Interprocedural Static Analysis Framework for Defects Detection in Binaries

Mariam Arutunian, Vahagn Vardanyan, Hayk Aslanyan, Grigor Keropyan, Seda Movsisyan

Laboratory of system programming YSU

2018

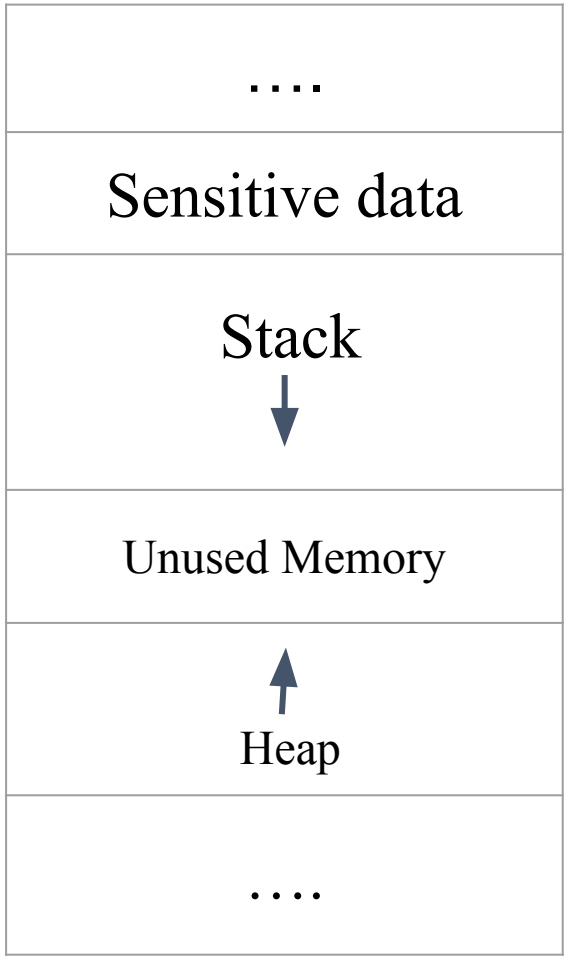
Program Defects examples

1. Format String
2. Buffer Overflow
3. Use After Free (Double Free)
4. Command Injection

Format String

```
int main() {
...
gets(userName)
printf (userName);
...
}
```

what if userName =”%s%s%s%s%s%s%s%s%s%s%s%s”



Buffer Overflow

```
int main(int argc, char **argv){
char Filename[256];
…
strcpy(Filename, argv[1]);

printf("Searching file: %s\n", Filename);
...
}
```

Use after free, Double free

```
char* ptr = (char*)malloc
(size);
   ...
   if (err) {
      abrt = 1;
      free(ptr); }
   ...
   if (abrt) {
      logError("operation
aborted before commit", ptr); }
```

```
char* ptr = (char*)malloc
(size);
   ...
   if (abrt) {
     free(ptr); }
   ...
   free(ptr);
```

Protect your software: Static Analysis

- Static code analysis is one of the common approaches for detecting defects. This approach is a program analyzing method that is performed by examining the code without executing the program.

- Through a complete analysis of syntax, semantics, control and data flow, static code analysis can find errors that are difficult or impossible to find in the testing phase of programs.

Binary file analysis is important

- The source code of the program is not always available, thus the use of source code analyzers becomes impossible.

- Not all compiler optimizations are safe, they can lead to errors in binary code that don’t exist in the source code.

- Analysis of a binary can provide more accurate information than a source-level analysis, because, for many programming languages, certain behaviors are left unspecified by the semantics

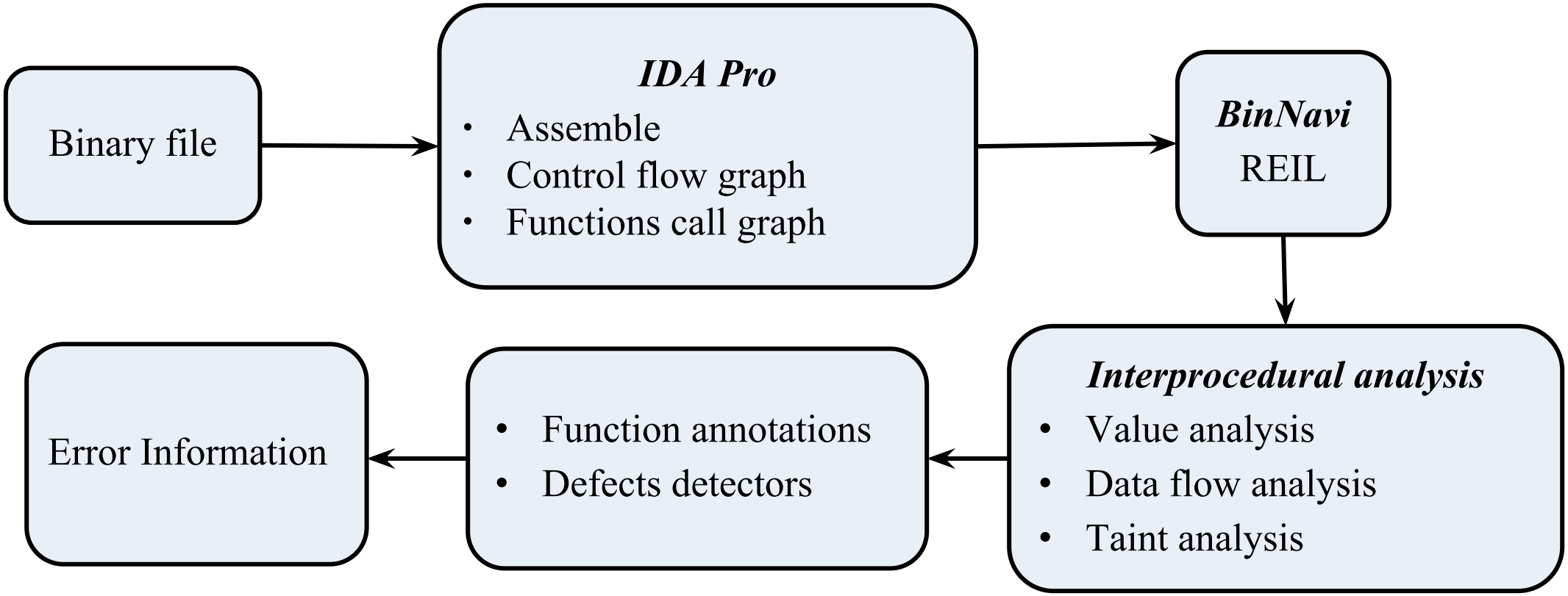
Our research areas and final results

We’ve done wide researches in almost all areas of software security

As a result we have developed:

- Framework for static analysis of binary code that is independent from the target architecture, scalable and easily extensible
- Methods of data flow analysis, value analysis, taint analysis for the binary code
- Methods for finding defects of Use after free, Double free, Format string, Buffer overflow and Command injection

Framework Architecture



REIL representation

- Platform independent
- 17 simple instructions (and, add, ldm, stm…)
- No side effects

Intraprocedural analysis

- Value analysis
- Taint analysis
- Dynamic memory analysis (tracing memory allocation and deallocation)
- Constructing DEF-USE and USE-DEF chains

Value analysis

At each program point computes all possible values that the given register or memory address can have

- Memory simulation in the stack
- Memory simulation in the heap
- Static memory and global variables

Taint analysis

```
...
char * s;
gets(s);
char * p = s;
printf(p);
...
```

```
...
mov     rax, [rbp+var_10]
mov     rdi, rax
call    _gets
mov     rax, [rbp+var_10]
mov     [rbp+format], rax
mov     rax, [rbp+format]
mov     rdi, rax        ; rdi is argument of printf
mov     eax, 0
call    _printf
...
```

Taint analysis

```
...
char * s;
gets(s);
char * p = s;
printf(p);
...
```

```
...
mov     rax, [rbp+var_10]
mov     rdi, rax
call    _gets
mov     rax, [rbp+var_10]
mov     [rbp+format], rax
mov     rax, [rbp+format]
mov     rdi, rax        ; rdi is argument of printf
mov     eax, 0
call    _printf
...
```

Taint analysis

```
...
char * s;
gets(s);
char * p = s;
printf(p);
...
```

```
...
mov     rax, [rbp+var_10]
mov     rdi, rax
call    _gets
mov     rax, [rbp+var_10]
mov     [rbp+format], rax
mov     rax, [rbp+format]
mov     rdi, rax        ; rdi is argument of printf
mov     eax, 0
call    _printf
...
```

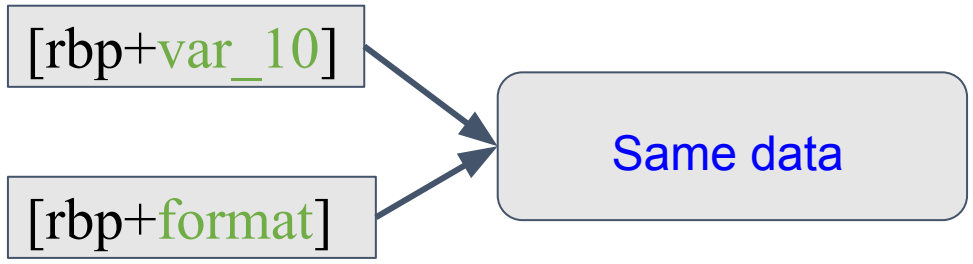
Taint analysis

```
...
char * s;
gets(s);
char * p = s;
printf(p);
...
```

```
...
mov     rax, [rbp+var_10]
mov     rdi, rax
call    _gets
mov     rax, [rbp+var_10]
mov     [rbp+format], rax
mov     rax, [rbp+format]
mov     rdi, rax        ; rdi is argument of printf
mov     eax, 0
call    _printf
...
```

Taint analysis

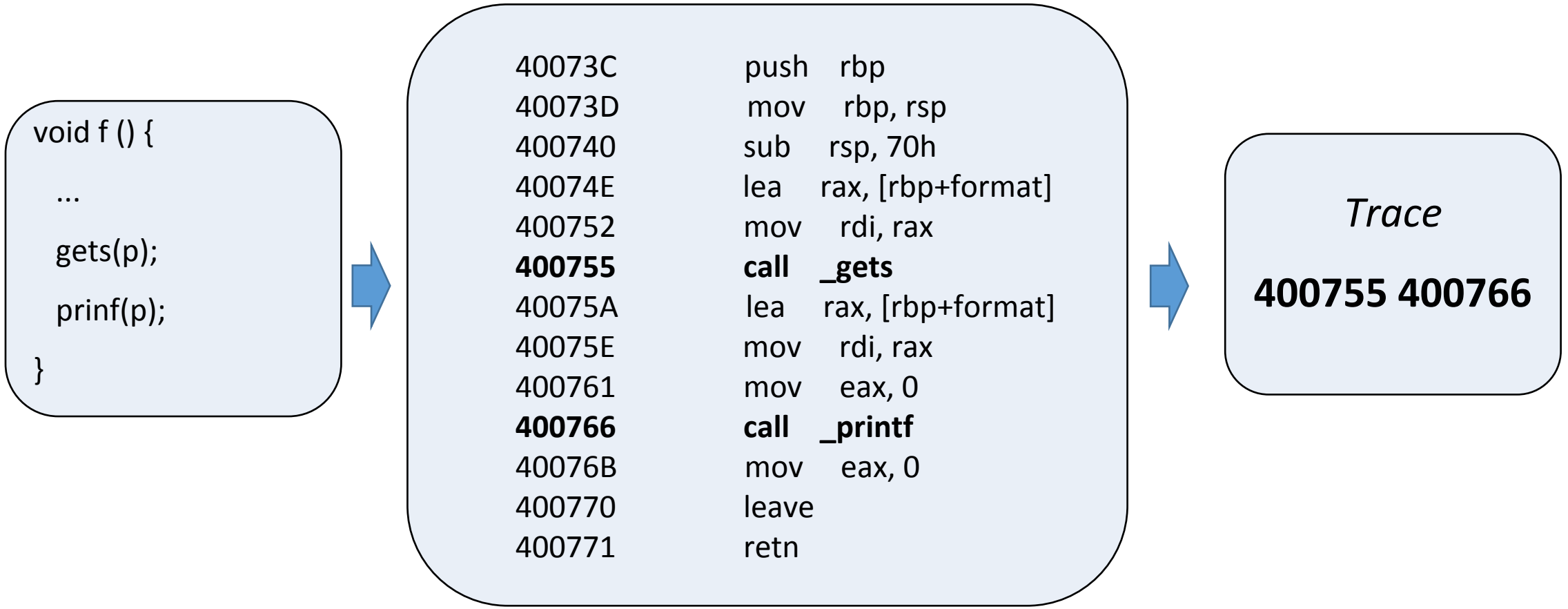
```
...
mov     rax, [rbp+var_10]
mov     rdi, rax
call    _gets
mov     rax, [rbp+var_10]
mov     [rbp+format], rax
mov     rax, [rbp+format]
mov     rdi, rax        ; rdi is argument of printf
mov     eax, 0
call    _printf
...
```



Interprocedural analysis

- Defects detectors
  - Use after free
  - Double free
  - Format string
  - Buffer overflow
  - Command injection

Defect detectors



Results (Use-After-Free, Double-Free)

Project	Architecture	Size	Analysis time	Number of found UAF and DF	Percentage of correct handling
accel_pppd 1.10.0	x86	232 KB	3m	4	100%
gnome-nettool 3.8.1	x86	336 KB	1m 40s	1	100%
slpd 1.2.1	x86	128 KB	50s	1	100%
libssh 0.5.2	x86	632 KB	3m	20	40%
jasper 1.900.1	x86	980 KB	11m 41s	8	13%
libtiff 4.0.3	x86	1 KB	2m 58s	3	67%
accel_pppd 1.10.0	x64	244 KB	4m 1s	1	100%
gnome-nettool 3.8.1	x64	436 KB	1m 50s	3	67%
libssh 0.5.2	x64	324 KB	3m 50s	23	26%
slpd 1.2.1	x64	128 KB	3m 1s	4	25%
pbs_server 2.4.8	x64	1.6 KB	11m 48s	1	100%

Results (comparison with GUEB)

Project	Working time of GUEB	Found UAF and DF with GUEB	Percentage of right handling of GUEB	Found UAF and DF	Percentage of right handling
gnome-nettool 3.8.1	16 s	4	25%	1	100%
gifcolor 5.1.2	21 s	15	6.7%	1	100%
jasper 1.900.1	4m 23s	255	1.2%	8	12.5%
accel-pppd 1.10.0	5m 5s	35	11.4%	4	100%

Results (Buffer overflow, Format String, Code injection)

Project	Architecture	Size	Analysis time	Number of found defects	Percentage of correct handling
dba 2.4.1	x86	312 KB	1m 40s	12	50%
httpd 0.5.0	x86	6.4 MB	6m 51s	22	90.9%
iwconfig 26	x86	44 KB	24 s	3	100%
mkfs 1.1.12	x86	56 KB	25 s	9	100%
pswdb 2.4.1	x86	300 KB	55 s	9	33%
hsolinkcontrol 1.0.118	x86	28 KB	2 s	22	100%
alsa_in 1.1.3	x86	28 KB	8 s	2	100%
htget 0.1	x64	28 KB	11 s	12	100%
mkfs 1.1.12	x64	56 KB	19 s	7	100%
libtorque 2.0.0	x64	892 KB	57 s	12	100%
alsa_out 1.1.3	x64	28 KB	10 s	2	100%
pbs_server 2.4.8	x64	320 KB	3m 20s	4	75%

Results (comparison with Loongchecker)

Project	Size	LoongChecker	Percentage of correct handling of LoongChecker	Number of found defects	Percentage of correct handling
Serenity.exe	19.6 MB	8	12.5%	2	50%
FoxPlayer.exe	33 MB	27	4%	2	100%

Thanks for attention!