

TECHNICAL PAPER

BIN Attacks

(Also Known as Enumeration Attacks or Credit Master Attacks)

Predicting and Preventing Financial and Reputational Loss

Last update: March 2026



Contents

Overview2

 What Is the Modus Operandi?..... 3

 What Is Your Defense?..... 4

Using SAS Fraud Decisioning to Detect BIN Attacks.....4

 Overview 4

 Create a Profile Variable Set 5

 Create Profile Variables 5

 Create Variable Rule 6

 Create the Decision Rule..... 7

Resources10

Relevant Products and Releases

- SAS® Fraud Decisioning

Overview

It is not rocket science to use a simple program to predict a valid credit card's primary account number (PAN). Thanks to the internet, many online bank identification number (BIN) verification databases are freely available where anyone can access and validate a given BIN. Using a program and the BIN database, fraudsters can effortlessly predict a valid PAN for a credit card and use this number fraudulently for financial gain. With the rise of online shopping, e-commerce offers a low-risk opportunity for people who are perpetrating the fraud. After all, this type of fraud eliminates face-to-face contact and supports higher volumes of transactions at a much lower cost because there are no requirements to produce a fake plastic card to commit fraudulent activities.

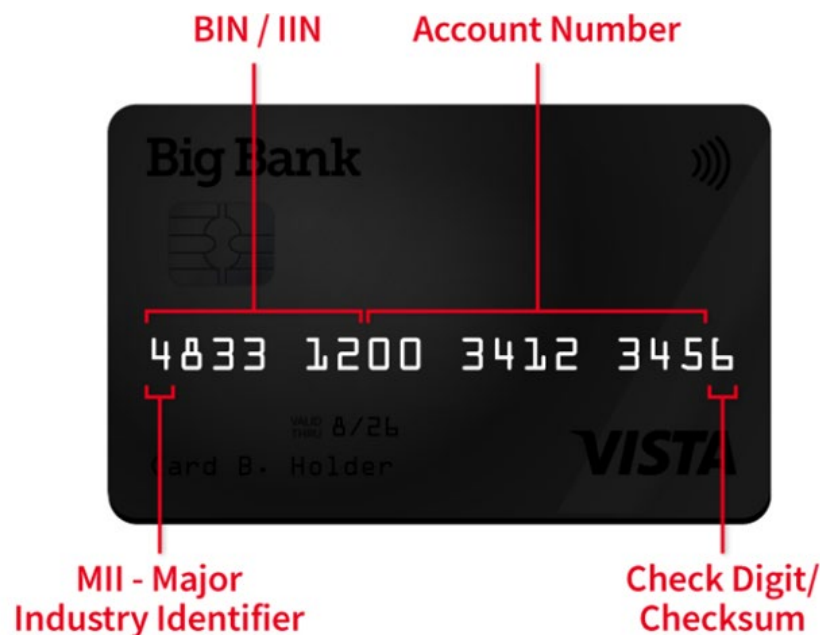


Figure 1. Credit Card Numbers

Note: The BIN can be 6 or 8 digits.

Other than the PAN, the values that are needed to complete an online transaction are the expiration (expiry) date and the [security code](#) (the code on the signature panel). Unfortunately, there are merchants who still do not enforce the need for the security code, which leaves only the expiration date.

Expiration dates are specific to the month and year only. Unfortunately, this is where banks hurt themselves. Banks have historically issued cards in sequence, so if fraudsters know the expiration date for one card, they know the expiration date is the same for the rest of the cards issued during that same month! All the fraudster needs to do is find the upper and lower boundaries for the card number range for each month.

It is the knowledge of the boundaries that results in a BIN attack. A [bot](#) is used to iterate through the card number range to issue low-value payments to the same merchant. Eventually, card numbers begin to be consistently blocked. One declined transaction could be due to cancellation or reissue of a specific card. However, a consistent range of subsequently declined transactions means that the fraudster can be confident that the expiration date has changed.

What Is the Modus Operandi?

There are only a few steps to trigger a BIN attack:

1. Gather a genuine BIN and expiration date from an internet compromise or from a discarded credit card transaction receipt. (The BIN is not typically masked.)
2. Generate the remaining 7 to 13 digits (the unique identifier of an account) with a program to create the required PAN. The PAN is typically 15 to 19 digits.
3. The last digit of the PAN is a checksum and there is **mod-10** logic to predict this number. (This logic is not covered in this paper.)
4. With steps 1-3, the fraudster creates a list of PANs. Combine these numbers with the expiration date.
5. Use a bot to process authorizations via an online merchant for a small value purchase (around USD 1). This purchase confirms that the PAN and expiration date combination is valid.
6. Start to purchase high-value goods.

From the preceding steps, you can see that an attempted BIN attack involves minimal effort and risk from a fraudster's perspective to gain financially.

BIN attacks are difficult to prevent. Although BIN attacks represent fraudulent attempts, they rarely result in a loss to the merchant. The authorization attempts are never posted. These BIN attacks are merely a test to gain a range of compromised and validated card details that can then be used to perpetrate future higher value card not present (CNP) transactions.

BIN attacks do morph and change. For example, a BIN attack can shift from traditional testing of a single merchant with high volumes to distributed testing across hundreds of merchants with only a couple of attempts each, although total volumes are high.

Although BIN attacks remain card-number focused, trends show overlap with the following methods:

- AI-powered identity synthesis
- Deepfake-driven account opening
- Industrialized cross-platform fraud networks

These broader changes reflect a shift into autonomous, algorithmic fraud ecosystems.

However, there are ways to mount a defense against BIN attacks with minimal interference to a customer's experience.

What Is Your Defense?

Here are three things that you can do to stop a BIN attack:

1. A BIN attack that uses a bot is sometimes over within minutes. A 100% real-time fraud detection system with dynamic profile updates is crucial to monitor transactions. Using this capability makes it easy to profile transactions on the same BIN, to the same merchant, and for the same amount within a short period of time.
2. When a bank observes a BIN attack, the bank can gather the cards that were exposed and mark them as an at-risk group in a hot list. Any transactions on the cards can then be assessed proactively to impose stricter measures for future online merchant purchases (for example, blocking all transactions from that channel for a certain period of time). Again, a sophisticated anti-fraud system that profiles all transaction records is required to enable such real-time analytics to be applied.
3. A sophisticated customer notification mechanism (for example, two-way SMS text messages) between the bank and card holders can help save legitimate customers from the impact of a BIN attack. Ideally, the SMS text message requires a simple Yes or No response to confirm the validity of a suspicious transaction. If the customer does not respond within a certain time period, a time-out setting should be defined to block the card. This approach is much more effective than a single SMS text message notification that requests the customer contact the card center if the transaction is not authorized, leaving the transaction in an unknown state for a longer period of time.

To accommodate for the latest trend of BIN attacks (a shift from a focused single-merchant attack to a distributed merchant attack), you can also take the following actions:

- Reduce focus on merchant-level data (merchant identifier and MCC code).
- Track authorization-system declines, especially those for invalid cards, across the entire BIN range.

BIN attacks, which have been used by fraudsters for a long time, will unfortunately continue. If banks staggered the sequencing when they issued cards, the consistency between sequential card numbers and the expiration date would be greatly reduced. Also, if merchants began to enforce additional security measures that are available, including demanding built-in card verification options and the use of 3D Secure 2.0, the opportunity to be defrauded would be reduced. Issuing banks must incorporate these mitigation steps for BIN attacks into their fraud detection systems.

Using SAS Fraud Decisioning to Detect BIN Attacks

Overview

The following procedure shows how to set up a rule within SAS Fraud Decisioning to protect your card base from BIN attacks.

Create a Profile Variable Set

This rule tracks information about CNP transactions, specifically, keyed and e-commerce point-of-sale entry mode transactions that occur for a merchant. You first need to create a profile variable set to uniquely identify the merchant.

Note: The profile variable set name that is used in the subsequent rule code in this paper is 'merchant'. If you use a different name or already have an existing merchant profile variable set with a different name, the rule code will need to be updated to replace merchant with the profile variable set name that you chose.

To ensure that the merchant is uniquely identified, select suitable key message schema variables for the profile variable set. You can select multiple message schema variables as long as the total number of combined key variables for the profile variable set is no more than 15 variables.

Note: Profile variables are stored in Redis, which does not allow locking of the profile variable sets. Because profile variable sets are not locked, one transaction can update a profile variable set record for a certain key and another transaction can simultaneously update the same profile variable set record. Although the processing of a transaction takes only milliseconds, there can be instances (particularly if the chosen profile variable key has low cardinality) where multiple transactions are trying to update the same profile variable set record within a short period of time. Because the transactions do not wait for the previous transactions to finish updating the record, the stored profile data in these instances can become inaccurate. This behavior might be observed when using a merchant-level profile and if so, this rule may fire later than expected due to the profile inaccuracies.

Introduction of a new profile variable set can have technical implications that are not apparent to the bank. Before creating the profile variable set, consult with your internal IT department and SAS experts.

Create Profile Variables

Now that you have created a profile variable set, you can create the profile variables within this profile variable set to use in the rule.

The profile variables are all created as arrays to store the following information from the last five CNP keyed or e-commerce transactions that have occurred at the merchant:

- card numbers (merchant.BA_PAN)
- transaction amounts (merchant.BA_amount)
- merchant country code (merchant.BA_Countrycode)
- merchant category code (MCC) (merchant.BA_MCC)
- expiry date of the card on the transaction message (merchant.BA_TransactionCardExpiry)
- date and time of the transaction (merchant.BA_DtTm)

Table 1. Profile Variables to Be Created within the Merchant Profile Variable Set

Name	Type	Array Length
BA_PAN	String (length 80)	5
BA_amount	Numeric	5
BA_Countrycode	String (length 3)	5
BA_MCC	String (length 4)	5
BA_TransactionCardExpiry	String (length 4)	5
BA_DtTm	Timestamp	5

Create a Variable Rule

The variable rule uses the previously created profile variables to store the following information:

- card numbers (message.creditcard.number or message.debitcard.number)
- transaction amounts (message.cardfinancial.amount)
- merchant country code (message.merchant.countryCode)
- merchant category code (MCC) (message.merchant.categoryCode)
- expiry date of the card on the transaction message (message.creditcard.messageExpirationDt or message.debitcard.messageExpirationDt)
- date and time of the transaction (message.request.messageDtTm)

This information is gathered from the last five CNP keyed or e-commerce transactions that have occurred at the merchant.

Copy the following rule code into your variable rule:

```
declare int i;
declare int k;

if message.solution.originationType in ('CC', 'DC')
and message.solution.activityType = 'CA'
then do;

    if message.cardfinancial.posEntryType in ('00','01','81','8E','8F')
    and message.cardfinancial.cardPresentInd = '1'
    then do;
```

```

do i = 5 to 2 by -1;
  k = i-1;
  profile.merchant.BA_PAN[i] = profile.merchant.BA_PAN[k];
  profile.merchant.BA_amount[i] = profile.merchant.BA_amount[k] ;
  profile.merchant.BA_countryCode[i] =
profile.merchant.BA_countryCode[k] ;
  profile.merchant.BA_MCC[i] = profile.merchant.BA_MCC[k] ;
  profile.merchant.BA_transactionCardExpiry[i] =
profile.merchant.BA_transactionCardExpiry[k];
  profile.merchant.BA_transactionDtTm[i] =
profile.merchant.BA_transactionDtTm[k] ;
end;

profile.merchant.BA_amount[1] = message.cardfinancial.amount;
profile.merchant.BA_countryCode[1] = message.merchant.countryCode;
profile.merchant.BA_MCC[1] = message.merchant.categoryCode;
profile.merchant.BA_transactionDtTm[1] = message.request.messageDtTm;

if message.solution.originationType = 'CC'
then do;
profile.merchant.BA_PAN[1] = message.creditcard.number;
profile.merchant.BA_transactionCardExpiry[1] =
message.creditcard.messageExpirationDt;
end;

if message.solution.originationType = 'DC'
then do;
profile.merchant.BA_PAN[1] = message.debitcard.number;
profile.merchant.BA_transactionCardExpiry[1] =
message.debitcard.messageExpirationDt;
end;

end;

end;

```

Create the Decision Rule

The Decision rule checks whether there have been five CNP transactions in a row at a merchant, specifically, keyed and e-commerce point-of-sale entry mode transactions. Each transaction is made with a different card, but the following conditions are met:

1. The cards have the same BIN. (Although a BIN is 6 or 8 digits, the number of digits used here to identify the BIN can be modified as needed to achieve optimal rule results. The rule code below uses the first 12 digits.)
2. All of the transactions are for the same amount.
3. The merchant has the same country code on all the transactions.
4. The merchant has the same MCC on all the transactions.
5. The expiry date is the same on each transaction.

6. All of the transactions occur within 5 minutes. (The time frame can be modified as needed to achieve optimal rule results.)

If the conditions are met, then the rule declines the transaction and creates an alert.

To exclude from the rule certain high-volume e-commerce merchants that might cause unnecessary false positives, use an advanced list.

Copy the following rule code into your decision rule.

When selecting the alert type, choose a merchant-level alert type, if available. A merchant-level alert view enables your analyst to see all the transactions that occur at the merchant. This level of visibility helps the analyst identify whether it is a BIN attack.

```
if message.solution.originationType in ('CC', 'DC')
and message.solution.activityType = 'CA'
then do;

    if message.cardfinancial.posEntryType in ('00','01','81','8E','8F')
    and message.cardfinancial.cardPresentInd = '1'
    /*and not lists.high_volume_merchants.contains(message.merchant.name) */
    then do;

        if
        /* each transaction is made with a different card*/
        ( COMPRESS(profile.merchant.BA_PAN[1]) ^=
        (COMPRESS(profile.merchant.BA_PAN[2]))
        or COMPRESS(profile.merchant.BA_PAN[1]) ^=
        (COMPRESS(profile.merchant.BA_PAN[3]))
        or COMPRESS(profile.merchant.BA_PAN[1]) ^=
        (COMPRESS(profile.merchant.BA_PAN[4]))
        or COMPRESS(profile.merchant.BA_PAN[1]) ^=
        (COMPRESS(profile.merchant.BA_PAN[5]))
        or COMPRESS(profile.merchant.BA_PAN[2]) ^=
        (COMPRESS(profile.merchant.BA_PAN[3]))
        or COMPRESS(profile.merchant.BA_PAN[2]) ^=
        (COMPRESS(profile.merchant.BA_PAN[4]))
        or COMPRESS(profile.merchant.BA_PAN[2]) ^=
        (COMPRESS(profile.merchant.BA_PAN[5]))
        or COMPRESS(profile.merchant.BA_PAN[3]) ^=
        (COMPRESS(profile.merchant.BA_PAN[4]))
        or COMPRESS(profile.merchant.BA_PAN[3]) ^=
        (COMPRESS(profile.merchant.BA_PAN[5]))
        or COMPRESS(profile.merchant.BA_PAN[4]) ^=
        (COMPRESS(profile.merchant.BA_PAN[5]))
        )

        /*Card range is the same for all transactions*/

        and substr(profile.merchant.BA_PAN[1],1,12) =
        SUBSTR(profile.merchant.BA_PAN[2],1,12)
        and substr(profile.merchant.BA_PAN[1],1,12) =
        SUBSTR(profile.merchant.BA_PAN[3],1,12)
        and substr(profile.merchant.BA_PAN[1],1,12) =
```

```

SUBSTR(profile.merchant.BA_PAN[4],1,12)
  and substr(profile.merchant.BA_PAN[1],1,12) =
SUBSTR(profile.merchant.BA_PAN[5],1,12)

/*Amount is the same for all transactions*/
  and profile.merchant.BA_amount[1] = profile.merchant.BA_amount[2]
  and profile.merchant.BA_amount[1] = profile.merchant.BA_amount[3]
  and profile.merchant.BA_amount[1] = profile.merchant.BA_amount[4]
  and profile.merchant.BA_amount[1] = profile.merchant.BA_amount[5]

/*Country Code is the same for all transactions*/
  and profile.merchant.BA_countryCode[1] = profile.merchant.BA_countryCode[2]
  and profile.merchant.BA_countryCode[1] = profile.merchant.BA_countryCode[3]
  and profile.merchant.BA_countryCode[1] = profile.merchant.BA_countryCode[4]
  and profile.merchant.BA_countryCode[1] = profile.merchant.BA_countryCode[5]

/*MCC is the same for all transactions*/
  and profile.merchant.BA_MCC[1] = profile.merchant.BA_MCC[2]
  and profile.merchant.BA_MCC[1] = profile.merchant.BA_MCC[3]
  and profile.merchant.BA_MCC[1] = profile.merchant.BA_MCC[4]
  and profile.merchant.BA_MCC[1] = profile.merchant.BA_MCC[5]

/*Expiry Date is the same for all transactions*/
  and profile.merchant.BA_transactionCardExpiry[1] =
profile.merchant.BA_transactionCardExpiry[2]
  and profile.merchant.BA_transactionCardExpiry[1] =
profile.merchant.BA_transactionCardExpiry[3]
  and profile.merchant.BA_transactionCardExpiry[1] =
profile.merchant.BA_transactionCardExpiry[4]
  and profile.merchant.BA_transactionCardExpiry[1] =
profile.merchant.BA_transactionCardExpiry[5]

/*DateTime is within 5 mins*/
  and profile.merchant.BA_transactionDtTm[1]
    - profile.merchant.BA_transactionDtTm[5] <= dhms(0,0,5,0)

  then do;
  detection.decline();
  detection.alert();
  end;

end;
end;

```

After you have created both rules, follow your normal procedures for testing and promoting the rules.

For more advice and support about preventing BIN attacks and implementing strategies, contact your local or global fraud expert at SAS.

Resources

SAS Fraud Decisioning documentation is intended for use by existing customers and requires an access key. You can obtain the access key from your SAS consultant or by contacting SAS Technical Support. To expedite your request, please include SAS Fraud Decisioning in the subject field of the form. Be sure to provide the SAS Site Number for your software license along with your request

Release Information

Content Version: 1.0 March 2026

Trademarks and Patents

SAS Institute Inc. SAS Campus Drive, Cary, North Carolina 27513

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. R indicates USA registration. Other brand and product names are registered trademarks or trademarks of their respective companies.

To contact your local SAS office, please visit: sas.com/offices

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries.

* indicates USA registration. Other brand and product names are trademarks of their respective companies. Copyright © SAS Institute Inc. All rights reserved.

