Cracking Credit Card Number Tokenization

Li-Hsiang Kuo
University of Wisconsin-Madison
lorderic@cs.wisc.edu

Abstract

As the popularity of online shopping grows, online payment security becomes an important issue. There are third-party service providers who aim to handle sensitive personal data for the merchants. Tokenization is one of the commonly use technology. The service providers take care of sensitive cardholder information, while tokens are stored in the merchants storage. In this paper we point out the weaknesses in tokenization nowaday, both in standard and in implementation. We study the standards defined by payment card industry. We study the token system of an online payment service provider called *Sage Pay*. From weak standard and flawed token system implementation, we provide a possible attack that can retrieve credit card number by its token, which breaks the goal of tokenization. We conclude that the payment card industry should strengthen their standards and the secure payment service providers also should avoid flawed implementation.

1 Introduction

As the popularity of online shopping grows, online payment security becomes an important issue. Millions of credit card numbers are stored and sent on the internet everyday. However, it leads to a great danger if these sensitive data are keep or transmitted insecurely. Malicious ones might break into a merchants server and obtain thousands of credit card numbers as well as corresponding information, e.g. cardholders name, expiry date, card verification value, etc. The payment card industries (PCI) foreseen this problem. They setup a standard called PCI Data Security Standards (PCI DSS) [1], which provides a framework for developing a robust payment card data security process. All entities involved in payment card processing, including merchants, processors, acquirers, issuers, and service providers, are asked to meet the standard. This increases the security of payment card transactions.

However, it is quite expensive for each merchant to maintain a PCI DSS compliance server. It is also risky if the server has security flaws. Therefore, new enterprises arise. They provide services for secure online payment gateway, which meets the requirement of PCI DSS. Instead of building its own secure server, the merchants rely on these service providers. The online transaction goes this way: While a customer shops online, the service provider deals with sensitive personal data. Instead of the payment card number, a token is generated by service provider and returned to the merchant for further transaction. Tokens are stored in the merchants system while the actual cardholder data are stored in a secure token storage system.

One should not be able to compute the actual payment card number by its token. However, we figure out that the standard does not meet the security level. Whats more, bad implementation in tokenization might also be harmful. In this paper we will point out these weaknesses. In section 2 we give a brief overview of tokenization standards. We focus on Visa best practices for tokenization[2] and discuss its vulnerability. Section 3 provides a case study of a secure online payment service provider called *Sage Pay*[3]. We indicate its protocol flaws and try to mount a possible attack. We discuss some related work in section 4, and this paper is concluded in section 5.

2 Tokenization Standard

2.1 PCI DSS

Payment Card Industry Data Security Standard (PCI DSS) [1] is a standard for entities that handle payment card information. It is defined by the Payment Card Industry Security Standards Council. The current version

of the standard is version 2.0, released on 26 October 2010. The standard specifies 12 requirements for compliance. Specifically, requirement #3 defines protecting stored cardholders data. The main idea is that primary account number (e.g. credit card number) should be unreadable in the storage. It could only be stored in the following approaches:

- One-way hashes based on strong cryptography
- Truncation
- Index tokens and pads
- Strong cryptography with associated key management processes and procedures

Requirement #4 defines encrypting transmission of cardholders data across open, public networks. It is required to use strong cryptography and security protocols (e.g. SSL/TLS, IPSEC, SSH) to safeguard sensitive cardholder data during transmission over public networks.

PCI DSS does not specify the security level on cryptographic systems. However, the payment card industries do have some practice on protecting cardholders data. The following section gives what Visa recommends for tokenization.

2.2 Visa Best Practices For Tokenization

Visa best practices for tokenization [2] version 1.0 is released on 14 July 2010. Two token generation methods are recommended:

- A known strong cryptographic algorithm
- A one-way irreversible function

Also tokens can be generated as a single- or multi-use value, the choice of which depends on business processes. For single-use token, it recommends hashing of the primary account number with a transaction-unique salt value using a unique sequence number, or encrypting with an ISO- or ANSI-approved encryption algorithm using a transaction-unique key. For multi-use token, it recommends hashing of the data using a fixed but unique salt value per merchant.

If a hash function is used to generate a token, the salt value should have minimum length of 64 bits. If an encryption algorithm is used, the minimum key length is defines as following:

- Two-key TDES: 112 bit
- AES: 128 bit
- RSA: 2048 bit
- ECC: 224 bit
- SHA: 224 bit

Moreover, while the number of transactions is more than 1 million, two-key TDES is not suitable. Instead, three-key TDES or AES is suggested.

2.3 Vulnerability

Since hash function is deterministic, in order to avoid dictionary attacks, a salt should be added to increase security. But for multi-use token, VISA recommends using the same salt per merchant. Re-use the same salt does not improve any computational complexity. If the salt for a merchant has been discovered, it is feasible to compute any payment card number by its hash value using dictionary attack. We will discuss more detail in section 3.3.

Second, a 64-bit salt might not be sufficient. NIST recommended a minimum security strength in 2010 is 80 bits [4]. ECRYPT II also comments 64 bit security should not be used for confidentially in new systems [5]. Since the computation power increases rapidly, it is possible that in a few years 64-bit security can be attacked in real-time by highly parallel systems.

3 Case Study

3.1 Sage Pay

Sage Pay [3] is a online payment service provider in United Kingdom. They deal with thousands of businesses by supporting online payment gateway that is PCI DSS compliance. They also provide token system for online merchants. Everyone can sign up on their website and download the development kit. Their protocol implementation documents are also available online. From these resource we discuss their token system protocol in the following.

3.2 Sage Pay Token System Protocol

Here is the step-by-step process of token registration in *Sage Pay* system.

1. While Shopper visit Merchants website, Shopper selects some product and checks out.
2. Merchant makes a HTTP POST to *Sage Pay* server, which includes information such as Merchant ID, type = TOKEN and a NotificationURL in either http or https header.

3. *Sage Pay* checks the message. If the message is verified, sends back an HTTP response to Merchant with status = OK, a key to generate further signature, and a NextURL which Merchant must redirect Customer to enter card details.

4. Customer is redirected to NextURL and enter sensitive personal information.

5. *Sage Pay* validates the entry of these data. If the data are valid, then *Sage Pay* will store the card number, expiry date, card verification value and generate a token.

6. *Sage Pay* sends a Notification POST to NotificationURL. Message includes status = OK, token, card type, last 4 digits, expiry date, and a signature of the message using the key in step 3.

7. Merchant replies with a RedirectURL.

8. *Sage Pay* redirects Customer to the RedirectURL.

When Customer makes a payment in the future, Merchant fetches the token in database and sends to *Sage Pay*. Sage pay will retrieve the card information corresponding to the token and continue the transaction with Merchants bank. We omit the step-by-step token using process here.

3.3 Possible Attack

Observe that a NotificationURL provided by Merchant can be either http or https. Which means the message including token, payment card type, last 4 digit of card number, expiry date could be all sent in plaintext. Although *Sage Pay* do not specify what mechanism they are using to generate a token, its possible that they follow VISA best practices for tokenization, i.e. hashing the data using a fixed salt value per merchant.

Hence an attacker can work this way: He registers his card information on Merchants website. By listening to the traffic of Merchant, he is able to get the token generated by *Sage Pay*. Once he has the payment card number, the hash function *Sage Pay* is using, and token generated by hashing with a Merchant unique salt, he can exhausted search for the 64-bit salt value using highly parallel system. It might take months or even years for now, but its possible be done in days in a near future.

After the Merchant unique salt is found, the attacker can crack any token for Merchant. A credit card number is in between 15-16 numerical digits. Last 4 digit could be known by Notification message, first 1-2 digits could be know by payment card type, and 1 digit is determined by checksum. That leads to less than 10 digits computing complexity, approximate 32 bits security. It can be easily cracked in real-time. [5]

4 Related Work

4.1 Breaking Point-of-Service Device

Murdoch et al. [6] found a vulnerability in the EMV point-of-service (POS) devices. They mount a man-in-the-middle attack between POS device and EMV card. The attacker tells the POS device that the PIN entered is correct, while on the other hand the attacker tells the card that the POS device does not have PIN verification. Since EMV support offline transaction, i.e. it is not connected to card issuer so that the transaction is not verified at the time payment happens, the attacker can cheat both POS device and the card and shop for free.

4.2 Breaking Protocol

Wang et al. [7] found that theres inconsistencies between the states of the online cashier service provider and the merchant. They present several attacks against real-world merchant applications, online stores and online cashier service provider. The result shows that by exploit these flaws an attacker can either purchase an item at a lower price, shop for free after paying for one item and even avoid payment.

5 Conclusion

In this paper we gave a brief overview of current standards for payment card industry. We discussed the vulnerability of Visa best practices of tokenization. We studied a real-world token system and pointed out its weakness. At the end we constructed a possible attack against this system. Our result showed that from weak standard and flawed token system, it is possible to crack one credit card token in months, afterwards break the rest in real-time. The payment card industry should strengthen their standards and the secure payment service providers also should avoid flawed implementation.

References

[1] PCI Security Standards Council. Payment card industry data security standards.

[2] VISA. Visa best practices for tokenization.

[3] Sage Pay. Sage pay. http://www.sagepay.com.

[4] NIST. Nist special publication 800-131a.

[5] ECRYPT II. Ecrypt ii yearly report on algorithms and keysizes, 2009-2010.

[6] Steven Murdoch, Saar Drimer, Rose Anderson, and Mike Bond. Chip and pin is broken. In *IEEE Symposium on Security and Privacy*, 2010.

[7] Rui Wang, Shuo Chen, XiaoFeng Wang, and Shaz Qadeer. How to shop for free online. In *IEEE Symposium on Security and Privacy*, 2011.