

# Ethical and Professional Concerns About Online Banking Fraud: A Whitehat Hacker's Perspective on Social Responsibility and Pointing Out the Stupidity and Intentional Ignorance of Others in Donald Trump's New American World Order

## Purpose and Intent of this Article

- definitions of “white hat” hacking and “white paper”;
- abuses of social contract and social / ethical responsibility (references);
- protections, freedom of information (example of the XboxLinux) and more recent technology laws;
- author's intent, implied accountability and honest use; The underlying assumption is that the contents of this article will be used with responsibility and ethical intent by its readers;
- Original framework:  $\pm$  \$12 service charge / convenience fee considered too greedy an approach by some to replicate fraudulent charges up to the unique ID numbers posted with the suspect transactions;
- It is hard to believe that BOA is unaware of these abuses and the daily potential for misuses in their proprietary banking systems;

## Know Your Rights

### Timelines for Effective Action Mandated by Federal Law

Compared to Bank of America's grossly underwhelming misadvertised timelines for their fiscal accountability with your sensitive financial data, underspecified at 10 to 60 days depending on who you ask, when you ask, and the phonetree worker routed through Mobile, AL from your cell phone, the [TODO] federal law provides an unlimited time to recover funds removed from your checking and/or savings account by fraudulent activity. This is functionally a statement mandated by US (inter)national law providing unwitting consumers an indefinite postponement of the process under any abnormal circumstance to protect the rights of customers and victims of identity theft regardless of social status or the financial means to hire teams of statisticians and lawyers to protect their assets and hard earned money.

In other words, the Bank of America corporation is unconditionally accountable to their customers independent of the convenient policies stated on their website and even in their print mass mailing notices of what they think is an appropriate timeline appropriate for them to act on your behalf. Moreover, Bank of America and other similar financial institutions are accountable to their customers who have not filed explicit claims of fraudulent activity and should have their paid expert technical analysts proactively finding other small errors executed on a larger scale without having to be asked to do so. This is a part of the implied social contract formed with these financial institutions for your business and their rights in managing the sensitive personal information required for the day-to-day operations of these large-scale corporations.

### Legal Rights for “Hacking” Your Own Bank Account to Detect and Discover Fraudulent Activity

According to jurisdiction under the same federal law, the consumer is granted protections to post online banking transactions of any nature to their own financial accounts. Again, to summarize and be clear, if someone employed by your bank tells you that “hacking” is a crime and you are to be arrested in the state of Florida for replicating suspect charges on your own account, please do proceed to issue them an appropriate however understated middle finger gesture, do it any way, and then call the police on them for criminal negligence and obstruction of justice. This law and the US constitution provide consumers a bare minimum

of these legal rights – even if you manage to “piss off” airheads and stupid people endowed with a title by Bank of America (or otherwise) in due legal process. In particular, knowledge is power, educated users should be aware of their rights, and simply possessing the technical know-how to exploit a vulnerability in the the flawed banking systems used to manage your own sensitive personal data does not make you an evil counterpart to the perpetrators of these criminal activities. Additionally, federal and Florida state laws (along with IL, MO, and WA depending on jurisdiction and the sources of the on-shore electronic wire transfers involved in routing these transactions) require a minimum of ten plaintiffs to file a class action lawsuit under sufficiently competent legal counsel for wrongdoings of this nature, so be prepared as an individual to substantiate your claims well by clear-cut demonstration of point example and to necessarily motivate the designated customer service representatives at Bank of America to investigate the fraudulent criminal charges to your account.

My experience consulting with Bank of America employees on staff for an on-site scheduled appointment was to have a print-out of their newly mailed policies (which I was mysteriously made unaware of up to that point) on correcting fraudulent charge activity printed, highlighted, and then handed to me by their most experienced statistician of financial engineering available to me for my visit at that time. This response, incidentally, may or may not be representative of their corporate policies in larger state-wide practice. When we met for my scheduled visit this time, I was told by my designated financial representative (Figure 1) to check whether it is actually worth it and in my best fiscal interest to request alternate copies of the prohibitively expensive transaction history associated with my account (now only available to me upon serious inquiry at the rate of \$5 per statement per month for the last 5–6 years) if, in fact, their in-house fraud department would be willing to help me at all.

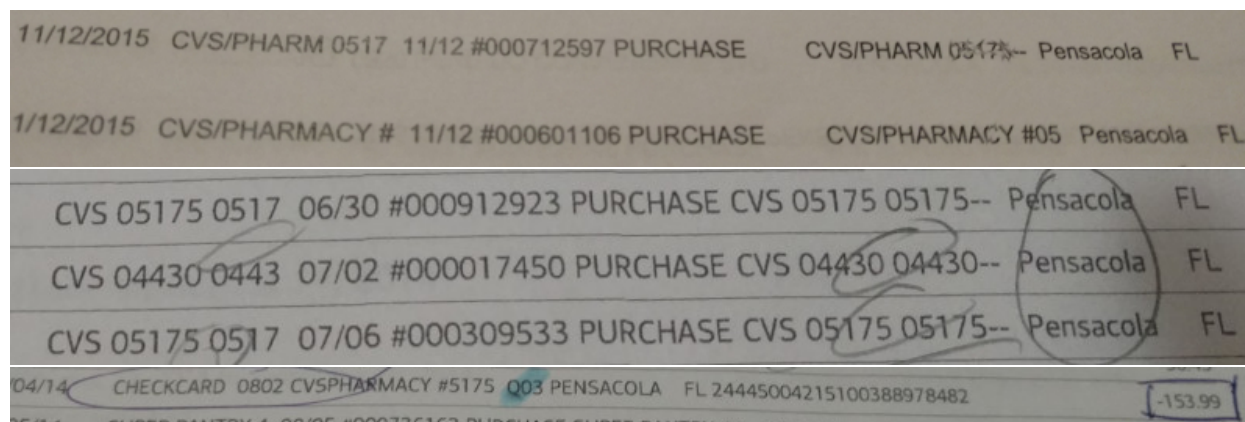


**Figure 1:** Contact Information for Bank of America’s Favorite Local Technology Expert and Financial Engineering Major at Large

## Pseudocode for the USAePay Mobile Banking Website

### Disclaimer

Before we proceed with the next information contained in the pseudocode source listings of Figure and opinionology a priory to be attributed strictly to the author, please note that I do not regularly, and have barely ever, been able to write functional source code in Visual Basic (or VBScript). My disgust with stumbling onto the Vericheck and USAePay websites in my search for information about current technology laws has led to our forays into the open source freeware code on the next pages to be written in the ugly, partially functional, however still disturbingly accurate, demonstrations given below. Any of the languages supported by a modern installation of Microsoft’s Visual Studio IDE should also suffice to make this point



**Figure 2:** CVS Pharmacy Transactions and Separate Payment Charges in Florida

(C# / Java, XML, or just plain old javascript will do).

(Explanation and Documentation of this Light API)

What this API (or Application Programming Interface) effectively lets you do is enable a mildly experienced novice “*script kiddie*” with a “*would do, but never ever should do*” mentality (and more to the understated point), rather than more mastered technical expertise in the form of seamless “*can do*” hacking magic, to use an internet-enabled Windows computer to easily manipulate the sensitive personal financial data of millions in the greater Florida area.

## Explanations of Suspicious Charges By Example

See Figures 2 – 4 referenced on the previous pages.

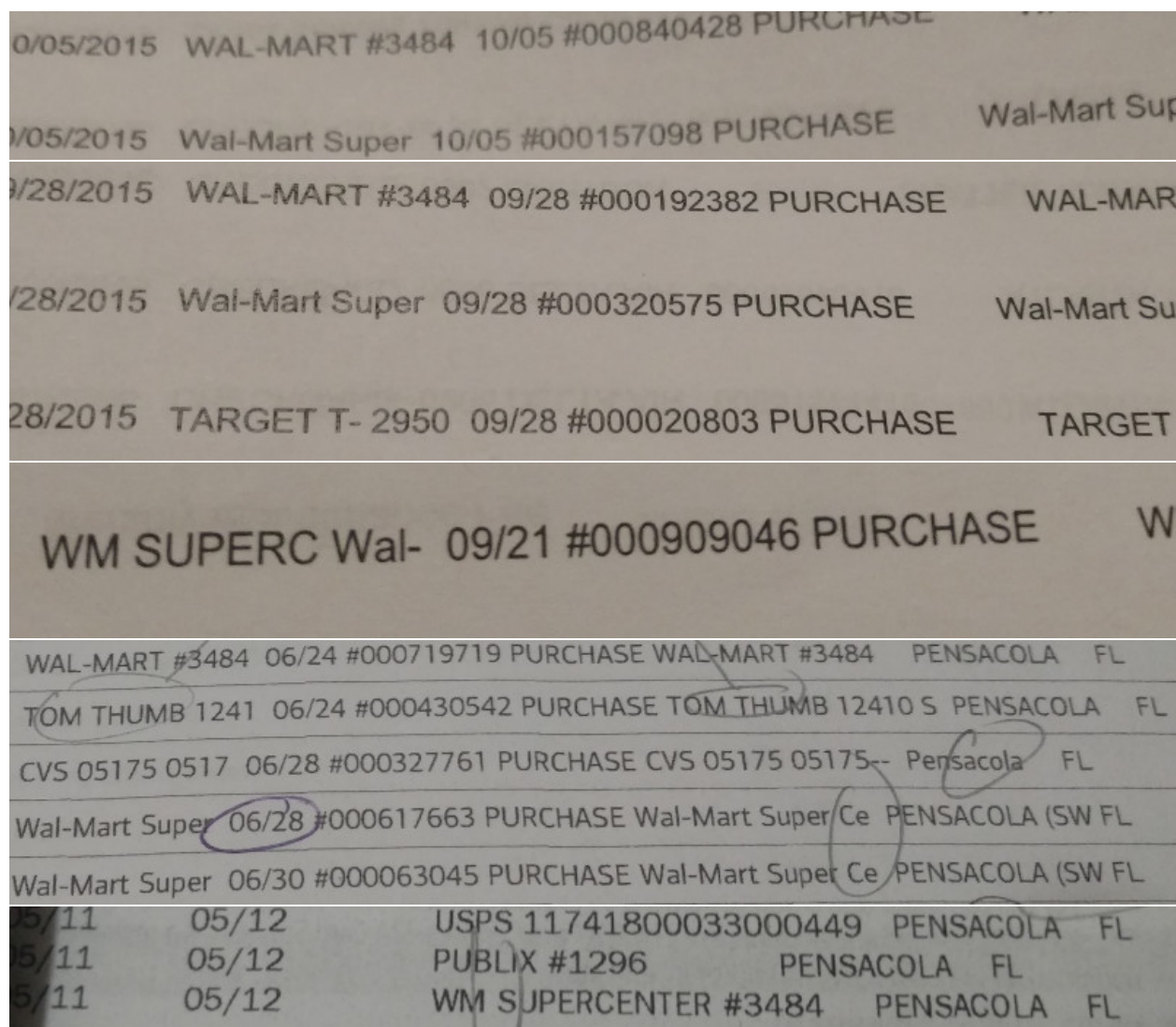
## More Information and Links to Online Reading Material

### Homograph attacks

- Unicode varieties of the “-” (dash) characters, uses of capital letters, and hash marks: It is typically considered good programming practice to be cautious with transaction record string containing multiple similar or visually indistinguishable markings with unicode character sets, for example, to avoid unnecessary bank fraud (Figure 5). This is one argument, in addition to non-standardized font possibilities, for why your printed BOA transaction logs alone are inadequate to detect all possibilities of fraudulent transactions.
- Hash marks, italicized print characters, 0x0999, fixed width font variations, etc.

### Undocumented API Functionality

- Brute-force discovery of semi-random customer data points See the verbose descriptions of the error codes returned by the API documented on the “ACH Return Codes” and “NOC Codes” on the Vericheck pages, and by the Wikipedia documentation for Address Verification System (AVS) return code entries.
- Repeat or recurring transactions differing by small amounts. There are at least a couple of references available online to thinly veiled documentation of small charges and service upgrade fees charged to implement transactions of this type.
- Updating routing numbers of failed scanned check (RDC) transactions using the ACH return codes from above. Since the bank routing number information is all that is required to access a customer’s unique sensitive financial data, we observe that this loophole effectively provides an attacker with perpetual access to your funds with BOA’s flawed proprietary systems as they are currently implemented in practice.
- Ability to lookup previous transaction records in full by names, unrelated unique identifiers or other customers, and by transaction description strings.



**Figure 3:** Examples of At Least Four Distinct Walmart Transaction Headers (and One Assumed At Least Partially Legitimate CC Bill Entry) Originating From the Same Local Store and Same Two Cash Registers in Pensacola, FL

- There is sandbox access available to testers available before posting transaction strings to live accounts for processing.

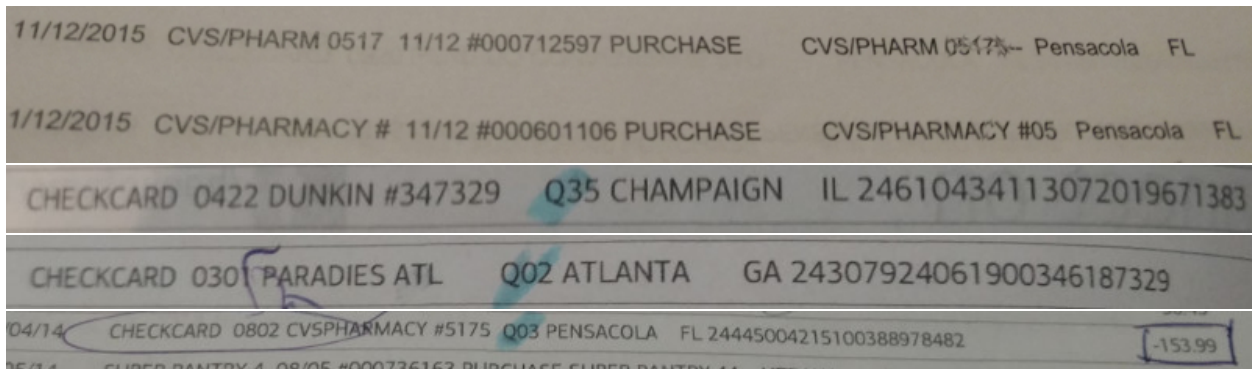
## Other Suspicious Transaction Record Patterns

Other suspicious activities to look out for on your account include:

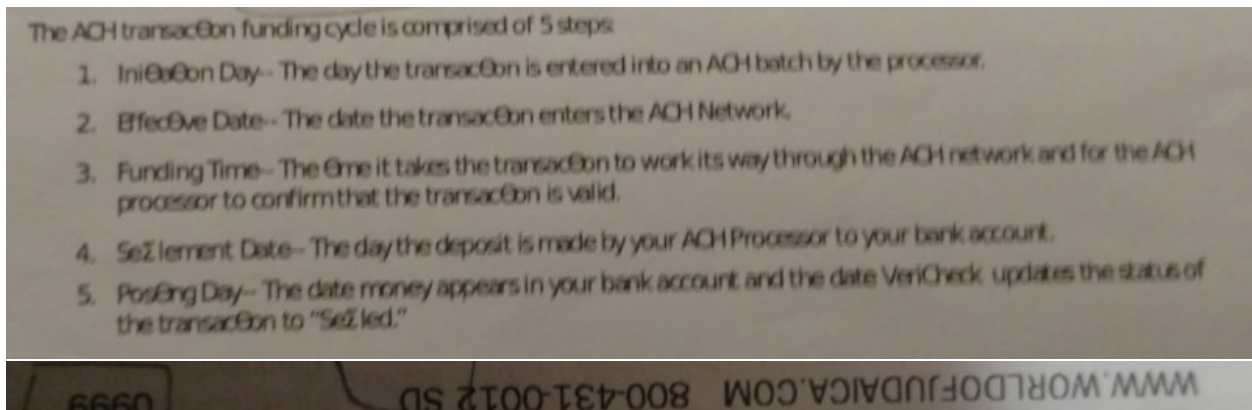
- Transaction entries for hard-wired, non-checkcard records not posted on the same day should throw an immediate red flag in auditing suspicious activity on an account (See the tables provided on the ACH Funding at a Glance page in Figure 6).
- ARC, BOC, and NSF transaction header oddities in the listings typically contained in separate listings on BOA's monthly statement printouts (Figure 7).

## About the Authors

Donald Eugene writes no code, but has a Ph.D. in Social Psychology from the University of Washington in Seattle. The author has a Masters degree in Computer Science from the University of Illinois at Urbana-Champaign, a nationally ranked top five engineering school by *US News and World Reports*, whose academic



**Figure 4:** Dunkin Donuts in Champaign, IL and A Few Other Examples of Mysteriously Concatenated Magic Number Strings in Suspicious Transaction Headers



**Figure 5:** Printouts of PDF to Text Translations Illustrating Our Findings With Regard to the Homograph Attack Framework

ambitions have more recently been upgraded to the status of Ph.D. candidate in Mathematics at a large public university in the Pacific Northwest.



Enter Day	Settlement Day	Posting Day
Monday	Thursday	Friday
Tuesday	Friday	Saturday*
Wednesday	Monday	Tuesday
Thursday	Tuesday	Wednesday
Friday	Wednesday	Thursday
Saturday	Thursday	Friday
Sunday	Thursday	Friday

\*NOTE: Many banks do not update statements on Saturday; if your bank falls into this category, you will see the deposit post on Monday. The transaction will always be updated to "Settled" in VeriCheck on Saturday.

## 5 Day Funding

Enter Day	Settlement Day	Posting Day
Monday	2nd Monday	2nd Tuesday
Tuesday	2nd Tuesday	2nd Wednesday
Wednesday	2nd Wednesday	2nd Thursday
Thursday	2nd Thursday	2nd Friday
Friday	2nd Friday	2nd Saturday*
Saturday	2nd Monday	2nd Tuesday
Sunday	2nd Monday	2nd Tuesday

NOTE: Many banks do not update statements on Saturday; if your bank falls into this category, you will see the deposit post on Monday. The transaction will always be updated to "Settled" in VeriCheck on Saturday.

Figure 6

```

1 .....
2  '' Source Code References Constructed From Consulting the
3  '' USAePay API and Online Verisign Documentation on 11/20/2015:
4  ''   (1) http://docs.vericheck.com/v1.0/docs/obtaining-credentials
5  ''   (2) http://wiki.usaepay.com/developer/statuscodes
6  ''   (3) http://www.vericheck.com/ach-return-codes/
7  ''   (4) http://wiki.usaepay.com/developer/soap-1.4/methods
8  ''   (5) Supporting documentation from the online WIKI guide references
9  ''   "What is Vericheck?" and "ACH Funding at a Glance"
10 .....
11  '' The following provides a listing of pseudocode, or more loosely speaking,
12  '' informal untested functional technical documentation prepared to educate the
13  '' reader. This sourcecode, and moreover the information prepared in this entire
14  '' document without the contained contributing author information referenced on the
15  '' first page, is informally released under a tentative open source creative-commons-
16  '' like
17  '' license for non-commercial use and greater-good-type public education purposes.
18  '' What this means in non-technical speak is that you may blatantly copy the
19  '' next source code and freely distribute the information contained in this
20  '' whitepaper writeup to educate yourself and to prevent this fraudulent misuse of

```

NSF Fee: Schmidt, Maxie7ZTJGQPDLA5	\$30.00	\$100.00
Unable To Locate Account (R03)	( \$1,347.26 )	( \$1,347.26 )
WelcomeHome ACH payment -- Maxie Schmidt	( \$1,347.26 )	( \$1,347.26 )
rent	\$1,275.00	\$1,275.00
1036628830 TRAS 03/01/2015-03/31/2015	\$10.00	\$10.00
1036628830 MWBD 03/01/2015-03/31/2015	\$55.89	\$55.89
1036628830 GAS 03/01/2015-03/31/2015	\$6.37	\$6.37
Bulk Processing - Maxie Schmidt	( \$135.71 )	( \$135.71 )

Figure 7

```

20  '' bank transactions from happening again to your neighbor's checking and savings
    account.
21  '' The technically annoying references to the author's credentials are optional, and
22  '' may be omitted, or freely modified at will in the event that you can beat me.
23
24  ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
25  '' Definitions of Constants and Other Magic Numbers in the Source Code:
26  ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
27  On Error GoTo ThrowUserError;
28  Dim DEFAULT_ERROR_CODE As Unsigned Integer = 0xdeadbeef;
29  Dim OurLocalBranchExperience As Integer = 11 << 6 Xor 19 << 4 Xor 2015;
30  Dim uniqueSessionIDStr = "ABCDEFGH-%{RemoteOSSessionIDStr}-%{strftime}" % \
31      ("1234", "LinuxMintIsWindows9ImprovedManPages");
32
33  ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
34  '' Declare Variables to Fill the Requisite Transaction Fields in the API:
35  ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
36  Dim merchantID, transSummatoryAmount, transDesc, transType, transSEClass As String;
37  merchantID = "IWouldntButSomeoneElseWouldAndMightHave_sdkf02322512";
38  transSummatoryAmount = "153.99";

```

```

39 ' From my favorite local pharmacy apparently submitted to pay for a
40 ' stock, recurring, and otherwise routine prescription to be filled last summer
41 transDesc = "Something Routine, Overly Simplistic, and Most-Plausible-Probable-Cause-
    Type String Handling";
42 transType = "debit";
43 transSEClass = "POP";
44 ' Also: TEL, CCD, BOC, or WEB
45 ' See "Types of Vericheck Transactions and Funding" at the online WIKI site
46 % TODO: Implement full setup of the transaction request object for use immediately below
47 .....
48 '' Transaction Objects Defined by the API:
49 .....
50 Dim malwareAPIClientInstance As usaepay.usaepayService;
51 malwareAPIClientInstance = New usaepay.usaepayService;
52
53 Dim localUniqueTransactionToken As ueSecurityToken;
54 localUniqueTransactionToken = API.CreateToken(uniqueSessionIDStr, 0xcafebabe);
55
56 %Dim ourProofOfConceptTransaction As usaepay.TransactionRequestObject;
57 %ourProofOfConceptTransaction = New usaepay.TransactionObject;
58 %ourProofOfConceptTransaction.Details = New usaepay.TransactionDetail;
59 %ourProofOfConceptTransaction.Details.Amount = transSummatoryAmount;
60 %ourProofOfConceptTransaction.Details.AmountSpecified = true;
61 %ourProofOfConceptTransaction.Details.Invoice = "232431";
62 %ourProofOfConceptTransaction.AuthCode = "ReturnedAPICall";
63 %ourProofOfConceptTransaction.RefNum = "AnotherReturnedAPICall";
64
65 .....
66 '' Submit and Process Our Untested Replicate Transaction:
67 .....
68 %Dim processedTransactionResponse As usaepay.TransactionResponse;
69 %processedTransactionResponse = New usaepay.TransactionResponse;
70 %processedTransactionResponse = malwareAPIClientInstance.captureTransaction(
    localUniqueTransactionToken, \
71 % ourProofOfConceptTransaction.RefNum, ourProofOfConceptTransaction.Details.Amount);
72 statusMessage = processedTransactionResponse.ResultCode = "A" ? \
73 "Approved, Reference Number is " & processedTransactionResponse.RefNum : \
74 "Declined for " & processedTransactionResponse.Error;
75 System.StrError.printf("Status Code %c: %s\n", statusMessage,
    processedTransactionResponse.ResultCode);
76 %'' And so on
77
78 .....
79 '' Some Good Programming Practices for Default Error Handling,
80 '' Both Stock and Implied by Descriptive Suggestion:
81 .....
82 ThrowUserError:
83 GoTo BankOfAmerica
84 End Sub
85
86 BankOfAmerica:
87 ' Default descriptive expectation of privacy and accountability
88 On Error GoTo Hell
89 Err.raise OurLocalBranchExperience + DEFAULT.ERROR.CODE, -, \
90 "Error Class (None): Don't Shoot the Programmer, Blame Someone Else's System", \
91 "Error Message: For default expectations of privacy and accountability " + \
92 "in Hell, consult the next lines"
93 End Sub
94
95 Hell:
96 Debug.print
97 "Infinite Loop: " + \
98 "Rinse, Lather, Repeat (Install Linux and Kill Process to be Sure)"
99 GoTo ThrowUserError
100 End Sub
101
102 .....
103 '' Other untested noteworthy API functionality for handling
104 '' debit / credit card transactions, bulk transaction uploads, and the
105 '' processing of credit card batches:
106 '' (*) Methods for Transactions: See API WIKI
107 '' (*) Methods for Bulk Transaction Uploads:

```



[illegible]

Date	Descrip
11/17/14	CHEC
11/19/14	CHEC
11/19/14	SHEL
11/20/14	CHEC
11/20/14	11 BA
11/21/14	CHEC
11/21/14	CHEC
11/21/14	PETC
11/21/14	SAFE
11/24/14	CHEC
11/24/14	CHEC
11/24/14	CHEC
11/24/14	CHEC
11/24/14	CHEC
11/28/14	CHEC
12/02/14	SHEL
12/02/14	Cardt

**Figure 8:** Example Date Ranges of Bulk Batch Processing of Transaction Records