Secure Coding: Phase 4 Report

Elias Tatros
elias.tatros@cs.tum.edu

Chair XXII Software Engineering
Department of Computer Science, Technische Universität München

Abstract. short description/overview

Keywords: CAN, automotive security

1 Introduction

About the structure of this report
Short overview over the structure of contents

2 Memory Tampering Vulnerabilities

- Identify possible stale memory locations
- Possible to change memory contents based in specific location?
- Recommend address space layout randomization
- Recommend further checks for memory tampering

2.1 PHP & JavaScript Code Review

- PHPSecAudit: no luck
- RATS: didn't get it to work
- RIPS: found a few false positives, maybe 1 of interest, see screenshots

2.2 CWE-329: Use of static IV with Rijndael CBC

Next9 bank makes use of a 512 bit hash value scs_{seed}. This value is used to generate TANs for the smart card simulator. It is recalculated after each successful transaction. The scs_{seed} is generated on the server side and then encrypted using the Rijndael algorithm with a block size of 128 bits and a 128 bit key k (according to the AES-128 standard):

$$enc_{k_{enc}}(scs_{seed}).$$

The encrypted 512 bit hash is saved to a file, which is then offered to the user for download.

The user is supposed to download the file and use it as input for the smart card simulator, which uses it to generate a TAN that can be used to commit transactions. The user is asked to enter his 6 digit SCS PIN, which is utilized by the SCS to generate the decryption key k_{dec}. The key is then used to decrypt the 512 bit hash provided by the user. The original scs_{seed} is obtained if $k_{enc} = k_{dec}$.
- The IV is static (fedcba9876543210)
- A randomly generated IV needs to be used

```
$iv_size = mcrypt_get_iv_size(MCRYPT_RIJNDAEL_128, MCRYPT_MODE_CBC);
$iv = mcrypt_create_iv($iv_size, MCRYPT_RAND);
```

quote:
"Things are less dire if you use CBC. In CBC, the data itself is broken into 16-byte blocks. When a block is to be encrypted, it is first XORed with the previous encrypted block. The IV has the role of the -1 block (the previous encrypted block for the first block). The main consequence of reusing the IV is that if two messages begin with the same sequence of bytes then the encrypted messages will also be identical for a few blocks. This leaks data and opens the possibility of some attacks.

To sum up, do not do that. Using the same IV with the same key ever and ever defeats the whole purpose of the IV, the reason why a chaining mode with IV was used in the first place."

2.3 Weakness in key generation and padding

The effectiveness of a symmetric encryption algorithm in terms of the provided confidentiality is always depended on the key size. An advanced encryption standard, such as AES is not a silver bullet. The security of symmetric encryption algorithms is quickly compromised if configured with insufficient care.

The next9 bank encryption key is generated by taking the users SCS password (a 6 character number) and then padding it with the character sequence "0000000000". The key is not derived from the password in any way, but rather the key is the actual password plus 80 bits of 0 padding. Therefore the meaningful bits of the key are reduced to the size of the password, which is 6 characters or 48 bits long. Even if all bits were perfectly random and uniformly distributed, we would already consider this to be a pretty weak key.

However, we also know that the SCS password can only consists of digits, which drastically reduces the entropy of the key even further. A 48 bit key can result in 2^{48} different values. When doing encryption it is important that every single one of these values has the same probabilty of ocurrence. This is what CSPRNGs (cryptographically secure pseudo-random number generator) are usually used for. In the optimal case a 48 bit key produced by a CSPRNG would have an entropy of 48 bits.

However, in this case the bits in the key are not uniformly distributed. In fact, for a 6 digit number each digit can take on a value between 0 to 10, resulting in a total of 10^6 or 1 million values. Such a string gives 20 bits of entropy in the

best case, meaning if we assume that each digit in the SCS password is chosen completeley randomly.

The method of key generation used significantly reduces the strength of the key and consequently the effectiveness of the entire encryption. A 48 bit key with only 20 bits of entropy does not measure up to todays encryption standards.

A quick fix for this issue would be to remove the 0 padding and generate the key by running the SCS password through a cryptographic hash function, such as HMAC. In a final step the output would be reduced to the first 128 bits, resulting in a more secure key.

References

1. Robert Bosch GmbH: CAN main specifications, at www.can.bosch.com, (2005)
2. K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Schacham, and S. Savage: Experimental security analysis of a modern automobile. In D. Evans and G. Vigna, editors, IEEE Symposium on Security and Privacy. IEEE Computer Society, (2010)
3. OpenGarages and C. Smith: Car Hackers Handbook, pages 26-27, (2014), at http://opengarages.org/handbook/
4. S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner and T. Kohno: Comprehensive Experimental Analyses of Automotive Attack Surfaces In USENIX Security, August 10–12, (2011).
5. M. Wolf, A. Weimerskirch and T. Wollinger: State of the Art: Embedding Security in Vehicles, In EURASIP Journal on Embedded Systems, Volume 2007, Article ID 74706, (2007).
6. P. Papadimitratos , L. Buttyan , T. Holczer , E. Schoch , J. Freudiger , M. Raya , Z. Ma , F. Kargl , A. Kung and J.-P. Hubaux: Secure vehicular communication systems: Design and architecture, In IEEE Commun. Mag., vol. 46, no. 11, pp.100 -109, (2008)
7. M. Wolf, A. Weimerskirch, C. Paar, L. Kerstin: Secure In-Vehicle Communication, In Embedded Security in Cars, Springer Berlin Heidelberg, ISBN 978-3-540-28384-3, pp 95-109, (2006)i
8. F. Kargl, P. Papadimitratos, L. Buttyan, M. Muter, E. Schoch, B. Wiedersheim, T.-V. Thong, G. Calandriello, A. Held, A. Kung and J.-P. Hubaux: Secure vehicular communication systems: implementation, performance, and research challenges, In IEEE Communications Magazine, vol. 46, no. 11, pp. 110-118, (2008)
9. T. Ziermann, S. Wildermann and J. Teich: Can+: A new backward-compatible controller area network (CAN) protocol with up to 16x higher data rates, In Design, automation & test in Europe conference & exhibition, p. 1088-1093, (2009)
10. D.K. Nilsson, L. Sun and T. Nakajima: A framework for self-verification of Firmware updates over the air in vehicle ECUs, In Proc. IEEE GLOBECOM Workshops, pages 1-5, (2008)
11. A. Van Herrewege, D. Singelee and I. Verbauwhede: CanAuth - a simple, backward compatible broadcast authentication protocol for CAN bus, In: 9th Embedded Security in Cars Conference, (2011)
12. B. Groza, S. Murvay, A. Van Herrewege and I. Verbauwhede: LiBrA-CAN: A Lightweight Broadcast Authentication Protocol for Controller Area Networks In Lecture Notes in Computer Science, Springer Berlin Heidelberg, p. 185-200, (2012)

13. O. Henniger, A. Ruddle, H. Seudie, B. Weyl, M. Wolf and T. Wollinger: Securing vehicular on-board IT systems: The evita project, In 25th VDI/VW Automotive Security Conference, Ingolstadt, Germany, (2009)
14. A. Hazem and Hossam A. H. Fahmy: LCAP - A Lightweight CAN Authentication Protocol for Securing In-Vehicle Networks, In Embedded Security in Cars Conference, (2012)
15. M. Pleil and M. Busse: D1.2-10 data exchange concepts for gateways, Technical report, EASIS1, (2006)