Practical Challenges of Type Checking in Control Flow Integrity

Reza Mirzazade Farkhani; Sajjad Arshad; Saman Jafari
Northeastern University

Problem Statement

- Lack of memory management in unsafe programming languages such as C/C++ has been introducing significant threats to the applications.
- It has been shown that defenses such as ASLR and DEP can be bypassed by motivated attackers[1].
- Control Flow Integrity (CFI) is introduced to enforce the application's control flow to adhere to the statically generated Control Flow Graph (CFG).[2]
- The effectiveness of CFI depends on the ability to construct an accurate CFG.
- Type checking only allows control transfers if the types of the caller and the callee match [3][4].

Type checking

- Type checking, indeed, faces numerous practical challenges for deployment in C and C++ such as **type collision**, **type diversification** and **covariant return type**.
- There are some types such as *void ** that can be matched with any other type.
- Resolving collisions requires global type diversification which complicates dynamic loading of libraries and separate compilation.

```
typedef void( * fptr)(void);
int new_proc = 0;
char p;
void foo1(void) {
    printf("Indirect call of foo1\n");
}
void foo2(void) {
    printf("Indirect call of foo2\n");
}
void spawn_process(void) {
    initial_proc();
}
int initial_proc(void) {
    if (new_proc == 1) shell_execute(p);
}
int shell_execute(char * proc) {
    system(proc);
}
void vulnfunc(char * input) {
    fptr fIndirectCall;
    if (strcmp(input, "1") == 0) fIndirectCall = & foo1;
    else fIndirectCall = & foo2;
    printf(input) fIndirectCall();
}
```

Figure 1. Sample vulnerable source code

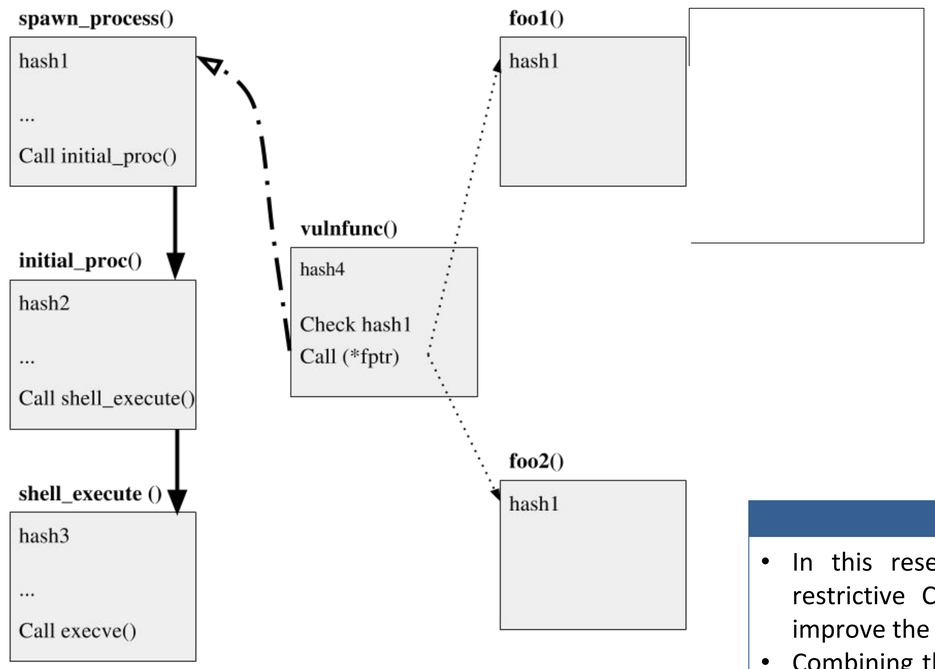

Figure 2. CFG of the program based on type

Results

App	Version	FP	Function	Collision		
				FP	ICS	Function
nginx	1.10.1	84	1299	48	34	121
httpd	2.4.25	248	2800	64	101	483
lighttpd	1.4.45	27	899	10	47	40
exim	4.90	43	968	17	179	319

Table 1. Type collision in popular applications

Chart 1. Type collision with glibc

Conclusions

- In this research, we have studied the implications of creating a restrictive CFI with type matching and propose some solutions to improve the accuracy of the CFG.
- Combining the result of points-to analysis and type checking can result in a more precise CFG.
- By pruning the CFG with type matching, a more precise CFG would be available. This purging decreases the chance of a practical attack on CFI, but it faces numerous practical deployment challenges.

Contact
Reza Mirzazade Farkhani
Northeastern University
Email: reza@iseclab.org

References

1. Shacham, Hovav, et al. "On the effectiveness of address-space randomization." Proceedings of the 11th ACM conference on Computer and communications security. ACM, 2004.
2. Abadi, Martn, et al. "Control-flow integrity." Proceedings of the 12th ACM conference on Computer and communications security. ACM, 2005.
3. van der Veen, Victor, et al. "A tough call: Mitigating advanced code reuse attacks at the binary level." Security and Privacy (SP), 2016 IEEE Symposium on. IEEE, 2016.
4. Tice, Caroline, et al. "Enforcing Forward-Edge Control-Flow Integrity in GCC & LLVM." USENIX Security Symposium. 2014.