FASTCash and Associated Intrusion Techniques

By Kevin Perlow

Key Points

- ISO 8583 is a standard that applies to card payments and provides a common set of data fields and a common format for these fields during these financial transactions.
- Since at least 2018, a DPRK-nexus threat group has used malware that incorporates this standard to perform large-scale fraudulent cash withdrawals against a small group of victims.
- Public reporting refers to this technique – and the malware used to carry it out – as FASTCash.

Executive Summary

Since at least 2016, a DPRK-nexus threat group[1] has conducted financially-motived intrusions against companies within or associated with the banking sector. Traditionally, these attacks have involved complicated, long-term workflows and culminate in large SWIFT transactions to threat group-owned accounts held at other banks.

Since 2018 – and possibly as early as 2016 – this threat group has also conducted a novel type of attack in which it injected malware into a bank's payment switch, forcing it to approve fraudulent transactions originating from attacker-controlled Automated Teller Machines (ATM). This technique relies on the manipulation and fraudulent creation of fields within ISO 8583 messages sent to and from these payment switches. Public sources refer to this technique and its associated malware as FASTCash.[i]

This whitepaper supplements a portion of a BlackHat 2020 presentation that details the ISO8583 standard and the FASTCash malware. It offers two key additions to the presented slides. First, this paper provides additional technical details regarding each known FASTCash variant. Second, this paper provides details regarding additional tools from this threat group.[2]

[1] Different vendors and government agencies classify this activity using different – and often conflicting – monikers and clusters. To avoid confusion, this paper does not use any of these designations and focuses solely on a set of specific tools that share code-level characteristics or can be linked through information available on public and semi-public platforms such as VirusTotal.

[2] Due to the sensitive nature of the operational context in which the threat group used (and continues to use) these tools, this paper limits its data to sourcing available to the general public and broader security community; however, the author notes that sensitive source information strongly corroborates the operational relationships shared by these tools. This paper also limits its discussion to tooling used through 2019 to prevent the adversary from strengthening its own operational security posture.

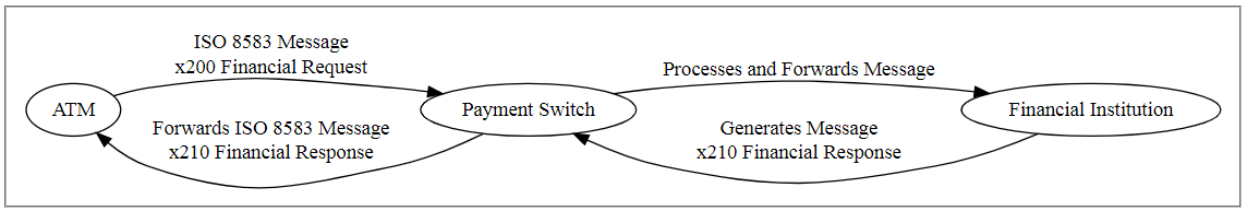
ISO 8583 and FASTCash

ISO 8583

ISO 8583 is a widely-adopted financial standard that specifies a common set of fields and a common format for these fields during card transactions. In short:

- Card transactions at ATMs and Point of Sale (PoS) devices generate ISO8583 messages.
- These messages contain key transaction information, such as the user's account number.
- These messages are sent to a payment switch controlled by the institution that issued the card.
- The payment switch processes these messages and sends them to the card issuer (bank).
- The issuer generates an ISO 8583 response message and sends it to the payment switch.
- The payment switch processes this response and sends it back to the ATM or PoS that originated the message.

The payment switch is the critical point in this process, as financial institutions and financial software and hardware developers implement ISO 8583 slightly differently.



Representative workflow for ISO 8583 transactions

ISO 8583 Message Format

ISO 8583 messages contain information that a bank needs to approve or reject a transaction. There are over one hundred possible fields that specify information such as a user's account number, the currency code used for a balance inquiry, or the amount for a transaction.[ii] [iii]

ISO 8583 messages contain a:

1) **Message Type Identifier**, consisting of:
    a. ISO 8583 Version
    b. Message Classification (e.g. Authorization, Financial, Chargeback)
    c. Message Function (e.g. Request, Response)
    d. Message Source

2) **Bitmap**, which specifies which data fields are present in a message. Because there are over one hundred possible data fields, messages only contain the fields used. The bitmap (and, if needed, secondary bitmap) allows the payment switch to understand what to look for.

3) **Data Elements**, containing the actual data indicated by the bitmap. The data elements make up the majority of the message.[3]

[3] The FASTCash malware contains numerous functions that refer to these as "fields." Data Element is the "official" term, but sources often use these terms interchangeably.

The snippet below represents a sample ISO 8583 message, with different message parts and data elements highlighted.

0200420004000000000216123456789012345606091730300114567 89ABC1000123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789

Deconstructed, this message indicates the following:

- **MTI** – 0200
  - Version: 0 (1987)
  - Classification: 2 (Financial Message),
  - Function: 0 (Request)
  - Source: 0 (Acquirer)

- **Bitmap** – 4200040000000002
  - Indicates data elements 2, 7, 22, 63

- **Data Elements**
  - DE 2 (PAN) – 2 digits indicating length + PAN
    - 16 + 1612345678901234 56
  - DE 7 (Transmission Date/Time)
    - 0609173030 = 06-09 17:30:30 UTC
  - DE 22 (Point of Service Entry Mode)– 2 digits for mode + 1 digit indicating PIN availability
    - 011 = 01 (Manual Entry) + 1 (PIN entry available at terminal)
  - DE 63 – Reserved for private use

The actual process for decoding a bitmap is out of scope for this paper, but an excellent resource discussing the structure and methodology for doing so is available online,[iv] as are several tools that perform the decoding automatically.[v]

Different payment switches implement slightly different requirements for each data field; for example, an older publicly available ISO 8583 document for MasterCard Debit Switch indicated that all 02xx (1987, Financial) messages require DE 43, the Card Acceptor Name and Location, to "satisfy national regulatory requirements."[vi] This is therefore a custom (and practical) requirement, rather than a part of the standard.

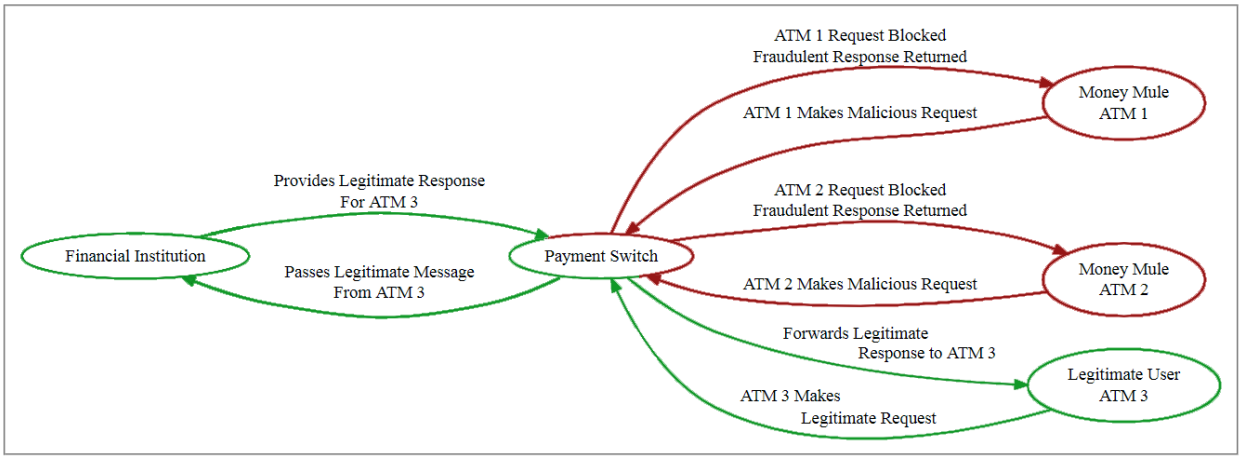
Similarly, different financial institutions use or opt not to use several "private fields." For example, DE 127 is referenced in the Windows version of the FASTCash malware. MasterCard Debit Switch documentation defers to proprietary Customer Processing Systems for this field's use,[vii] Star Northeast expects it to contain data used for certain types of preauthorization,[viii] and RuPay expects it to contain biometric data in a tag-length-value (TLV) format.[ix] These differences limit visibility into the purpose of the particular reference within the malware.

FASTCash

FASTCash refers to a malware family and technique that relies on the ISO 8583 standard to authorize fraudulent ATM withdrawals. The attackers inject the FASTCash malware into the process on the payment switch responsible for receiving these transactions. The malware intercepts incoming messages by hooking the send and recv functions:

- If the transactions meet a certain criteria, they are approved without ever reaching the bank. The malware can support both a withdrawal request *and* a balance inquiry request.

- If they do *not* meet the criteria, they are passed to and from the bank as a regular transaction.



The United States Department of Homeland Security (DHS) first publicly disclosed the FASTCash malware in 2018 in a Malware Analysis Report (MAR) containing brief descriptions for two variants[4] of this malware as well as descriptions of associated tools. This report examines workflows for:

MD5 Hash	Classification	Description
46b318bbb72ee68c9d9183d78e79fb5a	AIX Type 1	A likely earlier version of the malware that contains IP whitelisting but no card blacklisting.
d790997dd950bb39229dc5bd3c2047ff	AIX Type 2	A version of the malware with a placeholder for a blacklisting function and a consolidated message generating structure. This version is likely a "bridge" between the AIX Type 1 and Windows variants.
c4141ee8e9594511f528862519480d36	Windows	A version of the malware from 2019 designed for Windows systems. Appears to be a "port" and update of AIX type two, with a fully functional blacklisting feature and similar message generating logic. This version was uploaded to VirusTotal in 2019 and first referenced publicly in 2020.[x 5]

[4] DHS actually included three hashes in its disclosure, but this whitepaper considers two of the three FASTCash files in the DHS report to be the same variant.

[5] Several aspects of this public report contain technical inaccuracies, but the report correctly lists a hash for the Windows FASTCash malware.

AIX Type 1

The first FASTCash sample is presumed to be the oldest, and contains several core libraries that are implemented or replicated within newer versions of the malware. The AIX versions of this malware contain descriptive function names that indicate ISO 8583 interaction, including the following:

- DL_ISO8583_MSG_GetField_Str
- DL_ISO8583_MSG_SetField_Str
- CopyMsgFieldStr

These functions operate with a consistent pattern. Prior to the function calls, the malware loads an integer into register 3. This integer indicates the field being copied, retrieved, or set. For the "SetField_Str" call, the malware also loads a string into register 0. This string represents the data that will be placed in a field.

```
ld      r0, LC..43_TC # _eg64.rw+0xB0 # (0110, authorization response)
addi    r9, r31, 0x8B0
li      r3, 0         # Field 0: MTI
mr      r4, r0
mr      r5, r9
bl      .DL_ISO8583_MSG_SetField_Str
nop
mr      r0, r3
std     r0, 0x88(r31)
```

The Field 0[6] value being set to the string "0100" indicating an authorization response message

```
addi    r9, r31, 0x8A8
addi    r0, r31, 0x80
li      r3, 2         # Field 2 (Primary Account Number)
mr      r4, r0
mr      r5, r9
bl      .DL_ISO8583_MSG_GetField_Str
nop
mr      r0, r3
std     r0, 0x78(r31)
```

Retrieval of the Field 2 (PAN) value

```
lwz     r0, 0xBC(r31)
clrldi  r9, r0, 32
lbz     r0, 0xC8(r31)
clrldi  r0, r0, 56
ld      r11, 0xC0(r31)
addi    r10, r31, 0xC9
ld      r3, LC..131_TC # _eg64.rw+0x278
mr      r4, r9
mr      r5, r0
mr      r6, r11
mr      r7, r10
bl      .out_dump_log
nop
li      r0, 0xFF
stw     r0, 0xAC(r31)
b       loc_10009AEC
```

```
_eg64.rw: .byte 0xA, 0, 0, 0, 0, 0, 0, 0, 0x25, 0x73, 0x2D,
                                # DATA XREF: .data:_eg64.r
                                # .data:LC..4_TC↓o
          .byte 0x20, 0x2D, 0x2D, 0x2D, 0x2D, 0x2D, 0x2D, 0x
          .byte 0x73, 0, 0, 0, 0x25, 0x30, 0x32, 0x78, 0x20,
          .byte 0x2D, 0x2D, 0x2D, 0x2D, 0x2D, 0x2D, 0x2D, 0x
          .byte 0x30, 0x32, 0x58, 0, 0, 0, 0, 0x30, 0x31, 0x
          .byte 0x31, 0x32, 0x64, 0x25, 0x63, 0x25, 0x63, 0x
          .byte 0x38, 0x39, 0x21, 0x28, 0x70, 0x23, 0x79, 0x
          .byte 0x61, 0x28, 0x73, 0x68, 0x72, 0x5F, 0x36, 0x
          .byte 0x25, 0x6C, 0x6C, 0x58, 0x2C, 0x20, 0x74, 0x
          .byte 0x69, 0x74, 0x5F, 0x69, 0x73, 0x6F, 0x5F, 0x
          .byte 0x64, 0, 0, 0, 0, 0x55, 0x6E, 0x6C, 0x6F, 0x
          .byte 0x54, 0x5F, 0x53, 0x49, 0x47, 0x4E, 0x41, 0x
```

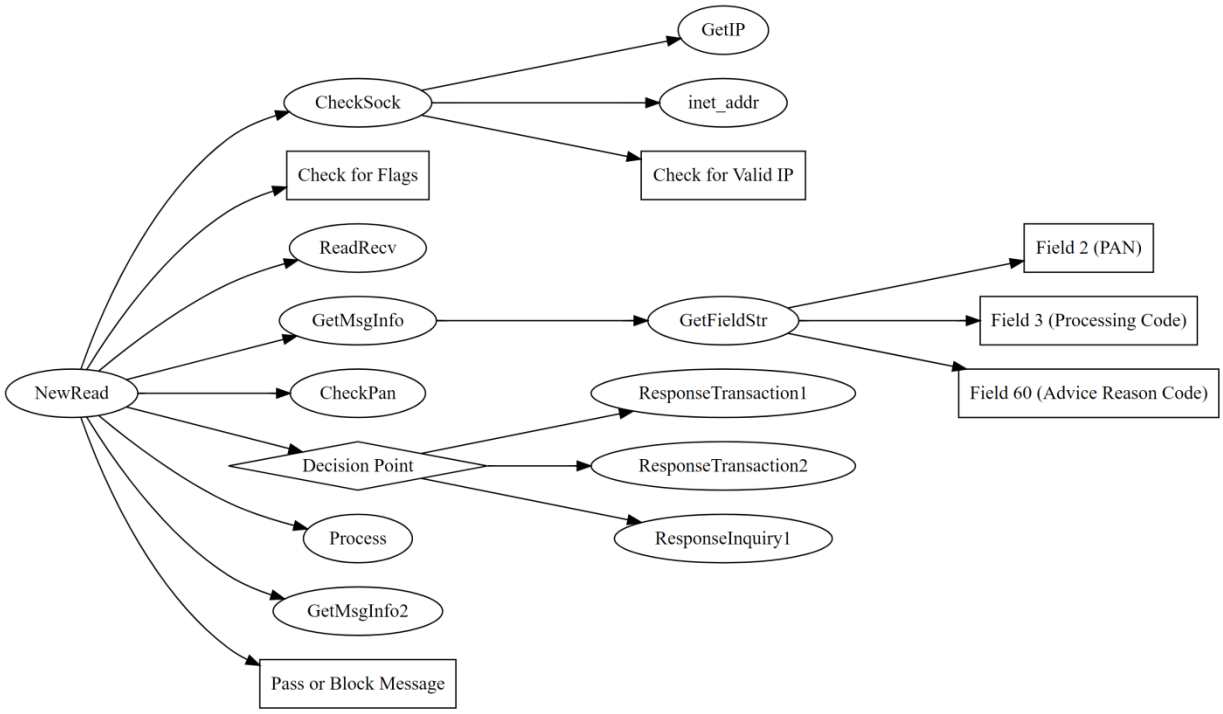
```
.......%s--------------- ISO858
3 MSG DUMP ---------------%s....
[%03d : %03X] ..%-60s...%02x ...
   .....------------------------
-----------------------%s%s....
%08lxh .%02X....0110....51......
00......%0*d....%c%c01%3sC%012d%
c%c02%3sC%012d..356.....dkfjy)1*
290(yY89!(p#y!@pURH1;2EH89Pu*KF(
p3RHH89.libc.a(shr_64.o)........
read....write...Load....func=%ll
X, toc=%llX.....init_hashmap suc
c.......init_iso_handle succ....
HF_Initialize failed....Unload..
[recv] len=%d...SET_SYNC........
```

```
Blocked Message(msg=%04x, term=%02x, pcode=%06x, pan=%s)
```

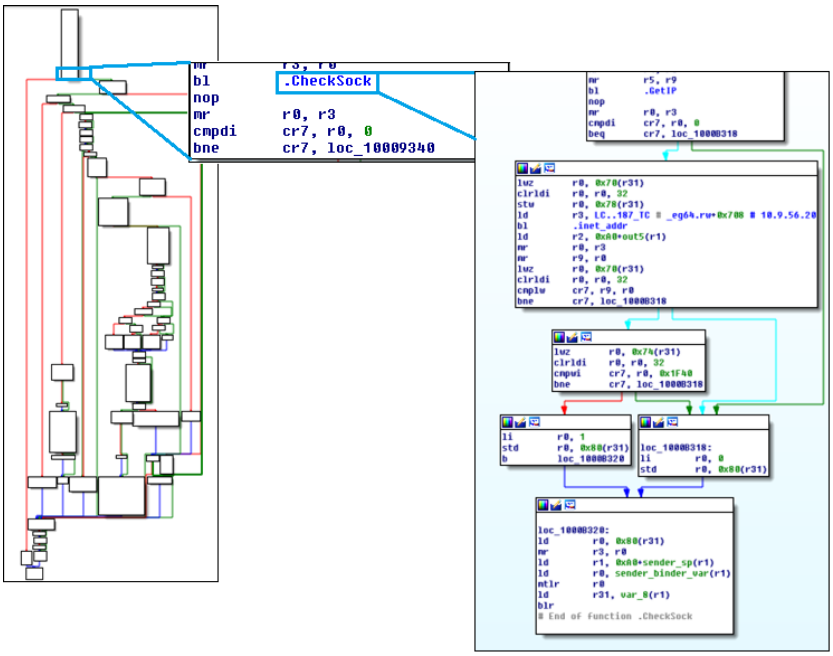
Strings within the AIX malware

[6] Although the ISO 8583 standard does not describe a field designated as "0", FASTCash considers "Field 0" to be the MTI, likely for practical purposes.

With these core concepts in place, the following graph depicts the general workflow for the AIX Type 1 version of the FASTCash malware. Ovals represent an actual function name defined within the malware, and rectangles represent an action taken or set of data retrieved.



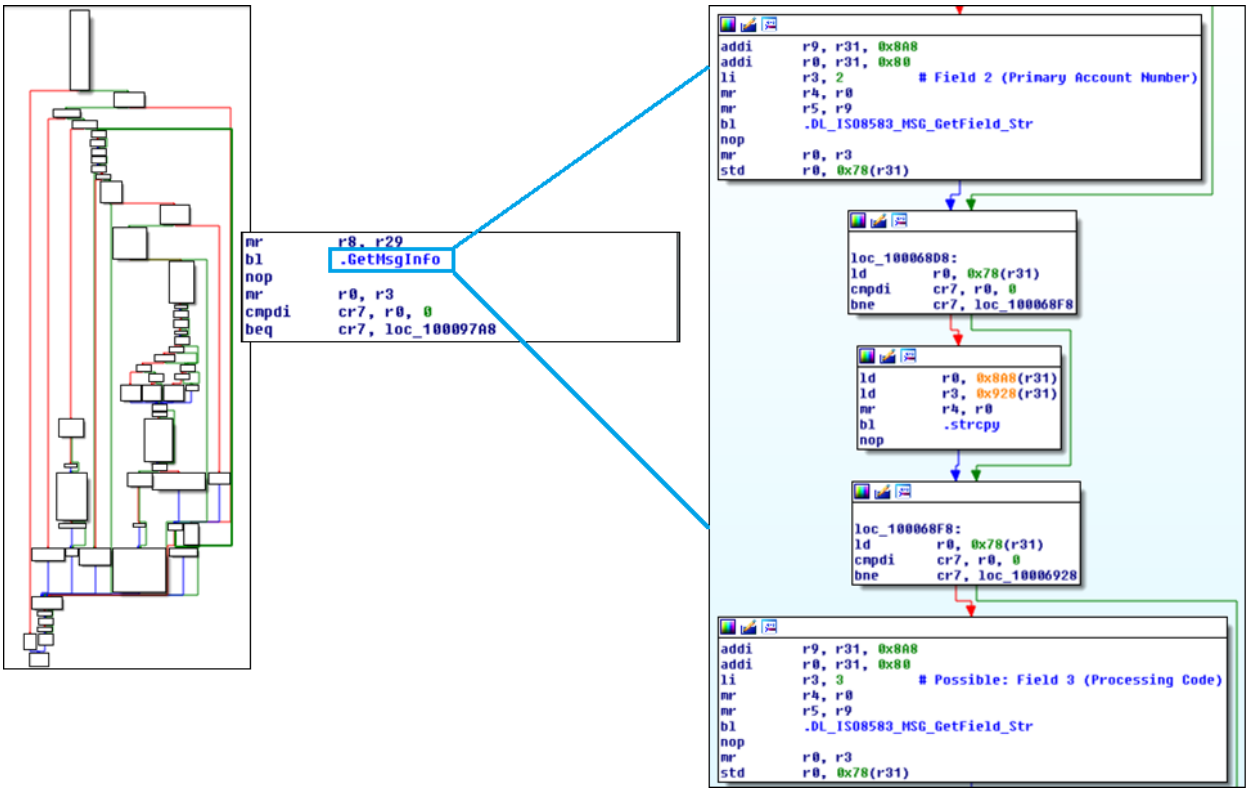
The AIX malware begins with an IP address check called from the NewRead workflow, shown in the image below. The malware expects valid messages to be sent from a specific internal IP address.



NewRead function (left) calling CheckSock (middle and right) to validate message origination

The malware performs an apparent set of “flag” checks; although their purpose is unconfirmed, debugging strings suggest that these ensure that a “block” flag isn’t set and that memory was properly allocated prior to continuing.

If these checks are passed, the malware calls a function named GetMsgInfo to retrieve Field 2 (PAN), Field 3 (Processing Code), Field 60, and the MTI from the message. Field 60 is a “Private/Reserved” field. Examples of its use include holding the “Advice Reason Code”[xi] or “Additional EMV Information.”[xii]



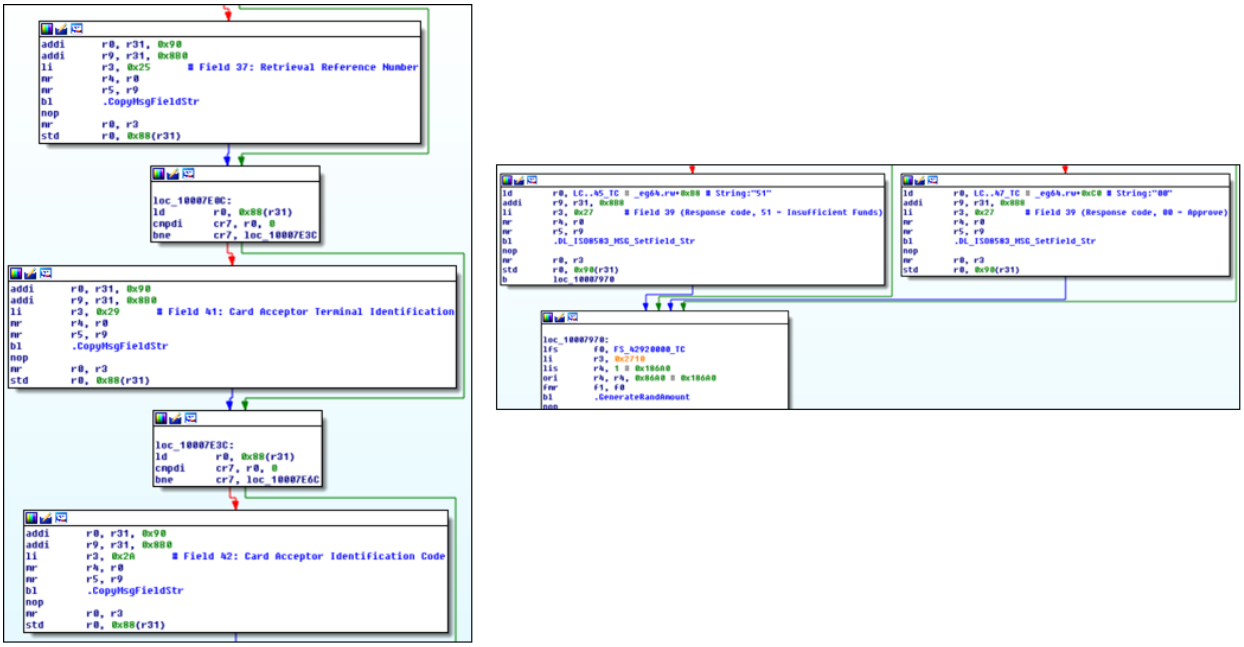
NewRead function (left) calling GetMsgInfo, which uses GetFieldStr to retrieve message fields

The malware then checks the PAN against a whitelist of cards. If it finds a match, it can take one of three options:

- GenerateResponseTransaction1
- GenerateResponseTransaction2
- GenerateResponseInquiry1

The first two provide a mechanism for FASTCash to approve a fraudulent withdrawal request from attacker-controlled cards. Combine with a whitelist, this allows the attackers to ensure that *only* their cards cause this behavior. A full-scale cash-out would dispense money to non-attackers and would likely result in earlier detection.

In each of these two functions, the malware uses CopyMsgFieldStr to copy data from the original message to construct a portion of a response. The malware sets one of two response codes in Field 39 – response code 51 (“Insufficient Funds”) in the event of an error, or response code 00 (“Approved”) for successful construction of a new message. The malware also generates a random withdrawal amount as part of the message.



Portions of the "GenerateResponseTransaction1" Workflow

The two GenerateResponseTransaction functions are nearly identical, with the primary difference residing in the fields that each one copies. One possibility for this discrepancy is that the environment in which the malware operated supported multiple types of ISO 8583 transactions, including some that did not require the Track 2 data omitted in the second transaction type; unfortunately, without access to the environment this remains speculation.

Transaction 1

- 2 – PAN
- 3 – Processing Code
- 4 – Amount, Transaction
- 7 – Transaction Date and Time
- 11 – System Trace Audit Number
- 14 – Date, Expiration
- 19 – Acquiring Country Code
- 22 – POS Entry Mode
- 25 – POS Condition Code
- 32 – Acquiring Identification Code
- 35 – Track 2 Data
- 37 – Retrieval Reference Number
- 41 – Card Acceptor Terminal ID
- 42 – Card Acceptor ID
- 44 – Additional Response Data
- 49 – Currency Code, Transaction
- 62 – INF Data (binary)
- 63 – Network Data (binary)

Transaction 2

- 2 – PAN
- 3 – Processing Code
- 4 – Amount, Transaction
- 7 – Transaction Date and Time
- 11 – System Trace Audit Number
- ~~14 – Date, Expiration~~
- 19 – Acquiring Country Code
- ~~22 – POS Entry Mode~~
- 25 – POS Condition Code
- 32 – Acquiring Identification Code
- ~~35 – Track 2 Data~~
- 37 – Retrieval Reference Number
- 41 – Card Acceptor Terminal ID
- 42 – Card Acceptor ID
- 44 – Additional Response Data
- 49 – Currency Code, Transaction
- 62 – INF Data (binary)
- 63 – Network Data (binary)

The final possible action, GenerateReponseInquiry1, allows the malware to fraudulently construct a response to a balance inquiry issued by an attacker. Structurally, this workflow is similar to the workflows within the ReponseTransaction routines; however, this action clearly serves a specific purpose. While the attacker's intent cannot be certain, a likely explanation is that this allows the attacker to test if the malware is properly intercepting ISO 8583 messages without actually performing a cash withdrawal.

From a threat intelligence standpoint, this function contains valuable information: the results of a balance inquiry are placed in Field 54 (Additional Amounts) using a format string:

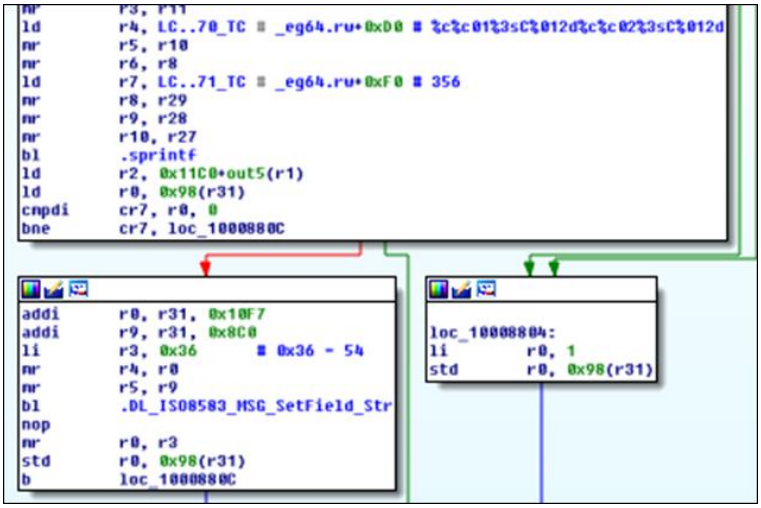
cc01356Cddddddddddddd

This format string consists of four key segments:[xiii] [xiv]

- Account Type – Two digits (e.g. 10 for a Savings Account or 20 for a Checking Account)[xv]
- Amount Type – The type of balance shown to the user (01, Ledger Balance)
- Currency Code – The currency units for the balance shown (356, Indian Rupee)
- Balance – Type Digit (0, C, D) + Amount (12 digits)

This sample of the malware will *always* return a Credit Amount in units of the Indian Rupee. From a threat intelligence standpoint, this information hints that that victim is located in India or a country that would be expected to use this currency. As with the transactions, the returned value is randomized.

During legitimate behavior, this type of workflow would be expected when a user opts to check their account balances at an ATM preceding or following a transaction. However, because this malware intercepts the message before it reaches the bank and randomizes the response, the value has no actual bearing on anything tied to the account or card.



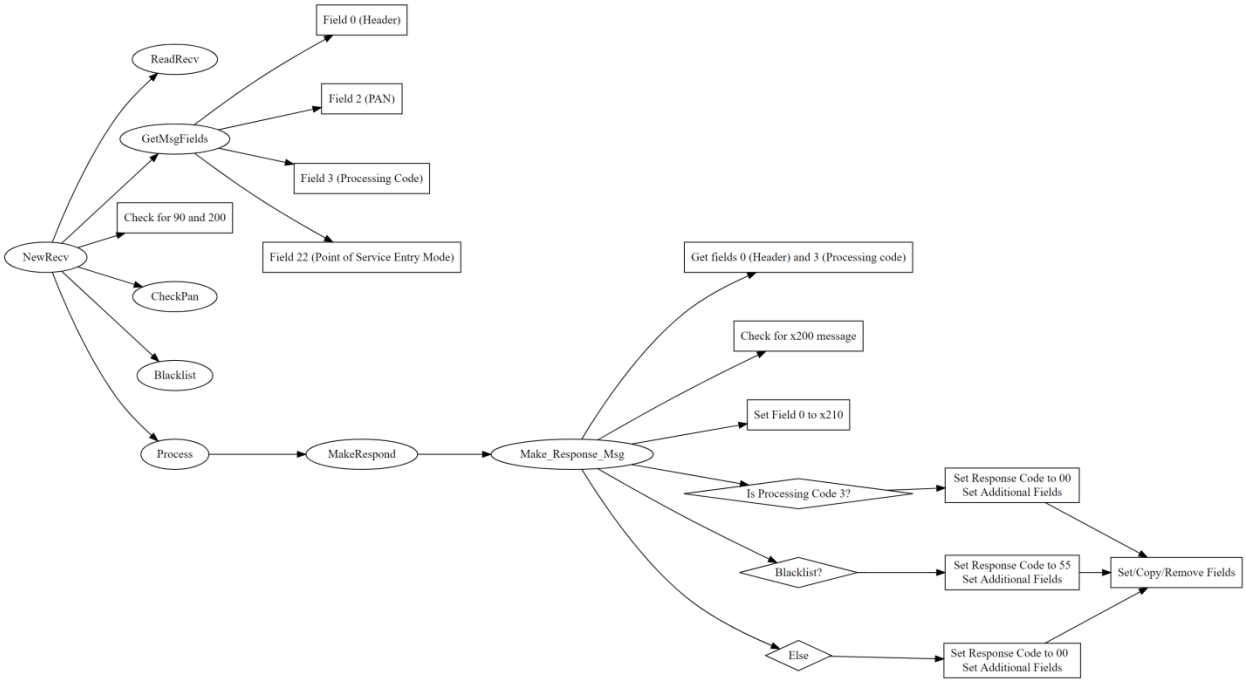
Format string and currency constant within GenerateResponseInquiry1 function

Following these three branches, the malware finalizes the message and logs that it has successfully blocked it (or, if these workflows were bypassed due a PAN not being on the whitelist, logs that it passed) before sending the response back to the original source.

AIX Type 2

The second AIX variant contains a similar set of function names and at a high level serves the same purpose: the malware either fraudulently approves withdrawal requests or responds to attacker-owned balance inquiry requests. However, this version contains two significant changes:

- An incomplete “blacklist” function (later completed for the Windows version)
- A more streamlined workflow for responding to balance inquiries and transaction requests



As the workflow graph shows, this AIX variant foregoes the three separate response functions and consolidates them into a single function call. This creates several efficiencies for the malware developers: rather than writing nearly identical code for three slightly different functions, this version applies one set of logic to all three possible response options.

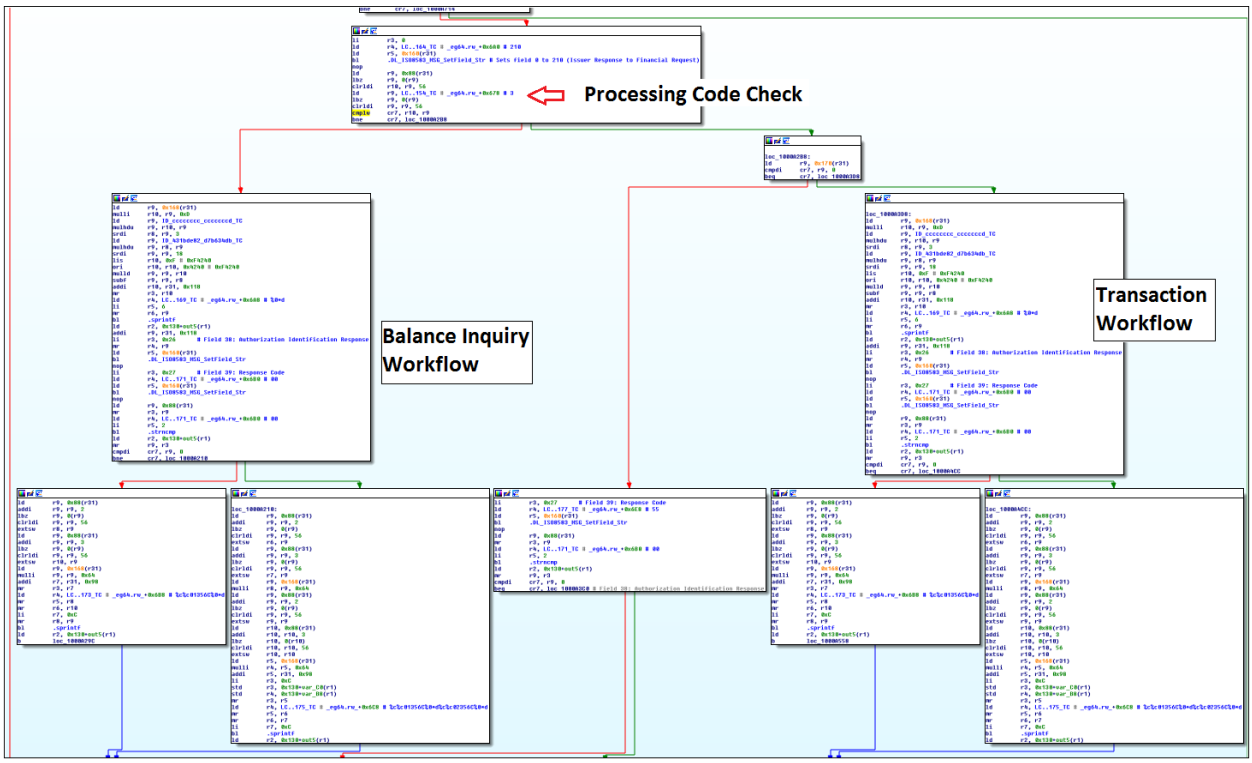
As part of this function, the malware checks that the message type is an x200 (Financial Request) message and sets the response to x210 (Financial Response).

- If the Processing Code starts with 3, the malware will return a balance inquiry response using the same currency code (356, Indian Rupee) as the AIX Type 1 variant.

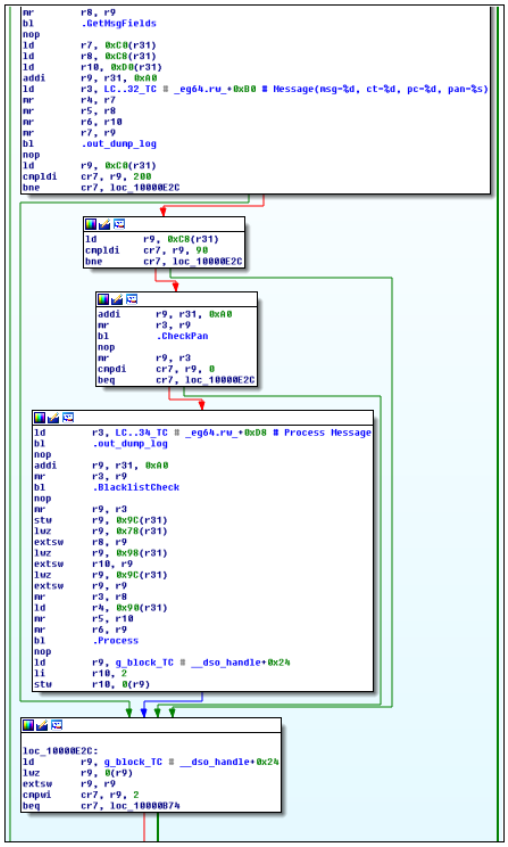
- If the Processing Code does *not* start with 3, the malware will fraudulently approve the transaction request.

- There is a third workflow available that open source reporting[xvi] suggests is affiliated with a blacklist function, although this is unconfirmed given that the blacklist function is incomplete in this version.

This logic aligns with high level details provided in a Symantec report in November 2018,[xvii] although the researchers did not list an analyzed hash to verify that this was the file examined.



Consolidated response logic in AIX Type 2 variant



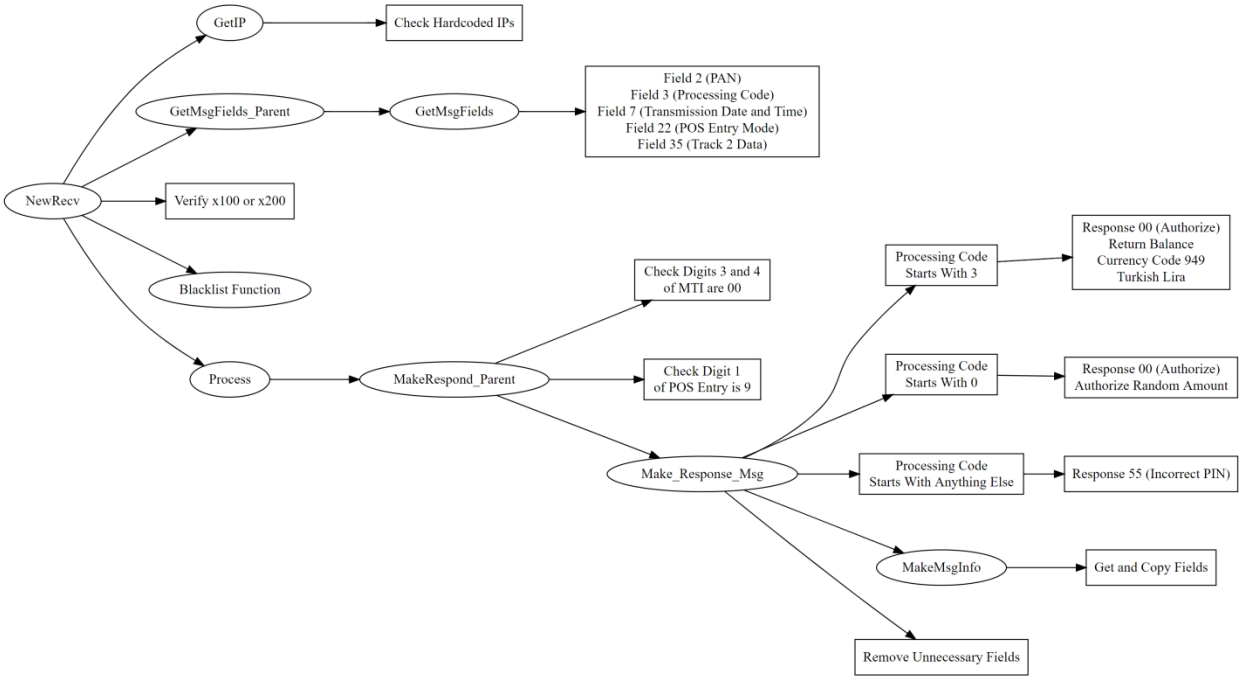
Overhead workflow for performing 0x200, 90, and blacklist checks

Windows

The Windows variant of FASTCash closely resembles the AIX Type 2 version, with the following changes:

- A completed blacklist function
- Additional checks prior to generating a response message
- Stripped function names (due to the programming language change)
- Support for Field 127, a Private/Reserved field likely specific to the target environment

Despite the function name stripping, many functions within this variant are nearly identical to their AIX counterparts. The graph below therefore uses the same or similar names where appropriate.



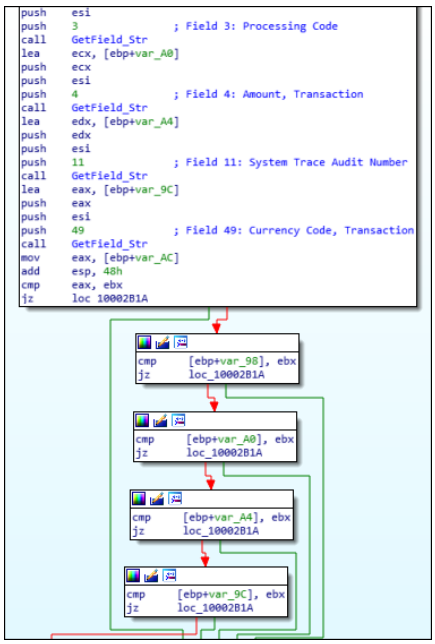
The malware performs an IP address check to determine if the message originates from one of two hardcoded IP addresses. It then retrieves an initial set of fields, including the MTI and verifies that the message is either an x100 (Authorization) or x200 (Financial) request. After validating the PAN against both a whitelist *and* a blacklist, the malware proceeds to a "Process" function similar to AIX Type 2.

The Process function calls a parent function that performs three additional actions:

1) The parent function calls GetField_Str to retrieve Field 0 (MTI) and Field 22 (POS Entry Mode)
2) The parent function checks that digits 3 and 4 in the MTI are "00" (ensuring it is a "request")
3) The parent function checks if the POS Entry Mode begins with a "9" (indicating a card transaction)

Following these checks, the malware calls the function responsible for generating the response message.

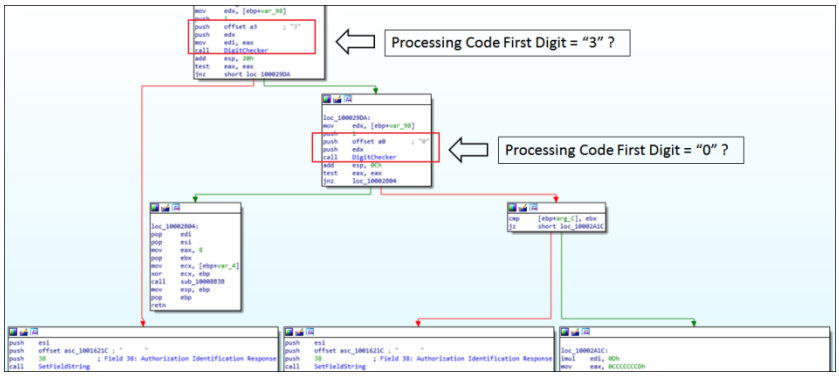
Within the response message function, the malware once again validates several fields. The malware retrieves Fields 3, 4, 11, and 49 plus the MTI – if it cannot retrieve one of these fields, the malicious workflow exits.



Validation of field data within MakeResponse_Msg function

If these checks pass, the malware follows a workflow similar to the AIX Type 2 version. It will:

- Generate a balance response inquiry if the first digit in the processing code is a 3
- Generate a fraudulent withdrawal if the first digit in the processing code is a 0
- Generate an “Invalid PIN” response if the first digit of the processing code is neither 0 nor 3



Decision workflow

The malware once again uses a format string for the balance inquiry; however, this time the currency code is set to 949, the Turkish Lira. As with the previous two variants, the malware uses randomized amounts for these workflows.

Following this routine, the malware finalizes the response message and transmits it back to the originating device, using functions similar to the previous two variants.

Tooling Clusters

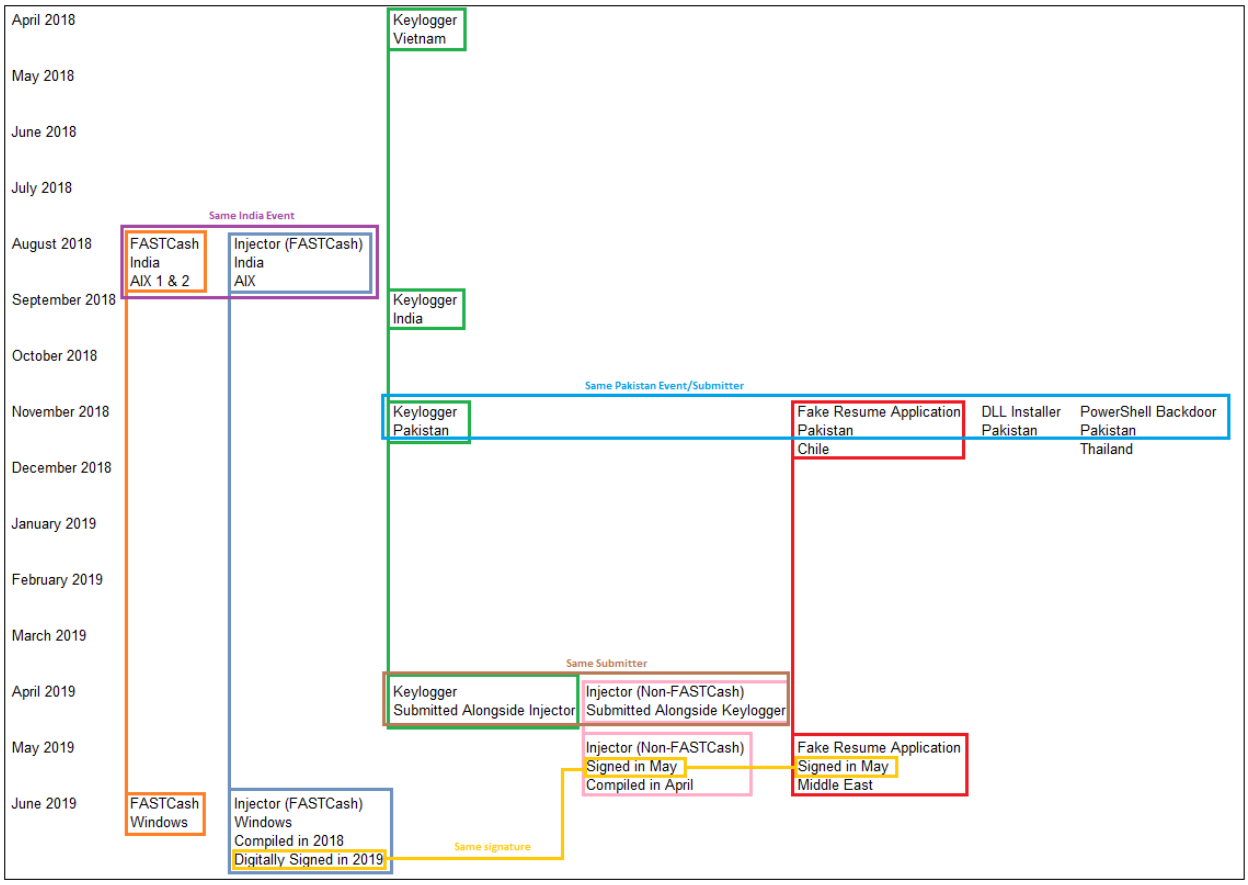
Timeline

FASTCash can be associated with other tools from the same threat group based on shared code characteristics and inferred operational relationships through VirusTotal submissions. Discussing individual operations is outside of the scope of this paper; however, sensitive sourcing corroborates these relationships at an operational level.

During its FASTCash and SWIFT operations, the threat group also uses:

- A keylogger
- PowerShell-based backdoors
- PowerShell-based installers
- Process injectors

The graph below demonstrates how these tools are related at a code and operational level. This graph (and the subsequent tool descriptions) omits a select set of tools that the attacker continues to use as of July 2020 that have not been publicly described. Hashes for the listed files are included at the end of the paper.



Fake Resume Generator and Secondary Payload

The Fake Resume Generator likely represents the one of the group's entry-stage tools to gain a foothold on a victim's network. This tool typically purports to belong to a legitimate company, often in the payment processing or interbank network sectors.

There are multiple versions of this tool, but in general this program performs the following steps:

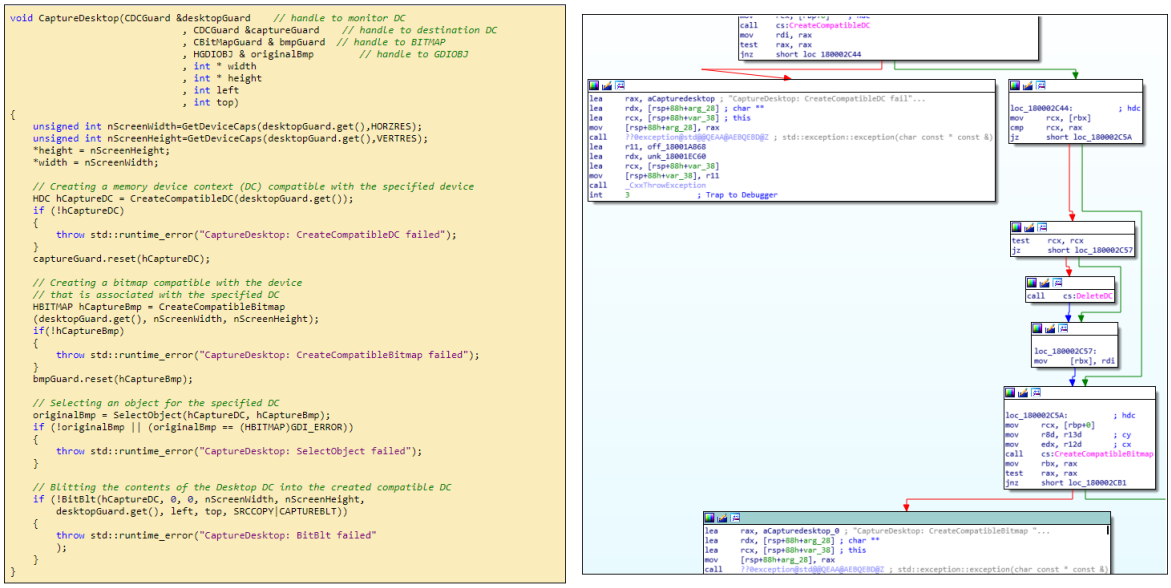
1) Downloads or runs a PowerShell-based backdoor
2) Opens a window for the user to enter resume-related information
3) Generates a PDF with this information

The malicious actions occur regardless of whether or not the victim actually generates the PDF. One second-stage payload, likely related to a late-2018 attack based on relationship information available on VirusTotal,[xviii] supported C2-initiated command-line execution, secure file deletion, and file writing/running of additional payloads. This tool also installed itself as either a scheduled task or a service for persistence.

PSLogger Keylogger/Screenshot Grabber

The keylogger and screen grabber is delivered via PowerShell and executable injectors, likely on targets identified as being both higher priority and highly-used. There are DLL and EXE versions of this tool.

The screenshot functionality of PSLogger likely derives from open source code examples: the code strongly resembles example content available online[xix] for generating multi-monitor image files. This content is zipped using code resembling the publicly available XZip library or a similar code set.[xx]



Open-source screenshot code (left) resembling threat group keylogger code (right)

The filenames and locations for zipped screenshots and logged keystrokes have typically changed from version to version, but are usually stored in the victim's AppData\Local\Temp directory or a subdirectory of this folder. The screenshot filename will typically contain a timestamp (e.g. tmp_userA_0121_142748).

PowerShell Backdoor

For a roughly one-year period, the threat group deployed a PowerShell backdoor at victim sites alongside several of the other tools discussed in this report. A VNCERT report[xxi] was among the first to publicly disclose hashes for this malware, some of which had been uploaded to VirusTotal with other tools from an event in Pakistan. The attackers have likely largely phased out this tool following its public disclosure and replaced it.

The samples initially disclosed to the public were likely "incomplete," as they contained several empty functions. Based on the decoded function names present within the backdoor, a completed version would be expected to support:

- File Transfer
- Screen Grabbing (through Base64-encoded script supplied by the C2)
- File Compression
- Command Execution
- Keylogging
- Registry Modification
- Collecting and Sending System Information

The attackers went through significant efforts to obfuscate the text: nearly every function named has been replaced with an MD5 hash, which significantly slows down analysis. However, the contents of certain command variables are only Base64-encoded, which facilitates identifying these functions' intent.

PowerShell DLL Installer

The DLL installer was also uploaded alongside several tools in Pakistan in late 2018 and early 2019.[xxii] This script has a configuration that accepts five input variables: a .dll, .dat, and .cfg file alongside a service name and a delete setting. This installer:

1) Moves the input .dll file (e.g. AppChk.dll) from the current working directory to the System32 directory and names it using the service name variable (e.g. appchk.dll)

2) Moves the input .cfg file (e.g. AppChk.cfg) from the current working directory to the System32 directory and names it using the service name variable and a .mui extension (e.g. appchk.dll.mui)

3) Moves the input .dat file (e.g. AppChk.dat) from the current directory to c:\programdata\microsoft\ using the service name variable, a generated number, and a .ldx extension (e.g. appchk6.ldx)

4) Creates and starts a service pointing to the DLL file

If the deletion switch is set to "1" when the script is run, the malware terminates the service, deletes the service, overwrites the bytes in all three written files, and then removes all three files from the disk. This deletion function shares code with at least one other PowerShell tool not listed in this report and shares principles with a secure delete function included in the "non-FASTCash Injector" tool.

Injector (Non-FASTCash)

The threat group uses at least two types of injectors (in addition to PowerShell loaders that run payloads in memory). One of these is specifically for the FASTCash malware and the other injects arbitrary payloads into memory. One possibility is that this injector is used specifically for the keylogger payload, as one sample was uploaded to VirusTotal directly alongside it; however, this remains speculation.

The non-FASTCash injector accepts four possible parameters:

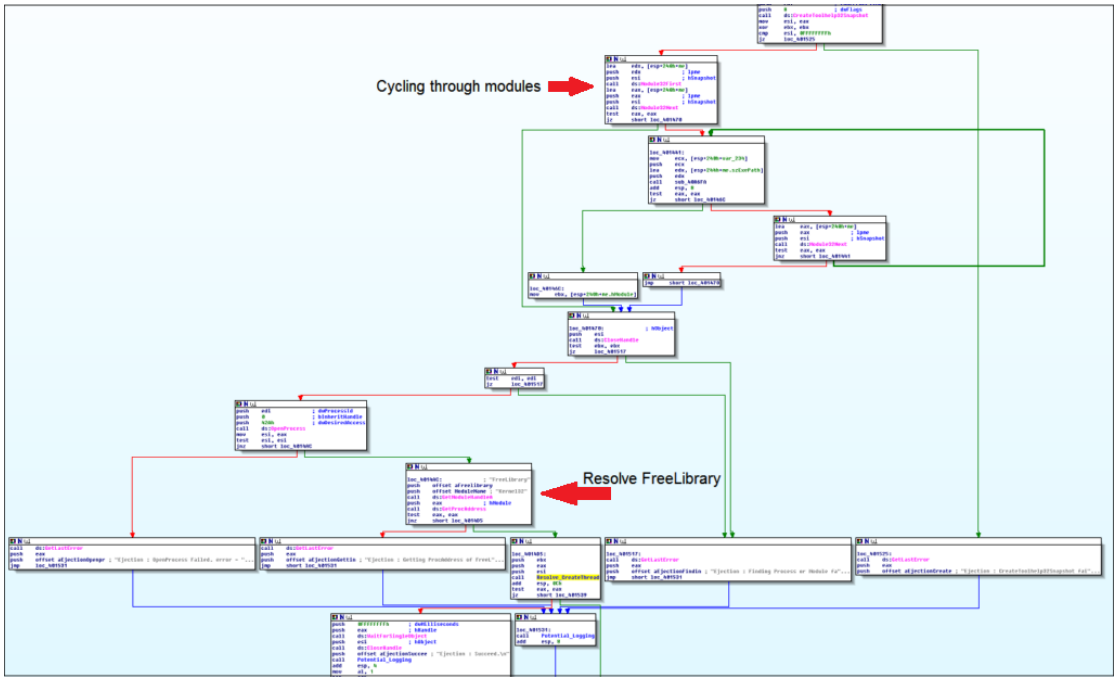
- File path (required, points to payload to be injected)
- -S (causes the malware to sleep before injection, also causes secure deletion of payload)
- -D (causes the malware to securely delete the payload)
- -E (skips the deletion step)

The injector writes the specified payload into the process memory of explorer.exe and then executes it.

Injector (FASTCash)

The FASTCash injector is likely designed specifically to inject the FASTCash payload into the memory of a payment switch application. The Windows version of this tool expects several command line arguments:

- The path to the injector (under normal circumstances, automatically included at runtime)
- An integer value (1 or 2) that specifies the operational mode (inject or eject)
- A PID that specifies the target process for injection
- A path to the DLL to be injected into the target process



Depending on the specified operational mode, the malware will either write the specified DLL into the target process and create a threat or resolve and call the FreeLibrary API to remove it from the process.

Conclusions

The FASTCash technique and malware represents a novel way for its operators to conduct large-scale cash-outs against financial institutions, but several complexities inherent in this approach limit the frequency that these attacks can be carried out:

- Each network will have a slightly different implementation of the ISO 8583 standard
- Each network will have different security controls on critical applications
- This attack requires a network of money mules to perform the cash withdrawals

These three factors are likely significant drivers for the apparent relative rarity of successful cash-outs through this technique. Still, the technique is a proven one and makes use of the fact that the ATMs targeted are largely operating “as expected” and without being manipulated directly; put more simply, the attackers can conduct an ATM cash-out without installing malware on these devices (either remotely or via physical access).

The threat group’s supplemental toolset is effective, but largely relies on TTPs that are significantly less distinct: the group often uses PowerShell-based tools, scheduled tasks, and malicious services to gain access to and maintain a foothold on target networks. When the attackers combine these techniques with their intricate knowledge of financial networks, however, the threat created is significantly larger.

File Hashes Referenced

Keylogger	d13c15016b5ea2a88434d427bb47110d (India)
	d45931632ed9e11476325189ccb6b530 (api)
	34404a3fb9804977c6ab86cb991fb130 (Pakistan)
	3122b0130f5135b6f76fca99609d5cbe (Vietnam)
Injector (FASTCash)	89081f2e14e9266de8c042629b764926 (Windows)
	b3efec620885e6cf5b60f72e66d908a9 (AIX)
Injector (Non-FASTCash)	b9ad0cc2a2e0f513ce716cdf037da907 (signed)
	a042e53edd734b6a96ef9ab82bec8193 (unsigned)
Fake Resume Application	d1d779314250fab284fd348888c2f955 (Middle East)
	b484b0dff093f358897486b58266d069 (Chile)
	4c26b2d0e5cd3bfe0a3d07c4b85909a4 (Pakistan)
DLL Installer	a827d598b4d13005526839473f38a01b (Pakistan)
PowerShell Backdoor	b12325a1e6379b213d35def383da2986 (Thailand)
	7c651d115109fd8f35fddfc44fd24518 (Pakistan)
FASTCash	c4141ee8e9594511f528862519480d36 (Windows)
	46b318bbb72ee68c9d9183d78e79fb5a (AIX Type 1)
	d790997dd950bb39229dc5bd3c2047ff (AIX Type 2)

[i] "HIDDEN COBRA – FASTCash Campaign" United States Department of Homeland Security, 2 October 2018. https://us-cert.cisa.gov/ncas/alerts/TA18-275A

[ii] "ISO8583 financial transaction message format" ADM Factory, https://www.admfactory.com/iso8583-financial-transaction-message-format/

[iii] "Financial Transaction Message Tools - ISO 8583 (1987) Data Fields" http://www.fintrnmsgtool.com/encode-iso87-bitmap.html

[iv] "ISO8583 – A layman's guide to understanding the ISO8583 Financial Transaction Message" http://www.lytsing.org/downloads/iso8583.pdf

[v] "Financial Transaction Message Tools" http://www.fintrnmsgtool.com/decode-iso87-bitmap.html

[vi] "ISO 8583–1987 Data Element Definitions", MasterCard, 2002. Retrieved from http://read.pudn.com/downloads305/doc/1357633/ISO8583_1987DE_Def.pdf

[vii] "ISO 8583–1987 Data Element Definitions", MasterCard, 2002. Retrieved from http://read.pudn.com/downloads305/doc/1357633/ISO8583_1987DE_Def.pdf

[viii] "STAR ISO 8583 MessageFormat Guide" FirstData, February 2011. https://www.scribd.com/document/119326455/STAR-ISO-8583-Message-Format-Guide-02-11

[ix] "RuPay – Online Switching Interface Specification" National Payments Corporation of India, 2013. Retrieved from https://www.scribd.com/document/254420544/RuPay-Online-Switching-Interface-Specification

[x] "APT38 Threat Analysis Report" ADEO, April 2020. https://adeo.com.tr/wp-content/uploads/2020/05/ADEO-Lazarus-APT38.pdf

[xi] "ISO 8583–1987 Data Element Definitions", MasterCard, 2002. Retrieved from http://read.pudn.com/downloads305/doc/1357633/ISO8583_1987DE_Def.pdf

[xii] "1LINK Message Format and Data Element Definitions." 1Link, 2018. Retrieved from https://www.scribd.com/document/418059282/1LINK-Technical-Document-Data-Element-Definitions-and-Message-Format-v5-7

[xiii] "What is the structure of Field No. 54 (P54) in the the ISO 8583 Standard" StackOverflow, March 2015. https://stackoverflow.com/questions/26119041/what-is-the-structure-of-field-no-54-p54-in-the-the-iso-8583-standard

[xiv] "ISO 8583 Field Açıklamaları (F54)" Arif Unal, 9 October 2018. http://unalarif.com/yazi/iso-8583-field-aciklamalari-f54/

[xv] "jPOS Common Message Format" jPOS.org. http://jpos.org/doc/jPOS-CMF.pdf

[xvi] "FASTCash: How the Lazarus Group is Emptying Millions from ATMs" Symantec, 8 November 2018. https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/fastcash-lazarus-atm-malware?om_ext_cid=biz_social_NAM_facebook_Asset%20Type%20%20-%20Blog,Threat%20Intelligence

[xvii] "FASTCash: How the Lazarus Group is Emptying Millions from ATMs" Symantec, 8 November 2018. https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/fastcash-lazarus-atm-malware?om_ext_cid=biz_social_NAM_facebook_Asset%20Type%20%20-%20Blog,Threat%20Intelligence

[xviii] https://www.virustotal.com/gui/file/a38c1e24eaf34c944c11d9968427c74b3412a2c1e82e31551cabd7d3e213bf31/details

[xix] "Creation of Multi-monitor Screenshots Using WinAPI" Code Project, 12 August 2010. https://www.codeproject.com/Articles/101272/Creation-of-Multi-monitor-Screenshots-Using-WinAPI

[xx] "XZip Demo" Code Project. https://www.codeproject.com/KB/cpp/xzipunzip/XZip_demo.zip

[xxi] "CẢNH BÁO MÃ ĐỘC APT TẤN CÔNG CÓ CHỦ ĐÍCH" VNCert, 31 January 2019. http://www.vncert.gov.vn/baiviet.php?id=109

[xxii] "New Lazarus DPRK installer" Twitter, 24 January 2019. https://twitter.com/KevinPerlow/status/1088501115245727744