Memory Safety And Cyber Security

The SAPPEUR Language as a Systemic Solution to Cybernetic Weaknesses

By Dipl. Ing.(BA) Frank Gerlach (frankgerlach.tai@gmx.de)

The Problem With C and C++ (Basics)

- Types of C-style errors:
  - buffer overflows
  - heap management errors (double frees, accessing freed heap memory)
  - non initialized pointers
  - Undetected integer underflows, overflows
  - Multithreading synchronization errors
- All human software engineers have a bad day and produce programming errors then and now
  - Happened to Unix, Windows, Linux, Yacc and many other systems
- Complex software is sometimes too hard to easily understand and to easily track down memory errors
  - Static Checking tools such as PC Lint, Coverity can provide some mitigation

The Problem With C and C++ (Effects)

- Real-World Effects have been devastating:
  - Maersk Intranet Destruction
  - Sony Pictures Intranet Destruction
  - RSA Security theft of keys for 100s millions of electronic key generators
  - Cyber Bank-robbery National Bank of Bangla Desh (damage $1000 0000 0000)
  - Widespread „crypto ransom“ attacks with „revenue“ in the $billions, hundreds of companies/institutions affected
- 70% of CVE database exploits are due to C and C++ style memory errors
  - This is the low hanging fruit to be reaped

The SAPPEUR Language Approach

- Use a special type system to avoid potentially dangerous C style constructions (e.g. void* pointers, unsafe casts,…)
- Model single- and multithreaded datastructures and algorithms using the type system
  - generate locking code to ensure safe access to multithreaded data
- Generate runtime checking code to turn buffer index errors into a determinstic, debuggable crash
- All pointers are smart pointers
  - Reference counting
- References cannot be turned into pointers

Real World SAPPEUR Programs / Results

- Gauss Web Server
  - Programming Errors were detected as expected
  - Stable Operation over months
  - Language already mature enough to build a useful, non trivial system
- SAPPEUR programs are very efficient in runtime and memory consumption
- SAPPEUR programs are soft-realtime capable
- Empirical results in terms of „exploitable bug“ statistics still outstanding

Conclusion

- Memory safe systems languages can improve cyber security dramatically by eliminating 70% of exploitable bugs
- Efficiency does not need to be impaired as much as garbage-collected languages do
- Other measures such as proper scanners, parsers and strict input validation are still required.

References

- http://sappeur.ddnss.de/
  - http://sappeur.ddnss.de/manual.pdf
- http://gauss.ddnss.de/
- Microsoft on Memory Safety
  - https://msrc-blog.microsoft.com/2019/07/18/we-need-a-safer-systems-programming-language/
- Google on Memory Safety
  - https://www.chromium.org/Home/chromium-security/memory-safety