


CORSAIRE WHITE PAPER
ATTACKING MAGSTRIPE GIFT CARDS
WITH A FOCUS ON UNAUTHORISED PURCHASES

Reference	Corsaire Whitepaper Gift Cards.doc
Author	Adrian Pastor
Date	21 October 2009
Distribution	Public Release

Attacking Magstripe Gift Cards

Table of Contents

Attacking Magstripe Gift Cards

Executive Summary

Credit card security has been steadily evolving in response to the various threats to the industry related to card fraud. The ubiquitous magstripe has been modified, revised and is becoming redundant in most situations as smartcards and technologies such as Chip and PIN are rolled out globally.

But the magstripe is far from retired and fraud against it is still present. Aside from the 'fallback' attacks plaguing the payment card industries in regions that have not moved to the securer alternatives, magstripes and their security flaws are finding new homes as virtual cash associated with store gift and loyalty cards.

Gift cards – sometimes referred to as *loyalty* or *stored-value* cards – are on the rise[i]-[iv]. Countries such as the US and UK provide great examples of markets where this type of currency is exploding.

Gift cards can be purchased from just about any type of merchant chain, and can be found everywhere; supermarkets, coffee-shops, restaurants, cinemas, department stores, beauty and hair-dressing chains. At the time of writing, one UK gift card collector[v] catalogued over 1600 different types of gift cards from more than 186 retailers.

However, as with any other form of currency, gift cards are subject to fraud.

"Magstripes and their security flaws are finding new homes as virtual cash… as with any other form of currency, gift cards are subject to fraud"

This paper is based on research conducted on a large number of UK gift cards. It has been created to complement the presentation "Stored Value Gift Cards: Magstripes Revisited", which was delivered at the EUSecWest security conference in London in May 2009. It concentrates on magnetic stripe (magstripe) gift card attack techniques and also provides a series of guidelines and tips for developers and systems architects who are involved in the process of implementing their own gift card technology.

Attacking Magstripe Gift Cards

Introduction

Like any other form of currency, gift cards are subject to fraud. Fortunately, there are significant restrictions that apply to gift cards which limit the potential for fraud, which do not apply to other types of prepaid cards.

Unlike other types of pre-paid cards (e.g. prepaid credit card), gift cards are **closed-loop cards**, which means that they can only be used within a certain retail chain. The exception to this rule is that some retailers have established partnerships with other retail chains which allow their customers to use the same gift cards within either retail chain (although in effect this is just a larger closed-loop).

Although there are some differences between stored-value loyalty cards and gift cards, from a functional perspective, they are the same: both are prepaid cards. Once purchased, the amount paid by the customer is associated with the card's number in back-end systems. For the purpose of this paper, the term "gift cards" will be used to refer to both, stored-value gift cards and loyalty cards.

Another factor that minimises fraud against gift cards is that issuers of gift cards do not generally allow for cash refunds in the Terms and Conditions of the contract with the customer[1]. This means that once a gift card has been purchased, it is not usually possible for a legitimate customer (or attacker) to turn the remaining credit into cash. Instead, the remaining balance can only be used to purchase goods available at the retail chain where the gift card was obtained from.

Nevertheless, this paper will demonstrate how gift cards are still subject to attacks, which are sometimes practical depending on how a particular gift card technology has been implemented.

This paper discusses ways to attack magnetic stripe (magstripe) gift cards. The reason why this research has focused on magstripe gift cards, as opposed to other types of gift cards (e.g. barcode-based), is because this is currently the most common type in the UK, the country of origin of the majority of gift-cards analysed from a number of different retail chains.

This paper explores traditional attacks such as skimming, and also, lesser-known attacks such as *crafting the track data (cloning) of gift cards without ever swiping magnetic stripes*. It also focuses on certain factors that can make attacks against magstripe gift cards more feasible, when compared to magstripes used for

[1] There might be instances where it is actually possible to indirectly turn the gift card balance into cash by purchasing goods and requesting a statutory refund. See http://www.dca.ca.gov/publications/legal_guides/s-11.shtml for further information.

other applications (e.g. parking tickets). It concludes with a series of recommendations to help service providers implement their store-value loyalty / gift card technology in a more secure manner.

The intended target audience of this paper is providers of gift card and loyalty card programs, and penetration testers who need to assess the security of a certain gift card technology and its underlying infrastructure. Familiarity with the basics of magstripe technology (defined in ISO 7811), and the common retailer card schemes would benefit the reader. Several online resources have been included under the *References* section which introduces readers to the basics of such topics.

General Findings

Some readers familiar with attacks against stored-value cards might expect this paper to discuss how to change the balance stored on the card, in order to perform unauthorised transactions when the card is swiped at the POS terminal. A recent example of this type of attack was presented against the CharlieTicket[vi] (used in the Boston metro system) where it was possible to change the card's balance by modifying the "balance" field on the magnetic stripe, and updating the checksum field. However, this type of attack is not the main focus of this paper.

Although some types of stored-value magstripe cards store the card's balance on the actual track data, the gift cards from the UK analysed for this paper did *not* store the card's current balance on the card. Therefore, in this scenario, it would not typically be possible for an attacker to purchase goods for free by counterfeiting a magstripe card with specially-crafted data on it. However, it is feasible that a card's balance could be changed by writing specially-crafted data to the magnetic stripe without a balance even being stored in the card; this is a separate class of attack and is discussed in the *Attack Classes* section of this paper.

There are several valid reasons why gift card vendors usually choose *not* to store the card's balance on the magnetic stripe, including:

- This design decision could introduce a serious security issue, where an attacker is able to fool the transaction system by changing the card's balance with a magstripe encoder/writer.

- Gift card vendors now promote the integration of their technology with existing POS terminals, to avoid the need for merchants to invest in additional magstripe writer equipment. These POS terminals only come with reading capabilities, which means that it would not be possible for them to update the card's balance by writing a new value on the magstripe. Instead, when the gift card is swiped, an **online transaction** is made with the back-end servers of the gift card program provider.

Factors that are Unique to Attacking Magstripe Gift Cards

There are several factors which are specific to magstripe gift cards which do not apply when attacking magstripe cards used in other applications such as hotel room keys or parking passes.

Many retailers provide gift cards to customers at an accessible location (store stand) for anyone to take before being activated. The reason why some retailers do not consider the loss of gift cards to be a security issue is because the cards are *not* activated unless purchased. The data on the magnetic tracks does *not* change before and after being activated with a certain balance. Instead, an online transaction is performed which causes the back-end servers of the gift card provider to update the balance for that given card number, and flag the card number as being activated (if the card had not previously been activated).

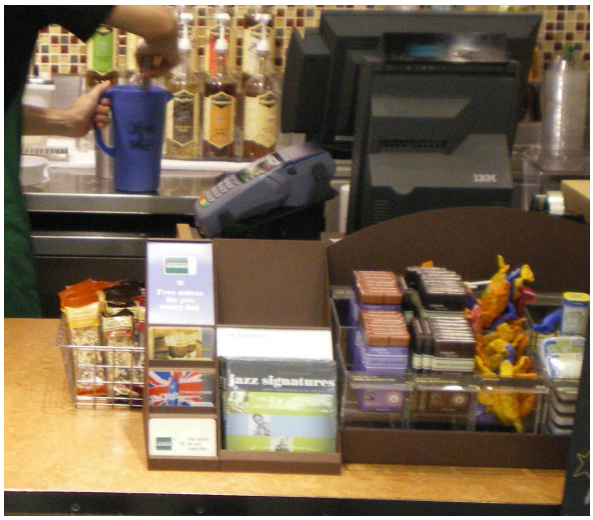


Figure 1 Gift cards stand at a food and beverage retailer

Figure 2 Gift cards stand at beauty products retailer

Figure 3 Gift cards stand at a clothing retailer

Even in cases when gift cards must be paid for before being taken, some types of gift cards can be activated for as little as £1 GBP which allows attackers to collect samples for a relatively low cost.

Also, sometimes different retail chains are using the same gift card provider. This is of course possible as most gift card providers offer their gift card technology to any retailer willing to pay the necessary fees for their services. The implication of outsourcing these services is that if an attacker found a security vulnerability in a given gift card implementation, they would be able to exploit the same vulnerability against any retailer using the same gift card technology. i.e. outsourcing to the same gift card provider.

Attacking Magstripe Gift Cards

There are other unique factors to take into account when attacking magstripe gift cards, many of which are summarised in the tables below, from an attacker's perspective:

Attack Advantages	***Attack Disadvantages***
Ease of access to the cards prior to purchase increases the likelihood of skimming (cloning by physically swiping the magstripe).	Attackers cannot exchange remaining balances into cash. Note: as an exception, it might be possible to indirectly turn the balance into cash as previously explained in the *Introduction* section.
Similarly, this ease of access allows writing data of cloned cards on them, which results in genuine-looking cards with original branding. This saves an attacker from spending money on specialised equipment such as thermal printers which would usually be required to create genuine-looking counterfeit cards. Note: data on magnetic stripes can be rewritten using appropriate magstripe-writing equipment.	Some gift cards automatically expire after a certain period from the time of purchase or after a certain period of inactivity, thus decreasing the attack time window.
If a security vulnerability is discovered in a given gift card implementation, the same issue might apply to several retail chains. This would be true in cases where the same gift cards' service provider is being used by several retail chains.	There are restrictions on the maximum amount allowed on a certain gift card. The maximum amount typically ranges anywhere from £100 to £500 in the UK allthough some retailers will allow up to £1000 on each card.
Cloning a gift card that has been activated might provide "free" goods for life to an attacker. This is possible because some gift card providers allow customers to perform **automatic top-ups** after registering their card online (credit card details need to be provided for this feature to be enabled).	Attackers need to ensure that the gift card which has been cloned is activated and still has credit left on it. Otherwise, items cannot be purchased.

Table 1 – Attack advantages and disadvantage to targeting magstripe gift cards

Attack Classes

Once again, this paper is based on research conducted on a large number of UK gift cards. The sample set of cards selected during the research shared a common characteristic: *none stored the card's balance on the track data*. This is a common security measure to avoid balance-manipulation attacks by tampering with the track data. The following attacks show that magstripe gift cards may be subjected to fraudulent use; the goal is always the same; to purchase goods for "free".

In-Band Cloning

This would typically be achieved by *skimming* a gift card by swiping the magnetic stripe.

This is perhaps the most obvious type of attack against a magstripe gift card. In short, an attacker swipes the targeted gift card and then produces a counterfeit version of it. The attacker's profile would most likely be a malicious employee, customer or other party who could gain physical access to a gift card. As many retailers make gift cards easily accessible before being purchased, it would be highly likely for a member of the public to conduct this attack.

The reason why gift cards are often available for anyone to take before being purchased is because unless they are activated, they have no monetary value associated to them. At this stage, retailers don't care if one is lost or stolen as the costs to replace one are negligible. And, even if a gift card was cloned, an attacker would not to be able to purchase goods as it was never activated with any credit.

Or so they think! To get around this an attacker simply puts the cloned card back on the stand. Then, with a bit of luck it will eventually be purchased and activated by someone else. At this point an attacker can purchase goods with the cloned card, which has ultimately been paid for by the victim customer.

The only challenge to overcome is that an attacker must be able to detect when the cloned card has been activated by a legitimate customer. However, there are several ways to do this. The first method is to simply visit one of the retail stores and ask a member of staff to check the balance on the card. The attacker can then find out if the card has been activated and, if it has been activated, what the remaining balance is. The second method involves querying balance-checking facilities available over the web. This is a feature that is currently being provided by most UK retailers that offer gift card programs. It is also interesting to note that some of the very few retailers that still do not offer this facility are planning to do so in the near future.

A possible challenge an attacker might face when checking the balance of the cloned card online is that a PIN is required. This is a feature that a number of gift cards implementations do not support. For those gift

cards that require a PIN to check the balance online, the PIN is usually located on the back of the card and covered with a protective coating which can be easily removed (scratched off). This effectively stops an attacker from checking the balance of the cloned gift card online since they would have to scratch off the protective coating on the legitimate card before putting it back on the retailer's stand. Of course, such obvious tampering is likely to alert the majority of customers when selecting or purchasing a gift card for their own use.

However, some online balance-checking facilities leak the fact that a given card number has been activated or not, even when a PIN is required, for example through variation in the error messages as described below. These changes of error messages can occur even when the attacker provides the wrong PIN.

This means that an attacker can sometimes still find out when the cloned card has been activated by a legitimate customer, even though he/she would still not know the remaining balance. By using common sense, an attacker could guess that a card that has recently been activated is more likely to have a remaining balance, than a card that was activated a significant period of time ago. An example of this attack is described in the *Cloning a gift card by abusing web balance-checking facilities* section where we demonstrate how an attacker can programmatically obtain numbers of cards which have been activated even when a PIN is required to check the card's balance.

Attack Advantages	***Attack Disadvantages***
Relatively easy to perform; the track data can be obtained with inexpensive standard ISO 7811-compatible equipment.	Physical access to the targeted gift card is required. However, this might not be a significant deterrent in cases where gift cards are easily accessible. Card reading is a low-cost attack, but writing cloned cards typically requires a higher investment.

Out-of-Band Cloning

In this case, closing the card would be achieved by obtaining the card number from a separate source, for example writing down the card number; skimming of the actual card would not be required.

There are also other ways an attacker could obtain a valid card number that are discussed in the following sections. For example, a valid card number could be obtained by simply examining the back of a targeted card before it is sold while still available at the store stand. Sometimes, unmasked card numbers can be found on disposed receipts or registration forms available at the store where the gift card was originally sold. Although most retailers mask gift card numbers partially on receipts, there are still some who do not. The following is an example:

CUSTOMER COPY

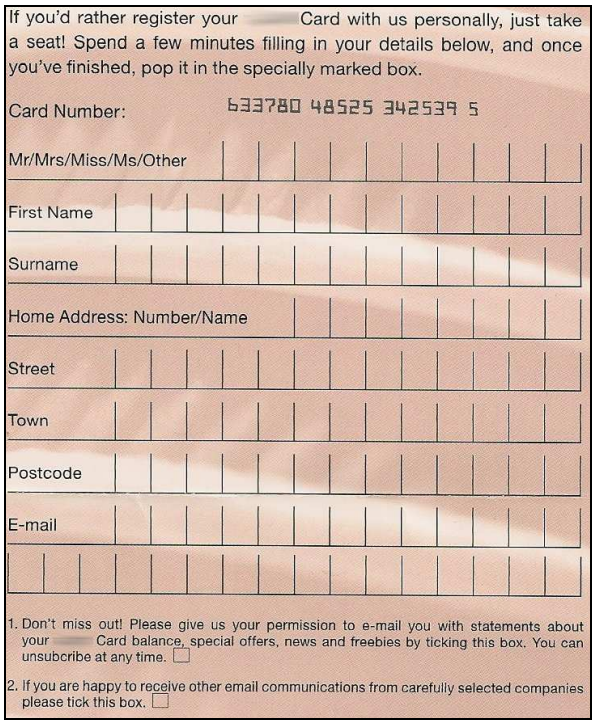
Assistant
06/06/09 18:46 36616

You have been served today by:

Gift Card 10.00
6050963192534632
Gift Card 10.00
6050963193599315

Figure 4 Receipt showing 2 complete gift card numbers without being masked

The following is a gift card registration form; once again the card number is fully shown without being partially masked:



If you'd rather register your Card with us personally, just take a seat! Spend a few minutes filling in your details below, and once you've finished, pop it in the specially marked box.

Card Number: 633780 48525 342539 5

Mr/Mrs/Miss/Ms/Other

First Name

Surname

Home Address: Number/Name

Street

Town

Postcode

E-mail

1. Don't miss out! Please give us your permission to e-mail you with statements about your Card balance, special offers, news and freebies by ticking this box. You can unsubcribe at any time. ☐

2. If you are happy to receive other email communications from carefully selected companies please tick this box. ☐

Figure 5 Registration form showing complete card number without being masked

In some gift card implementations it is possible to generate the complete track data from a given card number. In other words, *it is sometimes possible to clone a gift card without ever swiping the magnetic stripe of the targeted card*.

The simplest example of this type of implementation is a gift card which includes the card number in Track 2, plus other standard data which is usually stored on cards that follow the ISO 7811 standards for Track 2 (BCD encoding). Since Track 2 is usually read by default when swiping a card at POS terminals, an attacker does not need to worry about the data encoded in Track 1 (if applicable). The following is an example of Track 2 data found on a gift card:

```
;6036281797197487672 5?7
```

Note: card number shown in bold font.

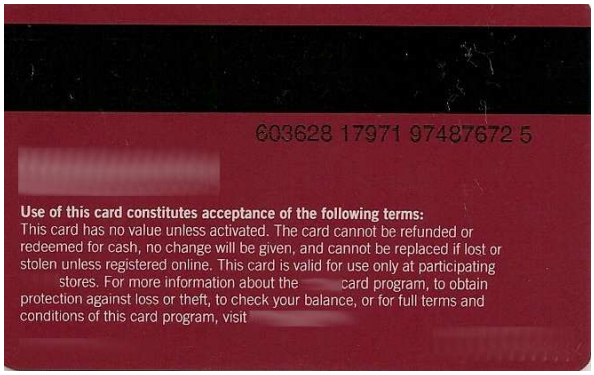


Figure 6 Back of card corresponding to previous track data

Visible from left to right is:

1. Start sentinel character: **;**
2. Card number: **6036281797197487672 5** (written on back of card)
3. End sentinel character: **?**
4. LRC (error-checking byte): **7** in this case but varies depending on rest of data

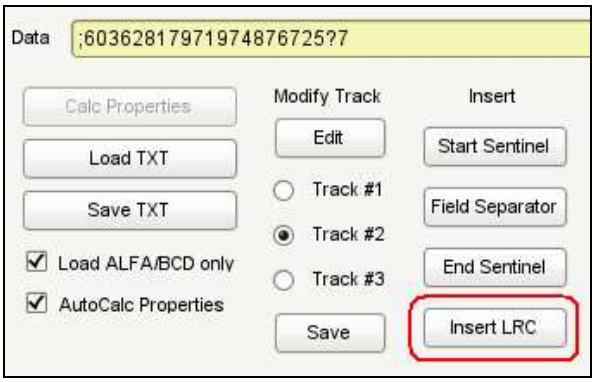


Figure 7 Cloning targeted card number without skimming using MAKStripeExplorer v1.22

Using the appropriate magstripe-writing software an attacker can craft all the Track 2 data from the card number, including the LRC character, as shown in the previous screenshot.

However, other gift cards implementations may not be so easily cloned. For instance, some implementations include a **magic number** in the track data. A magic number is just a random number or checksum value which cannot be predicted, or at least which is not trivial to predict. By including a random number in the track data, an attacker would be unable to craft the full track data from a given card number. Sometimes, the magic number is located in what ISO 7811 refers to as the "discretionary data" field.

The magic number would only add real security if *both* of the following requirements were met:

1. The back-end servers require the correct magic number for the transaction to be processed successfully

2. The magic number cannot be predicted by the attacker

For instance, if instead of including the correct magic number, an attacker decides to write a different number on Track 2 and the transaction still completes successfully, then the magic number would not add any additional security. Also, if the magic number can be predicted by an attacker – e.g. based on a publicly-known checksum algorithm – it would not add any security benefit either.

The following is an example of Track 2 of a gift card which includes a magic number:

5045075645502551155=1612110**93621576**?0

Note: card number shown in bold font; magic number shown in red font.

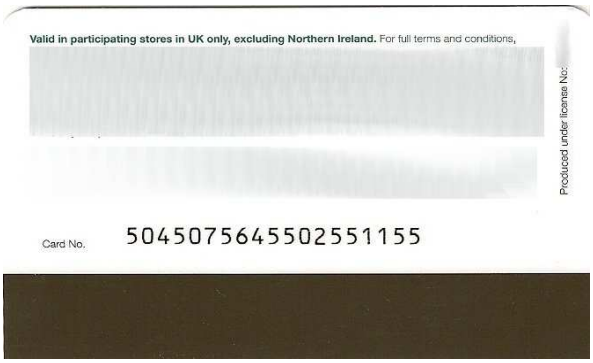


Figure 8 Back of card corresponding to previous track data

Visible from left to right is:

1. Start sentinel character: **;**
2. Gift card number: **5045075645502551155** (written on back of card)
3. Field separator: **=**
4. Expiry date: **1612** (seemed constant across different instances of same type of gift card)
5. Service code: **110** (also seemed constant)
6. Discretionary data (magic number): **93621576** (varies for each card number)
7. End sentinel character: **?**
8. LRC (error-checking byte): **0**

Attack Advantages	***Attack Disadvantages***
An attacker can simply record card numbers and does not require magstripe reading equipment. This could be beneficial in cases where several individuals are involved in carrying out the cloning activities. For instance, a malicious retail employee could sell a daily list of gift card numbers to someone who would then have the means to write the track data and detect when the targeted cards have been activated by legitimate customers.	Not all gift card implementations are vulnerable to this type of cloning attack. It is a more targeted attack.

Cloning a Gift Card by Abusing Web Balance Checking Facilities

This is perhaps the most elegant attack from a technical point of view. In some cases, it might actually be possible to *remotely clone a gift card* without swiping the magnetic stripe or even writing down the card number from the back of the card.

The requirements for this attack are as follows:

1. Understand the data stored on the magnetic track(s) and be able to craft track data from the start.

   The level of difficulty of this task might vary from trivial to challenging, or even impossible, depending on each implementation. For instance, a gift card might simply encode the card number in Track 2 plus other predictable data. If the card number was predictable, then counterfeiting a valid card would be trivial.

   Another example would be a card storing the card number, plus a random long integer in the magnetic stripe. Even if the card number was predictable, guessing a truly random and long integer which is only stored in the network back-end (i.e. cannot be manipulated by changing its value on the card) would render the attack impossible. Of course, the supposedly "random" number would have to be carefully analysed to ensure it is truly random, as opposed to being derived from a known value such as the card number. e.g. a checksum value, rather than a random value. Equally important is to ensure that this random number is required by the back-end system for the transaction to be considered valid. Otherwise an attacker might simply be able to include a dummy number with the same number of digits.

2. Clone a card which has been activated and has credit on it.

   Even if the format of the data on Track 2 could be predicted from the start, an attacker wouldn't want to clone a card without credit. After all, the whole point of the attack is to purchase goods which someone else will ultimately pay for! So, how can an attacker find out which cards have already been sold and topped-up by their respective owners? The explored method involves querying the card's vendor site without (almost) exploiting any bug, by simply (ab)using legitimate features such as balance-checking facilities. Additionally, taking advantage of checksum algorithms used to generate the card ID (e.g. Luhn/mod10), can be beneficial in order to reduce the number of HTTP requests that are required.

   Note that although some card numbers are not completely sequential, they can still be predicted by querying lists of pre-generated numbers against web-based balance-checking sites.

Figure 9 Example of gift card balance-checking site which only requires card number (no PIN is required)

Consider the following gift card numbers:

Sample #1	*Sample #2*
6033 5313 0758 3889 6033 5313 0759 3389 6033 5313 0760 2021 6033 5313 0761 2021 6033 5313 0762 5595 6033 5313 0763 6443 6033 5313 0764 4909 6033 5313 0765 0567 6033 5313 0766 6166 6033 5313 0767 6166 6033 5313 0768 1683 6033 5313 0769 0714 6033 5313 0770 7768 6033 5313 0771 3908 6033 5313 0772 4886 6033 5313 0773 5509	6041 1458 0871 8860 6041 1458 0872 3309 6041 1458 0873 0545 6041 1458 0874 6418 6041 1458 0875 2310

Both of the previous samples are from gift cards available at the same retailer.

Clearly visible are the card numbers which are separated in four chunks of four digits each. These numbers were obtained from cards which were produced adjacently by the manufacturer. This can be confirmed by analysing the first twelve digits which are fully sequential. The last four digits (shown in red), however, are *not* sequential. Instead they appear to be random (although entropy analysis against a large sample of numbers might show otherwise).

For some reason, these last four digits are repeated sometimes for adjacent card numbers in sample #1. For example, the first and second card numbers both end with "3389". The same is true for the third and fourth card numbers (both end with “2021”), and the ninth and tenth card numbers (both end with “6166”). This could be an indication that the last four digit number is derived from the current time. e.g. a pseudo-random number based on the current time at the time it was generated.

The following coding example simulates the generation of numbers following the same syntax as the previously shown samples (12 sequential digits plus four “random” digits):

```
#include <stdio.h>
#include <time.h>
#include <string.h>

short int foo() {

        short int i,j,d;
         char tmp[11]={'\0'},magic[5]={'\0'};

        // use current time as seed
        srand(time(NULL));
        time_t secs=rand();
        sprintf(tmp,"%ld",secs);

        // grab only first 4 digits from pseudo-random number
        for(i=0;i<4;++i)
                magic[i]=tmp[i];

        return atoi(magic);
}

int main(void) {

        short int i;
        short int a,b,c,d;
        a=6033;b=5313;c=0000;d=0;

        for(c=0;c<=9999;++c) {
                // delay number generation for a bit less than a sec
                usleep(950000);
                d=foo();
                printf("%.4d %.4d %.4d %d\n",a,b,c,d);
        }

        return 0;
}
```

Here the code would generate 16-digit numbers, always starting with “6033 5313”, continued by a four digit sequential number, and finally a pseudorandom number based on the current time. Since the time it takes to generate one pseudo-random number and the next is less than a second, sometimes two different card numbers would have the same last four pseudo-random digits (notice in the previous code that the current time expressed in seconds is being used to seed the “rand()” function via “srand(time(NULL))”):

```
6033 5313 0000 1298
6033 5313 0001 9960
6033 5313 0002 6869
6033 5313 0003 1463
6033 5313 0004 9009
6033 5313 0005 8434
6033 5313 0006 1616
6033 5313 0007 1320
6033 5313 0008 1320
6033 5313 0009 1014
6033 5313 0010 7024
6033 5313 0011 4026
6033 5313 0012 1178
6033 5313 0013 8748
6033 5313 0014 1630
6033 5313 0015 1334
6033 5313 0016 2097
6033 5313 0017 1805
6033 5313 0018 4197
6033 5313 0019 1153
6033 5313 0020 8784
6033 5313 0021 1658
6033 5313 0022 2767
6033 5313 0023 2126
6033 5313 0024 1829
6033 5313 0025 1525
6033 5313 0026 1213
6033 5313 0027 1984
6033 5313 0028 1984
6033 5313 0029 1681
6033 5313 0030 2984
```

For the purpose of this example, the assumption is that the last four digits are truly random. Therefore in order to "hit" a valid card number by brute-forcing HTTP requests against the balance-checking site, 10,000 requests would need to be submitted.

Sometimes, gift card numbers follow the popular Luhn's checksum algorithm (a.k.a. mod10). This means that the number of generated card numbers to query can be reduced by ten. For instance, take the following gift card numbers:

```
600176 0542 0007 60606
600176 0542 0007 60614
600176 0542 0007 60622
600176 0542 0007 60630
600176 0542 0007 60648
600176 0542 0007 60655
```

At first sight, it seems that the last digit is random, and so a maximum of 10 different numbers need to be queried in order to find a valid card number. However, further inspection reveals that the last digit is just the mod10 checksum digit. Thus the number of HTTP requests needed to find a valid card number can be reduced from ten to one.

An attacker can generate a list of target numbers. For instance, a list of 10,000 numbers from 600176054200076**0000** to 600176054200076**9999** can be created:

```
$ ./gennumbers
```

```
usage: ./gennumbers <prefix> <start_number> <end_number>
```

```
$ ./gennumbers 600176054200076 0000 9999 > numbers.lst
```

```
$ wc -l numbers.lst
10000 numbers.lst
```

And then the numbers which do follow Luhn's algorithm can be filtered out, in this case reducing the number of possible card numbers from 10,000 to 1,000:

```
$ ./luhncheck
usage: ./luhncheck <number_or_file> [mode]
mode:
       0 - print numbers that follow Luhn's algorithm
       1 - print numbers that follow Luhn's algorithm AND have a valid PAN's length
       2 - print numbers whose syntax match major credit card types
       3 - same as 0 but verbose
       4 - same as 1 but verbose
       5 - same as 2 but verbose
note: mode 0 is used by default
```

```
$ ./luhncheck numbers.lst 1 > valid_numbers.lst
```

```
$ wc -l valid_numbers.lst
1000 valid_numbers.lst
```

Once an attacker has figured out the syntax followed by the card numbers of the targeted implementation, and has generated a list of possible card numbers, he/she then needs to find out which numbers correspond to gift cards which have been activated by their respective owners.

As previously discussed in the section *In-band Cloning*, some balance-checking websites require users to enter a PIN in order to check the remaining balance on their gift cards. The following example shows how some of these sites allow attackers to identify valid numbers of gift cards which have been activated; even when the attacker does not have knowledge of the required PIN.

This particular gift card implementation uses a 16 digit card number and an eight digit PIN. When entering an invalid 16 digit card number (6034 5032 0240 5555, in this case) and an invalid eight digit PIN (12345678 in this case), an "Invalid Card number" error message is returned:

Figure 10 "Invalid Card number" message returned when submitting invalid card number and an invalid eight digit PIN

When entering a *valid* 16 digit card number which has not yet been activated by the customer (6035 9745 7485 1314, in this case) and an invalid eight digit PIN (12345678 used once again), a “Card has not been activated” message is returned:

Figure 11 "Card has not been activated" message returned when submitting a valid card number of a card that has not been activated yet (invalid eight digit PIN also submitted in this case)

However, when entering a valid card number of a card which has already been activated by its owner (6034 5032 0240 5779, in this case) and an invalid eight digit PIN (12345678 again), an “Invalid PIN” error message is returned:

Figure 12 "Invalid PIN" message returned when submitting number of an activated card plus an 8-digit invalid PIN

Thus in the third case, the attacker knows that 6034 5032 0240 5779 corresponds to a valid card number which has already been activated by the customer, even though he/she never had knowledge of the PIN. The changes in returned error messages can be exploited to remotely enumerate card numbers which correspond to activated gift cards. All an attacker needs to do in this case is brute-force card numbers whilst always submitting a dummy eight digit PIN.

The following example shows how an attacker can mount such an attack using a specialised tool such as Burp Intruder.

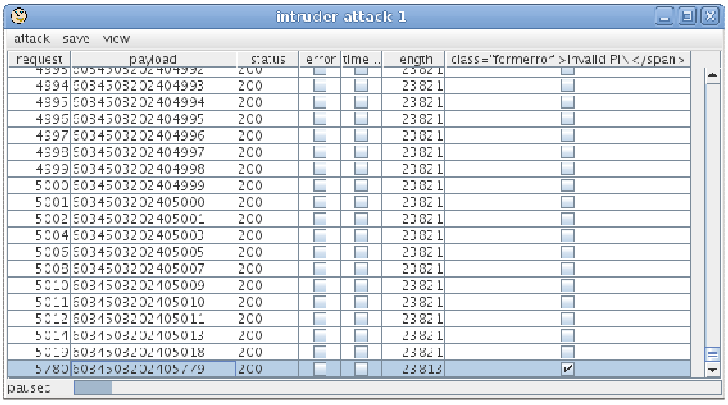


Figure 13 By looking for certain responses within the returned HTML pages, it is possible to find activated card numbers in an automated fashion

Attack Advantages	*Attack Disadvantages*

Attack Advantages	*Attack Disadvantages*
Physical access to the targeted gift card is not required; i.e. skimming is not required. There is also no need to write down the gift card number to be cloned, as this is predicted via a brute-forcing attack. However, in order to optimise the attack to always target cards that have been sold recently (i.e. those most likely to have a remaining balance), the attacker would sometimes visit the targeted retailer and write down one of the gift card numbers available at the store's stand.	Not all gift card implementations are vulnerable to this type of cloning attack as it is a more targeted attack. An attacker may therefore require additional technical expertise that was unnecessary in previously discussed attacks. If card numbers are not fully sequential, it could take many HTTP requests to find a valid card number; e.g. 10,000 or 100,000

Injection Attacks

As previously discussed, all of the UK gift cards analysed as research for this paper do *not* store the card's balance on the track data. The simplest way to verify that balance information is not stored on the magnetic tracks is to:

1. Dump (swipe) and save the binary data of tracks that "contain" any data (usually Track 1 and 2, and sometimes only Track 2).

2. Purchase a good using the gift card. The balance should then be updated in the back-end DB servers. Note: the card's current balance can usually be found on the transaction receipt, by requesting it online (not all gift card types support checking the current balance online, but many do), or via telephone.

3. Swipe the gift card again after a purchase (assuming a pre-purchase swipe was recorded), and compare the binary data on all tracks. If the data remains exactly the same, then the conclusion is that the current balance is *not* stored on the card.

In summary, it would not be possible to modify the card balance by altering the track data, as the balance value is not stored on the card. However, there are other types of attacks where it might be possible to change the card's balance by tampering with the track data, even when the card's balance is *not* stored on the track data.

An attacker could exploit weaknesses within the system by inserting malicious SQL statements that update the card's balance in the back-end DB servers when an online transaction is performed; i.e. when a gift card is swiped at a retail POS terminal. However, this type of attack would require specialist knowledge of the internal systems and, as such, would be much more complicated to realise.

ISO 7811 *does not* allow letter characters ([a-z],[A-Z]) on **Track 2**, which is the default read by POS terminals. According to the standards, the encoding type supported on Track 2 is Binary Coded Decimal (**BCD)** with five bits for each character, although only four bits are actual data (the fifth is the odd parity bit). This only gives a total of **16 different characters**. Due to physical limitations, Track 2 can only hold **up to 40 characters**. The following is the character set for BCD encoding used in Track 2:

- Ten digits (used for account number, discretionary data, etc):

```
0 1 2 3 4 5 6 7 8 9
```

- Three framing/field characters (used for start sentinel, field separator, end sentinel respectively):

```
; = ?
```

- Three control characters (mainly used for LRC byte):

```
: < >
```

Since there is a restriction imposed to **digits** and **special symbols** such as ';', '=', '?', ':', '<' and '>' a SQL injection attack would not be possible, as alpha-numeric strings are required to compose a well-crafted statement.

ISO 7811 imposes fewer restrictions on **Track 1** where **ALPHA** encoding is used providing seven bits for each character (one bit for the odd parity bit and six bits for the actual data). This means that up to four times the number of possible characters can be used than on Track 2 (i.e. **64 different characters as opposed to 16)**. Additionally, Track 1 can hold **up to 79 characters**, as opposed to 40 characters which is the maximum number of characters that can be stored on Track 2.

The following is the character set for ALPHA encoding used in Track 1:

36 alpha-numeric characters (digits and upper-case letters):

```
0 1 2 3 4 5 6 7 8 9 A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
```

Three framing/field characters (used for start sentinel, field separator, and end sentinel respectively):

```
% ^ ?
```

25 control/special characters:

```
 (space character) ! " # $ & ' ( ) * + , - . / : ; < = > @[ \ ] _
```

This means that malicious SQL statements on Track 1 may be inserted while the swiped card is still considered valid by the POS terminal, since the inserted data is ISO-compliant:

In summary, Track 1 offers more possibilities for a SQL injection attack due to holding a bigger maximum number of stored characters and also a bigger number of different characters:

Track	*Character set*	*Maximum number of characters stored on track*
Track 1 (7-bit ALPHA encoding)	64 different characters	79 characters
Track 2 (5-bit BCD encoding)	16 different characters	40 characters

The following Track 1 data is made of 75 characters (still within the limit of 79 maximum characters) and only contains characters that are allowed by ISO 7811 (ALPHA encoding).

```
%B1;UPDATE CARDS SET BALANCE=150 WHERE CARD_NUMBER=6337805586634252 45;#^^?+
```

Note: the malicious payload is shown in red. This is usually where the card numbers are located.

In this case the payload is inserted within the account number field. This usually holds the card number (PAN) and is located between the start sentinel and format code ('%B'), and the field separator ('^'). The cardholders name is usually found between both field separators ('^') but here it has been omitted for space-saving purposes. Finally, the expiration date, service code, and discretionary data would be located between the second field separator ('^') and the end sentinel character ('?'), but once again this data has been omitted due to space restrictions.

Let's suppose the gift card application executes the following SQL query when performing a transaction, in order to first verify that the swiped card is active and has sufficient balance to purchase the desired good:

```
SELECT * FROM CARDS WHERE CARD_NUMBER = $CARD_NUMBER;
```

Note: value controlled by the attacker shown in bold fonts.

The following is an example of the data that could be returned by the back-end DB:

```
+-------------------+---------+--------+------+
| CARD_NUMBER       | BALANCE | ACTIVE | ID   |
+-------------------+---------+--------+------+
| 633780558663425245 |    1.20 |      1 | 6456 |
+-------------------+---------+--------+------+
```

Provided that the value of the *$CARD_NUMBER* variable is not filtered by the application, it might be possible to alter the SQL statement into the following by crafting a card with the data shown in a previous example:

```
SELECT * FROM CARDS WHERE CARD_NUMBER = 1;UPDATE CARDS SET BALANCE=150 WHERE
CARD_NUMBER=633780558663425245;#;
```

This provides a fully legal SQL statement despite card number 1 not existing:

```
mysql> SELECT * FROM CARDS WHERE CARD_NUMBER = 1;UPDATE CARDS SET BALANCE=150 WHERE
CARD_NUMBER=633780558663425245;#;
Empty set (0.00 sec)

Query OK, 1 row affected (0.06 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

and results in the swiped card's balance being updated from 1.20 to 150:

```
mysql> SELECT * FROM CARDS WHERE CARD_NUMBER = 633780558663425245;
+--------------------+---------+--------+------+
| CARD_NUMBER        | BALANCE | ACTIVE | ID   |
+--------------------+---------+--------+------+
| 633780558663425245 |     150 |      1 |    1 |
+--------------------+---------+--------+------+
1 row in set (0.00 sec)
```

The SQL queries above are tailored to the specific test systems, and will not work on all implementations.

Because most POS terminals read Track 2 by default, an attacker would need to trick the terminal to read Track 1, which is where the malicious SQL statements are stored. It might be possible to cause the terminal to consider the data on Track 1 by damaging the data on Track 2, although *this has not been tested in a real environment*.

Attack Advantages	***Attack Disadvantages***
Due to the nature of this attack, gift card vendors or providers of such services may not have sufficient database auditing in place to detecting unauthorised updates from cards. In the event that this attack was successful, it may go unnoticed, unless fraud analysis detects discrepancies between gift card transactions and gift card sales.	Highly experimental attack. Very unlikely to be successful unless the attacker has privileged information of the targeted gift card implementation; e.g. malicious developer working for the gift card provider company.

Web Application Manipulation

Rather than cloning a victim's card, it may be simpler to compromise the system through the online administration consoles.

For example, an attacker might be able to perform fraudulent purchases by changing the balance of a card under their possession without even having knowledge of magstripe technology. Instead of targeting the data encoded in the magnetic stripes, an attacker could use research and web 'hacking' techniques to identify and compromise the admin interface used by retailers to manage their card programs.

These types of admin interfaces have been marketed by some prepaid card solutions providers as Gift Card Management Systems (GCMS) or Card Inventory Management Systems (CIMS). Unfortunately, they are sometimes accessible to anyone over the web, making them susceptible to the same types of attacks as any other Internet-facing systems. The following provides an example:

Figure 14 Gift card program management interface of anonymous service provider

If compromised, a gift card program admin console could allow an attacker to reset the balance of their own gift card. What would be possible for an attacker after compromising the admin interface would ultimately depend on specific implementation and configuration details of the targeted environment. For instance, the following factors would play a crucial role on determining what the attacker could do after breaking into the admin interface:

- Functionalities available by the application. e.g. reset card's balance, block/unblock card, etc
- Whether or not the web application is connected to the same back-end database which is queried by POS terminals to check a card's current balance when purchases are made

Attack Advantages	*Attack Disadvantages*
No need to use magstripe reading/writing equipment. Attacker simply changes/resets balance of a previously-purchased gift card – if functionality is available or back-end data can be manipulated (e.g. via SQL injection).	Expertise in web "hacking" techniques is required. It is necessary to ensure that the infiltration of the website is not discovered – at least not within a certain window to carry out the fraud.

Recommendations

A list of recommendations for gift card service providers to protect against attacks which target implementation flaws of the actual gift cards, and attacks which target vulnerabilities present on the actual infrastructure used to provide the gift cards service is provided below:

Threat	*Mitigations*
Automated retrieval of card numbers that have been activated and their respective balances.	CAPTCHA on balance-checking website; although CAPTCHA implementations can be broken, CAPTCHAs will make the task of the attacker significantly more difficult. Random long alphanumeric PIN (e.g. eight characters) for each card number.
Detecting activated gift card numbers, even when a PIN is required.	Generation of generic error messages by balance-checking web application.
Non-physical cloning attacks where an attacker can derive the full track data from a given card number.	Random long magic number in track data. e.g. "discretionary data" field.
Obtaining unauthorised management rights where an attacker can potentially reset the balance of their gift card, or any gift card they might have cloned.	Restricted gift card program management interface. For example only allow trusted IP addresses via firewalls and/or require two-factor authentication such as tokens or private keys.
Sampling large number of gift cards for free.	Locate the gift cards stand in a visible, but not reachable area, where only the store employees have access to them. Note that obtaining large number of gift cards is an essential step in reverse engineering the structure of the data encoded in the magnetic tracks.
Reusing or manipulating legitimate gift cards to counterfeit cloned track data using affordable equipment.	Ensure that track data is encoded on High Coercivity (HiCo) magstripes. This is not an infallible security feature as magstripe encoders that can write on HiCo magstripes exist in the public domain. However, this type of equipment is usually significantly more expensive, which can decrease the likelihood of a given gift card implementation being targeted.

Threat	*Mitigations*
Attacks against the gift card infrastructure where it is possible to perform gift card fraud without directly targeting the magstripe track data; e.g. by gaining unauthorised access to the back-end DB servers where the remaining balance for each card number is stored.	Perform regular security assessments on the infrastructure and applications providing the gift cards service; e.g. front-end balance-checking web applications and payment servers, back-end DB servers. Although this is more of a good security practice rather a security feature, it is highly recommended for gift card providers who are concerned with providing a secure service.

Conclusions

This paper demonstrates that there are multiple strategies in an attacker's arsenal to attempt to perform fraudulent purchases using gift cards. It has shown that these types of attacks are possible in the real world and also provided recommendations to help protect against such attacks.

In conclusion it is vital to randomise not just gift card numbers, but also the track data stored on them. Similar to the "discretionary data" present on most credit cards track data, gift cards should contain a non-predictable value encoded on their track data. This random value has been referred to as the "magic number" throughout this paper.

In the worst case scenario, not storing a random value on the gift card's track data could result in attackers cloning activated gift cards, even without having physical access to the targeted cards.

Internet-facing web interfaces should also be taken into account when designing a secure gift card environment. Examples of web interfaces that can be targeted to perform fraudulent purchases include administrative sites used by retailers to manage their gift card programs and websites visited by users to check their card balance.

Additionally, the data stored on the track data should never be trusted. Although encoding specially-crafted track data is not a popular input validation attack vector, nothing stops attackers from tampering with the data stored on gift cards. Should this data not be correctly validated by the gift card provider's payment servers, an attacker could potentially alter transaction data such as the card's balance.

Gift cards should be treated as any other type of currency. Service providers should follow the same good security practices which are usually recommended to other types of currency providers, such as credit card issuers.

References

[i] "Gift Card Sales Near $100 Billion in 2007"
http://www.emarketer.com/Article.aspx?R=1005802

[ii] "Gift-Card Sales Rise More Than Forecast, Boost U.S. Retailers "
http://www.bloomberg.com/apps/news?pid=10000103&sid=aruLgVF6Qfg4&refer=us

[iii] "All you need to know about gift and stored value cards "
http://www.the-logic-group.com/Downloads/giftex-report-excerpts.pdf

[iv] "Gift Vouchers - US VS UK "
http://www.prepaytec.com/resources/whitepapers/gift+vouchers+-+us+vs+uk

[v] "UK Gift Cards "
http://homepage.ntlworld.com/victaylor/UKretailers.html

[vi] "Anatomy of a Subway Hack "
http://tech.mit.edu/V128/N30/subway/Defcon_Presentation.pdf

The following documents provide further information on magstripes, including ISO 7811 standards usually used to encode track data among UK gift cards:

What is the Layout of Data on Magnetic Stripe Cards?
http://www.tech-faq.com/mag-stripe-cards.shtml

Card-O-Rama: Magnetic Stripe Technology and Beyond
http://www.phrack.com/issues.html?issue=37&id=6#article

Payment Card Technology
http://www.task.to/events/presentations/Payment%20Card%20Technology.pdf

Track format of magnetic stripe cards (tracks 1 and 2)
http://www.acmetech.com/documentation/credit_cards/magstripe_track_format.html

ISO Magnetic Stripe Card Standards
http://www.cyberd.co.uk/support/technotes/isocards.htm

Magnetic Stripe Card
http://en.wikipedia.org/wiki/Magnetic_stripe_card

Magnetic Stripe Technology
http://www.usna.edu/InfoTech/papers/MAGSTRIPE%20Readers3.doc

Forensic data recovery and examination of magnetic swipe card cloning devices
http://www.engagecf.co.uk/articles/magneticswipecardclone.pdf

Frequently Asked Questions (FAQ)

1. Q. *What hardware/software was used to perform the tests which this research is based on?*

   A. MAKStripe USB reader/writer with included MAKStripeExplorer v1.22 software running under Ubuntu Linux (other OS are also supported by this software), see:

   http://www.makinterface.de/makstusbe.php3

2. Q. *You have provided several examples of gift card numbers, track data and screenshots. Are all of these made up, or are they actually real examples?*

   A. They all are real examples. However, retailer and service provider-specific information has been removed in order to protect against any potential abuse.

3. Q. *Why did you specifically focus on researching magstripe gift cards?*

   A. Primarily because they're subject to fraud just like any other type of currency, even though there is little information on attacking them in the public domain.

4. Q. *You talked about cloning magstripe gift cards without swiping the magnetic stripe. Is this a theoretical attack only, or have you actually tested this?*

   A. This attack has been confirmed on at least two implementations. Other implementations are also suspected to be vulnerable, but further testing is required to confirm this.

5. Q. *I'm confused. You mentioned that none of the gift cards you analysed in the UK contain the card's balance in the (magnetic) track data. If so, how would it be possible for an attacker to purchase goods with a gift card without spending any money?*

A. Instead of changing the balance on the track data, the attacker might be able to predict the track data of a genuine card and also detect if such card has been activated by a legitimate customer. Alternatively, the attacker might be able to change the balance of his own card (whether legitimate or cloned) by indirectly attacking the gift card system. Instead of targeting the actual card implementation, it might be possible to change the card balance by compromising the provider's back-end DB servers or administrative interfaces. Fore more details, please refer to *Attacks Classes* section of this paper.

Acknowledgements

This paper was written by Adrian Pastor, Principal Consultant at Corsaire. The author of this paper would like to give special thanks to Dragos Ruiu (kyx.net) for inviting him to present his research at EUSecWest, to Major Malfunction (rfidiot.org) for the inspiration to research magstripes, and to Corsaire for sponsoring the research. Thanks also go to Zack Anderson (mit.edu), Rogan Dawes, David Ryan, Glyn Geoghegan and Petko D. Petkov for their invaluable feedback.

About The Author

Adrian Pastor, B.Sc. (Hons), is a Principal Security Consultant in Corsaire's Professional Services Team and has worked in IT Security since 2001. He has spent the last six years focused on Ethical Hacking, Security Assessment and Audit at Corsaire, ProCheckUp, NTA Monitor, and independently as a freelance consultant. He is a member of GNUCITIZEN, an information security research organisation and contributes to OWASP. He is a frequent speaker at industry events. Adrian's work has been featured in established media outlets such as BBC Radio 1, The Washington Post, Wired, Slashdot, PC Pro, The Register, PC World, CNET and many others. He is perhaps best known for finding critical vulnerabilities on the BT Home Hub, the most common Wi-Fi home/SOHO router in the UK.

About Corsaire

Corsaire are experts at securing information systems, consultancy and assessment. Through our commitment to excellence we provide a range services to help organisations protect their information assets and reduce corporate risk.

Founded privately in the United Kingdom in 1997, we operate on an international basis with a presence across Europe, Africa and the Asia-Pacific rim. Our clients are diverse, ranging from government security agencies and large blue-chip FTSE, DAX, Fortune 500 profile organisations to smaller internet start-ups. Most have been drawn from banking, finance, telecommunications, insurance, legal, IT and retail sectors.

They are experienced buyers, operating at the highest end of security and understand the differences between the ranges of suppliers in the current market place. For more information contact us at contact-us@corsaire.com or visit our website at http://www.corsaire.com.