RESEARCH PROPOSAL

JOSEPH HALLETT

A growing problem in the field of information-security is exfiltration. Put simply exfiltration is the removal of secure data from a supposedly secure system. A particularly interesting subproblem is how to stop people removing executable code from a secure system. Current techniques for preventing unauthorized software copying, such as DRM, utilize cryptographic techniques such as digital signatures and encryption to protect programs. Others utilise steganographic techniques, such as watermarking, to help protect their software.

Bytecode steganography is a novel technique proposed in 2010 for creating *Platform Independant Programs*[1]. The idea behind it is that between different computer architectures there is enough of an overlap between their compiled bytecode formats that you can construct programs that are valid for both architectures. One suggested application for this technique is as a defence against program exfiltration. The idea is that by making a secret modification to an existing virtual machine your program runs correctly on the modified architecture; but when run on an unmodified architecture the program behaves differently—perhaps deleting itself or phoning home. The advantages to using a steganographic system such as this over DRM would be that not only is the protection difficult to remove (it would require decompilation and binary translation to the unmodified architecture[2]) but that it isn't immediately obvious there is any protection at all as the program remains a valid program for the unmodified architecture.

There has been relatively few attempts to demonstrate bytecode steganography in practice. I would wish to extend the work from my Masters thesis (on discoving platform independant program snippets for a variety of architectures) and look at the possibility of using bytecode steganography for exfiltration protection. Specifically I wish to develop a toolchain to modify an existing architecture and create protected programs for it before auditing the generated executables to assess how much security the technique provides.

References

[1] Sang Kil Cha, Brian Pak, David Brumley, and Richard Jay Lipton. Platform-independent programs. In *CCS '10: Proceedings of the 17th ACM conference on Computer and communications security*. ACM Request Permissions, October 2010.

[2] C Cifuentes, M Van Emmerik, and N Ramsey. The design of a

resourceable and retargetable binary translator. *Reverse Engineering, 1999. Proceedings. Sixth Working Conference on*, pages 280–291, 1999.