Ethical and Professsional 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 propietary 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 activtity 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 simpliy posessing 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 fradulent 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 representive 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.

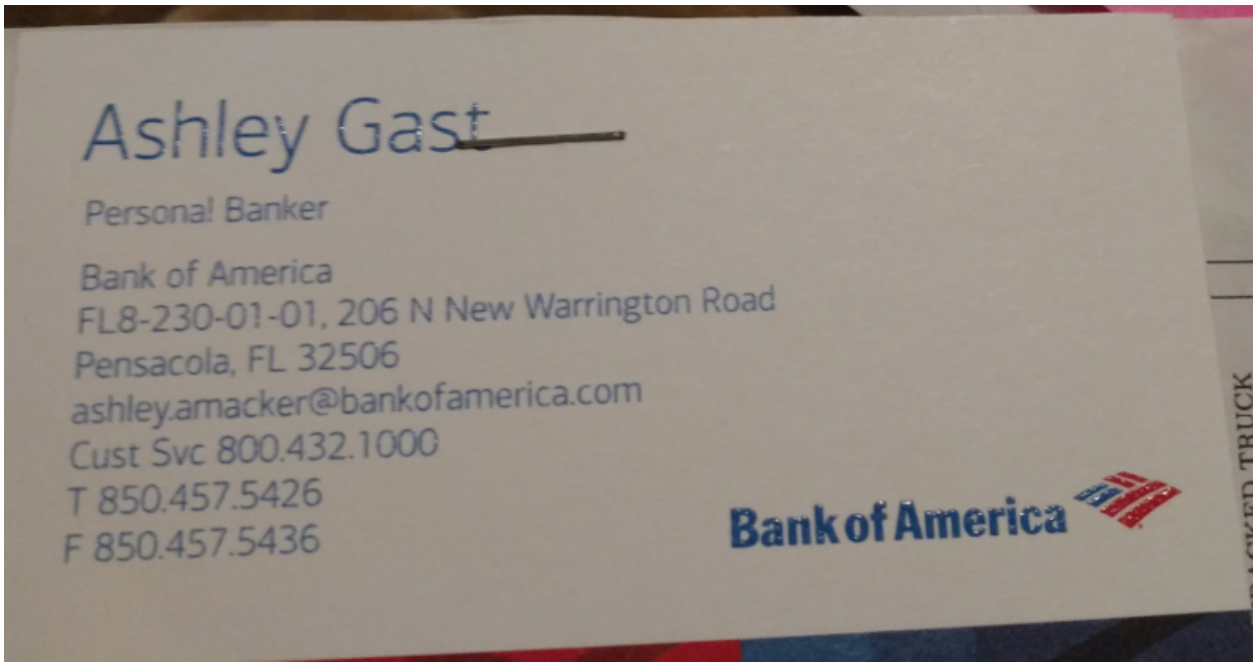


Figure 1: Contact Information for Bank of America's Favorite Local Technology Expert and Financial Engineering Major at Large

Pseudocode for the EPay USA 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

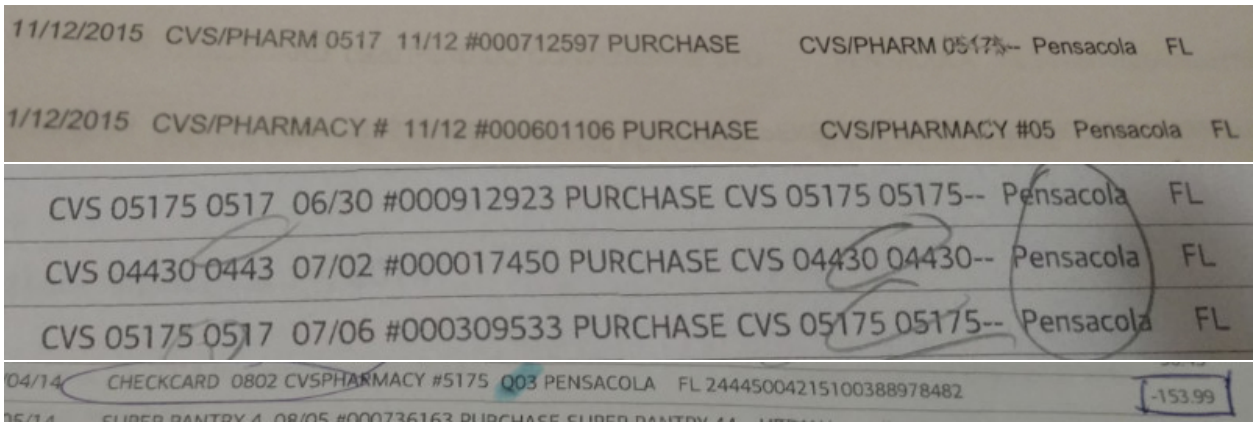


Figure 2: CVS Pharmacy Transactions and Separate Payment Charges in Florida

(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 mildy experienced novice "*script kiddie*" with a "*would do, but never ever should do*" mentaility (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

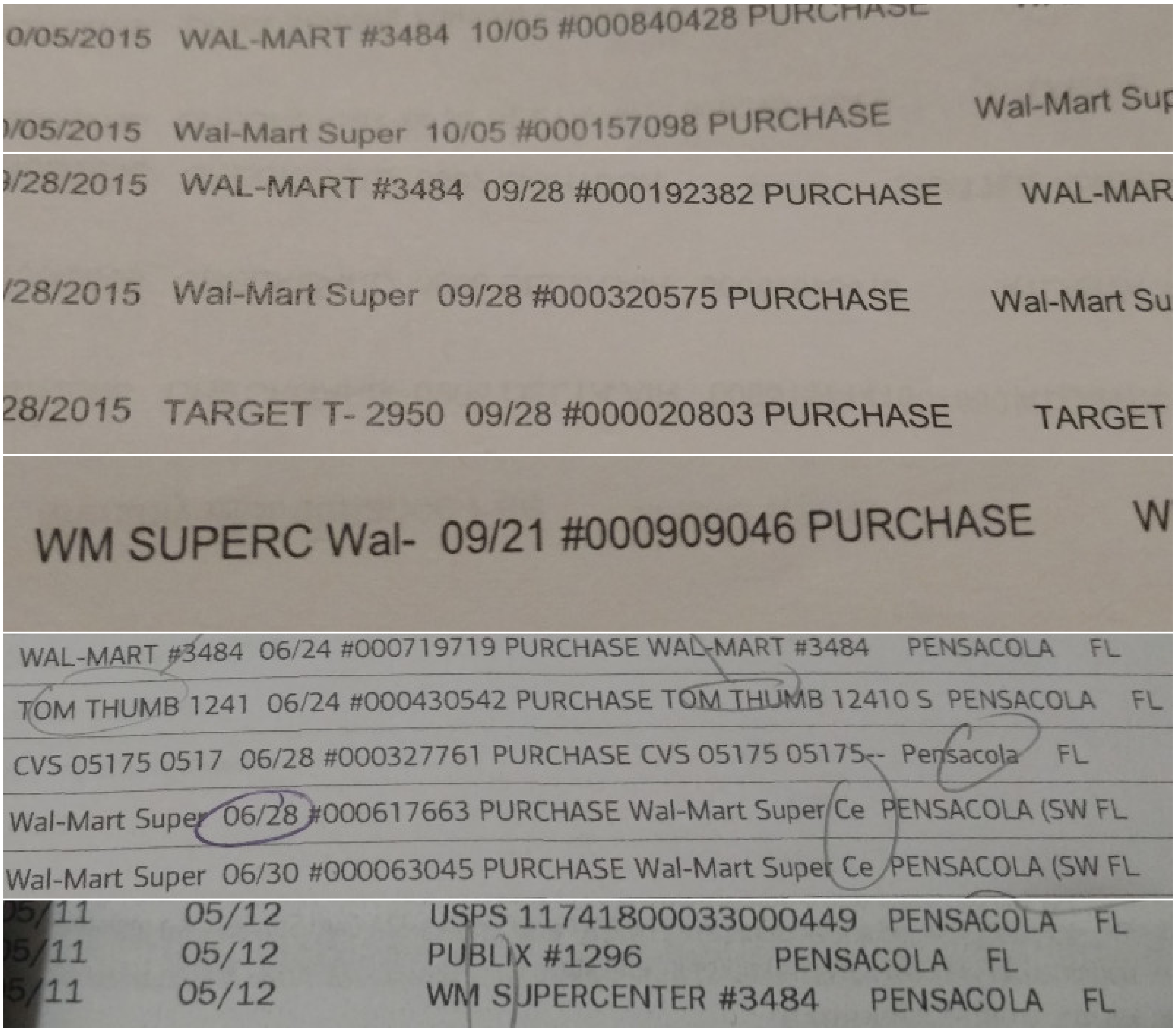
Homography attacks: unicode varieties of the "-" (dash) characters, good programming practice to avoid unnecessary bank fraud, and why your printed (see notes) BOA transaction logs are inadequate to detect all possibilities of fraudulent transactions. Uses of capital letters and hash marks, italicized print characters, 0x0999, fixed width font variations, etc.;

Undocumented API Functionality: Brute–force discovery of semi–random customer data points; Repeat transactions differing by small amounts (get a couple of references to online thinly veiled documentation of small charges for this); Updating routing numbers of failed check transactions; Can lookup a previous transaction record by customer, names, transaction descriptions, etc.;

Other suspicious activities to look out for on your account transaction records:

About the Authors

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



0/05/2015 WAL-MART #3484 10/05 #000840428 PURCHASE

/05/2015 Wal-Mart Super 10/05 #000157098 PURCHASE Wal-Mart Sup

/28/2015 WAL-MART #3484 09/28 #000192382 PURCHASE WAL-MAR

/28/2015 Wal-Mart Super 09/28 #000320575 PURCHASE Wal-Mart Su

28/2015 TARGET T- 2950 09/28 #000020803 PURCHASE TARGET

WM SUPERC Wal- 09/21 #000909046 PURCHASE W

WAL-MART #3484 06/24 #000719719 PURCHASE WAL-MART #3484 PENSACOLA FL

TOM THUMB 1241 06/24 #000430542 PURCHASE TOM THUMB 12410 S PENSACOLA FL

CVS 05175 0517 06/28 #000327761 PURCHASE CVS 05175 05175-- Pensacola FL

Wal-Mart Super 06/28 #000617663 PURCHASE Wal-Mart Super Ce PENSACOLA (SW FL

Wal-Mart Super 06/30 #000063045 PURCHASE Wal-Mart Super Ce PENSACOLA (SW FL

5/11 05/12 USPS 11741800033000449 PENSACOLA FL

5/11 05/12 PUBLIX #1296 PENSACOLA FL

5/11 05/12 WM SUPERCENTER #3484 PENSACOLA FL

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

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

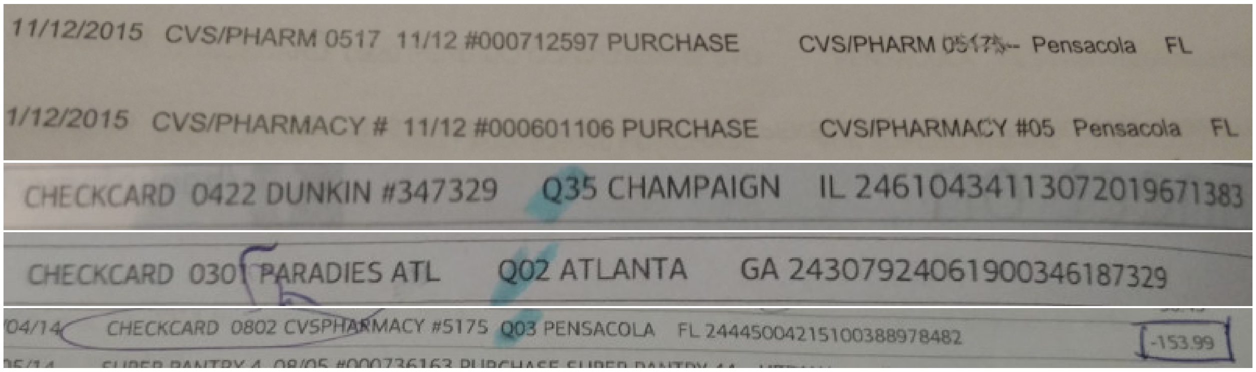
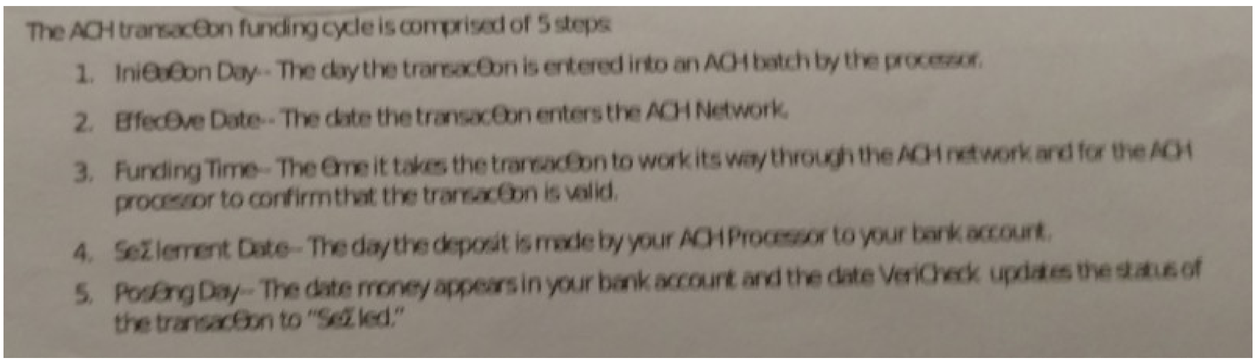


Figure 4: Dunkin Donuts in Champaign, IL and A Few Other Examples of Mysteriously Concatenated Magic Number Strings in Suspicious Transaction Headers



The ACH transacOon funding cycle is comprised of 5 steps

1. IniOaOon Day-- The day the transacOon is entered into an ACH batch by the processor.
2. EffecOve Date-- The date the transacOon enters the ACH Network.
3. Funding Time-- The Ome it takes the transacOon to work its way through the ACH network and for the ACH processor to confirm that the transacOon is valid.
4. SeZlement Date-- The day the deposit is made by your ACH Processor to your bank account.
5. PosOng Day-- The date money appears in your bank account and the date VeriCheck updates the status of the transacOon to "SeZled."

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

```
'' The technically annoying references to the author's credentials are optional, and
'' may be omited, or freely modified at will in the event that you can beat me.

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'' Definitions of Constants and Other Magic Numbers in the Source Code:
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
On Error GoTo ThrowUserError;
Dim DEFAULT_ERROR_CODE As Unsigned Integer = 0xdeadbeef;
Dim OurLocalBranchExperience As Integer = 11 << 6 Xor 19 << 4 Xor 2015;
Dim uniqueSessionIDStr = "ABCDEFG-%{RemoteOSSessionIDStr}-%{strftime}" % \
  ("1234", "LinuxMintIsWindows9ImprovedManPages");

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'' Declare Variables to Fill the Requisite Transaction Fields in the API:
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Dim merchantID, transSummatoryAmount, transDesc, transType, transSEClass As String;
merchantID = "IWouldntButSomeoneElseWouldAndMightHave_sdkf02322512";
transSummatoryAmount = "153.99";
  ' From my favorite local pharmacy apparently submited to pay for a
  ' stock, recurring, and otherwise routine prescription to be filled last summer
transDesc = "Something Routine, Overly Simplistic, and Most-Plausible-Probable-Cause-
    Type String Handling";
transType = "debit";
transSEClass = "POP";
  ' Also: TEL, CCD, BOC, or WEB
  ' See "Types of Vericheck Transactions and Funding" at the online WIKI site
% TODO: Implement full setup of the transaction request object for use immediately below
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'' Transaction Objects Defined by the API:
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Dim malwareAPIClientInstance As usaepay.usaepayService;
malwareAPIClientInstance = New usaepay.usaepayService;

Dim localUniqueTransactionToken As ueSecurityToken;
```

```
localUniqueTransactionToken = API.CreateToken(uniqueSessionIDStr, 0xcafebabe);

%Dim ourProofOfConceptTransaction As usaepay.TransactionRequestObject;
%ourProofOfConceptTransaction = New usaepay.TransactionObject;
%ourProofOfConceptTransaction.Details = New usaepay.TransactionDetail;
%ourProofOfConceptTransaction.Details.Amount = transSummatoryAmount;
%ourProofOfConceptTransaction.Details.AmountSpecified = true;
%ourProofOfConceptTransaction.Details.Invoice = "232431";
%ourProofOfConceptTransaction.AuthCode = "ReturnedAPICall";
%ourProofOfConceptTransaction.RefNum = "AnotherReturnedAPICall";

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'' Submit and Process Our Untested Replicate Transaction:
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
%Dim processedTransactionResponse As usaepay.TransactionResponse;
%processedTransactionResponse = New usaepay.TransactionResponse;
%processedTransactionResponse = malwareAPIClientInstance.captureTransaction(
     localUniqueTransactionToken, \
%  ourProofOfConceptTransaction.RefNum, ourProofOfConceptTransaction.Details.Amount);
statusMessage = processedTransactionResponse.ResultCode == "A" ? \
  "Approved, Reference Number is " & processedTransactionResponse.RefNum : \
  "Declined for " & processedTransactionResponse.Error;
System.StrError.printf("Status Code %c: %s\n", statusMessage,
     processedTransactionResponse.ResultCode);
%'' And so on

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'' Some Good Programming Practices for Default Error Handling,
'' Both Stock and Implied by Descriptive Suggestion:
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
ThrowUserError:
  GoTo BankOfAmerica
End Sub

BankOfAmerica:
  ' Default descriptive expectation of privacy and accountability
  On Error GoTo Hell
  Err.raise OurLocalBranchExperience + DEFAULT_ERROR_CODE, _, \
    "Error Class (None): Don't Shoot the Programmer, Blame Someone Else's System", \
    "Error Message: For default expectations of privacy and accountability " + \
    "in Hell, consult the next lines"
End Sub

Hell:
  Debug.print
    "Infinite Loop: " + \
    "Rinse, Lather, Repeat (Install Linux and Kill Process to be Sure)"
  GoTo ThrowUserError
End Sub

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'' Other untested noteworthy API functionality for handling
'' debit / credit card transactions, bulk transaction uploads, and the
'' processing of credit card batches:
'' (*) Methods for Transactions: See API WIKI
'' (*) Methods for Bulk Transaction Uploads:
''     API Functions to submit multiple transactions by single same-day-bulk
''     uploads to avoid higher overhead processing fees:
''     See API WIKI and examples in the attached image Figures [TODO]
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
```