Type-based analysis of real PKCS#11 devices*

Riccardo Focardi
DAIS, Università Ca' Foscari Venezia, Italy
focardi@dsi.unive.it

Flaminia Luccio
DAIS, Università Ca' Foscari Venezia, Italy
luccio@unive.it

PKCS#11 defines a widely adopted API for cryptographic tokens [10]. It provides access to cryptographic functionalities while providing some security properties. More specifically, the value of keys stored on a PKCS#11 device and tagged as *sensitive* should never be revealed outside the token, even when connected to a compromised host. Unfortunately, PKCS#11 is known to be vulnerable to attacks that break this property [1, 4, 5]. A recent work [1] has shown that many existing commercially available devices are vulnerable to these attacks; the secured ones, instead, prevent the attacks by drastically reducing functionalities. However, it has been shown that the API can be 'patched' without necessarily cutting it down so much [1, 3, 5, 6, 7, 8]. In [3] we have defined a simple imperative programming language, suitable to code PKCS#11 APIs for symmetric key management and we have presented a type system to statically enforce API security. We have then applied the type system to validate fixes proposed in the literature and a new one based on key-diversification [2].

We are presently extending the type system of [3] so to be applicable to real devices. The extension is being implemented in the opencryptoki simulator [9] in order to provide a proof-of-concept that the firmware of real devices might benefit of this kind of verification. In fact, even if we have never had access to the firmware source code of real devices, we can expect that the difficulty of type-checking it is comparable to the one of type-checking opencryptoki C source code. A full presentation of this work will be given at the ASA workshop, right after CSF.

References

[1] M. Bortolozzo, M. Centenaro, R. Focardi, and G. Steel. Attacking and fixing PKCS#11 security tokens. In *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS)*, pages 260–269. ACM, 2010.

[2] M. Centenaro and R. Focardi. Fixing PKCS#11 by key-diversification. In *Proceedings of the 5th International Workshop on Analysis of Security APIs (ASA)*, Paris, France, June 2011.

[3] M. Centenaro, R. Focardi, and F.L. Luccio. Type-based Analysis of PKCS#11 Key Management. In *POST*, volume 7215 of *Lecture Notes in Computer Science*, pages 349–368. Springer, 2012.

[4] J. Clulow. On the security of PKCS#11. In *5th International Workshop on Cryptographic Hardware and Embedded Systems (CHES 2003)*, volume 2779 of *Lecture Notes in Computer Science*, pages 411–425. Springer, 2003.

[5] S. Delaune, S. Kremer, and G. Steel. Formal analysis of PKCS#11 and proprietary extensions. *Journal of Computer Security*, 18(6):1211–1245, November 2010.

[6] S.B. Fröschle and N. Sommer. Reasoning with Past to Prove PKCS#11 Keys Secure. In *FAST*, volume 6561 of *Lecture Notes in Computer Science*, pages 96–110. Springer, 2010.

[7] S.B. Fröschle and N. Sommer. Concepts and Proofs for Configuring PKCS#11. In *FAST*, volume 7140 of *Lecture Notes in Computer Science*. Springer, 2011.

[8] S.B. Fröschle and G. Steel. Analysing PKCS#11 key management APIs with unbounded fresh data. In *ARSPA-WITS*, volume 5511 of *LNCS*, pages 92–106, York, UK, 2009. Springer.

[9] openCryptoki. http://sourceforge.net/projects/opencryptoki/.

[10] RSA Security Inc., v2.20. *PKCS #11: Cryptographic Token Interface Standard.*, June 2004.

*Work partially supported by the RAS Project "*TESLA: Techniques for Enforcing Security in Languages and Applications*".