According to the address sanitizer report, a global overflow occurred at line 22 in the /source/m3_compile.c. Let me view lines 17 to 27 in /source/m3_compile.c.

```
....
IM3OpInfo opinfo = GetOpInfo(opcode);
_throwif(unknownOpcode, opinfo == NULL);
if (opinfo->compiler) {
    (*opinfo->compiler) (o, opcode)
....
```

`opinfo` is an overflow pointer generated by GetOpInfo, I would like to locate its definition.

Human Expert

```
==35==ERROR: AddressSanitizer: global-buffer-overflow
READ of size 8 at 0x55bc18 thread T0
    #0 0x55969b in Compile_BlockStat /source/m3_compile.c:22
    #1 0x55c4c6 in Compile_Block /source/m3_compile.c:2277
    #2 0x55cbc3 in Compile_If /source/m3_compile.c:1648
    #3 0x5596ec in Compile_BlockStatement /source/m3_compile.c:2207
    #4 0x55ca29 in Parse_InitExpr /source/m3_parse.c:282
    ......
    #8 0x55d715 in LLVMFuzzerTestOneInput /app_fuzz/fuzzer.c:30
    #9 0x552e14 in fuzzer::Fuzzer::ExecuteCallback (BuildId: f0fdeb36a)
    ......
    0x555bc18 is located 88 bytes after global variable c_operations

    SUMMARY: AddressSanitizer: global-buffer-overflow
```

According to the address sanitizer report, a global overflow occurred at line 22 in the /source/m3_compile.c. Let me view lines 17 to 27 in /source/m3_compile.c.

....
IM3OpInfo opinfo = GetOpInfo(opcode);
_throwif(unknownOpcode, opinfo == NULL);
if (opinfo->compiler) {
  (*opinfo->compiler) (o, opcode)
....

`opinfo` is an overflow pointer generated by GetOpInfo, I would like to locate its definition.

The variable GetOpInfo is defined at line 5 in the /source/m3_compile.c.

Human Expert

```
==35==ERROR: AddressSanitizer: global-buffer-overflow
READ of size 8 at 0x55bc18 thread T0
    #0 0x55969b in Compile_BlockStat /source/m3_compile.c:22
    #1 0x55c4c6 in Compile_Block /source/m3_compile.c:2277
    #2 0x55cbc3 in Compile_If /source/m3_compile.c:1648
    #3 0x5596ec in Compile_BlockStatement /source/m3_compile.c:2207
    #4 0x55ca29 in Parse_InitExpr /source/m3_parse.c:282
    ......
    #8 0x55d715 in LLVMFuzzerTestOneInput /app_fuzz/fuzzer.c:30
    #9 0x552e14 in fuzzer::Fuzzer::ExecuteCallback (BuildId: f0fdeb36a)
    ......
    0x555bc18 is located 88 bytes after global variable c_operations

    SUMMARY: AddressSanitizer: global-buffer-overflow
```

According to the address sanitizer report, a global overflow occurred at line 22 in the /source/m3_compile.c. Let me view lines 17 to 27 in /source/m3_compile.c.

....
IM3OpInfo opinfo = GetOpInfo(opcode);
_throwif(unknownOpcode, opinfo == NULL);
if (opinfo->compiler) {
  (*opinfo->compiler) (o, opcode)
....

`opinfo` is an overflow pointer generated by GetOpInfo, I would like to locate its definition.

The variable GetOpInfo is defined at line 5 in the /source/m3_compile.c.

Great, now I want to view lines 5 to 16 in /source/m3_compile.c

Human Expert

```
==35==ERROR: AddressSanitizer: global-buffer-overflow
READ of size 8 at 0x55bc18 thread T0
    #0 0x55969b in Compile_BlockStat /source/m3_compile.c:22
    #1 0x55c4c6 in Compile_Block /source/m3_compile.c:2277
    #2 0x55cbc3 in Compile_If /source/m3_compile.c:1648
    #3 0x5596ec in Compile_BlockStatement /source/m3_compile.c:2207
    #4 0x55ca29 in Parse_InitExpr /source/m3_parse.c:282
    ......
    #8 0x55d715 in LLVMFuzzerTestOneInput /app_fuzz/fuzzer.c:30
    #9 0x552e14 in fuzzer::Fuzzer::ExecuteCallback (BuildId: f0fdeb36a)
    ......
    0x555bc18 is located 88 bytes after global variable c_operations

    SUMMARY: AddressSanitizer: global-buffer-overflow
```

According to the address sanitizer report, a global overflow occurred at line 22 in the /source/m3_compile.c. Let me view lines 17 to 27 in /source/m3_compile.c.

....
IM3OpInfo opinfo = GetOpInfo(opcode);
_throwif(unknownOpcode, opinfo == NULL);
if (opinfo->compiler) {
  (*opinfo->compiler) (o, opcode)
....

`opinfo` is an overflow pointer generated by GetOpInfo, I would like to locate its definition.

The variable GetOpInfo is defined at line 5 in the /source/m3_compile.c.

Great, now I want to view lines 5 to 16 in /source/m3_compile.c

GetOpInfo(m3opcode_t opcode) {
  switch (opcode >> 8) {
  case 0x00:
    return &c_operations[opcode]; ....

Human Expert

```
==35==ERROR: AddressSanitizer: global-buffer-overflow
READ of size 8 at 0x55bc18 thread T0
    #0 0x55969b in Compile_BlockStat /source/m3_compile.c:22
    #1 0x55c4c6 in Compile_Block /source/m3_compile.c:2277
    #2 0x
    #3 0                                                    2207
    #4 0
    ....
    #8 0
    #9 0x
    ......
    0x555bc18 is located 88 bytes after global variable c_operations

    SUMMARY: AddressSanitizer: global-buffer-overflow
```

The report indicates that the overflow occurred in c_operations, so I need to add a boundary check before performing pointer arithmetic.

According to the address sanitizer report, a global overflow occurred at line 22 in the /source/m3_compile.c. Let me view lines 17 to 27 in /source/m3_compile.c.

```
....
IM3OpInfo opinfo = GetOpInfo(opcode);
_throwif(unknownOpcode, opinfo == NULL);
if (opinfo->compiler) {
    (*opinfo->compiler) (o, opcode)
....
```

`opinfo` is an overflow pointer generated by GetOpInfo, I would like to locate its definition.

The variable GetOpInfo is defined at line 5 in the /source/m3_compile.c.

Great, now I want to view lines 5 to 16 in /source/m3_compile.c

```
GetOpInfo(m3opcode_t opcode) {
    switch (opcode >> 8) {
    case 0x00:
        return &c_operations[opcode]; ....
```

Human Expert

```
==35==ERROR: AddressSanitizer: global-buffer-overflow
READ of size 8 at 0x55bc18 thread T0
    #0 0x55969b in Compile_BlockStat /source/m3_compile.c:22
    #1 0x55c4c6 in Compile_Block /source/m3_compile.c:2277
    #2 0x
    #3 0                                                    2207
    #4 0
    ....
    #8 0
    #9 0x                                                   f0fdeb36a)
    ......
    0x555bc18 is located 88 bytes after global variable c_operations

    SUMMARY: AddressSanitizer: global-buffer-overflow
```

The report indicates that the overflow occurred in c_operations, so I need to add a boundary check before performing pointer arithmetic.

Final Patch

```
1   GetOpInfo(m3opcode_t opcode) {
2       switch (opcode >> 8) {
3       case 0x00:
4   +       if(opcode < sizeof(c_operations))
5               return &c_operations[opcode];
6   +       break;
7       case 0xFC:
8   +       if(opcode & 0xFF < sizeof(c_operations))
9               return &c_operationsFC[opcode & 0xFF];
10  +       break;
11      default:
12          return NULL;
13      }
14  }
```

According to the address sanitizer report, a global overflow occurred at line 22 in the /source/m3_compile.c. Let me view lines 17 to 27 in /source/m3_compile.c.

```
....
IM3OpInfo opinfo = GetOpInfo(opcode);
_throwif(unknownOpcode, opinfo == NULL);
if (opinfo->compiler) {
    (*opinfo->compiler) (o, opcode)
....
```

`opinfo` is an overflow pointer generated by GetOpInfo, I would like to locate its definition.

The variable GetOpInfo is defined at line 5 in the /source/m3_compile.c.

Great, now I want to view lines 5 to 16 in /source/m3_compile.c

```
GetOpInfo(m3opcode_t opcode) {
  switch (opcode >> 8) {
  case 0x00:
    return &c_operations[opcode]; ....
```

Vanilla Agent

```
==35==ERROR: AddressSanitizer: global-buffer-overflow
READ of size 8 at 0x55bc18 thread T0
    #0 0x55969b in Compile_BlockStat /source/m3_compile.c:22
    #1 0x55c4c6 in Compile_Block /source/m3_compile.c:2277
    #2 0x55cbc3 in Compile_If /source/m3_compile.c:1648
    #3 0x5596ec in Compile_BlockStatement /source/m3_compile.c:2207
    #4 0x55ca29 in Parse_InitExpr /source/m3_parse.c:282
    ......
    #8 0x55d715 in LLVMFuzzerTestOneInput /app_fuzz/fuzzer.c:30
    #9 0x552e14 in fuzzer::Fuzzer::ExecuteCallback (BuildId: f0fdeb36a)
    ......
    0x555bc18 is located 88 bytes after global variable c_operations

    SUMMARY: AddressSanitizer: global-buffer-overflow
```

According to the address sanitizer report, a global overflow occurred at line 22 in the /source/m3_compile.c. So I want to view lines 21 to 23 in /source/m3_compile.c.

Vanilla Agent

```
==35==ERROR: AddressSanitizer: global-buffer-overflow
READ of size 8 at 0x55bc18 thread T0
    #0 0x55969b in Compile_BlockStat /source/m3_compile.c:22
    #1 0x55c4c6 in Compile_Block /source/m3_compile.c:2277
    #2 0x55cbc3 in Compile_If /source/m3_compile.c:1648
    #3 0x5596ec in Compile_BlockStatement /source/m3_compile.c:2207
    #4 0x55ca29 in Parse_InitExpr /source/m3_parse.c:282
    ......
    #8 0x55d715 in LLVMFuzzerTestOneInput /app_fuzz/fuzzer.c:30
    #9 0x552e14 in fuzzer::Fuzzer::ExecuteCallback (BuildId: f0fdeb36a)
    ......
    0x555bc18 is located 88 bytes after global variable c_operations

    SUMMARY: AddressSanitizer: global-buffer-overflow
```

According to the address sanitizer report, a global overflow occurred at line 22 in the /source/m3_compile.c. So I want to view lines 21 to 23 in /source/m3_compile.c.

```
_throwif(unknownOpcode, opinfo == NULL);
if (opinfo->compiler) {
  (*opinfo->compiler) (o, opcode)
```

Vanilla Agent

```
==35==ERROR: AddressSanitizer: global-buffer-overflow
READ of size 8 at 0x55bc18 thread T0
    #0 0x55969b in Compile_BlockStat /source/m3_compile.c:22
    #1 0x55c4c6 in Compile_Block /source/m3_compile.c:2277
    #2 0x55cbc3 in Compile_If /source/m3_compile.c:1648
    #3 0x5596ec in Compile_BlockStatement /source/m3_compile.c:2207
    #4 0x55ca29 in Parse_InitExpr /source/m3_parse.c:282
    ......
    #8 0x55d715 in LLVMFuzzerTestOneInput /app_fuzz/fuzzer.c:30
    #9 0x552e14 in fuzzer::Fuzzer::ExecuteCallback (BuildId: f0fdeb36a)
    ......
    0x555bc18 is located 88 bytes after global variable c_operations

    SUMMARY: AddressSanitizer: global-buffer-overflow
```

According to the address sanitizer report, a global overflow occurred at line 22 in the /source/m3_compile.c. So I want to view lines 21 to 23 in /source/m3_compile.c.

```
_throwif(unknownOpcode, opinfo == NULL);
if (opinfo->compiler) {
  (*opinfo->compiler) (o, opcode)
```

It appears that the overflow occurs in opinfo. I would like to locate the definition of opinfo.

Vanilla Agent

```
==35==ERROR: AddressSanitizer: global-buffer-overflow
READ of size 8 at 0x55bc18 thread T0
    #0 0x55969b in Compile_BlockStat /source/m3_compile.c:22
    #1 0x55c4c6 in Compile_Block /source/m3_compile.c:2277
    #2 0x55cbc3 in Compile_If /source/m3_compile.c:1648
    #3 0x5596ec in Compile_BlockStatement /source/m3_compile.c:2207
    #4 0x55ca29 in Parse_InitExpr /source/m3_parse.c:282
    ......
    #8 0x55d715 in LLVMFuzzerTestOneInput /app_fuzz/fuzzer.c:30
    #9 0x552e14 in fuzzer::Fuzzer::ExecuteCallback (BuildId: f0fdeb36a)
    ......
    0x555bc18 is located 88 bytes after global variable c_operations

    SUMMARY: AddressSanitizer: global-buffer-overflow
```

According to the address sanitizer report, a global overflow occurred at line 22 in the /source/m3_compile.c. So I want to view lines 21 to 23 in /source/m3_compile.c.

```
_throwif(unknownOpcode, opinfo == NULL);
if (opinfo->compiler) {
  (*opinfo->compiler) (o, opcode)
```

It appears that the overflow occurs in opinfo. I would like to locate the definition of opinfo.

The variable opinfo is defined at line 20 in the /source/m3_compile.c.

Vanilla Agent

```
==35==ERROR: AddressSanitizer: global-buffer-overflow
READ of size 8 at 0x55bc18 thread T0
    #0 0x55969b in Compile_BlockStat /source/m3_compile.c:22
    #1 0x55c4c6 in Compile_Block /source/m3_compile.c:2277
    #2 0x55cbc3 in Compile_If /source/m3_compile.c:1648
    #3 0x5596ec in Compile_BlockStatement /source/m3_compile.c:2207
    #4 0x55ca29 in Parse_InitExpr /source/m3_parse.c:282
    ......
    #8 0x55d715 in LLVMFuzzerTestOneInput /app_fuzz/fuzzer.c:30
    #9 0x552e14 in fuzzer::Fuzzer::ExecuteCallback (BuildId: f0fdeb36a)
    ......
    0x555bc18 is located 88 bytes after global variable c_operations

    SUMMARY: AddressSanitizer: global-buffer-overflow
```

According to the address sanitizer report, a global overflow occurred at line 22 in the /source/m3_compile.c. So I want to view lines 21 to 23 in /source/m3_compile.c.

```
_throwif(unknownOpcode, opinfo == NULL);
if (opinfo->compiler) {
  (*opinfo->compiler) (o, opcode)
```

It appears that the overflow occurs in opinfo. I would like to locate the definition of opinfo.

The variable opinfo is defined at line 20 in the /source/m3_compile.c.

Great, now I want to view lines 17 to 20 in /source/m3_compile.c

Vanilla Agent

```
==35==ERROR: AddressSanitizer: global-buffer-overflow
READ of size 8 at 0x55bc18 thread T0
    #0 0x55969b in Compile_BlockStat /source/m3_compile.c:22
    #1 0x55c4c6 in Compile_Block /source/m3_compile.c:2277
    #2 0x55cbc3 in Compile_If /source/m3_compile.c:1648
    #3 0x5596ec in Compile_BlockStatement /source/m3_compile.c:2207
    #4 0x55ca29 in Parse_InitExpr /source/m3_parse.c:282
    ......
    #8 0x55d715 in LLVMFuzzerTestOneInput /app_fuzz/fuzzer.c:30
    #9 0x552e14 in fuzzer::Fuzzer::ExecuteCallback (BuildId: f0fdeb36a)
    ......
    0x555bc18 is located 88 bytes after global variable c_operations

    SUMMARY: AddressSanitizer: global-buffer-overflow
```

According to the address sanitizer report, a global overflow occurred at line 22 in the /source/m3_compile.c. So I want to view lines 21 to 23 in /source/m3_compile.c.

```
_throwif(unknownOpcode, opinfo == NULL);
if (opinfo->compiler) {
  (*opinfo->compiler) (o, opcode)
```

It appears that the overflow occurs in opinfo. I would like to locate the definition of opinfo.

The variable opinfo is defined at line 20 in the /source/m3_compile.c.

Great, now I want to view lines 17 to 20 in /source/m3_compile.c

```
m3opcode_t opcode;
read_opcode(&opcode,&o);
IM3OpInfo opinfo = GetOpInfo(opcode);
```

Vanilla Agent

```
==35==ERROR: AddressSanitizer: global-buffer-overflow
READ of size 8 at 0x55bc18 thread T0
    #0 0x55969b in Compile_BlockStat /source/m3_compile.c:22
    #1
    #2
    #3                                                          7
    #4
    ..
    #8
    #9                                                  b36a)
    ......
    0x555bc18 is located 88 bytes after global variable c_operations

    SUMMARY: AddressSanitizer: global-buffer-overflow
```

To address the global overflow vulnerability in the opinfo structure, you can introduce bounds checking before dereferencing.The patch ensure that opinfo points to a valid memory location

According to the address sanitizer report, a global overflow occurred at line 22 in the /source/m3_compile.c. So I want to view lines 21 to 23 in /source/m3_compile.c.

```
_throwif(unknownOpcode, opinfo == NULL);
if (opinfo->compiler) {
  (*opinfo->compiler) (o, opcode)
```

It appears that the overflow occurs in opinfo. I would like to locate the definition of opinfo.

The variable opinfo is defined at line 20 in the /source/m3_compile.c.

Great, now I want to view lines 17 to 20 in /source/m3_compile.c

```
m3opcode_t opcode;
read_opcode(&opcode,&o);
IM3OpInfo opinfo = GetOpInfo(opcode);
```

Vanilla Agent

```
==35==ERROR: AddressSanitizer: global-buffer-overflow
READ of size 8 at 0x55bc18 thread T0
    #0 0x55969b in Compile_BlockStat /source/m3_compile.c:22
    #1
    #2
    #3                                                          7
    #4
    ..
    #8
    #9                                                    b36a)
    ......
    0x555bc18 is located 88 bytes after global variable c_operations

    SUMMARY: AddressSanitizer: global-buffer-overflow
```

To address the global overflow vulnerability in the opinfo structure, you can introduce bounds checking before dereferencing.The patch ensure that opinfo points to a valid memory location

Final Patch

```
1  M3Result Compile_BlockStat(IM3Compilation o) {
2      m3opcode_t opcode;
3      Read_opcode (&opcode, &o);
4  +   if (opcode < 0 || opcode > MAX_OPCODE)
5  +       return unknwonOpcode;
6      IM3OpInfo opinfo = GetOpInfo(opcode);
7      _throwif(unknownOpcode, opinfo == NULL);
8      if (opinfo->compiler) {
```

According to the address sanitizer report, a global overflow occurred at line 22 in the /source/m3_compile.c. So I want to view lines 21 to 23 in /source/m3_compile.c.

```
_throwif(unknownOpcode, opinfo == NULL);
if (opinfo->compiler) {
    (*opinfo->compiler) (o, opcode)
```

It appears that the overflow occurs in opinfo. I would like to locate the definition of opinfo.

The variable opinfo is defined at line 20 in the /source/m3_compile.c.

Great, now I want to view lines 17 to 20 in /source/m3_compile.c

m3opcode_t opcode;
read_opcode(&opcode,&o);
IM3OpInfo opinfo = GetOpInfo(opcode);

Vanilla Agent

Too small code range

```
==35==ERROR: AddressSanitizer: global-buffer-overflow
READ of size 8 at 0x55bc18 thread T0
    #0 0x55969b in Compile_BlockStat /source/m3_compile.c:22
    #1
    #2
    #3
    #4
    ..
    #8
    #9
    ......
    0x555bc18 is located 88 bytes after global variable c_operations

    SUMMARY: AddressSanitizer: global-buffer-overflow
```

To address the global overflow vulnerability in the opinfo structure, you can introduce bounds checking before dereferencing.The patch ensure that opinfo points to a valid memory location

According to the address sanitizer report, a global overflow occurred at line 22 in the /source/m3_compile.c. So I want to view lines 21 to 23 in /source/m3_compile.c.

```
_throwif(unknownOpcode, opinfo == NULL);
if (opinfo->compiler) {
    (*opinfo->compiler) (o, opcode)
```

It appears that the overflow occurs in opinfo. I would like to locate the definition of opinfo.

The variable opinfo is defined at line 20 in the /source/m3_compile.c.

Great, now I want to view lines 17 to 20 in /source/m3_compile.c

```
m3opcode_t opcode;
read_opcode(&opcode,&o);
IM3OpInfo opinfo = GetOpInfo(opcode);
```

Final Patch

```
1  M3Result Compile_BlockStat(IM3Compilation o) {
2      m3opcode_t opcode;
3      Read_opcode (&opcode, &o);
4  +   if (opcode < 0 || opcode > MAX_OPCODE)
5  +       return unknwonOpcode;
6      IM3OpInfo opinfo = GetOpInfo(opcode);
7      _throwif(unknownOpcode, opinfo == NULL);
8      if (opinfo->compiler) {
```

Vanilla Agent

```
==35==ERROR: AddressSanitizer: global-buffer-overflow
READ of size 8 at 0x55bc18 thread T0
    #0 0x55969b in Compile_BlockStat /source/m3_compile.c:22
    #1
    #2
    #3                                                          7
    #4
    ..
    #8
    #9                                                  b36a)
    ......
    0x555bc18 is located 88 bytes after global variable c_operations

    SUMMARY: AddressSanitizer: global-buffer-overflow
```

To address the global overflow vulnerability in the opinfo structure, you can introduce bounds checking before dereferencing.The patch ensure that opinfo points to a valid memory location

Final Patch

```
1  M3Result Compile_BlockStat(IM3Compilation o) {
2      m3opcode_t opcode;
3      Read_opcode (&opcode, &o);
4  +   if (opcode < 0 || opcode > MAX_OPCODE)
5  +       return unknwonOpcode;
6      IM3OpInfo opinfo = GetOpInfo(opcode);
7      _throwif(unknownOpcode, opinfo == NULL);
8      if (opinfo->compiler) {
```

Too small code range

According to the address sanitizer report, a global overflow occurred at line 22 in the /source/m3_compile.c. So I want to view lines 21 to 23 in /source/m3_compile.c.

```
_throwif(unknownOpcode, opinfo == NULL);
if (opinfo->compiler) {
    (*opinfo->compiler) (o, opcode)
```

It appears that the overflow occurs in opinfo. I would like to locate the definition of opinfo.

The variable opinfo is defined at line 20 in the /source/m3_compile.c.

Great, now I want to view lines 17 to 20 in /source/m3_compile.c

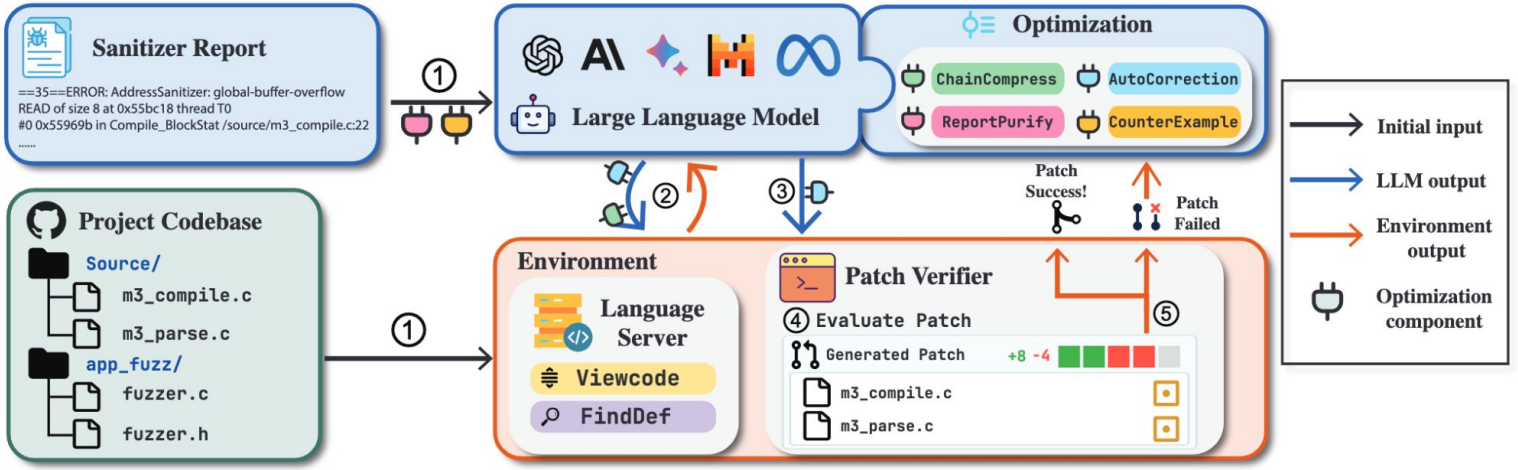
```
m3opcode_t opcode;
read_opcode(&opcode,&o);
IM3OpInfo opinfo = GetOpInfo(opcode);
```

Why not continue exploring related function?

Agenda

- Motivation Example
- PatchAgent Design
- Evaluation and Results

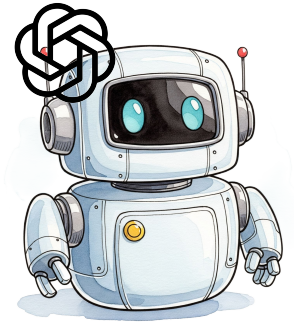
Patch Agent



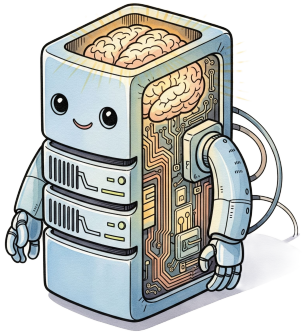
The process begins with the **sanitizer report** and the **project codebase** (1). The LLM retrieves the code context using the **viewcode** and **find_definition** APIs (2) and then **generates a patch** (3). The patch is subsequently validated by the patch verifier (4). If the patch is incorrect, the agent will **refine the patch or gather additional context** (5), iterating until a correct patch is generated or the budget is exhausted.

Auto Correction

Common Tool Call Process



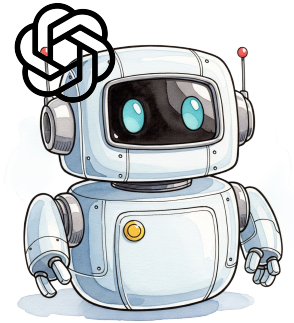
LLM



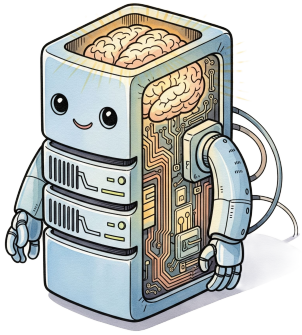
Environment
(Language Server)

Auto Correction

Common Tool Call Process



```
{
    “action”: “viewcode”,
    “path”: “src/m3_compile.c”,
    “start_line”: 21,
    “end_line”: 23
}
```

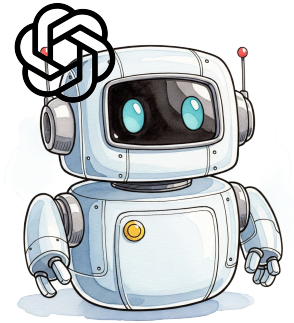


LLM

Environment
(Language Server)

Auto Correction

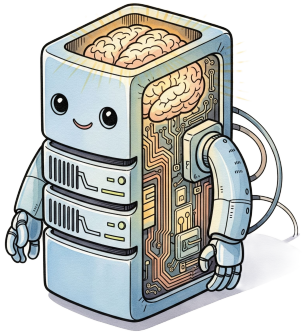
Common Tool Call Process



LLM

```
{
    “action”: “viewcode”,
    “path”: “src/m3_compile.c”,
    “start_line”: 21,
    “end_line”: 23
}
```

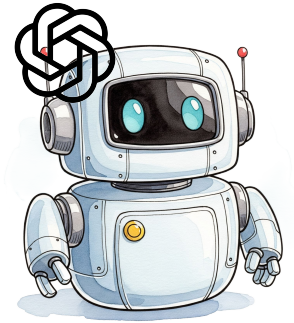
```
21| _throwif(unknownOpcode, opinfo == NULL);
22| if (opinfo->compiler) {
23|     (*opinfo->compiler) (o, opcode)
```



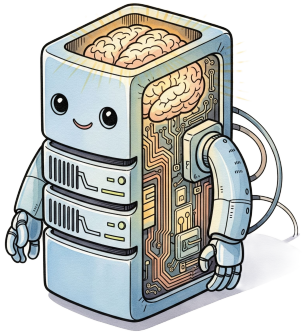
Environment
(Language Server)

Auto Correction

Tool Call Process (with Auto Correction)



```
{
    "action": "viewcode",
    "path": "src/m3_compile.c",
    "start_line": 21,
    "end_line": 23
}
```

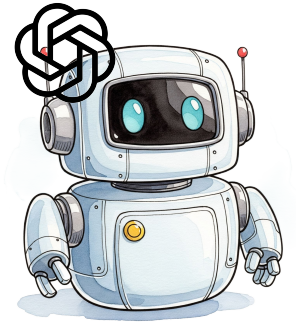


LLM

Environment
(Language Server)

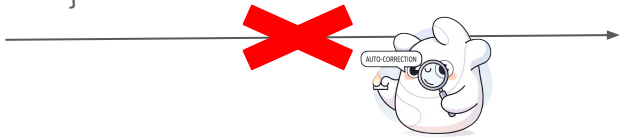
Auto Correction

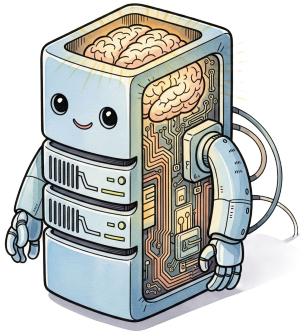
Tool Call Process (with Auto Correction)



LLM

```
{
    "action": "viewcode",
    "path": "src/m3_compile.c",
    "start_line": 21,
    "end_line": 23
}
```

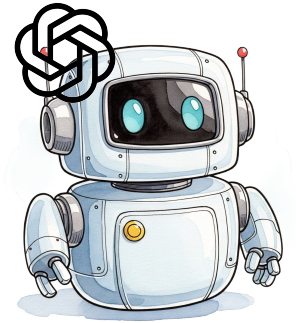




Environment (Language Server)

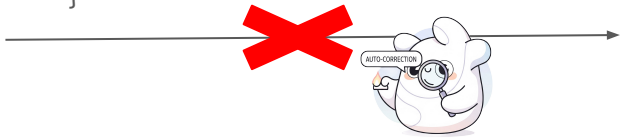
Auto Correction

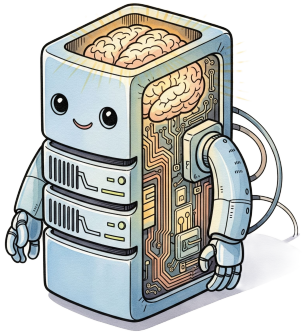
Tool Call Process (with Auto Correction)



LLM

```
{
    "action": "viewcode",
    "path": "src/m3_compile.c",
    "start_line": 17,
    "end_line": 27
}
```

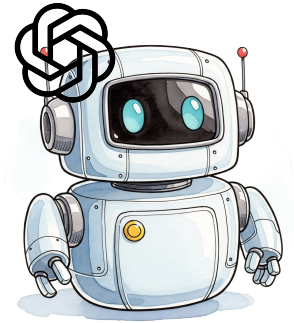




Environment (Language Server)

Auto Correction

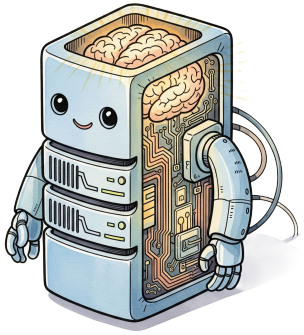
Tool Call Process (with Auto Correction)



LLM

```
{
    "action": "viewcode",
    "path": "src/m3_compile.c",
    "start_line": 17,
    "end_line": 27
}
```

```
17| Compile_BlockStat(IM3Compilation o) {
18|     m3opcode_t opcode;
19|     Read_opcode (&opcode, &o);
20|     IM3OpInfo opinfo = GetOpInfo(opcode);
21|     _throwif(unknownOpcode, opinfo == NULL);
22|     if (opinfo->compiler) {
23|         (*opinfo->compiler) (o, opcode)
24|     } else {
25|         Compile_Operator (o, opcode);
26|     }
27| }
```



Environment (Language Server)

Auto Correction

Code Viewing Enhancement

Expanded code range visibility and corrected file path references

- Resolved issues with narrow code display windows that limited readability
- Fixed incorrect file path typos that were breaking navigation links

Symbol Definition Resolution

Enhanced alias symbol detection and lookup

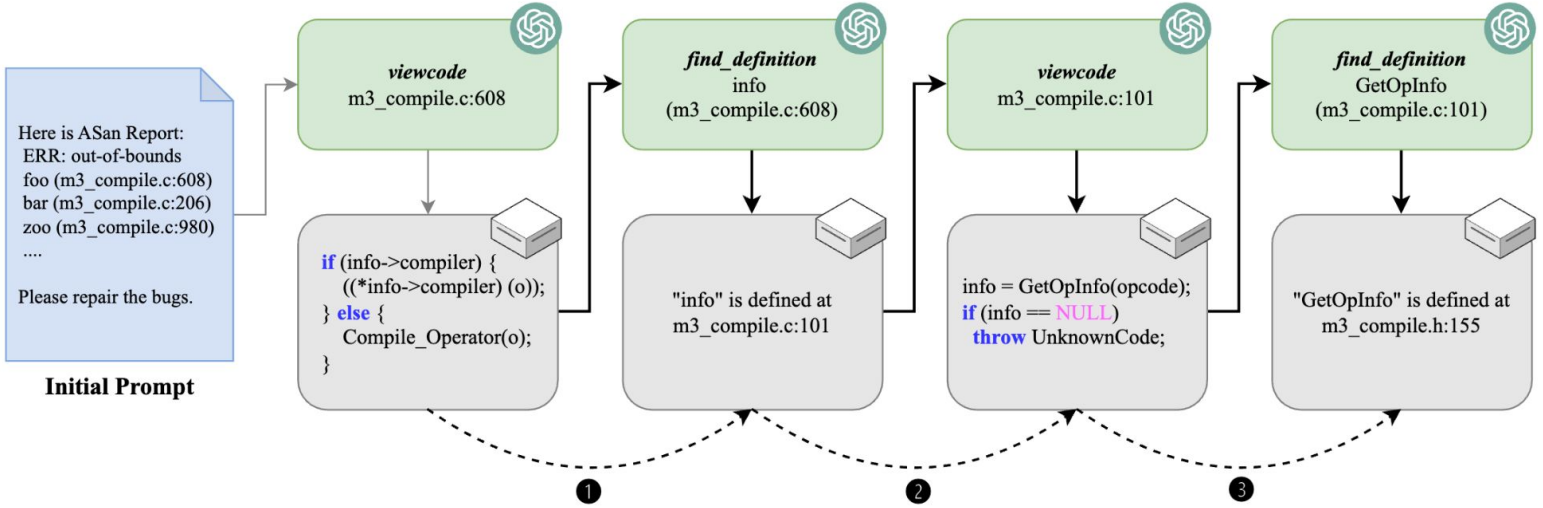
- Improved the "Find Definition" feature to properly handle aliased symbols
- Resolved cases where symbol aliases were not being recognized or linked correctly

Validation System Updates

Standardized patch format processing

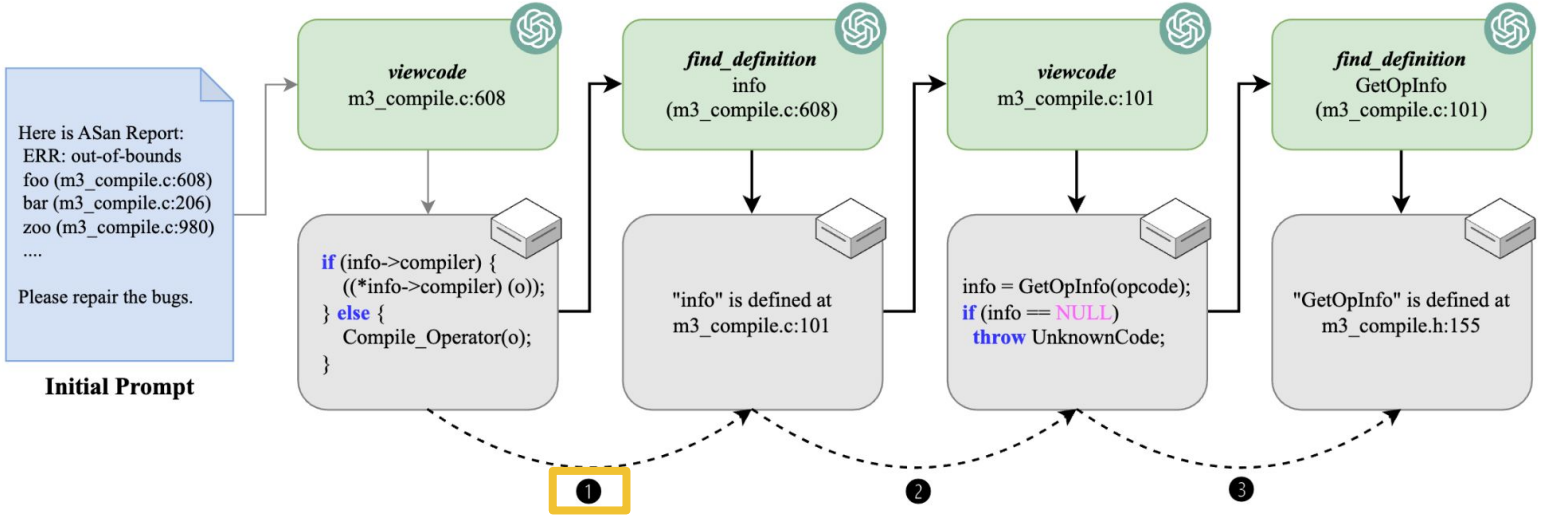
- Fixed validation errors related to inconsistent patch formats
- Ensured all patch submissions now follow proper formatting standards

Chain Compression



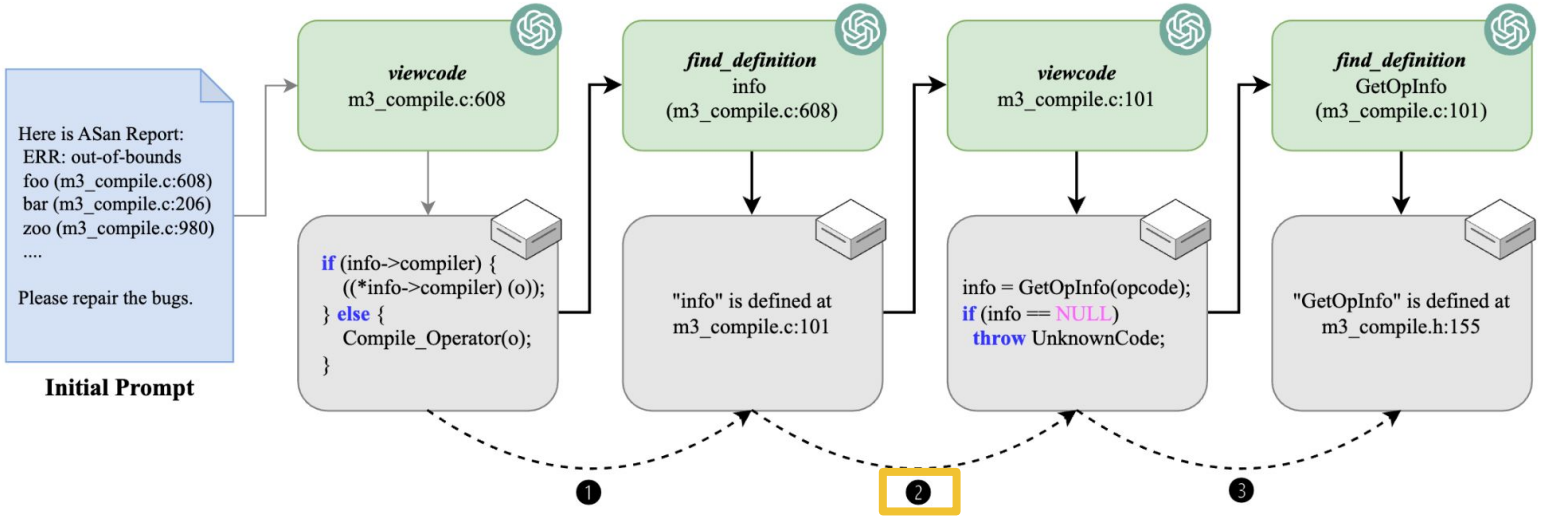
The LLM takes the initial prompt as input and starts interacting with the language server. The black bold arrows illustrate the interaction without chain compression, while the black dashed arrows represent the **compressed interaction process**. The original interaction **chain of length four** was compressed into a **single interaction**.

Chain Compression



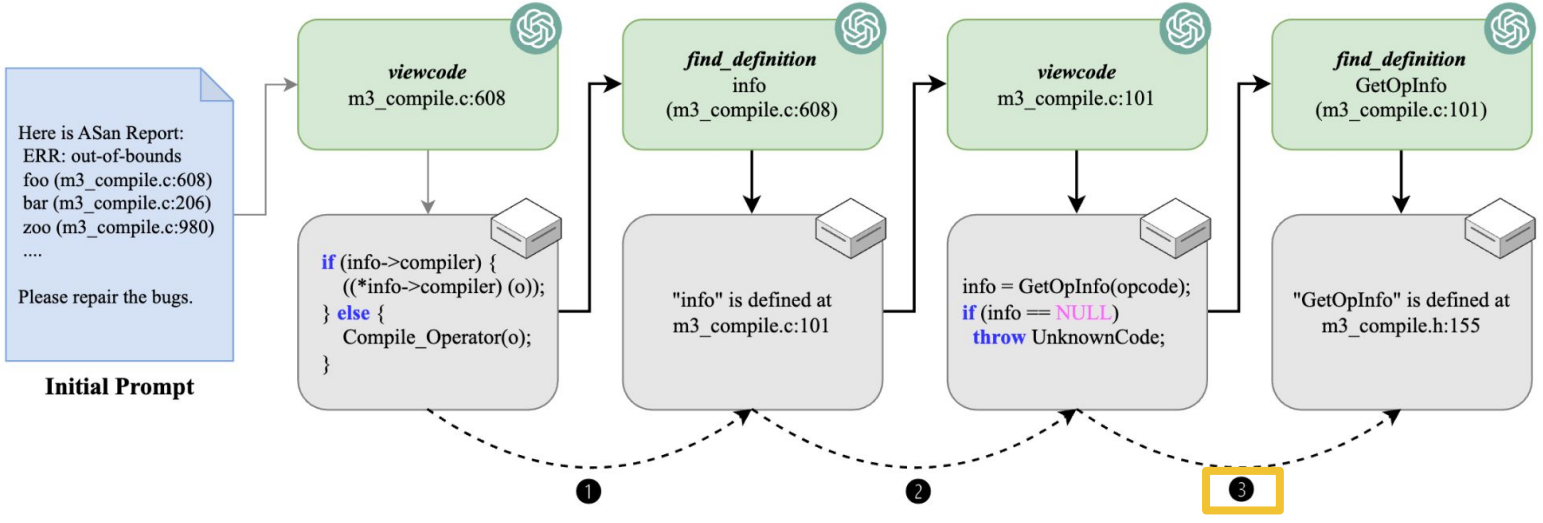
1. After the LLM sends a viewcode action, the mechanism determines that the crash is caused by the dereference of info and the line where info located **appears in both the viewed code snippet and the sanitizer report.** This indicates that it is a valuable symbol to explore.

Chain Compression



2. Using only the find_definition action to locate the definition of info is **insufficient to reveal its complete information.** Therefore, the mechanism first generates another viewcode action to obtain the definition code snippet of info.

Chain Compression



3. Then, it identifies that **the variable relies on another symbol**, GetOpInfo, and **recursively** finds its definition location.

Agenda

- Motivation Example
- PatchAgent Design
- Evaluation and Results

Dataset Overview

We created a dataset comprising 178 cases sourced from OSS-Fuzz , Huntr and ExtractFix on **9 distinct bug types**: stack overflow, heap overflow, integer overflow, use-after-free, double free, global overflow, divide by zero, invalid free, and null dereference.

Project	Lang	Source	LoC	#Vulns	#Test
assimp	C++	OSS-Fuzz	347.0K	3	474
c-blosc	C	OSS-Fuzz	88.8K	2	1643
c-blosc2	C++	OSS-Fuzz	117.1K	7	1284
h3	C	OSS-Fuzz	17.2K	1	124
hoextdown	C	OSS-Fuzz	7.1K	1	83
hostap	C	OSS-Fuzz	438.0K	4	19
htslib	C	OSS-Fuzz	66.5K	1	159
hunspell	C++	OSS-Fuzz	83.9K	11	128
irssi	C	OSS-Fuzz	64.4K	3	5
krb5	C	OSS-Fuzz	301.6K	1	125
libplist	C	OSS-Fuzz	12.1K	3	34
libsndfile	C	OSS-Fuzz	56.4K	5	141
libtpms	C	OSS-Fuzz	115.0K	1	6
libxml2	C	OSS-Fuzz	200.4K	10	3272
lz4	C	OSS-Fuzz	18.6K	2	22
md4c	C	OSS-Fuzz	8.0K	4	24
openexr	C++	OSS-Fuzz	227.8K	3	111
sleuthkit	C	OSS-Fuzz	196.2K	5	2
wasm3	C	OSS-Fuzz	22.8K	8	35062
zstd	C	OSS-Fuzz	93.4K	5	28
gpac	C	Huntr	743.7K	32	711
libmobi	C	Huntr	19.1K	5	12
mruby	C	Huntr	62.2K	10	61
radare2	C	Huntr	841.3K	20	858
yasm	C	Huntr	132.1K	3	44
binutils	C	ExtractFix	666.8K	2	20
coreutils	C	ExtractFix	86.1K	4	573
jasper	C	ExtractFix	44.7K	2	16
libjpeg	C	ExtractFix	46.9K	4	530
libtiff	C	ExtractFix	85.9K	11	74
libxml2	C	ExtractFix	200.4K	5	3272

Effectiveness Evaluation

Model	Temporal Error	Spatial Error	Null Dereference	Numeric Error	Total
GPT-4o	13/23 (56.52%)	96/125 (76.80%)	23/23 (100.00%)	7/7 (100.00%)	139/178 (78.09%)
GPT-4 Turbo	11/23 (47.83%)	87/125 (69.60%)	21/23 (91.30%)	7/7 (100.00%)	126/178 (70.79%)
Claude-3 Opus	14/23 (60.87%)	108/125 (86.40%)	22/23 (95.65%)	7/7 (100.00%)	151/178 (84.83%)
Claude-3 Sonnet	8/23 (34.78%)	77/125 (61.60%)	17/23 (73.91%)	6/7 (85.71%)	108/178 (60.67%)
Claude-3 Haiku	9/23 (39.13%)	93/125 (74.40%)	19/23 (82.61%)	7/7 (100.00%)	128/178 (71.91%)
Union	20/23 (86.96%)	114/125 (91.20%)	23/23 (100.00%)	7/7 (100.00%)	164/178 (92.13%)

This table compares the effectiveness of **PatchAgent** when utilizing different LLMs to repair vulnerabilities. The **Union** row represents the combined results of **PatchAgent** across all models, demonstrating the overall improvement in repair accuracy achieved through the collaborative use of multiple models.

Github Pull Requests

Fix vuln OSV-2024-947 #1699

Merged **seladb** merged 8 commits into `seladb:dev` from `oss-patch:patch-OSV-2024-947` on Mar 29

Fix vuln OSV-2023-77 #5210

Merged **lrknox** merged 7 commits into `HDFGroup:develop` from `oss-patch:patch-OSV-2023-77` on Jan 15

Fix buffer overflow in MD3Loader #5763

Merged **kimkulling** merged 2 commits into `assimp:master` from `cla7aye15I4nd:fix-5616` on Sep 10, 2024

kex: fix null pointer dereference in `diffie_hellman_sha_algo()` #1508

Merged **willco007** merged 1 commit into `libssh2:master` from `oss-patch:patch-crash-c7c664759c840b3f1c438b7232f713b756ed640f` on Feb 28

Fix Heap-buffer-overflow in __parse_options #1678

Merged **tigercosmos** merged 5 commits into `seladb:dev` from `oss-patch:patch-crash-7d18f37e1f05e0ff4aa4dfa2f67dd738340ad9cf` on Feb 2

Fix vuln OSV-2024-390 #5201

Merged **lrknox** merged 1 commit into `HDFGroup:develop` from `oss-patch:patch-OSV-2024-390` on Jan 3

Fix vuln OSV-2024-384 #1061

Merged **rurban** merged 1 commit into `LibreDWG:master` from `oss-patch:patch-OSV-2024-384` on Dec 24, 2024

Fix vuln OSV-2024-343 #1680

Merged **seladb** merged 2 commits into `seladb:dev` from `oss-patch:patch-OSV-2024-343` on Jan 11

LIBSSH2

HDF5

LibreDWG

PcapPlusPlus

Conclusion

- We propose a novel LLM-based program repair agent that leverages a language server and patch verifier to analyze programs, generate patches, and validate them.
- We introduce **four interaction optimizations** to enhance the repair performance of PatchAgent. An ablation study demonstrates their effectiveness in improving repair performance.
- We evaluate our prototype and provide in-depth analysis, demonstrating the effective and efficient of PatchAgent, including on vulnerabilities that LLMs have never encountered before.
- PatchAgent has made an impact in the real world.

Thanks

Home Page

Github (star it!)