Secure Coding Guideline en BASE24

OCELOT

METABASE Q

Index

Introduction

Today's payment systems are facing an ever-changing reality. New forms of payment, new channels for making purchases and new validations of the cardholder when making transactions, are forcing authorizing systems (POS, ATM, e-commerce, e-wallet, etc.) to be stricter with the validations and rules that must be carried out when processing a transaction, as well as having perfectly defined parameters that allow or prohibit a transaction to be authorized.

During this process, unexpected results can lead to catastrophic consequences for financial institutions. A simple confusion between the rules that apply to the validation of a magnetic stripe card against a contactless card, or a confusion in the authorizing party (switch or bank), can be enough to generate unexpected authorizations that cause the loss of millions of dollars for banks or banking switches.

A review of the best practices of secure coding requires a review of logical errors with a meticulous inspection through each step of the code during the validation and routing of a transaction. This requires a team with experience in this type of processes, knowledge of both the TAL language and payment systems (BASE24, Connex), and, of course, experience in identifying vulnerabilities in environments that allow proactive identification of attacks, as well as short- and long-term correction of security holes.

This guide is intended to set a precedent and lay the groundwork for establishing better programming practices in writing legacy code, specifically in TAL language, and especially that which runs on Electronic Funds Transfer (EFT) systems such as BASE24 and Connex. The secure coding errors presented in this guide are the result of a thorough analysis performed on BASE24 system, which has been modified by third party vendors other than the software owner (ACI Worldwide). On many occasions the system is modified by different vendors over the years without following any specific standard, guided only by personal experience, practicality or speed, and sometimes without acknowledging the rules of the TAL language or of transactionality established by the BASE24 system itself.

The errors shown in this guide are classified by categories depending on the type of impact they may have and their severity. The recommendations to follow in order to avoid this type of errors, as well as diagrams to better understanding the modular scheme used by BASE24 and the modules in which the vulnerability may occur are included. A diagram is also presented to provide the user with a better understanding of the memory management within the code written in TAL.

Credits

Metabase Q team that participated in the creation of this guide, in alphabetical order:

Alejandro Gómez Bucio

Dana Paola Valle Arroyo

José Antonio Alcalá López

Bad Memory

Errors that can cause memory corruption, leading to rejections, abends or malfunctioning of processes.

Bad Memory	Bugs	Risk and Impact
1.0	1.1 Bad Index	Errors where not all the occurrences of a table are considered, causing some elements to be discarded or invading the limits of others. In the first case, if there is an omission of any occurrence, the transaction in process can be affected, resulting in a rejection and affecting **availability.** In the case of readings that exceed the number of occurrences loaded in the table, the problem is that if during the reading the data being sought is not found, the reading will continue until an "out of bounds" error occurs. In other words, information that is available at the next memory address may be brought in, which may cause data corruption of other variables, causing a rejection or abend of the process and affecting its **availability.**
	1.2 Bad Token	Result validation errors in the operations performed with the tokens may cause rejections, memory corruption or process crashes, affecting the **availability** of the service and the **integrity** of the information.
	1.3 Bad Variables	Errors related to mishandling of variables: data corruption or validating the errors returned by the routines and still continuing with the flow of the transaction, which may cause rejections due to invalid data and affect the **availability** of the service.

Bad Index Recommendations

- In the case of readings that exceed the number of occurrences loaded in the table, the problem is that if during the reading the data being sought is not found, the reading will continue until an "out of bounds" error occurs, causing an abend in the RTAU. The LCONF parameters indicated in the analysis should also be reviewed to validate how much real data may be omitted from the program logic.

- Another good practice is to use an index field within the table that will be incrementing at the time of loading into the memory. This will help us have an overview of how many occurrences we have loaded in memory, regardless of whether the table has been defined larger for future use or in anticipation of the increase of information.

Bad Token Recommendations

- Given the existing handling of tokens implemented by ACI, it is recommended to adhere to these standards every time tokens are manipulated, validating the type of error and acting accordingly; avoiding mishandling of memory pointers, log messages in case of failure and finally, making sure that the information sent to the interfaces is correct in order to avoid rejections that are difficult to diagnose.

Bad Variables Recommendations

- The correct way and the **best practice** to make variable movements to avoid data inconsistencies, loss of important information for the transaction, or buffer overflow is to **always use** the TAL **$len** utility, which takes the exact length of the target field and moves at most to the target field. On the other hand, it is important to initialize (clean) before and after the buffers in memory.

Bad File

Errors in file operations that can lead to transaction malfunctions, file corruption and process crashes.

Bad File	Bugs	Risk & Impact
1.0	**1.1 Closing file**	Errors related to closing files after manipulation or after a failure occurs, causing unused copies to remain open. Within BASE24 there is a table in charge of adding or deleting the files used by the modules. The number of files that this file table can handle is determined by the Processing File Segment (PFS). These files are recognized by a file number (number by which a file is identified within the tandem processes). If a file is not explicitly closed, it will continue to use a previously designated space within the table, eventually causing memory exhaustion. As there are multiple copies of the same file that was not closed, it will reach a point where the table fills up and the process crashes. This can lead to **a denial of service**.
	1.2 LCONF read	Poor reading of the parameters or assignments in the LCONF file can cause incomplete or erroneous information to be brought in, affecting transactionality, causing **rejections** or strange behavior during the transaction process. The problem with this type of reading, which has become a standard in RPQ/CSMs performed by external providers within the code, is that it does not follow the best coding practices in TAL nor those established by ACI for handling this file. The risk lies in the fact that when the same parameters are added up to the position being read and then change value, it will not be known with certainty which parameter was actually read, which can lead to **data inconsistencies** or even **denial of service** due to **rejections**. Since the parameters are only loaded when starting the process or executing a Warmboot, the information brought incorrectly will be present all the time.
	1.3 Keypostion	It doesn't validate if the pointer was positioned correctly for reading in a record, which could cause **abends** in the programs, affecting the **availability of the service**. If the keyposition is performed without validating if there was an error in the routine or if the error is different from EOF (End of File), it would mean that something happened that needs to be taken into consideration with the file. Therefore, there would be a problem at the time of performing the read that is not being validated. The severity and the considerations to be taken will also depend on the importance of the information contained in the file. For more information go to the section **General information on file management related to the bad file family**.
	1.4 Read, Write, Open	The result of these operations is not validated in files, assuming that everything happens as expected, which can lead to abends, corruption or blocking of records, affecting the **availability of the service**.

Closing File Recommendations

- The recommendation is to close each file previously opened, or, depending on the case, to add a Warmboot routine for the file. This way, every time a Warmboot is executed the file in the file table will be restarted as well.

LCONF Read Recommendations

- It is recommended to adjust the reading of the parameters and to be consistent with the standard managed by ACI.

Keyposition Recommendations

- When doing a Keyposition it is recommended to validate if there was any error. Depending on the error number, if something different from EOF is detected, then it is recommended to make a file info to know the details of the error. In any other case, to put a logic to handle it and notify the failure in the EMS log.

Read, Write, Open Recommendations

- It is recommended that whenever a call is made to the **open** routine, the result of the variable where the error of the operation is returned should be evaluated.
- It is recommended that whenever a call is made to the **write/writex** routine, it should be evaluated whether the record has been modified or written correctly.

PCI Violation

Written code lacking the best security management practices, which could impact the confidentiality of cardholder data that is being processed, which could be stored on disk without being masked or encrypted (based on the PCI DSS 4.0 standard).

PCI Violation	Bugs	Risk & Impact
1.0	**1.1 PCI Violation Security Control 3.2**	By sending the number in clear, sensitive data or other transaction information such as the sequence number, you are **violating PCI regulations**. This is sensitive data that if intercepted, could be misused for **fraudulent purposes**. In addition to the fact that the institution that has the BASE24 system could incur some type of sanction for this type of failure, the main issue is that the **confidentiality** of the client is at risk.

Recommendations

- The recommendation is to mask the card number or any other cardholders' sensitive information before saving it in a log or file on disk

Abends

Unexpected or uncontrolled interruptions in BASE24 processes (known as "abending") which affect availability, loss of transactions and business continuity.

Abends	Bugs	Risk & Impact
1.0	1.1 By Token	The **process is stopped** when an error occurs in some operation with the tokens inside the message. It is usually related to serious memory problems. The problem is that BASE24 processes should "abending" only if it is absolutely necessary, in order to avoid affecting the availability of the service.
	1.2 By memory corruption	Use of pointers that are not correctly initialized, that point to invalid memory addresses or that point to information different from what they are supposed to point to. This is due to the fact that the result of routines that perform operations with them is not validated and it is assumed that the operation was satisfactory, continuing with the transaction process that can cause **breaks** which are difficult to correct and thus affecting the **availability** of services and even the **continuity of the business**.
	1.3 By file upload	The **process is stopped** when a file that is **not essential** to the operation of the process cannot be loaded correctly into the memory.

Recommendations by Token

- BASE24 programs should “abend” in the minority of possible scenarios, in order to avoid affecting the service and/or causing transaction queuing. It is recommended to rely on the tools already available in the program.

Recommendations by memory corruption

- When reading tokens, it is recommended to validate if there was no fatal error. Then, it is possible to identify if there is a problem of memory corruption or bad formatting of the message. This will prevent the error from snowballing into a major problem. From rejections to the process being stopped due to memory corruption.

Recommendations by file upload

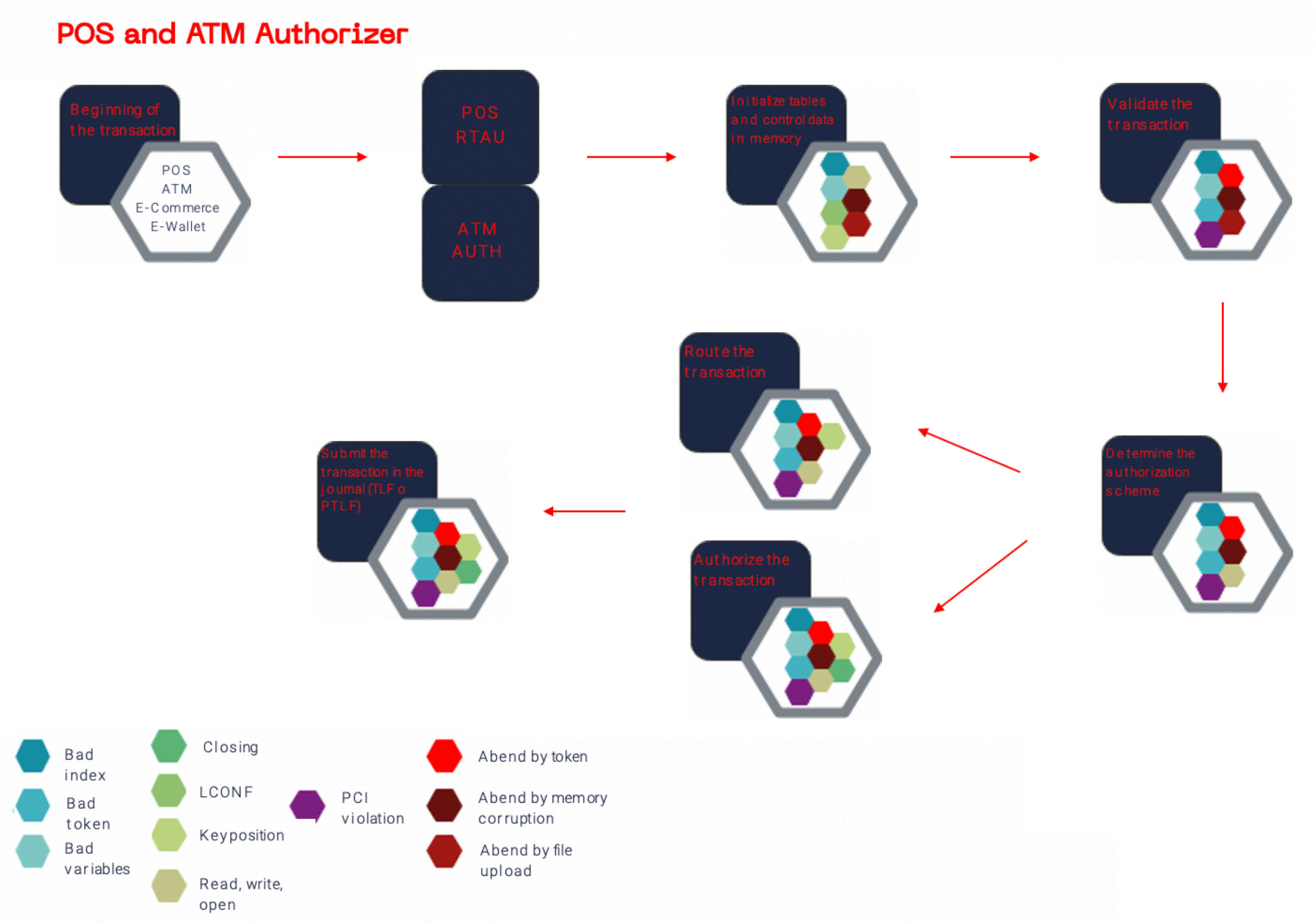
- It is recommended to avoid the abend so that the program does not close or stop running.

Now that we have described the different families of bugs that we validated during the review; it is very important to locate their impact within the different transactional flows, as shown in the following sections.

POS and ATM Authorizer

During any phase of the authorization process, several of the failures described in the following diagram may occur. From the initialization of the authorizing process, to the response of a transaction to the device handler. Within the authorizers is where the result of a failure can be **more critical** and where more attention to the code implemented by third parties is required.

Below there is a graphic that provides a general description of the RTAU and AUTH process and a brief context of the type of impact that may occur depending on the stage.


POS and ATM Authorizer
Beginning of the transaction
POS
ATM
E-Commerce
E-Wallet
POS RTAU
ATM AUTH
Initialize tables and control data in memory
Validate the transaction
Determine the authorization scheme
Route the transaction
Authorize the transaction
Submit the transaction in the journal (TLF o PTLF)
Bad index
Bad token
Bad variables
Closing
LCONF
Keyposition
Read, write, open
PCI violation
Abend by token
Abend by memory corruption
Abend by file upload

Initialize tables and control data in memory

- Incorrect data may be loaded or initialized in tables.
- May cause incorrect transaction processing based on incorrect data.

Validate the transaction

- An initial transaction rejection will occur, probably without knowledge of the reason for the rejection or failure.
- The authorizing process may crash as soon as files are initialized, or data is loaded into memory.

Determine authorization scheme

- The method by which the transaction is to be processed can be affected, which may result in the transaction being lost or not knowing to whom it should be routed, or even in unwarranted authorizations.
- Crashes may occur in the authorization process.

Route the transaction

- In the case of Level 1 and 2 authorization, incorrection information may fail or be sent to the host. It may even cause the rejection of the transaction or the authorization process to crash without the host knowing what happened.

Authorize the transaction based on account, institution card or track data

- In the case of Level 2 and 3 authorization, cardholder information can be compromised when handling sensitive data that is exposed in the EMS when the corresponding validations are performed. Can provoke unwarranted authorizations.

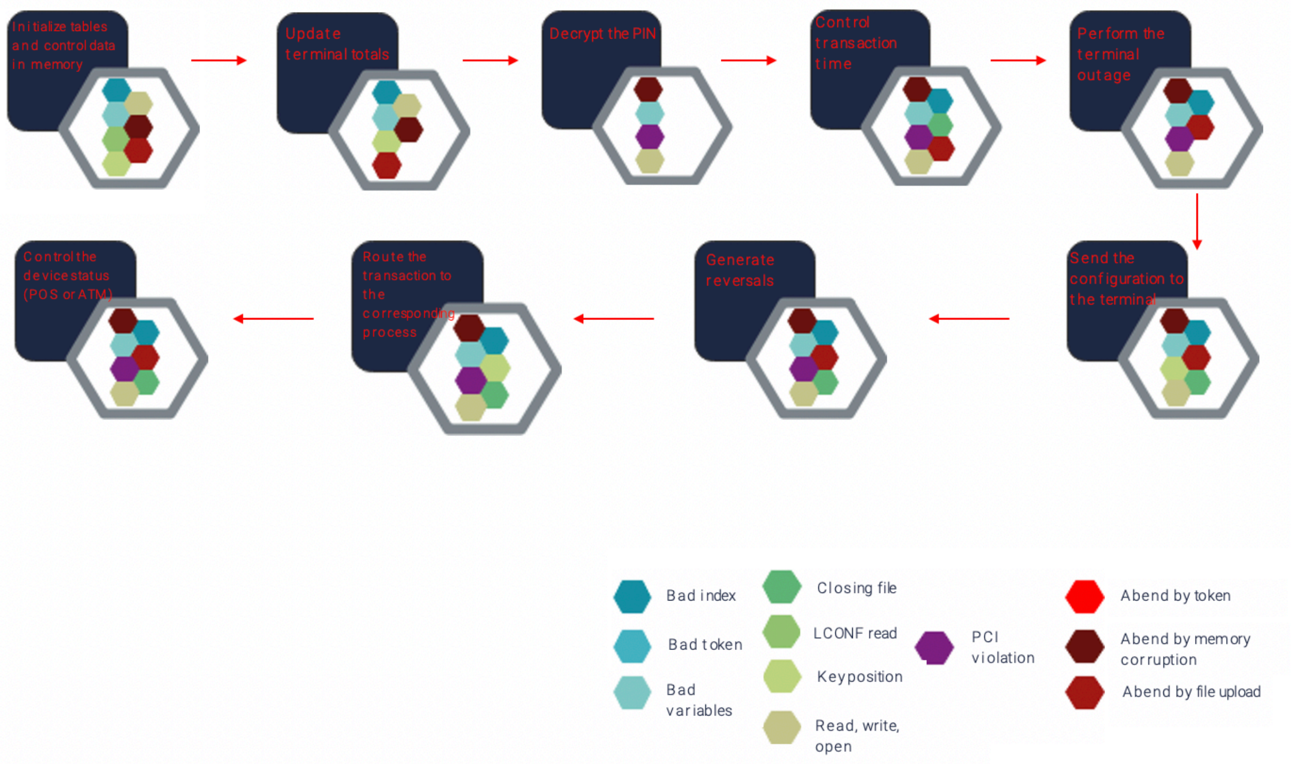
Submit the transaction in the journal (TLF o PTLF)

- Incorrect token information that should be submitted in the journal may be submitted.
- Information that is not consistent with the final result of the transaction may be submitted.
- Journals may be corrupted or blocked.
- Compensation may be affected.

Device Handlers

Device Handlers are the primary interface between the device (POS or ATM) and BASE24. Their main function is to convert the message formats between the device and BASE24. Within these processes, failures such as those described in the following diagram may occur during any phase of its functionality. From the moment the device receives the message, until the reply is sent back. It must be considered that one of the points of high criticality is that the terminal configuration files (ATM -> ATD or POS -> PTD) can be corrupted, causing serious problems in the business operation.

Device Handlers POS and ATM



Initialize tables and control data in memory

Update terminal totals

Decrypt the PIN

Control transaction time

Perform the terminal outage

Send the configuration to the terminal

Generate reversals

Route the transaction to the corresponding process

Control the device status (POS or ATM)

- Bad index
- Bad token
- Bad variables
- Closing file
- LCONF read
- Keyposition
- Read, write, open
- PCI violation
- Abend by token
- Abend by memory corruption
- Abend by file upload

The possible consequences of failure happening in these processes are discussed in more detail below.

Initialize tables and control data in memory

- Incorrect data may be loaded or initialized in the tables.
- May cause incorrect transaction processing based on incorrect data.

Update terminal totals

- The device's detailed counters may be updated with incorrect values affecting the accounting and possible limitations established in the services (maximum or minimum limits).

Decrypt the PIN

- Incorrect calculations of encrypted data may result in transactions being rejected due to incorrect PINs, causing poor customer service.

Control transaction time

- If these values are lost due to some kind of error, the control of the reverses may be lost, affecting the client and giving a bad image.

Perform the terminal outage

- If the outage fails or is performed incorrectly, it causes problems in transactions on the accounting date, resulting in rejections and poor customer service.

Control the device status

- By not having the status of each element of the devices (hardware and software elements) rightly controlled, it offers services that do not end correctly because the element does not have the correct status.

Route the transaction to the corresponding process

- Can cause transactions to be sent to incorrect or non-existent processes, resulting in poor service on the device and inconsistent transactions.

Generate reversals

- Due to the control with the device and its elements, if not operated correctly, it impacts a certain type of transactions that directly affect the client's account.
- Customer dissatisfaction.

Send the configuration to the terminal

- If there is not a correct control exchange with the device, it may be operating incorrectly and processing transactions erroneously.

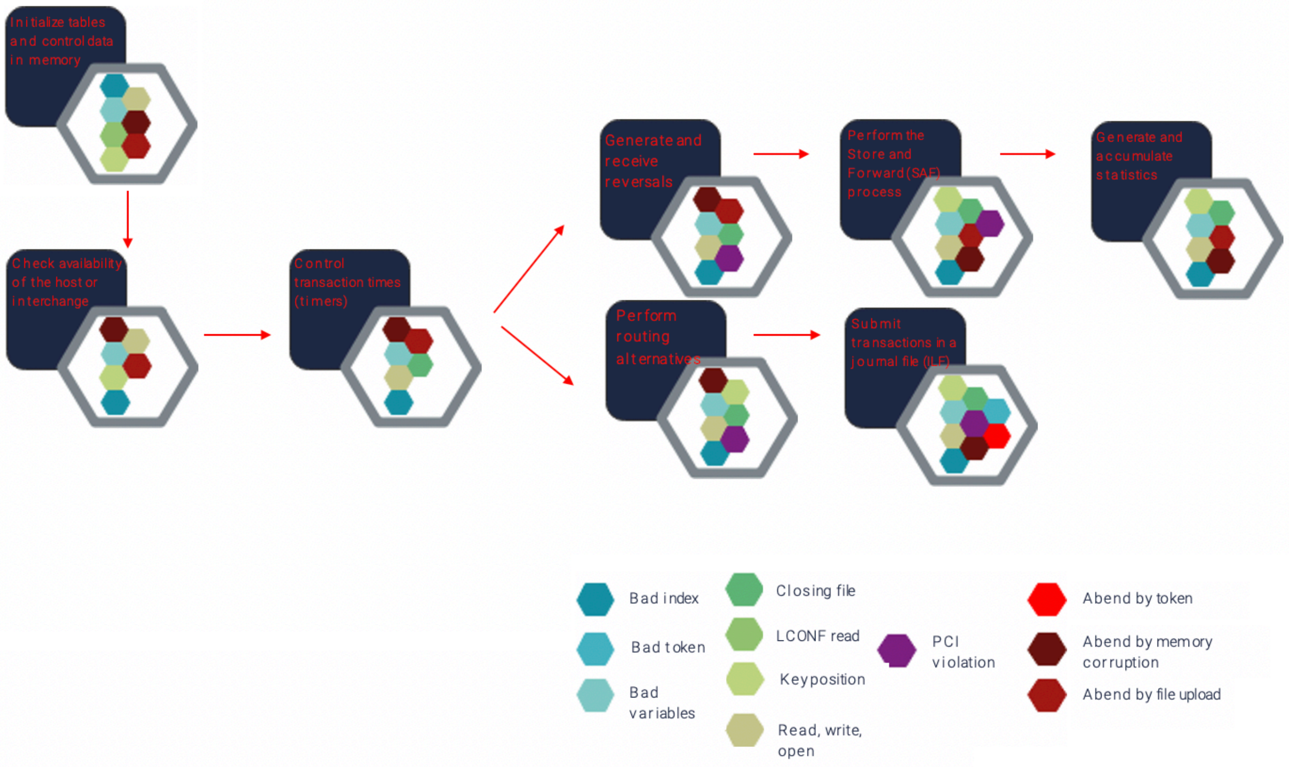
Host and Interchange Interface

Host interface (ANSI or ISO) is responsible for controlling the traffic of messages to and from the host, as well as converting message formats (internal and external) between hosts and BASE24.

Interchange interface is responsible for controlling the traffic of messages to and from the interchange, as well as converting message formats (internal and external) between interchanges and BASE24.

Within these processes, failures such as those described in the following diagram may occur during any phase of its functionality. From the moment BASE24 receives the message, until the reply is sent back. In this case, the most critical affectation produced by a vulnerability in the process is that it can cause incorrect answers that translate into unexpected authorizations or rejection.

Host and Interchange Interface


Initialize tables and control data in memory
Check availability of the host or interchange
Control transaction times (timers)
Generate and receive reversals
Perform routing alternatives
Perform the Store and Forward (SAF) process
Submit transactions in a journal file (ILF)
Generate and accumulate statistics
Bad index
Bad token
Bad variables
Closing file
LCONF read
Keyposition
Read, write, open
PCI violation
Abend by token
Abend by memory corruption
Abend by file upload

The possible consequences of failure happening in these processes are discussed in more detail below.

Initialize tables and control data in memory

- Incorrect data may be loaded or initialized in the tables.
- May cause incorrect transaction processing based on incorrect data.

Check availability of the host or interchange

- Not having control may cause the loss of transactions or give incorrect answers to the service instead of taking possible schemes altern to authorization.

Control transaction times (timers)

- It may cause unrequired reversals by losing control of limited transaction times or responses as a poor service.

Generate and receive reversals

- Due to control with the host, if not operated correctly, it impacts certain types of transactions that directly affect the client's account.
- Customer dissatisfaction.

Perform the Store and Forward (SAF) process

- This is necessary in order to have balance with the host. Failure of this control will result in data loss, seeking to achieve information balance with the host.

Generate and accumulate statistics

- The behavioral information with the host that allows monitoring that everything is efficiently processed is affected.

Perform routing alternatives

- Incorrect processing of possible transaction solution alternatives may be applied, resulting in poor service.

Submit transactions in a journal file (ILF)

- If there is not a correct submission in the journal, it directly impacts the compensation with the host.

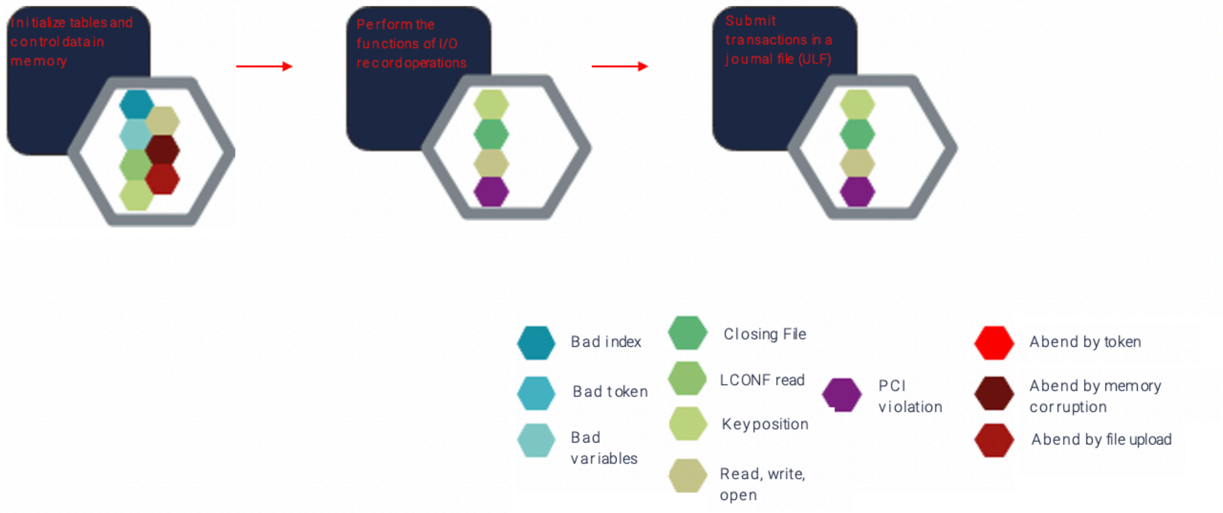
From Host Maintenance

From Host Maintenance (FHM) is responsible for enabling users to perform online updates to BASE24 authorization files based on host files or tables.

The update operations that are performed with the BASE24 database files are: query, add, delete and modify records. These file operations are performed through ISO messaging with the host and with support of the host interface module. The main files in the BASE24 database that operate with this scheme are authorization files: CAF (Cardholder file), PBF (positive Balance File) and NEG (Negative File).

Within these processes, failures such as those described above may also occur during any phase of its functionality. From the moment the message is received with the request for maintenance of the database in BASE24, until it is confirmed with the reply. In this case, the most critical affectation caused by a vulnerability in the process is that inconsistencies can be generated in the BASE24 database when performing incorrect update operations

From Host Maintenance



initialize tables and control data in memory

Perform the functions of I/O record operations

Submit transactions in a journal file (ULF)

Bad index

Bad token

Bad variables

Closing File

LCONF read

Keyposition

Read, write, open

PCI violation

Abend by token

Abend by memory corruption

Abend by file upload

The consequences of a failure in these processes are detailed below.

Initialize tables and control data in memory

- Incorrect data may be loaded or initialized in tables.
- May cause incorrect processing of database updates based on incorrect data and thus impacting the authorization process of financial transactions.

Perform the functions of I/O record operations

- Incorrect query updates, additions, deletions and updates may be performed, causing corruption of the database and resulting in loss of integrity.

Submit the transactions in a journal file (ULF)

- An incorrect record in the journals directly impacts the integrity and change control in the database.

Best practices for memory management

Direct addressing

Although direct addressing is limited to the amount of memory it can reference, it is more efficient and safer than indirect addressing. Therefore, direct addressing of memory should be used whenever possible.

For example, suppose a routine expects the use of a reference parameter in several calculations within the routine before returning the result to the caller. A good practice would be to move the parameter that is indirectly addressed to a variable in a local memory and then use that copy to perform the calculations. This avoids performing constant operations with memory, reducing the risk of corruption (bad variable or abend due to memory corruption). This allows to return the final result, already validated, to the original parameter. Although some overhead may occur in the process of copying from a parameter to a local variable and vice versa, the management is safer because the references used by this variable use direct addressing.

Indirect addressing

Considers arrays, structures and pointers. The advantage of using indirect addressing in the case of arrays and structures, is that the compiler generates a pointer, reserves the necessary space and initializes the pointer at the beginning of the structure or array. This eliminates the need to manipulate large amounts of data within the local memory; however, the manipulation of indirect array or structure information must be handled with care, since in case of error or information corruption, it will go out of the scope of the routine and may affect other parts of the program and cause availability errors (abend due to memory corruption). To use declared pointers, it is necessary to initialize and manage the space that will be used by the data to which the pointer will be pointing.

Extended indirect addressing

This type of addressing is used when the program requires additional information storage beyond what is already available in the user data segment. In the case of memory management for this type, the complier takes care of automatically allocating and evicting the necessary size in the extended segment. Therefore, the security and memory management considerations for this type of addressing **are the same as those for indirect addressing**. Also, the operations performed in the extended segment are not as fast as those used in the user segment. **This type of addressing should be used only when absolutely necessary.**

Best practices for file management

The following information is intended to give a general context on the use of files by the system and programs, as well as to provide considerations regarding the memory usage and possible consequences resulting from **a bad file** being found within a program.

The maximum number of files that can be opened at any given time depends on the space available for the control blocks:

- Access Control Blocks (ACBs)
- File Control Blocks (FCBs)
- Open Control Blocks (OCBs)

The space available for control blocks is mainly determined by the physical memory of the system. The maximum space for **ACB**s is determined by the size of the Processing File Segment (**PFS**). The available size of the PFS is **32 MB** in H-, J- and L- series.

Taking into account all of the above, we must analyze the possible effects that bug **bad file** can produce within the code. For example, if multiple **opens** are found within the same file, a **deadlock** may be produced. This means that the file could be blocked using the file number associated to the next **open**. Even if there are three or more opens, and if a **write** or **update** operation is executed in the file, it could associate the operation with any of the other opens, which could block any of the files, preventing access to the rest of the files or other file numbers and affecting the availability of the process. Hence the importance of always validating the status of operations that are executed within the files.

Another aspect to consider is that multiple **opens** in one or different files without a **close** can affect other external processes that make requests to these processes, as in the case of **replication tools** or **"tokenization" of sensitive data** in journals, preventing it from performing its task or causing **time-outs** or **crashes of the processes**.

Who are we?

Metabase Q

Metabase Q is a cybersecurity managed services company focused on securing Latin American organizations from cyberattacks. We offer custom-designed cybersecurity solutions and services designed to protect companies of various industries and sizes. Our end-to-end capabilities span CISO-as-a-Service consulting, technological innovation and integration, and offensive security services.

Ocelot

Ocelot, by Metabase Q, is the leading Offensive Security team in Latin America. We've centralized the world's top talent in advanced persistent threat simulations, PoS and NFC payment systems, BASE24, ATMs, IoT, and ICS to ensure we can serve as an ally to you in your most complicated problems. Ocelot threat intelligence, research, and offensive skills power Metabase Q's cybersecurity managed solutions.

Join us, join the revolution

contact@metabaseq.com
+52 55 2211 0920