PtrSplit:Supporting general pointers in automatic program partitioning

Shen Liu Gang Tan Trent Jaeger

Abstract: Partitioning a security-sensitive application into least-privileged components and putting each into a separate protection domain have long been a sought-after goal of security practitioners and researchers. However, a stumbling block to automatically partitioning C/C++ applications is the presence of general pointers in these applications. In this paper, we propose a set of techniques for supporting general pointers in automatic program partitioning. The resulting system, called PtrSplit, can automatically generate executable partitions for C applications that contain pointers.

System Overview

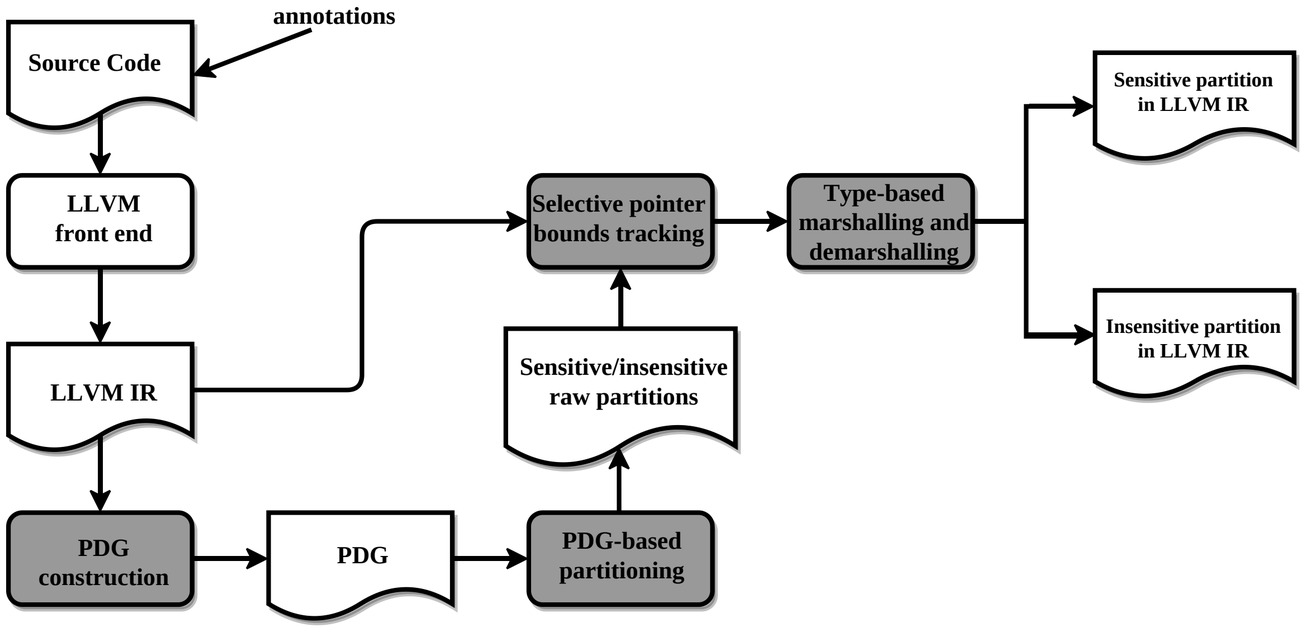

Figure 1. The workflow of our automatic program-partitioning framework (gray components belong to PtrSplit)

Selective Pointer Bounds Tracking (SPBT)

- For marshalling and demarshalling it is necessary to perform only bounds tracking, but not bounds checking. So, SPBT needs to :
  - compute a set of Bounds-Required (BR) pointers given a partitioning of the program
  - instrument the program to track the bounds of those BR pointers
- SPBT instrumentation
  - the SPBT implementation is based on SoftBound, which is an LLVM-based code transformation for enforcing spatial memory safety
  - instrument the program to track the bounds of those BR pointers

The PDG and Partitioning

```
char __attribute__((annotate("sensitive")))
*key;
char* ciphertext;
unsigned int i;

void greeter(char* str){
  printf(str); printf(", welcome!\n");
}

void initkey (int sz) {
  key = (char*)(malloc (sz));
  // init the key randomly; code omitted
  for (i=0; i<sz; i++) key[i]= ...;
}

void encrypt (char *plaintext, int sz) {
  ciphertext = (char*)(malloc (sz));
  for (i=0; i<sz; i++)
    ciphertext[i]=plaintext[i] ^ key[i];
}

void main (){

  char username[20], text[1024];

  printf("Enter username: ");
  scanf("%19s",username);

  greeter(username);

  printf("Enter plaintext: ");
  scanf("%1023s",text);

  initkey(strlen(text));

  encrypt(text, strlen(text));

  printf("Cipher text: ");
  for (i=0; i<strlen(text); i++)
    printf("%x ",ciphertext[i]);
}
```

Figure 2. A toy C program

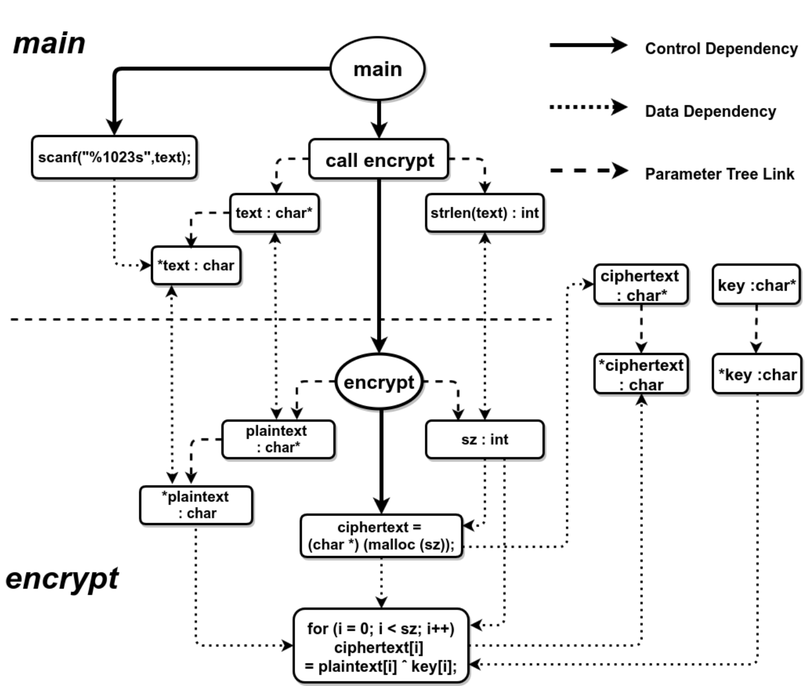

Figure 3: A PDG snippet for our toy program. For clarity, the graph uses a single node for the entire for loop in encrypt; in the implementation, our PDG construction breaks a loop into multiple LLVM IR instructions and has one node for each IR instruction. The graph also omits labels on data-dependence edges.

Type Based Marshalling and Demarshalling

- Since partitions are loaded into separate processes, some function calls are turned into Remote Procedure Calls (RPCs).
- During an RPC, arguments from the caller are marshalled into a data buffer and sent to the callee, which demarshalls the data buffer and recreates the values for the parameters in the callee process.
  - straightforward for values of most data types(integers, arrays of fixed sizes, and structs).
  - difficult for pointers.
- Pointer Processing
  - deep copying.
  - in the sender process, we encode each value v of type t into a list of bytes, and decode it in the receiver process.

Evaluation and Results

Benchmark	SLOC	Sensitive data and type	# of functions/ sensitive functions	Total/BR Pointers	SPBT overhead	Total Overhead
lbm	1,156	LBM_Grid* srcGrid	19/5	695/131	17%	24%
mcf	2,686	struct network_t* net	24/24	n/a	n/a	n/a
libquantum	4,358	struct quantum_reg* lambda	115/3	1690/128	11%	179%
bzip2	8,393	char* progName	100/6	4356/8	3%	5%
sjeng	13,547	char* realholdings	144/5	3415/81	9%	15%
milc	15,042	double[] path_coeff	235/2	5001/0	0%	2%
spihnx3	25,090	char** liveargs	369/3	9491/37	4%	7%
hmmer	35,992	int ser_randseed	538/7	17692/175	6%	27%
h264ref	51,578	int[] FirstMBInSlice	590/5	32212/461	7%	16 %

Table 1: Partitioning results for SPECCPU 2006 benchmarks.

Benchmark	SLOC	Sensitive data	# of functions/ sensitive functions	Total/BR Pointers	SPBT overhead	Total Overhead
chfn	146	password file	5/1	53/3	15%	37%
ping	502	request packet	8/2	149/5	9%	41%
thttpd	21,925	authentication file	145/5	3068/119	12%	14%
wget	61,216	user ftp password input	797/7	14939/371	18%	22%

Table 2: Partitioning results of security-sensitive programs.

Contact

Web: http://www.cse.psu.edu/~gxt29/
Office: 344C IST Building
Email: sxl463@cse.edu.edu