ETSI TR 103 642 V1.1.1 (2018-10)

CYBER;
Security techniques for protecting software in a white box model

Reference

DTR/CYBER-0029

Keywords

security, software

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

The present document can be downloaded from:
http://www.etsi.org/standards-search

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the only prevailing document is the print of the Portable Document Format (PDF) version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at
https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

If you find errors in the present document, please send your comment to one of the following services:
https://portal.etsi.org/People/CommiteeSupportStaff.aspx

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

Foreword

This Technical Report (TR) has been produced by ETSI Technical Committee Cyber Security (CYBER).

Modal verbs terminology

In the present document "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Introduction

When one tries to protect some sensitive data or code in a device like a mobile, one has to consider that the environment may be compromised (or if it is not yet, it will be soon). The attacker, trying to get secrets of an application, is able to read a lot about the software code and applications, and overrule the security measures, etc. This situation makes much harder the protection of the assets of a mobile application.

To address this, the main strategy is to apply obfuscation and anti-tampering techniques on the software (to hide code, data, key, etc.) and to deploy on the server side assurance function to detect fraudulent behaviour of the application and/or the device. One needs to put in place functions to diversify the software, the protection techniques, the assets (per user or per instance of the application) in order to prevent large scale attacks and functions for quick redeployment, replenishment of the software.

This methodology relies on two pillars:

- White Box Cryptography (which is about protecting the cryptographic functions in the code, together with the key); and
- Code and data protection (which is about making sure that the code, used to run the application cannot be understood, tampered, extracted, exploited, and the data cannot be retrieved and/or modified).

By combining those countermeasures, (and having a correct security design for the application), the level of resistance of a software application to software attack is increased.

1 Scope

The present document reports on the application of techniques for protecting software implementations, in the form of applications and content, using software resident security techniques. The present document makes recommendations for the application of specific techniques including white box cryptography (WBC), code obfuscation, and other techniques denoted as anti-xxx and including anti-tampering, anti-reversing, anti-debugging, anti-cloning, etc. These techniques address the threats presented by attackers of the forms outlined in the present document.

2 References

2.1 Normative references

Normative references are not applicable in the present document.

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] C. Collberg, C. Thomborson and D. Low: "A taxonomy of obfuscating transformations", Technical Report #148, University of Auckland, 1997.

NOTE: Available at http://profs.sci.univr.it/~giaco/download/Watermarking-Obfuscation/Obfuscation%20Taxonomy.pdf.

[i.2] N. Eyrolles, PhD: "Obfuscation with Mixed Boolean-Arithmetic Expressions: Reconstruction, Analysis and Simplification Tools", Cryptography and Security, Université Paris-Saclay, 2017.

NOTE: Available at https://tel.archives-ouvertes.fr/tel-01623849/.

[i.3] C. Collberg, C. Thomborson and D. Low: "Manufacturing cheap, resilient, and stealthy opaque constructs", University of Auckland, 1998.

NOTE: Available at https://www2.cs.arizona.edu/~collberg/content/research/papers/collberg98manufacturing-clean.pdf.

[i.4] E. Biham and A. Shamir: "Differential cryptanalysis of DES-like cryptosystems", A. Menezes and S. A. Vanstone, Eds., CRYPTO, LNCS 537, pp. 2-21. Springer, 1990.

[i.5] A. Biryukov, C. De Cannière, A. Braeken, and B. Preneel: "A toolbox for cryptanalysis: Linear and affine equivalence algorithms", in E. Biham, Ed., EUROCRYPT, LNCS 2656, pp. 33-50. Springer, 2003.

[i.6] O. Billet, H. Gilbert, and C. Ech-Chatbi: "Cryptanalysis of a White Box Implementation", in H. Handschuh and A. Hasan, Eds., Selected Areas in Cryptography 2004, LNCS 3357, pp. 227-240. Springer, 2005.

[i.7] L. Tolhuizen: "Improved cryptanalysis of an AES implementation", Proceedings of Symposium on Information Theory in the Benelux 2012.

[i.8] T. Lepoint, M. Rivain, Y. De Mulder, P. Roelse, and B. Preneel: "Two attacks on a white-box AES implementation", in T. Lange, K. Lauter and P. Lisonek, Eds., Selected Areas in Cryptography 2013, LNCS 8282, pp. 265-285, Springer, 2014.

[i.9] W. Michiels, P. Gorissen, H.D.L. Hollmann: "Cryptanalysis of a generic class of white-box implementations", Proceedings of Selected Areas in Cryptography (SAC) 2008, LNCS 5381, pp. 414-428, 2009.

[i.10] Y. De Mulder, B. Wyseur, and B. Preneel: "Cryptanalysis of a perturbated white-box AES implementation", in G. Gong and K. C. Gupta, Eds., INDOCRYPT 2010, LNCS 6498, pp. 292-310. Springer, 2010.

[i.11] Y. De Mulder, P. Roelse, and B. Preneel: "Cryptanalysis of the Xiao-Lai white-box AES implementation", in L. R. Knudsen and H. Wu, Eds., Selected Areas in Cryptography 2012, LNCS 7707, pp. 34-49. Springer, 2012.

[i.12] P. Kocher, J Jaffe, and B. Jun: "Differential Power Analysis", Advances in Cryptology - Crypto '99 Proceedings, LNCS 1666, M. Wiener, Ed., pp. 388-397, Springer 1999.

[i.13] E. Biham and A. Shamir: "Differential fault analysis of secret key cryptosystems", Advances in Cryptology - Crypto '97 Proceedings, LNCS 1294, B. Kaliski, Ed., pp. 513-525, Springer 1997.

[i.14] J. Bos, C. Hubain, W. Michiels, and P. Teuwen: "Differential Computation Analysis: Hiding Your White-Box Design is Not Enough", Cryptology ePrint Archive, Report 2015/753, 2015.

NOTE: Available at https://eprint.iacr.org/2015/753.

[i.15] M. Jacob, D. Boneh, E.W. Felten: "Attacking an obfuscated cipher by injecting faults. Proceedings of security and privacy in Digital Rights Management (DRM)", LNCS 2696, pp. 16-31, Springer, 2002.

[i.16] CHES 2017: "Capture the Flag Challenge", organised by the ECRYPT-CSA project, Submission server developed by CryptoExperts, Submission server, hosted by TU Eindhoven.

NOTE: Available at https://ches.2017.rump.cr.yp.to/a905c99d1845f2cf373aad564ac7b5e4.pdf.

[i.17] E. Diehl: "Securing Digital Video - Techniques for DRM and Content Protection", Springer, 2012. ISBN 978-3-642-17344-8.

[i.18] Motion Picture Laboratories Inc.: "MovieLabs Specification for Enhanced Content Protection", Version 1.1, 2015.

[i.19] H. Benoit: "Digital Television - Satellite, Cable, Terrestrial, IPTV, Mobile TV in the DVB Framework", 3rd edition, Elsevier, 2008. ISBN 978-0-240-52081-0.

[i.20] ETSI TS 100 289 (V1.1.1): "Digital Video Broadcasting (DVB); Support for use of the DVB Scrambling Algorithm version 3 within digital broadcasting systems".

[i.21] CI Plus LLP, CI Plus Specification: "Content Security Extensions to the Common Interface", v1.4.2, 2016.

[i.22] EMVCo: "EMV® Mobile Payment - Software-based Mobile Payment Security Requirements", Version 1.0, December 2016.

NOTE: Available at https://www.emvco.com/wp-content/EMVCo-Software-based-Mobile-Payment-Security-Requirements_V1.0_20161213.pdf.

[i.23] ISO SC27: "Study Period on Security requirements, test and evaluation methods for White Box Cryptography (WBC)" October 2016.

[i.24] ISO SC27: "Study Period on Security properties, test and evaluation guidance for White Box Cryptography (WBC)", April 2018.

[i.25] FIDO Alliance, FIDO Authenticator certification program.

NOTE: Available at https://fidoalliance.org/certification/authenticator-certification-levels/.

[i.26] OWASP, Mobile TOP 10 attacks, 2016.

NOTE: Available at https://www.owasp.org/index.php/Mobile_Top_10_2016-Top_10.

[i.27] Visa Mobile publications, Cloud-Based Payments Program.

NOTE: Available at https://technologypartner.visa.com/Mobile/MobilePublications.aspx#59.

3 Definitions of terms and abbreviations

3.1 Terms

For the purposes of the present document, the following terms apply:

NOTE: The definitions below are split in two paragraphs, to distinguish software protection techniques from attacks.

anti-cloning: software protection technique to prevent execution of a binary on a non-genuine device

NOTE: Strictly speaking the binary knows fingerprints of the execution environment like the MAC address, CPU ID, HDD serial number, etc. and verifies its integrity during execution. If the fingerprints do not match, the device is seen as "cloned" and the program can react in several ways. This technology can help to prevent code lifting.

anti-debugging: software protection technique to prevent debugging of an application through a debugger

NOTE: Example of debuggers are GDB, WinDBG. If a debugger is detected the program can react in several ways like notification of the user or abortion of the execution.

anti-disassembly: method to prevent disassembling of the binary through disassemblers

NOTE: Disassemblers like IDA Pro, Binary Ninja, etc. Anti-Disassembly can be applied directly on assembler code to let disassemblers generate wrong opcodes or on control flow level to hide the connection of the basic blocks of a function.

BGE: named after its authors (Billet, Gilbert and Ech-Chatbi), a realistic (in terms of work factor) algebraic attack on the white-box AES design by Chow, Eisen, Johnson and van Oorschot

NOTE: It has been further generalized to all Substitution-Permutation Network ciphers.

cloning: making an illegal copy of the contents (of the device) to a new device or for analysis

code anti-tampering: software protection technique that prevents modification of a binary

NOTE 1: If a modification of the binary is detected, the program can react in several ways like notification of the user or abortion of the execution.

NOTE 2: Code anti-Tampering can be divided into two groups:

- Static Anti Tampering: Before execution of the binary the integrity of the file will be verified and loaded into memory if and only if the checksum is valid.
- Dynamic Anti Tampering: During execution of the binary random integrity checks will be executed to self-check if the binary was tampered.

code/data anti-reversing: set of software techniques aiming to protect an attacker having access to a program to understand its implementation or having access to sensitive data in clear form

NOTE: See also Obfuscation.

code-flow/-pointer integrity: technique which prevents attacks that try to take control of the execution flow of the program

NOTE: During compilation the compiler injects guards that will verify if the destination address is valid. If the destination address is not valid the program can detect that and abort the execution.

code lifting: method where the attacker tries to reuse part of the binary code for executing it in an unauthorized way

code obfuscation: combination of several software based techniques that transform the source or machine code into code that is very difficult to understand for humans

NOTE: The purpose of obfuscation is to harden the resistance against attacks like reverse engineering or tampering of the code. Obfuscation can be broken but the amount of time and the expertise to apply attacks on obfuscation can be significant and is done by skilled experts.

code-pointer separation: technology to verify the integrity of code pointers by splitting the memory into safe memory and regular memory.

NOTE: Code pointers are placed in the safe memory and the integrity of the memory will be verified during execution of the binary. This technique prevents attacks where an attacker tries to take control of the execution flow of the application and has a performance advantage over Code-flow integrity.

code watermarking: technique to deeply embed a unique watermark in a binary and that is very difficult to detect and remove

NOTE: The watermark can be used to authenticate a genuine software but also to trace back illegal copies.

data anti-tampering: software protection technique that prevents modification of data (e.g. integrity checksum, redundancy, etc.)

NOTE: Data anti-tampering can be static or dynamic.

data encodings: reversible method to transform data without the requirement of a secret key

NOTE 1: Typically used in WBC.

NOTE 2: They are external encodings applied at the input and at the output of a crypto functions. They are internal encodings applied to internal variables and/or tables. The purpose is to manipulate data not in clear form and to avoid software side channel attacks. They are static encodings that are fixed at compilation time and they are dynamic encodings that are changing at execution time.

data lifting: method where the attacker tries to reuse data in an unauthorized way, on behalf of the genuine data owner

NOTE: Typically re-using a WBC dynamic key from one instance to another.

data obfuscation: set of software protection techniques to hide data against an attacker trying to reverse it (i.e. find, detect, and/or localise)

NOTE: This includes hiding sensitive data (encryption/encoding), making, splitting input data into smaller chunks or using steganography.

data protection: techniques that try to hide data like cryptographic keys, debug strings, application strings, etc. against an attacker and tools like "strings" or an (hex-) editor

data watermarking: technique to deeply embed a unique watermark in a data that is very difficult to detect and remove

NOTE: An example is a movie file watermarked to track back illegal copies.

device binding: software and/or a hardware technique to bind a data or code to a device, meaning if one tries to use the bound item in another device, it will fail

Differential Computation Analysis (DCA): side-channel attack derived from DPA aiming at key extraction from a WBC where measuring noisy power consumption is replaced by noiseless intermediate computation value collection during a cryptographic software executions

Differential Fault Analysis (DFA): family of attacks that relies on the analysis of the difference in outputs when a perturbation is injected during a cryptographic computation

NOTE: Such a perturbation can be a laser shot on a hardware chip, or a value modification in the context of a software implementation.

Differential Power Analysis (DPA): attack that exploits the information leakage of the manipulation of a secret value by a hardware, through its power consumption

NOTE 1: Information is collected for multiple variable inputs with an oscilloscope. Difference of means is then computed, after splitting the dataset based on a guessed value computed from the actual inputs and a key hypothesis. The good key stands out as it correctly splits the data.

NOTE 2: Power consumption is not the only side-channel that can be used. The same attack based on electro-magnetic radiation would be called DEMA, for example.

hooking: range of techniques used to alter or augment the behaviour of an operating system, of applications, or of other software components by intercepting function calls or messages or events passed between software components

stack cookies: method used to protect against buffer overflows

NOTE: The compiler integrates known values, "stack cookies", between a buffer and the control data. When the stack cookie gets overwritten during a buffer overflow the code can detect that by verifying the stack cookie with the hardcoded value in the code and abort the execution of the program.

White Box Cryptography (WBC): set of software protection techniques aiming at protecting software implementations of cryptographic algorithms against key recovery

NOTE: The specificity of a white-box cryptography attacker is that he is assumed to have full control over the whole execution of an implementation. In this highly unfavourable setup, security by obscurity was the initial industry response, with the deployment of private algorithms in real-world WBC. The academic world effort has increased over the past years, with a focus on standard cryptographic algorithms.

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

AES Advanced Encryption Standard
Anti-xxx All methods starting with "anti-"

NOTE: It includes at least anti-tampering, anti-cloning, anti-reversing, and anti-debugging.

BGE Billet, Gilbert and Ech-Chatbi

NOTE: From the name of the authors of a well-known attack against a white-box implementation of AES.

CA Conditional Access
CA/DRM Conditional Access/Digital Rights Management
CBP Cloud Based Payment
CHES Cryptographic Hardware and Embedded Systems

NOTE: A conference on cryptographic hardware and embedded systems.

CPE Customer Premises Equipment
CPUID derived from CPU (Central Processing Unit) IDentification
DCA Differential Computation Analysis
DES Data Encryption Standard
DFA Differential Fault Analysis
DPA Differential Power Analysis
DRM Digital Rights Management
DVB Digital Video Broadcasting
ECI Embedded Common Interface
EMV Europay Mastercard Visa
IC Integrated Card
ISG Industry Specification Group
MAC Medium Access Control
MBA Mixed Boolean-Arithmetic
OS Operating System
OTT Over-The-Top
SE Secure Element

SIM	Subscriber Identity Module
SoC	System on Chip
VCK	Virtual Car Key
WBC	White Box Cryptography

4 Threat security model

There are different threat security models used in the field of software security.

The classical threat model involves a malicious third party Eve attempting to decrypt the communication between Alice and Bob. In this case, Alice and Bob are friendly parties communicating, and not attacking the system (Eve is an external attacker). There are situations where this is not true, and where the attacker may be one of the two communicating parties.

In a "Black-box model" (see figure 1), the attacker is assumed to have access to inputs and outputs, to be able to observe and modify the communication (read, intercept and/or substitute). He has no access to the encryption keys, or more generally to the system performing cryptographic operations. He has no access to the software binary or to the execution environment internals, and may abuse the software functionality.
The implementation stays opaque ("black").

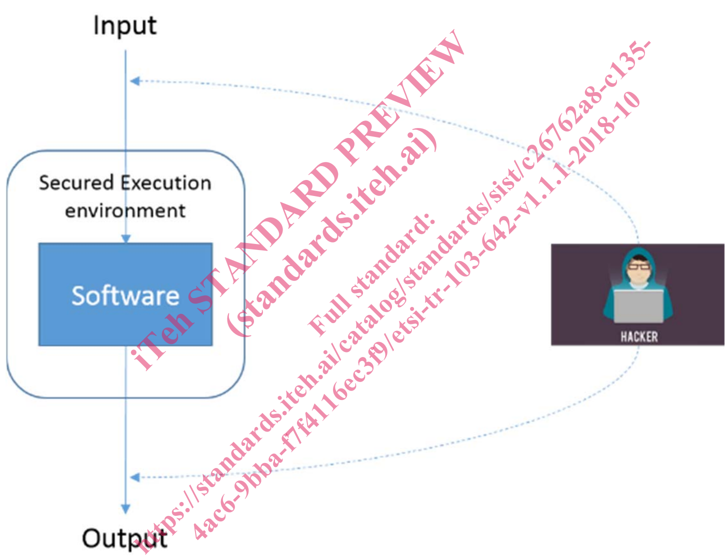

Figure 1: Black-box threat model

In a "Grey-box threat model" (see figure 2), the attacker has the same power as in the black-box model, but he also partially knows and/or have access to the internal structure of the system performing cryptographic operations (e.g. access to the documentation of the internal data structure, the cryptographic algorithms used). He has an indirect access, through side channel analysis, to execution environment internals and may disrupt the software execution through faults. In this model, the attacker does not have access to the keys and/or cannot tamper with the cryptographic algorithms.