Trust model in presence of trusted hardware

Jan Cappaert, Thomas Herlea, Dries Schellekens, Brecht Wyseur

March 20, 2007

Abstract

In this paper, the trust model for RE-TRUST WP3 is discussed.

1 Introduction

2 Trusted components

These are the components we trust:

2.1 Trusted execution environment

An adversary has total control over the untrusted client platform. A malicious user or program (i.e., malware) are assumed to have full privilege access to the system, in their attempts to attack the original program P and additional software M monitoring the integrity of P.

2.1.1 Trusted server

The trusted server is not controlled by the adversary. A simple solution to circumvent the remote entrusting problem is server side execution of the program P. This however only protects the integrity and confidentiality of the software functionality (i.e., program code), but not the input and output data; for instance, key loggers will still be able to log passwords and other sensitive data. Server side execution also puts a lot of processing load on the trusted server, which is undesirable in certain applications.

2.1.2 Trusted hardware

The trusted hardware can be trusted to perform computations on data, in such a way that the attacker does not learn anything about the data processed and/or the operations executed. The computational power and storage capacity of the secure hardware are limited and thus running the full functionality of the original program P is impossible; again I/O behavior (i.e., communication between untrusted platform and trusted hardware) can always be observed and manipulated by an attacker.

2.1.3 Software splitting

Because execution of the original software P on either the server, or the trusted hardware is undesirable, maybe even impossible, one could opt to only execute security sensitive functionality in a trusted execution environment.

Approaches have been proposed to determine how the split software in open and hidden components, that need to execute on the untrusted environment and trusted environment respectively [24]. In some highly reactive application, such as online games, splitting the execution of P between the untrusted client and trusted server is unacceptable if the interactions with the remote server are synchronous (i.e., the application blocks waiting for a response from the server). Virtual leashing is proposed as an asynchronous solution to overcome this potential problem [11].

2.2 Protecting data

Various cryptographic primitives exist to protect the integrity and/or confidentiality of data. Cryptology has historically focussed on protecting data during transmission. In the white-box security model, where an attacker has full control over the execution environment, traditional cryptographic algorithms are not so strong; e.g., it is rather straightforward to extract an encryption key in computer memory [21] or to deduce a key from cache timing information.

2.2.1 White-box cryptography

White-box cryptography [7, 8, 15] covers a set of techniques to implement a block cipher in order to obstruct the extraction of the embedded secret key. The main idea is to implement the block cipher as a network of lookup tables. All the operations such as the key addition are embedded in the lookup tables, which are then randomized to obfuscate their behavior. Typically, external encodings are applied to increase the level of security, transforming the block cipher E_k into an encoded implementation $G \circ E_k \circ F^{-1}$. Research in this topic is still ongoing, as several cryptanalysis have been presented [4, 12, 23] .

2.2.2 Observable cryptography

Physical observable cryptography is a new methodology mainly used for public key primitives. This research however is still in an early stage.

2.2.3 Computing with encrypted data

The use of privacy homomorphisms as a technique to process encrypted is quite old [1, 18]. Homomorphic functions are functions that have a relation between operations in the (remote) encrypted domain and the local domain, i.e., there is a relation $y = f(x) \rightarrow E(y) = f'(E(x))$. Research is ongoing to achieve more complicated processing in the encrypted domain.

2.3 Protecting software

Several techniques address these problems and try to create self-protecting software [10, 17]. For example, code obfuscation transforms code while preserving its functionality such that analysis becomes hard, expensive and time consuming. In the meantime, related techniques are being developed to protect against software tampering. These mechanisms typically also protect the data software is handling to a certain extent.

2.3.1 Software obfuscation

Code obfuscation to slow down the adversary in attempts to analyze and subsequently tamper software.

Code encryption...

Self modifying code as way to obfuscate software...

[9, 16, 22]

Hardware/software co-obfuscation?

2.3.2 Tamper resistant software

As code obfuscation aims to twart code analysis, self-checking code tries to protect against tampering. Protecting code against tampering can be considered as the problem of data authenticity, where in this context 'data' refers to the program code. Aucsmith [3] was the first ot propose a scheme to implement tamper-resistant software. Chang *et al.* [5] proposed a scheme based on software guards. Their protection scheme relies on a complex network of software guards which mutually verify each other's integrity and that of the program's critical sections. The security of the scheme relies partially on hiding the obfuscated guard code and the complexity of the guard network. Horne *et al.* [13] elaborated on the same idea and proposed 'testers', small hashing functions that verify the program at runtime. Other related research is oblivious hashing [6] which interweaves hashing instructions with program instructions and which is able to prove whether a program operated correctly or not.

2.3.3 Software watermarking

2.3.4 Software interlocking

2.3.5 Software replacement

[14]

Different functionality.

2.3.6 Software diversity

Code diversity [2] to produce substantially different implementations of the same software functionality.

2.3.7 Computing with encrypted functions

Encrypted functions that can be executed without prior decryption have been described in [19, 20], and are often also refered to as function hiding. Sander *et al.* describe how encrypted programs can be used to achieve protection of algorithms against disclosure and can give way to suprisingly solutions for seemingly unsolvable problems of software protection. The key point is to encrypt functions such that they remain executable, but provide encrypted results $E(y) = E(f)(x)$.

References

[1] Niv Ahituv, Yeheskel Lapid, and Seev Neumann. Processing Encrypted Data. *Communications of the ACM*, 30(9):777–780, 1987.

[2] Bertrand Anckaert, Bjorn De Sutter, and Koen De Bosschere. Software Piracy Prevention through Diversity. In Aggelos Kiayias and Moti Yung, editors, *Proceedings of the 2004 ACM Workshop on Digital Rights Management 2004, Washington, DC, USA, October 25, 2004*, pages 63–71. ACM, 2004.

[3] David Aucsmith. Tamper Resistant Software: An Implementation. In Ross J. Anderson, editor, *Information Hiding, First International Workshop, Cambridge, U.K., May 30 - June 1, 1996, Proceedings*, volume 1174 of *Lecture Notes in Computer Science*, pages 317–333. Springer, 1996.

[4] Olivier Billet, Henri Gilbert, and Charaf Ech-Chatbi. Cryptanalysis of a White Box AES Implementation. In Helena Handschuh and M. Anwar Hasan, editors, *Selected Areas in Cryptography, 11th International Workshop, SAC 2004, Waterloo, Canada, August 9-10, 2004, Revised Selected Papers*, volume 3357 of *Lecture Notes in Computer Science*, pages 227–240. Springer, 2004.

[5] Hoi Chang and Mikhail J. Atallah. Protecting Software Code by Guards. In Tomas Sander, editor, *Security and Privacy in Digital Rights Management, ACM CCS-8 Workshop DRM 2001, Philadelphia, PA, USA, November 5, 2001, Revised Papers*, volume 2320 of *Lecture Notes in Computer Science*, pages 160–175. Springer, 2001.

[6] Yuqun Chen, Ramarathnam Venkatesan, Matthew Cary, Ruoming Pang, Saurabh Sinha, and Mariusz H. Jakubowski. Oblivious Hashing: A Stealthy Software Integrity Verification Primitive. In Fabien A. P. Petitcolas, editor, *Information Hiding, 5th International Workshop, IH 2002, Noordwijkerhout, The Netherlands, October 7-9, 2002, Revised Papers*, volume 2578 of *Lecture Notes in Computer Science*, pages 400–414. Springer, 2002.

[7] Stanley Chow, Philip A. Eisen, Harold Johnson, and Paul C. van Oorschot. A White-Box DES Implementation for DRM Applications. In Joan Feigenbaum, editor, *Security and Privacy in Digital Rights Management, ACM CCS-9 Workshop, DRM 2002, Washington, DC, USA, November 18, 2002, Revised Papers*, volume 2696 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2002.

[8] Stanley Chow, Philip A. Eisen, Harold Johnson, and Paul C. van Oorschot. White-Box Cryptography and an AES Implementation. In Kaisa Nyberg and Howard M. Heys, editors, *Selected Areas in Cryptography, 9th Annual International Workshop, SAC 2002, St. John's, Newfoundland, Canada, August 15-16, 2002. Revised Papers*, volume 2595 of *Lecture Notes in Computer Science*, pages 250–270. Springer, 2002.

[9] Stanley Chow, Yuan Gu, Harold Johnson, and Vladimir A. Zakharov. An Approach to the Obfuscation of Control-Flow of Sequential Computer Programs. In George I. Davida and Yair Frankel, editors, *Information Security, 4th International Conference, ISC 2001, Malaga, Spain, October 1-3, 2001, Proceedings*, volume 2200 of *Lecture Notes in Computer Science*, pages 144–155. Springer, 2001.

[10] Christian S. Collberg and Clark D. Thomborson. Watermarking, Tamper-Proofing, and Obfuscation-Tools for Software Protection. *IEEE Transactions on Software Engineering*, 28(8):735–746, 2002.

[11] Ori Dvir, Maurice Herlihy, and Nir Shavit. Virtual Leashing: Internet-Based Software Piracy Protection. In *25th International Conference on Distributed Computing Systems (ICDCS 2005), 6-10 June 2005, Columbus, OH, USA*, pages 283–292. IEEE Computer Society, 2005.

[12] Louis Goubin, Jean-Michel Masereel, and Michael Quisquater. Cryptanalysis of White Box DES Implementations. Cryptology ePrint Archive, Report 2007/035, 2007. `http://eprint.iacr.org/`.

[13] Bill Horne, Lesley R. Matheson, Casey Sheehan, and Robert Endre Tarjan. Dynamic Self-Checking Techniques for Improved Tamper Resistance. In Tomas Sander, editor, *Security and Privacy in Digital Rights Management, ACM CCS-8 Workshop DRM 2001, Philadelphia, PA, USA, November 5, 2001, Revised Papers*, volume 2320 of *Lecture Notes in Computer Science*, pages 141–159. Springer, 2001.

[14] Markus Jakobsson and Michael K. Reiter. Discouraging Software Piracy Using Software Aging. In Tomas Sander, editor, *Security and Privacy in Digital Rights Management, ACM CCS-8 Workshop DRM 2001, Philadelphia, PA, USA, November 5, 2001, Revised Papers*, volume 2320 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2001.

[15] Hamilton E. Link and William D. Neumann. Clarifying Obfuscation: Improving the Security of White-Box DES. In *International Symposium on*

Information Technology: Coding and Computing (ITCC 2005), Volume 1, 4-6 April 2005, Las Vegas, Nevada, USA.

[16] Cullen Linn and Saumya K. Debray. Obfuscation of Executable Code to Improve Resistance to Static Disassembly. In Sushil Jajodia, Vijayalakshmi Atluri, and Trent Jaeger, editors, *Proceedings of the 10th ACM Conference on Computer and Communications Security, CCS 2003, Washingtion, DC, USA, October 27-30, 2003*, pages 290–299. ACM, 2003.

[17] Gleb Naumovich and Nasir D. Memon. Preventing Piracy, Reverse Engineering, and Tampering. *IEEE Computer*, 36(7):64–71, 2003.

[18] Ronald L. Rivest, Leonard Adleman, and Michael L. Dertouzos. On Data Banks and Privacy Homomorphisms. In R. DeMillo, D. Dobkin, A. Jones, and R. Lipton, editors, *Foundations of Secure Computation*, pages 169–180. Academic Press, 1978.

[19] Tomas Sander and Christian F. Tschudin. On Software Protection via Function Hiding. In David Aucsmith, editor, *Information Hiding, Second International Workshop, Portland, Oregon, USA, April 14-17, 1998, Proceedings*, volume 1525 of *Lecture Notes in Computer Science*, pages 111–123. Springer, 1998.

[20] Tomas Sander and Christian F. Tschudin. Protecting Mobile Agents Against Malicious Hosts. In Giovanni Vigna, editor, *Mobile Agents and Security*, volume 1419 of *Lecture Notes in Computer Science*, pages 44–60. Springer, 1998.

[21] Adi Shamir and Nicko van Someren. Playing "Hide and Seek" with Stored Keys. In Matthew K. Franklin, editor, *Financial Cryptography, Third International Conference, FC'99, Anguilla, British West Indies, February 1999, Proceedings*, volume 1648 of *Lecture Notes in Computer Science*, pages 118–124. Springer, 1999.

[22] Enriquillo Valdez and Moti Yung. Software DisEngineering: Program Hiding Architecture and Experiments. In Andreas Pfitzmann, editor, *Information Hiding, Third International Workshop, IH'99, Dresden, Germany, September 29 - October 1, 1999, Proceedings*, volume 1768 of *Lecture Notes in Computer Science*, pages 379–394. Springer, 1999.

[23] Brecht Wyseur, Wil Michiels, Paul Gorissen, and Bart Preneel. Cryptanalysis of White Box DES Implementations. Cryptology ePrint Archive, To appear, 2007. `http://eprint.iacr.org/`.

[24] Xiangyu Zhang and Rajiv Gupta. Hiding Program Slices for Software Security. In *1st IEEE / ACM International Symposium on Code Generation and Optimization (CGO 2003), 23-26 March 2003, San Francisco, CA, USA*, pages 325–336. IEEE Computer Society, 2003.