DEAR CRYPTO MAG READERS,

WE ARE GLAD THAT WE HAVE MANAGED TO PUBLISH THE FIRST ISSUE OF CRYPTOMAG. WE STRUGGLED TO MAKE IT HAPPEN FOR THE LAST TWO MONTHS AND FINALLY, WE REACHED THE DESTINATION. THIS ISSUE IS TOTALLY FREE AND HAS BEEN MADE OF THE BEST CRYPTO ARTICLES WRITTEN BY OUR ASSOCIATES WRITING MAINLY FOR HAKIN9. WE GENUINELY HOPE THAT OUR MAGAZINE WILL BE PUBLISHED REGULARLY. FIRST, WE AIM AT 1 MAGAZINE A MONTH. LATER ON, WE DEFINITELY WILL HAVE THE OPPORTUNITY TO PUBLISH MORE REGULARLY WHEN WE GROW UP IN POWER.

IF YOU HAVE ANY SUGGESTIONS, OR YOU WOULD LIKE TO KNOW MORE ABOUT CRYPTOMAG, OR BECOME A COLLABORATOR – PLEASE WRITE TO US:

M.WISNIEWSKI@SOFTWARE.COM.PL

Managing:
Michał Wiśniewski
m.wisniewski@software.com.pl

Senior Consultant/Publisher:
Paweł Marciniak

Editor in Chief:
Grzegorz Tabaka
grzegorz.tabaka@hakin9.org

Art Director:
Mateusz Jagielski

DTP:
Marcin Ziółkowski GDStudio

Production Director:
Andrzej Kuca
andrzej.kuca@hakin9.org

Marketing Director:
Grzegorz Tabaka
grzegorz.tabaka@hakin9.org

Proofreaders:
Dan Dieterle, Michael Munt,
Michał Wiśniewski

Betatesters:
Ruggero Risone,
David von Vistauxx,
Dan Dieterle,
Johnette Moody,
Nick Baronian,
Dan Walsh,
Sanjay Bhalerao,
Jonathan Ringler,
Arnaud Tijssen,
Patrik Gange

Publisher: Hakin9 Media Sp. z o.o. SK
02-682 Warszawa, ul. Bokserska 1
www.hakin9.org/en

Whilst every effort has been made to ensure the high quality of the magazine, the editors make no warranty, express or implied, concerning the results of content usage. All trade marks presented in the magazine were used only for informative purposes. All rights to trade marks presented in the magazine are reserved by the companies which own them. To create graphs and diagrams we used program by Mathematical formulas created by Design Science MathType™

DISCLAIMER!

The techniques described in our articles may only be used in private, local networks. The editors hold no responsibility for misuse of the presented techniques or consequent data loss.
TO BLACK BOX CRYPTOGRAPHY AND BEYOND…

Introduction: Traditionally, cryptography has offered a means of communicating sensitive (secret, confidential or private) information while making it unintelligible to everyone except for the message recipient. Nowadays, the process of protecting sensitive (secret or confidential) data embedded in one’s code while attempting to hide portions of text from malicious eyes is of an ever-growing concern.

The threat: Replay emulators are software applications that attempt to mimic (duplicate) the behavior of one system onto another while attempting to be as close to the original as possible. In effect emulators are a form of a man-in-the-middle (MITM) attack scenario.

IN THE CONTEXT OF SOFTWARE PROTECTION DONGLE EMULATORS, WE REFER TO THE FOLLOWING SCENARIOS:

Replay emulators (Partial emulators) – this type of emulator can only record and replay the communication between the protected application and the dongle. If the communication is encrypted, and different in every new session, this type of emulator cannot understand the communication therefore rendering it ineffective. If the protected application uses random data strings then the encrypted communication is always different again deeming it ineffective.

Full emulation – this type of emulator can fully understand the communication between the application and the dongle by decrypting the encrypted communication and mimicking functionality such as memory read/write, license verification, and in some cases even replay encrypted blocks seemingly overcoming the encryption. Given the strength of today’s hardware and the complexity of the used algorithms, this scenario is rarely seen.

One of the most effective tools in combating the abovementioned emulation techniques lies in the secure communication channel, created between the protected application and the hardware key. Nowadays, most dongles circumvent emulators by implementing a secure communication channel.

HERE ARE SOME OF THE CONCEPTS INVOLVED:

1. Each string of data passing through the channel is encrypted preferably using a different (random) key.
2. Data “recorded” from one secure communication channel session cannot be replayed in another session – similar technique as in TCP/IP packet counting.
3. Data communicated between the protected application and the dongle should “appear” different every time even if raw data is the same.
To black box cryptography and beyond…

Beyond the Black Box: Traditional cryptography assumes the attacker has no physical access to the key (i.e., the algorithm performing the encryption or decryption) or any internal workings, and instead can only observe external information and behavior (i.e., inputs and outputs). This methodology attempts to hide the cryptographic keys by putting them out of reach of the attacker.

Advances of attacks on hardware have shown that the previously assumed black box performing the cryptography is not completely black to the hacker anymore – it literally became a shade of grey. Grey box was found to be leaking secret information from within the box which is discovered through various means such as DPA and other side channel attacks. This has increased the awareness of hardware manufacturing companies on the need to increasing the security of crypto-processors themselves, in addition to spurring a different thinking about symmetric cryptography. Is there a way to hide a secret key even if the attacker has some access into a supposedly black box?

Symmetric cryptography needs to be performed in pure software only, running in non trustworthy environments - software running on a machine operated by an attacker is the perfect example. To model the threat correctly you need to assume the attacker has full control over the machine and can monitor any step the application performs. This is also called a White box scenario (e.g. a software application exposed to debugging). In contrast with previously described scenarios, this needs to handle far more severe threats while assuming hackers have full visibility and control. Hackers can freely observe dynamic code execution and internal algorithm details are completely visible and alterable at will.

<table>
<thead>
<tr>
<th>Cryptographic method</th>
<th>Attributes</th>
</tr>
</thead>
</table>
| **Black Box**        | Attacker is assumed to have:  
  - No access to the cryptographic key or the firmware code executing the cryptographic algorithm – zero visibility on code during execution  
  - External information such as plaintext (input) or ciphertext (output) only  
  - Considered secure as long as the cipher has no cryptographic weaknesses |
| **Gray Box**         | Attacker is assumed to have:  
  - Partial physical access to the cryptographic key as a result of the cipher leaking side channel information  
    - Electromagnetic radiation analysis  
    - Current/power consumption analysis.  
    - Operation timing analysis  
  - Most ciphers have published side channel information that fully reveal the cryptographic key or reduce its key size |
| **White Box**        | Attacker is assumed to have:  
  - Full visibility – inputs, outputs, memory (using debuggers), and intermediate calculations  
  - Access to the algorithms while watching how they are carried out  
  **Traditional cryptography** is not secure while running in a White box model  
  **White box cryptography** integrates the cipher in a way that does not reveal the key and is therefore highly secure |
THE FULLY TRANSPARENT ENVIRONMENT WAS THEREFORE CALLING FOR A NEW TYPE OF CRYPTOGRAPHIC APPROACH ALSO CALLED WHITE BOX CRYPTOGRAPHY THAT RAISED THE FOLLOWING CHALLENGES:

• How to encrypt or decrypt content without directly revealing any portion of the key?
• How to perform strong encryption mechanisms knowing that hackers can observe and alter the code during execution?

White box cryptography went head to head with traditional security models. With traditional black box implementations, the attacker only had access to inputs and outputs and to the cryptographic algorithm under attack, but had zero visibility into internal workings. White box cryptography provides full visibility instead. It aims to protect implementations of cryptographic algorithms in software against key recovery attacks conducted while operating in a fully transparent potentially hostile execution environment.

How is it then possible to securely “hide” the key within the executed code assuming that one can fully monitor and alter each and every instruction? Abstractly speaking, this is achieved by combining the secret key with data derived from a mathematical operation that is virtually impossible to invert1 .

This implementation allows developers to build a system that operates much like a full public/private key scheme, but much closer to a standard symmetric cipher in performance. The decryption function can be implemented inside the distributed application, but the key itself cannot be extracted and the decryption cannot be reversed, which enforces the encryption operation correctly.

Although the white box scenario is considered unsuitable for security-related tasks, White box cryptography shuffles all the cards and provides a highly secure method for performing encryption while operating in a fully transparent environment. Both encrypt and decrypt operations maintain sensitive data without revealing any portions of the key. In addition, White box cryptography permits the execution of strong encryption mechanisms (in conjunction with other techniques), whilst assuming malicious eyes constantly scrutinize the code during execution.

Conclusion: Security comes at a certain cost and as a direct result cannot be air tight. One must understand that the overall security of a protected application is highly dependent on the implementation itself, i.e., solely using a strong cryptographic algorithm does not provide any security if it is not used in the context it was designed for. Using traditional cryptography in a white box environment greatly increases the risk of attackers reverse-engineering the protected software and overcoming its “secure channel” and as a result are they able to extract the session key from memory and “peak” into the encrypted channel.

In the past, most common attacks have attempted to exploit software security flaws and not weaknesses in the cryptographic algorithms, but lately attackers have recognized the vulnerability of classical cryptography in the open PC environment, and are exploiting them. It is implicit that software protection must receive specific attention throughout the design and implementation stages in addition to being continuously enhance as part of the product life cycle and the release of new versions. In addition to White box cryptography, additional complementary security measures should be used to further strengthen the overall protection scheme.

The article is written by Assaf Ragev of SafeNet, which is a leading global provider of data protection. For over 25 years, Fortune 500 global corporations and government agencies have turned to SafeNet to secure and protect their most valuable data assets and intellectual property. SafeNet’s data-centric approach focuses on the protection of high value information throughout its lifecycle, from the data center to the cloud. More than 25,000 customers across commercial enterprises and government agencies trust SafeNet to protect and control access to sensitive data, manage risk, ensure compliance, and secure virtual and cloud environments.

Assaf is part of SafeNet’s software monetization solutions team; SafeNet offers the industry’s strongest, most flexible software licensing and management solutions. The Sentinel portfolio provides award-winning IP protection, software licensing, and entitlement management technology that minimizes software piracy risks and enables flexible licensing, pricing, and packaging models that help software developers create new revenue opportunities and improve customer satisfaction. To learn more about SafeNet’s cloud-based licensing and entitlement management service, visit www.sentinelcloud.com. To learn more about SafeNet’s full portfolio of software monetization solutions, visit www.safenet-inc.com/sentinel.

1. Amitabh Saxena, Brecht Wyseur, and Bart Preneel, Towards Security Notions for White Box Cryptography