RESEARCH PROPOSAL

JOSEPH HALLETT

Exfiltration is an information security problem studying the removal of data from a secure system. Stopping executable code being removed from a secure system is a sub-problem in the field. Currently DRM and encryption is used to prevent unauthorised copying. Others use steganography, such as watermarking, to help protect their software.

In 2010 a new technique was discovered for creating programs that ran on multiple computer architectures[2]. Amongst different architectures there is an overlap between their byte-code formats. By studying the instruction sets you can construct programs that are valid for both architectures. *Cha et al* found that by applying the technique to small enough pieces of code it could be used to perform byte-code steganography. A suggested application for this technique is as a defence against program exfiltration.

By making a secret modification to an existing virtual machine you can create an executable that hides two different programs. On the modified architecture the program behaves correctly. When run on an unaltered architecture the program takes action to defend against its theft.

There are many advantages to using a steganographic system such as this over DRM. The protection would be difficult to remove as it requires decompilation and translation to the unmodified architecture[3]. Any platform independent program generated is always valid for any architecture it is compiled for. The original program will remain hidden unless someone inspects the binary and can untangle the obfuscation.

There has been relatively few attempts to demonstrate this form of byte-code steganography in practice[2]. For my Masters thesis I looked at finding platform independent programs for various architectures. I wish to extend the work and look at using byte-code steganography for exfiltration protection. I will develop a tool-chain to modify an existing architecture and create steganographically protected programs. Then I will audit the generated executables to see how much security is provided by the technique.

Existing research has worked on detecting when exfiltration is happening[4] as well as protecting the data after it has been stolen. With generic data strong encryption will always provide a good level of protection against theft. But with executable code we have more options and the potential to add counter-measures to let us know when the data has been stolen. A really nice advantage of using steganography to provide this is that the cover-text can be this defensive program. The cover-text could be used to install malware, or behave as an inefficient implementation of the hidden program.

This topic is interesting for any group interested in security auditing, and the software and hardware necessary to implement secure systems. Byte-code steganography and platform independent programs is also of interest for groups looking at the security of instruction sets. It is also interesting for people studying compiler support for steganography. Recent work by people inside Bristol University's Cryptography and Information Security group suggest some interest in steganography[5]. Polymorphism can be used by platform independent programs to hide their presence. Looking into polymorphic coding techniques is something else the group has been interested in[1].

I believe that studying steganographic solutions to exfiltration is an interesting topic. There are opportunities within it to research a variety of sub-problems. The subject combines the study of hardware, software, steganography and information security. Further research into exfiltration protection is necessary to fully explore its potential. I wish to research this at Bristol University as part of my post-graduate studies.

References

[1] Antoine Amarilli, Sascha Müller, David Naccache, Daniel Page, Pablo Rauzy, and Michael Tunstall. *Can Code Polymorphism Limit Information Leakage?* Springer Berlin Heidelberg, 2011.

[2] Sang Kil Cha, Brian Pak, David Brumley, and Richard Jay Lipton. Platform-independent programs. In *Proceedings of the 17th ACM conference on Computer and communications security*, 2010.

[3] 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.

[4] Yali Liu, C Corbett, Ken Chiang, R Archibald, B Mukherjee, and D Ghosal. SIDD: A Framework for Detecting Sensitive Data Exfiltration by an Insider Attack. In *System Sciences, 2009.*, pages 1–10, 2009.

[5] Alexandros Zaharis, Adamantini Martini, Theo Tryfonas, Christos Ilioudis, and G Pangalos. Lightweight Steganalysis Based on Image Reconstruction and Lead Digit Distribution Analysis. *International Journal of Digital Crime and Forensics*, 3(4):29–41, 2011.