Update on Q: Exploit Hardening Made Easy

Edward J. Schwartz Thanassis Avgerinos David Brumley

February 21st, 2012

After *Q: Exploit Hardening Made Easy* [2] appeared in Usenix Security 2011, we noticed a discrepancy in our results. Our experiments showed that we could create a ROP payload to call statically linked functions in 80% of programs larger than 20KB, and additionally dynamically linked functions in 80% of programs larger than 100KB. The only difference between the two experiments is that the latter experiment uses the GOT overwriting technique from Roglia et al. [1].

Roglia, et al. estimate that gadgets needed for GOT overwriting are available in 96% of x86 executables larger than 20KB. One would then expect that the two Q experiments would have similar results, rather than the dynamic case requiring a five times increase in code size to achieve a similar success rate. When we investigated this discrepancy, we found a bug in Q that sometimes prevented it from finding the gadgets described by Roglia et al. We have since corrected this bug and rerun the experiments. Figure 1 shows the updated version of Figure 4 in our original paper. There is now a negligible difference between the static and dynamic experiments as expected.

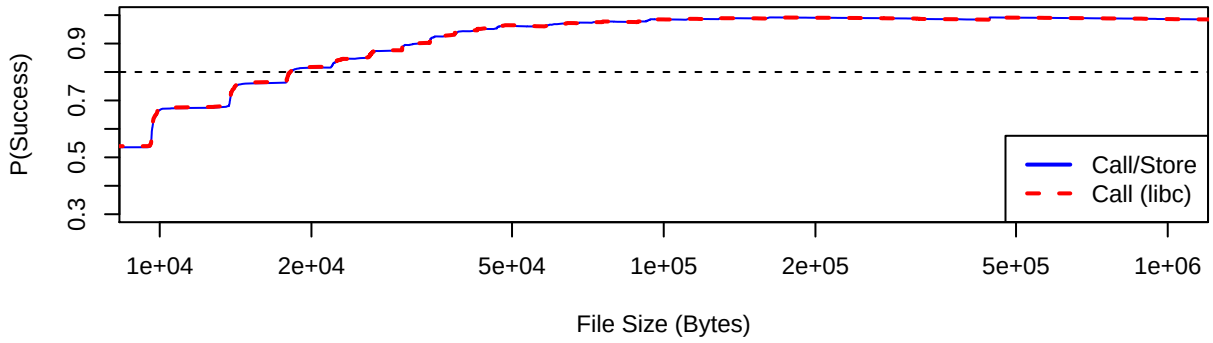

Figure 1: The probability that Q can generate various payload types, shown as a function of source file size. As expected, the probability grows with file size. The percentage is calculated over non position independent executables. Q can call linked functions in 80% of programs that are 20KB or larger, and can call any function in linked shared libraries in 80% of programs that are at least 20KB in size.

References

[1] G. F. Roglia, L. Martignoni, R. Paleari, and D. Bruschi. Surgically returning to randomized lib(c). In *Proceedings of the Annual Computer Security Applications Conference*, pages 60–69, 2009.

[2] E. J. Schwartz, T. Avgerinos, and D. Brumley. Q: Exploit hardening made easy. In *Proceedings of the USENIX Security Symposium*, 2011.