The Extraction of Cryptocurrency Transaction Information in Memory : CryptoScan

Team : BTC(BoB Tracer of Coin)

Role	Name
Document	**MinTaek Lim**
Development	**JeongYoon Kang, MoonGyu Lee**
Research	**HyeonDeok Jeong, JunSung Park**

Github: https://github.com/BoB10th-BTC/CryptoScan/blob/master/cryptoscan.py

Memory dump files: https://drive.google.com/drive/folders/12DS8Ua0IwXC57-NDCg-ZApUJqx_BgQUS?usp=sharing

ABSTRACT

Cryptocurrency is mined according to a cryptographic algorithm in a P2P network, and network participants, including miners, sometimes cash out through exchanges. Due to the peculiarity of cryptocurrencies that guarantee anonymity, they are often used for hiding and laundering criminal funds, and in particular, the usage rate of hardware wallets that can directly occupy and own cryptocurrencies is increasing. However, there are no forensic tools to analyze and track the use of these hardware wallets, it is difficult to investigate cryptocurrency. Therefore, in this paper, we develop and propose a forensic tool that analyzes, and extracts artifacts related to cryptocurrency transactions recorded on disk and memory for tracking and analysis of hardware wallet usage. Analyzing the extracted cryptocurrency transaction behavior will be a meaningful digital forensic data, and these hardware wallet artifacts can be utilized for investigations related to cryptocurrency tracking.

Ⅰ. Background of Tool Development

Bitcoin and Ethereum are virtual cryptocurrencies using blockchain technology. Cryptocurrency operates as a P2P (Peer-to-Peer) network and has the characteristic that transactions between individuals are possible without the control and management of a central institution. Cryptocurrency can be owned either by performing cryptocurrency mining directly or by purchasing it through a cryptocurrency exchange. Korea has an environment where mass mining is impossible, so most people purchase cryptocurrencies through exchanges.

Cryptocurrency technology attracted people's attention because it guarantees anonymity, but in Korea, cryptocurrency-related corporation 'Act on the Reporting and Use of Specific Financial Transaction Information' requires real-name authentication to purchase cryptocurrency through an exchange. If a criminal converts the cryptocurrency used for the crime into cash after real-name authentication on the exchange, the investigative agency can execute a search and seizure warrant on the exchange to track it. For this reason, the use of hardware wallets that provide security and anonymity in cryptocurrency transactions is increasing.

A hardware wallet is a USB-type cryptocurrency wallet and has its own wallet address. Information generated while trading cryptocurrency through a wallet address is called a transaction, and the transaction-related traces left on the PC can be divided into hardware wallet-related and cryptocurrency-related artifacts. Among them, artifacts related to hardware wallets include wallet addresses, mnemonic codes, PIN codes, and wallet connection traces.

In the case of cryptocurrencies stored on exchanges, users have just bonds. On the other hand, hardware wallets have the feature of being able to store their own cryptocurrencies in a USB wallet. To connect these hardware wallets to the network, hardware wallet manufacturers provide PC applications dedicated to hardware wallets for users, and users can trade cryptocurrency without using an exchange through the application.

In addition, hardware wallets and PC applications provide a Swap function to exchange one cryptocurrency for another. When this function is used, the sender sends cryptocurrency to the exchange in charge of the swap, and the swap takes place inside the exchange. Therefore, information generated during the exchange process other than the TXID that sent and received cryptocurrency does not remain on the user's PC, so it is difficult to track cryptocurrency without the cooperation of the exchange. Similarly, to use a cryptocurrency tracking solution such as Chainalysis's 'Reactor', you must know the TXID or wallet address in advance to be able to search. The information may not be available, and the solution may not be available.

Therefore, in this paper, when using a hardware wallet, the information of the hardware wallet recorded on the memory file is extracted to help track and investigate cryptocurrency. When using a hardware wallet, information such as TXID, wallet address, type of sending/receiving cryptocurrency, cryptocurrency balance, and Swap ID are recorded in the memory file. By extracting this information, we propose the development of efficient measures and tools for tracking cryptocurrency transactions.

For the study, two hardware wallets, Ledger Nano S (hereinafter referred to as Ledger) and Trezor One (hereinafter referred to as Trezor) were analyzed, and similarities were confirmed by additionally analyzing Exodus, a software wallet. Bitcoin (BTC), Ethereum (ETH), and Ripple (XRP) were selected as the cryptocurrencies sent and received to generate transactions. This is because it has the highest value when converted into cash and is often used to hide criminal funds.

Ⅱ. Background Knowledge and Forensic Value

2.1 Hardware Wallet

A hardware wallet is a means designed to protect personal assets when trading cryptocurrency more safely. Unlike software wallets or web exchanges, hardware wallets are not always connected to the Internet, so they must be used together by installing related applications on a personal PC. To connect with the PC application, a different application for each cryptocurrency must be installed in the hardware wallet to generate a wallet address, and the wallet address can be restored through reinstallation even if the cryptocurrency application in the hardware wallet is deleted until the hardware wallet is completely initialized.

When a hardware wallet is connected to a PC, a connection trace is left in the registry, and you can check the connection trace of the hardware wallet through analysis. Both Ledger and Trezor used for research could find connection traces in the HKLM₩SYSTEM₩ControlSet₩Enum₩USB registry path, and hardware wallet connection records could be found in the event log. These external device connection artifacts are information that is always left in a hardware wallet in the form of a USB.

Figure 1 Hardware wallet connection identified in registry and event log

2.2 Wallet Setting Information (Mnemonic Code and PIN code)

A mnemonic code is a word with the property of being easily remembered and is a set of 12 or 24 English words randomly assigned out of 2048 words when creating a personal wallet. Since a personal wallet can be fully cloned and restored through a mnemonic code, hardware wallet manufacturers require that it be written down on paper, etc. and stored separately in a place that only the owner of the wallet can access. If the user saves the mnemonic code in the PC as a file such as a document for convenience, it is also a way to duplicate the hardware wallet owned by the investigation target without PIN code authentication.

Among the hardware wallets, Trezor can be created by choosing between a regular wallet and a hidden wallet. A normal wallet can restore and clone a wallet by entering a mnemonic code, but a hidden wallet is not possible. Ledger did not provide a hidden wallet function, so it was possible to clone a hardware wallet by entering a mnemonic code.

A PIN code is a 4-10 digits number used as a means of authentication like a password. This is especially required when trading cryptocurrencies in a hardware wallet or changing related settings. If Ledger authenticates an incorrect PIN code multiple times, the use of a hardware wallet is impossible. In this case, you need to enter a mnemonic code to recover or reset your wallet. Trezor protects the wallet by extending the re-enter time as the PIN code is entered incorrectly.

2.3 Cryptocurrency Trading Information

2.3.1 Wallet Address

A wallet address is a value for distinguishing a hardware or software wallet for a specific cryptocurrency. A cryptocurrency wallet address can be used by generating a private key and public key pair. In addition, the wallet can create transactions. A hardware wallet has a unique wallet address for each device, and you need to install a different application for each type of cryptocurrency in the wallet to create a wallet address. Since USB type hardware wallets have different memory capacities for each manufacturer, Ledger can create 2-3 cryptocurrency wallet addresses, and Trezor can create 3-4 wallet addresses. The wallet address of cryptocurrency has a different form for each type. Bitcoin is an address generated by Bech32 encoding. This address type has a characteristic that starts with 'bc1', Ripple starts with 'r', and Ethereum starts with '0x'. Therefore, you can use a different regular expression for each address to identify the wallet address.

Cryptocurrency	Wallet address formet	Regular expression
Bitcoin	bc1q9tqzryx7vmv30c0xkfa7vlve4t60js0299qg50	₩b(bc(0([ac-hj-np-z02-9]{39}|[ac-hj-np-z02-9]{59})|1[ac-hj-np-z02-9]{8,87})|[13][a-km-zA-HJ-NP-Z1-9]{25,35})₩b
Ripple	raQwCVAJVqjrVm1Nj5SFRcX8i22BhdC9WA	r[0-9a-zA-Z]{33,35}
Ethereum	0x8496A6b1805346c3daF69790B898ADf72906Aaf5	0x[a-fA-F0-9]{40}

Table 1 Wallet address formet and regular expression

2.3.2 Cryptocurrency Transaction

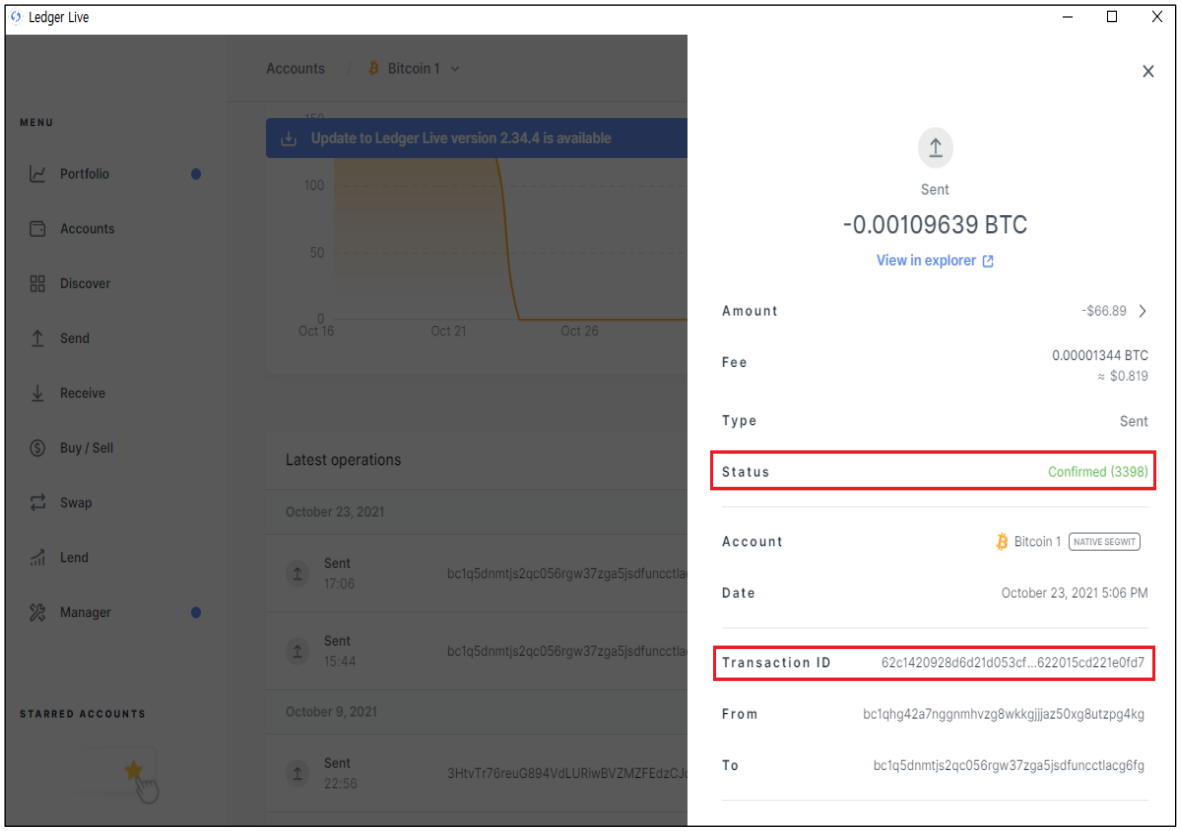

Figure 2 Transaction history recorded on Ledger Live

A transaction usually refers to a unit of work that changes the state of a database, and a transaction in cryptocurrency is defined as signature information written for the purpose of transferring ownership owned by one individual to another. Currently, TXID is the identifier of each transaction, and consists

of a combination of 64 lowercase letters and numbers. Through TXID inquiry, it is possible to check whether the transaction has been successfully executed.

After recording the amount of cryptocurrency for remittance, TXID, and the recipient's wallet address, you can generate a transaction by applying for remittance. Cryptocurrency transaction information is input to a block chain-based block at regular intervals and recorded on the network. After that, when the blockchain network confirms the transaction and connects to the previous blockchain block, the cryptocurrency transaction is completed through the approval process. In Ledger live (a dedicated PC application for Ledger Nano S), when a transaction is completed, the status is changed to 'confirmed' as shown in [Figure 2].

name	Byte	description
Version	4	transaction version
input count	compression size	number of inputs
previous output	32	TXID of previous output
output index	4	The number of the previous output that will use the balance
Script length	compression size	script length
Script sig	Var	Signature script digital signature
Sequence	4	sequence number
output count	compression size	number of outputs
value	8	amount to send
Script length	compression size	script length
Script Pubkey	Var	Recipient's public key hash value
Lock time	4	less than 500million block height or Unix timestamp

Table 2 Transaction details recorded in the block

When a Bitcoin transaction occurs, detailed data as shown in [Table 2] is recorded in the block. The number indicated in each field is a fixed size, Var is a variable size, and the compressed size is determined by the range of values. In the JSON format, TXID, sending/receiving wallet address, hash value, key information, etc. are recorded, so TXID and wallet address can be analyzed in memory through regular expressions. The details of these transactions are different for each cryptocurrency, but since most of the contents contain the same type of data, you can search for a JSON-formatted string when extracting.

2.3.3 Swap

Swap is a function that allows you to exchange a specific cryptocurrency for another cryptocurrency at a cryptocurrency exchange contracted with a hardware manufacturer. The swap process ends when the user sends the desired cryptocurrency to the wallet address of the swap exchange, exchanges the cryptocurrency for the amount inside the exchange, and sends it to the user's desired wallet address.

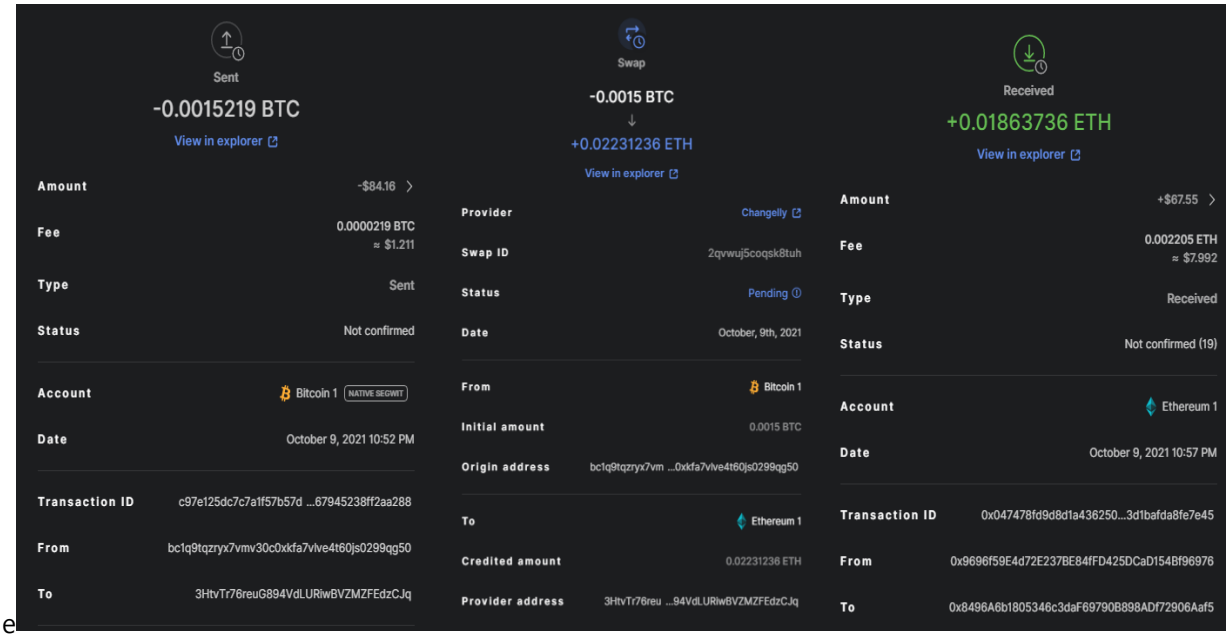

Figure 3 send, swap, received transaction history

Swaps can make it difficult to trace the origin, like a mixer, in the process of exchanging cryptocurrencies. [Figure 3] shows the swap transaction recorded on Ledger Live. From the left of [Figure 3], there are transaction records, swap records, and reception records sent to the swap exchange. When sending and receiving both swap exchanges and transmissions in a single hardware wallet, a lot of information such as the relevant TXID, wallet address, and cryptocurrency type was recorded in the memory, enabling analysis of all processes.

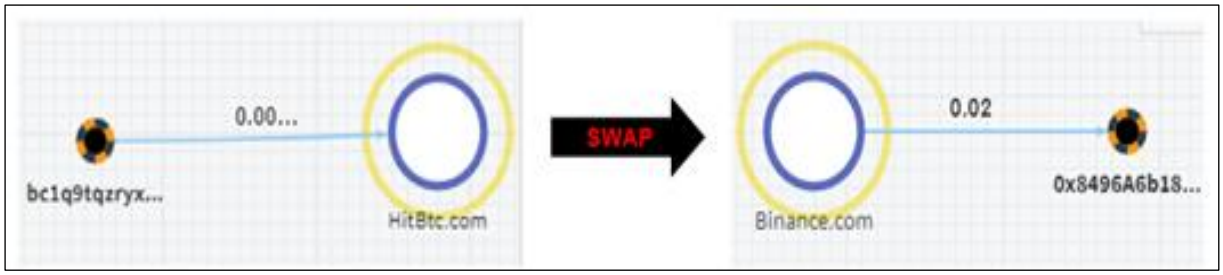

Figure 4 Swap process confirmed by Reactor (BTC -> ETH)

However, as a result of the Reactor search, as shown in [Figure 4], the exchange that sent Bitcoin to the HitBtc exchange to swap it with Ethereum and sent the Ethereum back after the swap was completed was Binance. After sending to the exchange for swap, it was confirmed that tracking using only the solution was difficult. Therefore, it is necessary to compare and analyze the transaction information remaining in memory with the solution results such as Reactor.

Ⅲ. Data Collection and Analysis Process

A memory file is a dump file for performing memory forensics, a series of processes for analyzing physical RAM using various forensic software. In this paper, 'Belka Live RAM Capturer', a live memory capture tool of Belkasoft that operates at the kernel level, was used to create a memory file. Since this research and tool development environment is Windows 10 Education 19043.1237 version, the memory file in this paper means a memory dump file of the Windows 10 19043.1237 version environment using the 'Belka Live RAM Capturer' tool. Cryptocurrencies that generated transactions for analysis are Bitcoin (BTC), Ethereum (ETH), and Ripple (XRP).

Category	PC Info
OS Version	Windows 10 Pro Education
OS Build	19043.1237
RAM Capacity	4.00GB
SYSTEM	64 Bit

Table 3 Analysis environment

Program Name	Program Version
Belka Live RAM Capturer	1.0
Volatility 3	3-1.0.1
HXD	2.5.0.0
Ledger LIVE	2.32.2
Trezor Suite	21.10.2

Table 4 Analysis Tool

TXID	Send	Received	Amount
c97e125dc7c7a1f57b57d0e71150c01deddb3789da83195a67945238ff2aa288	bc1q9tqzryx7vmv30c0xkfa7vlve4t60js0299qg50	bc1qdyvnm7ayrvxu5x4ljefpa6yu7xeq69u9ld7zmh	0.00363581 BTC
9d0fb1247c31fd7d154f030a77458e68034e30d81b5c3609f7b122f0af6f8b10	351yZhnPPbV8Y9C4NBaK1w4GSoRkUzUaU7	bc1qus52ul03xll06yxdyjy5q9gdgdq20jvrgu0zze	20.96877012 BTC
53fa46959eedf0cbe15672b6f19e0f5f76e0e13eb8fd3da410eb038e39485ec7	bc1qus52ul03xll06yxdyjy5q9gdgdq20jvrgu0zze	bc1q9tqzryx7vmv30c0xkfa7vlve4t60js0299qg50	0.00365373BTC

Table 5 Bitcoin transactions

TXID	Send	Received	Amount
17c2819183151c3f9150893cbcfc20f2a8bffd46638122d303ca1523a6fa4eae	0x8496a6b1805346c3daf69790b898adf72906aaf5	0x25672b04b810c67112bcf81ce152be35588a26d8	0.01684186 ETH

Table 6 Ethereum(ETH) transactions

TXID	Send	Received	Amount
31A88C6685422785FF6C7CB2A768AEA918D2E9D6BFA9218E438B64E0A1D78A3	rUNzcGi4eZUmEcprhmAAKto4fTJLsNQBEb	raQwCVAJVqjrVm1Nj5SFRcX8i22BhdC9WA	10.00001 XRP
5A86F9D6820264B34F8801FA36C6C45DC72FFBEF02FBFA2EDAA9C33FC10B2AF0	rshRbDTDVUA38vQxax9T7jBC1Bb3H7xQTR	rUNzcGi4eZUmEcprhmAAKto4fTJLsNQBEb	20.66 XRP
ECFA57394ADF5570F836BDFFA47385324BA66FF8BED3EB94D2035F18D7524B33	rUNzcGi4eZUmEcprhmAAKto4fTJLsNQBEb	rshRbDTDVUA38vQxax9T7jBC1Bb3H7xQTR	30 XRP
1F67F10BCB4396D8B905A0F4936E8166F34CECFF4975C3FC290956035C48FC98	rHuULof8mk1m7wffrmsBAVB3g6yAHivbmQ	rUNzcGi4eZUmEcprhmAAKto4fTJLsNQBEb	40.67 XRP

Table 7 Ripple (XRP) transaction

3.1 Transection ID Extraction

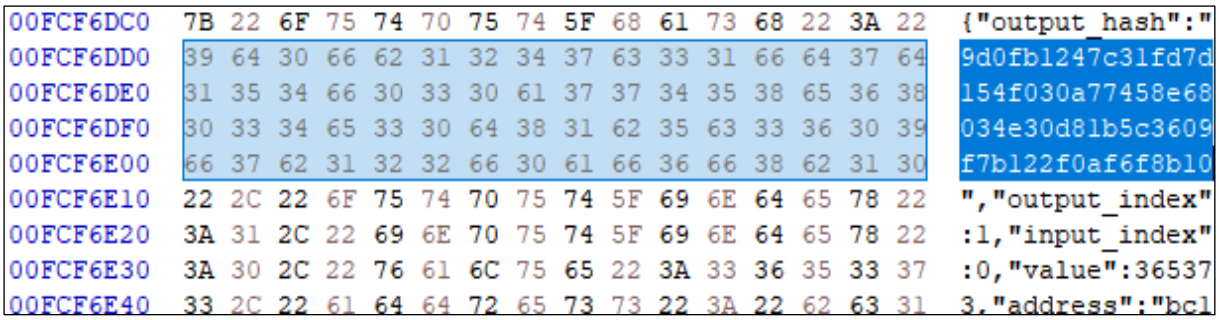

Figure 5 JSON's TXID extracted from memory

Figure 6 The full contents of the extracted JSON

Inputs	Result
output_hash	9d0fb1247c31fd7d154f030a77458e68034e30d81b5c3609f7b122f0af6f8b10
value	365373
address	bc1qus52ul03xll06yxdyjy5q9gdgdq20jvrgu0zze
Outputs	Result
address	bc1q9tqzryx7vmv30c0xkfa7vlve4t60js0299qg50
value	363581

Table 8 Contents of extracted JSON

After generating a transaction in the Ledger hardware wallet with the 'Belka Live RAM Capturer' tool, looking at the memory capture result, it was confirmed that the transaction record remains in the memory in JSON format as shown in [Figure 5]. The contents of JSON included 'id', 'hash', 'received', 'fees', 'input data', 'output data', and 'block', and 'input data' and 'output data' contain the data of the sending and receiving wallet, respectively. Detailed data were recorded in the form shown in [Table 2], and the extracted data was as shown in [Table 5]. Based on the entire JSON content in [Figure 6], we

verified whether the data exists by using the Reactor solution of Chainalysis.

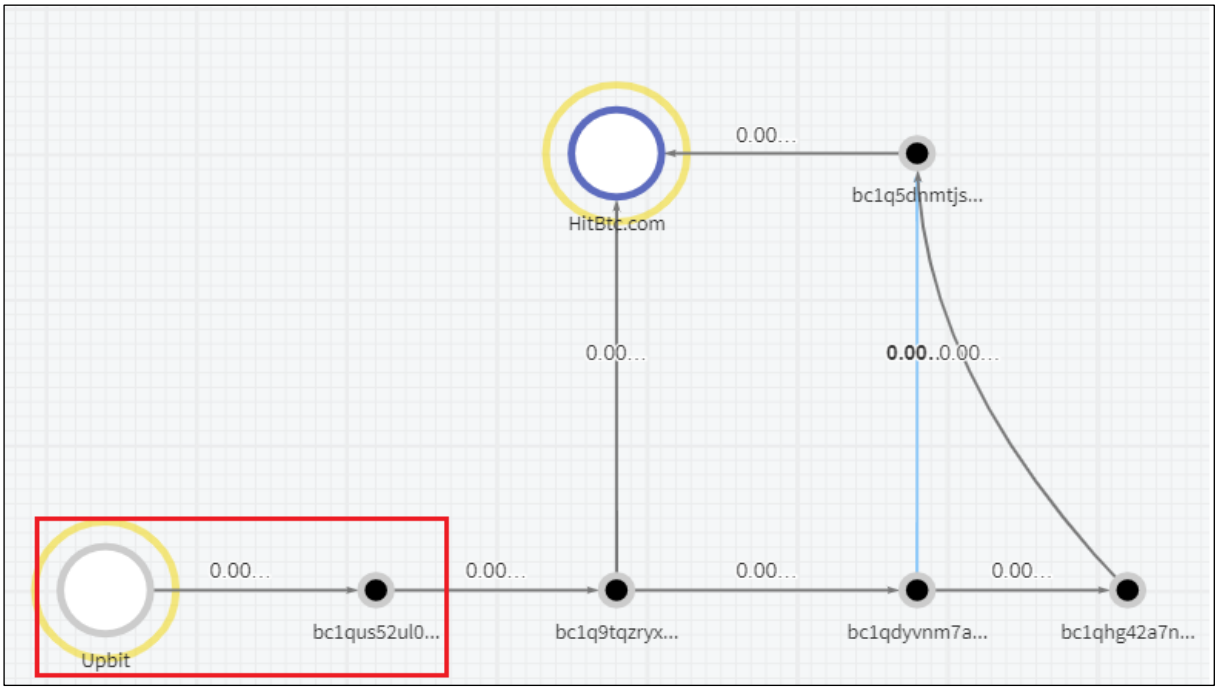

Figure 7 Reactor search results

In [Figure 7], you can check the sending/receiving bitcoin address as information about the transaction. In addition, it was possible to check all the exchange addresses used for the swap, including the amount of cryptocurrency sent. Therefore, if the transaction information recorded in the memory can be extracted, the flow of cryptocurrency traded from the extracted wallet address can be known. Like Ledger, Trezor also confirmed that transaction records remain in JSON format, and additionally analyzed software wallet Exodus also confirmed that it remains in memory in JSON format.

3.2 Wallet Address Extraction

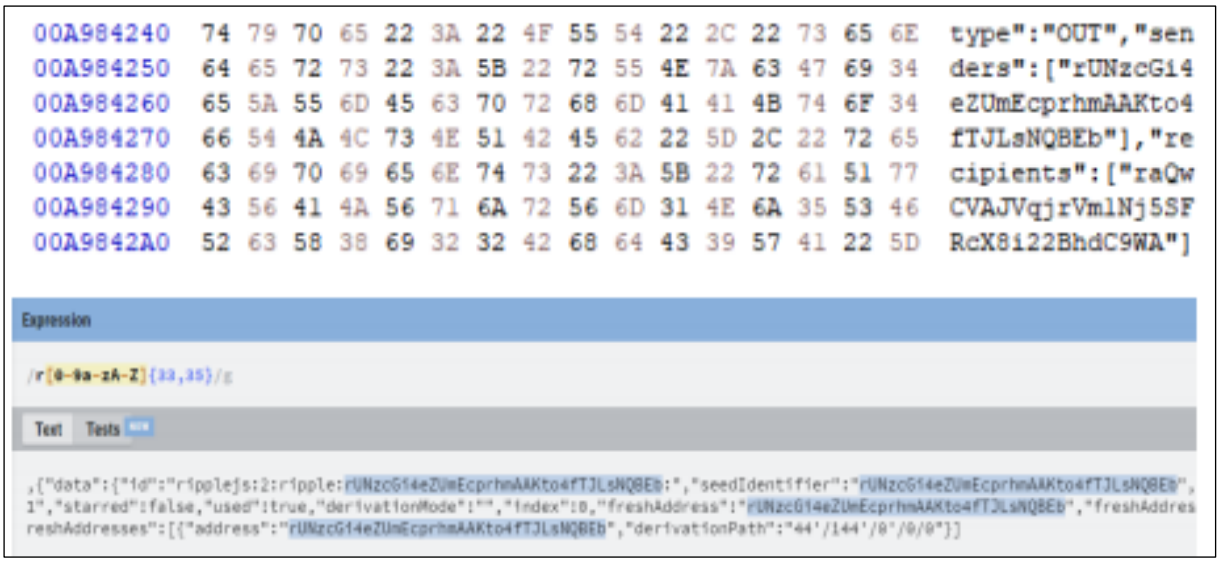

Figure 8 Wallet address extracted from memory and regular expression

Category	Value
sender	rUNzcGi4eZUmEcprhmAAKto4fTJLsNQBEb
recipients(received)	raQwCVAJVqjrVm1Nj5SFRcX8i22BhdC9WA

Table 9 Extracted wallet address

As a result of moving to the physical location of the actual memory and verifying it, JSON-type data could be confirmed at the offset, and both the type and address of the sent cryptocurrency were confirmed. In the memory dump file, JSON-type data identified by the ripple address exists. If you check the contents of the JSON data, the sending address is rUNzcGi4eZUmEcprhmAAKto4fTJLsNQBEb and the receiving address is raQwCVAJVqjrVm1Nj5SFRcX8i22BhdC9WA. This address was extracted using the regular expression in [Table 1], and it was the same as the actual transaction record in [Table 7].

We confirmed that both Bitcoin and Ethereum other than Ripple were recorded in the memory in JSON format, and it was confirmed that the transaction data extraction result is the same as the actual transaction. Since this extraction result is plaintext data that is not encrypted, the extracted information can be identified and processed and utilized.

3.3 Mnemonic Code Extraction

If the wallet is restored through the mnemonic code, all 24 mnemonic codes of the wallet remain in the PC memory as shown in [Figure 9]. Once the mnemonic code is known, the wallet can be fully restored, but extracting the mnemonic code from the memory is possible in a limited situation when the hardware wallet user restores and initializes the wallet with the mnemonic code. Therefore, in case

there is no mnemonic code remaining in the memory, it cannot be excluded to find the mnemonic code that may be stored as a file in the PC.

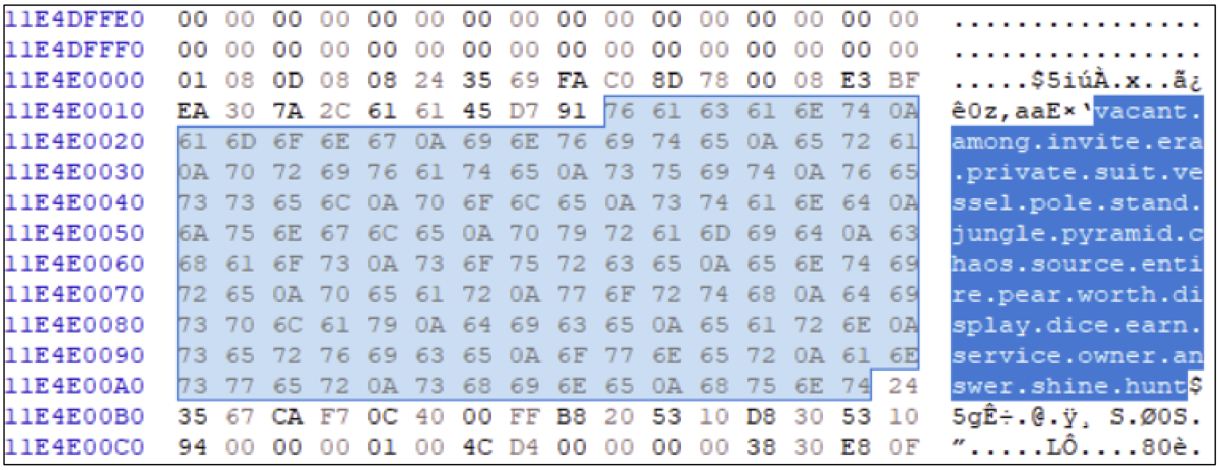

Figure 9 Ledger's mnemonic code extracted from memory

Category	Value
Mnemonic Code Regular Expression	exr = re.compile('[₩₩₩₩n]?[a-zA-Z]{3,8}₩₩₩₩n[a-zA-Z]{3,8}')

Table 10 regular expression of the mnemonic code

The mnemonic code extraction script written directly in this study searches for mnemonic codes using regular expressions targeting drive and memory dump files. Since two or more texts in the form of a mnemonic code can be extracted, the mnemonic code is extracted in a format that outputs the list with the highest probability of being a mnemonic code after evaluating the accuracy using the characteristics of the randomly generated mnemonic code. The extracted mnemonic code is compared with the published list of 2048 mnemonic codes to output the result.

Figure 10 Extract the mnemonic code from memory

Like extracting the mnemonic code from the memory, if the user saves the mnemonic code in the form of a txt file in the PC, it is possible to extract the mnemonic code based on "₩n" using disk forensics. In this study, the mnemonic code was saved in text format in "mnemonic code.txt" and extracted through the prepared script. Since the extracted mnemonic code can recover/replicate the hardware wallet, it can be usefully used in cryptocurrency-related investigations.

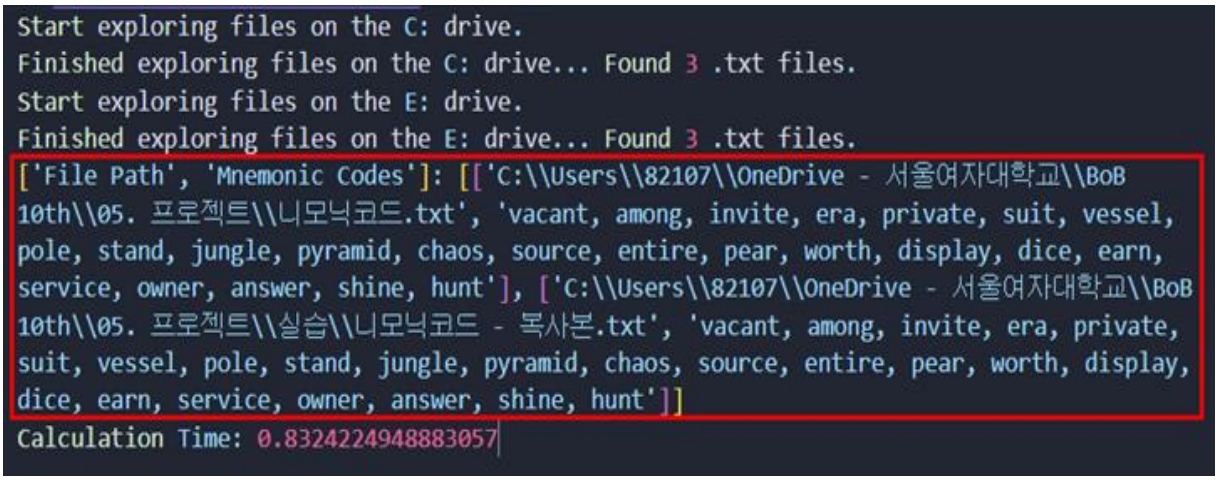

Figure 11 Extract the mnemonic code from disk

IV. Proposal of Cryptocurrency Transaction Extraction Tool

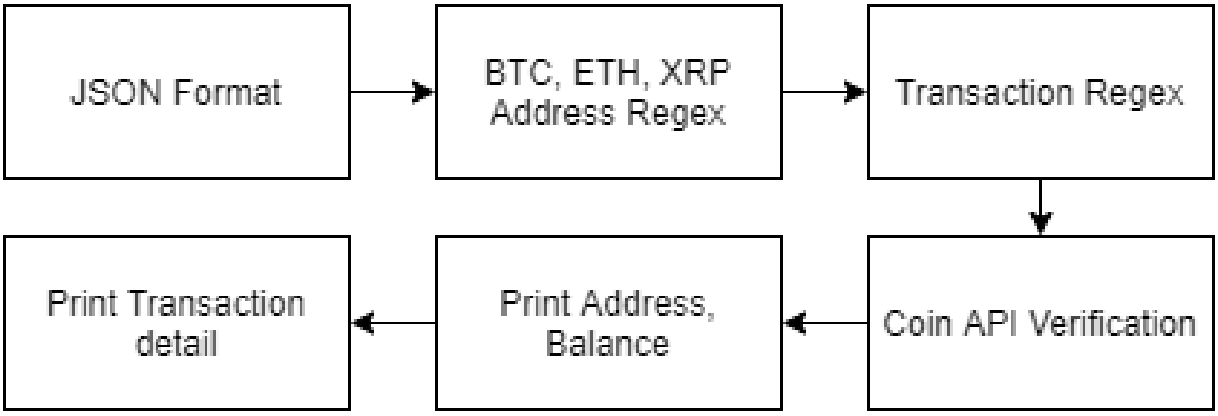

Figure 12 CryptoScan diagram

To have the effectiveness of on-site investigations, this paper was developed using the most used Windows 10 and the latest version of Volatility, a memory analysis tool.

Category	PC Info
OS Version	Windows 10 Pro Education
OS Build	19043.1237
RAM Capacity	4.00GB
SYSTEM	64 Bit
Language	Python3
Library	logging, re, requests, json, time, datetime
IDE	Visual Studio Code

Table 11 Tool development environment

A cryptocurrency transaction was generated in the environment shown in [Table 11], and a memory dump was performed with the 'Belka Live RAM Capturer' tool. The Volatility 3-1.0.1 memory analysis tool operating in Windows 10 was used to extract transaction information, including the sending and receiving addresses of cryptocurrencies (Bitcoin, Ethereum, Ripple) checked in the memory dump file. The goal was to extract cryptocurrency sending/receiving addresses and transaction information by creating a plug-in 'CryptoScan' that works in the memory analysis tool.

In Volatility, the hardware wallet client program can be identified via the pslist plugin. Before executing the CryptoScan plugin, you need to check the PID (Process ID) of the hardware wallet PC application through the pslist plugin as shown in [Figure 14]. CryptoScan divided options for each type of cryptocurrency so that you can search for cryptocurrency wallet addresses and transaction information in PID units of hardware wallet applications. In addition, it was created so that the type of cryptocurrency to be explored can be specified as an option.

Example)

1) python ./vol.py -f [memory.mem] windows.cryptoscan --pid [Process ID] --[btc,eth,xrp] --xrpapi [xrpapi key value] --ethapi [ethereum api key value] --pdf

2) python ./vol.py -f [memory.mem] windows.cryptoscan --pid [Process ID] --swap

3) python ./vol.py -f [memory.mem] windows.cryptoscan --pid [Process ID] --mnemonic

The following is a class of 'CryptoScan'.

```
requirements.TranslationLayerRequirement(name = 'primary',
                                         description = 'Memory layer for the kernel',
                                         architectures = ["Intel32", "Intel64"]),
requirements.SymbolTableRequirement(name = "nt_symbols", description = "Windows kernel symbols"),
requirements.PluginRequirement(name = 'pslist', plugin = pslist.PsList, version = (2, 0, 0)),
requirements.IntRequirement(name = 'pid',
                            description = "Process ID to include (all other processes are excluded)",
                            optional = True),
requirements.BooleanRequirement(name = 'btc',
                                description = "btc : search bitcoin address, transaction",
                                default = False,
                                optional = True),
requirements.BooleanRequirement(name = 'xrp',
                                description = "search ripple address, transaction",
                                default = False,
                                optional = True),
requirements.BooleanRequirement(name = 'eth',
                                description = "search ethereum address, transaction",
                                default = False,
                                optional = True),
requirements.BooleanRequirement(name = 'mnemonic',
                                description = "search mnemonic in the specified process",
                                default = False,
                                optional = True),
```

```
        requirements.BooleanRequirement(name = 'pdf',
                                        description = "create report",
                                        default = False,
                                        optional = True),
        requirements.BooleanRequirement(name = 'swap',
                                        description = "search swapId",
                                        default = False,
                                        optional = True),
        requirements.StringRequirement(name = 'ethapi',
                                        description = "It is essential to add personal api key(etherscan.io).",
                                        default = False,
                                        optional = True),
        requirements.StringRequirement(name = 'xrpapi',
                                        description = "It is essential to add personal api key(developers.cryptoapis.io).",
                                        default = False,
                                        optional = True),
        requirements.BooleanRequirement(name = 'usage',
                                        description = "usage example: python ./vol.py windows.cryptoscan --pid 1112 --eth --
ethapi xadad12akddad112sa --pdf | python ./vol.py windows.cryptoscan --pid 1112 --btc --pdf | python ./vol.py windows.cryptoscan --
pid 1112 --swap | python ./vol.py windows.cryptoscan --pid 1112 --mnemonic",
                                        default = False,
                                        optional = True)
    ]
```

If you enter the option you want, the generator method searches all transaction-related data through the regular expression below. The PID is needed because the search is done in the memory area of the process. PC applications can exchange data directly with the blockchain network to find transaction-related data in the memory area of the process.

The following is a generator of 'Cryptoscan'.

```
def _generator(self, procs):


        for proc in procs:

            pid = "Unknown"
```

```
            try:

                pid = proc.UniqueProcessId

                proc_layer_name = proc.add_process_layer()

                proc_layer = self.context.layers[proc_layer_name]

            except exceptions.InvalidAddressException as excp:

                vollog.debug("Process {}: invalid address {} in layer {}".format(pid, excp.invalid_address,

                                                                                 excp.layer_name))

                continue



            file_handle = self.open("pid.{}.dmp".format(pid))

            with file_handle as file_data:



                eth_apikey = self.config.get('ethapi',None)

                xrp_apikey = self.config.get('xrpapi',None)



                json_Reg = re.compile(r'₩{.*₩:₩{.*₩:.*₩}₩}') #json

                ripple_reg = re.compile(r'r[0-9a-zA-Z]{33,35}')

                btc_reg = re.compile(r'₩b(bc(0([ac-hj-np-z02-9]{39}|[ac-hj-np-z02-9]{59})|1[ac-hj-np-z02-9]{8,87})|[13][a-
km-zA-HJ-NP-Z1-9]{25,35})₩b')

                eth_reg = re.compile(r'0x[a-fA-F0-9]{40}')

                transactions_reg = re.compile(r'[A-Fa-f0-9]{64}')



                mnemonic_reg = re.compile('[a-z]{3,8}')

                swap_id_reg = re.compile(r'("|)(swapId)("|:)(:| )("[a-zA-Z1-9]{10,20}")')

                swap_op_id = re.compile(r'("operationId":")([a-z1-9]{2,10}:[1-9]:[a-z]{3,8})')

                swap_receive_id = re.compile(r'("receiverAccountId":")([a-z1-9]{2,10}:[1-9]:[a-z]{3,8})')
```

address_count = 0

tx_count = 0

duplicated_str = []

printed_str = []

check_pdf_list = []

transaction_list = []

rippple_transaction_list = []

swap_list = []

swap_op_id_list = []

swap_receive_id_list = []

d = enchant.PyPWL("wordlist.txt")

backup_offset = 0

backup_mapped_offset = 0

backup_mapped_size = 0

check_error = 0

t_backup_offset = 0

t_backup_mapped_offset = 0

t_backup_mapped_size = 0

This part is an important code to find the transaction, sending/receiving address, swap, and mnemo nic.

```
try:
                    data = proc_layer.read(offset, size, pad = True)
                    #file_data.write(data) --> mnemonic
                    #file_data.write(data)
                    buf = ''
                    for b in data:
                        buf += chr(b)

                    if self.config['swap']:
                        if 'swap' in buf:
                            if swap_id_reg.search(buf):
                                for j in swap_id_reg.findall(buf):
                                    if j not in swap_list:
                                        if j not in duplicated_str:
                                            swap_list.append(j)
                                            duplicated_str.append(j)

                            if swap_op_id.search(buf):
                                for j in swap_op_id.findall(buf):
                                    if j not in swap_op_id_list:
                                        if j not in duplicated_str:
                                            #print(buf)
                                            swap_op_id_list.append(j)

                            if swap_receive_id.search(buf):
```

```
                    for j in swap_receive_id.findall(buf):
                        if j not in swap_receive_id_list:
                            if j not in duplicated_str:
                                #print(j)
                                swap_receive_id_list.append(j)

    if json_Reg.search(buf):
        if self.config['xrp']:
            for j in ripple_reg.findall(buf):
                if j not in ripple_recv_list:
                    if j not in duplicated_str:
                        ripple_recv_list.append(j)
                        duplicated_str.append(j)

        if self.config['btc']:
            if 'address' in buf and 'bitcoin' in buf:
                for j in btc_reg.findall(buf):
                    if j not in btc_recv_list:
                        if j not in duplicated_str:
                            btc_recv_list.append(j)
                            duplicated_str.append(j)

        if self.config['eth']:
            if 'address' in buf:
                for j in eth_reg.findall(buf):
                    if j not in eth_recv_list:
                        #print(j)
```

```
                    if j not in duplicated_str:
                        eth_recv_list.append(j)
                        duplicated_str.append(j)

if self.config['xrp']:
    if transactions_reg.search(buf):
        if 'hash' in buf and 'ripple' in buf:
            for j in transactions_reg.findall(buf):
                if j not in rippple_transaction_list:
                    if j not in duplicated_str:
                        rippple_transaction_list.append(j)
                        duplicated_str.append(j)

if transactions_reg.search(buf):
    for j in transactions_reg.findall(buf):
        if j not in transaction_list:
            if j not in duplicated_str:
                transaction_list.append(j)
                duplicated_str.append(j)
```

When ‘CryptoScan’ extracts the transaction ID, wallet address, and mnemonic code, it verifies using the public API. The APIs we used are ‘cryptoapis’ and ‘blockchain.com’. Also, the mnemonic code is verified in 2048 mnemonic code lists. When the verification of the transaction data through the API is completed, finally, the balance of the wallet address extracted and transaction details data are output from the result of API.

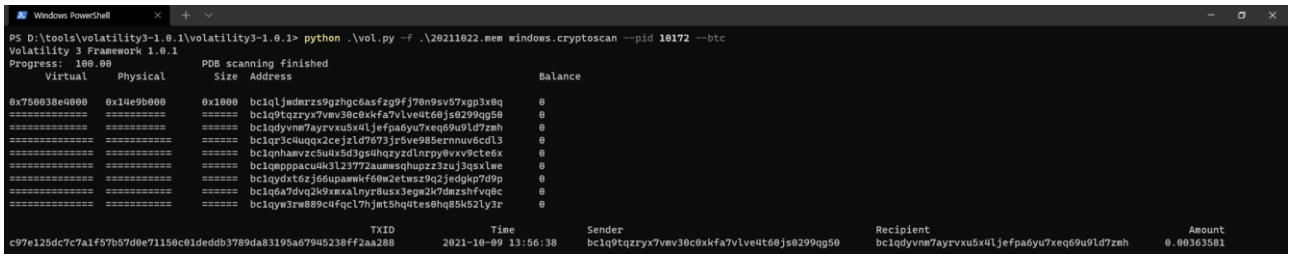

Figure 13 CryptoScan's --btc option output

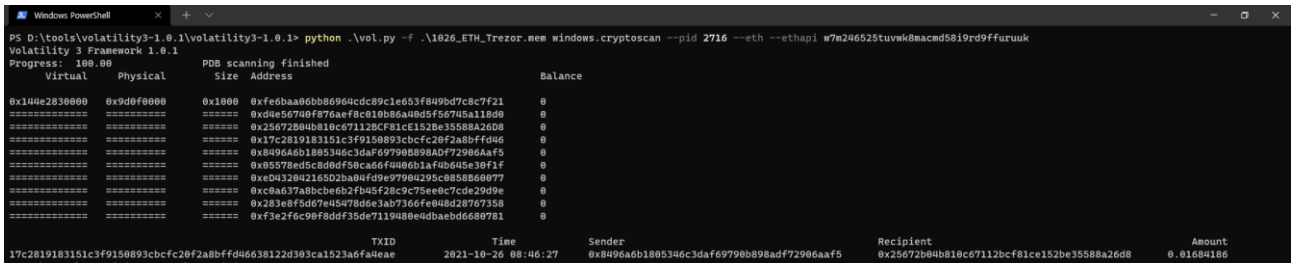

Figure 14 CryptoScan's --eth --ethapi option output

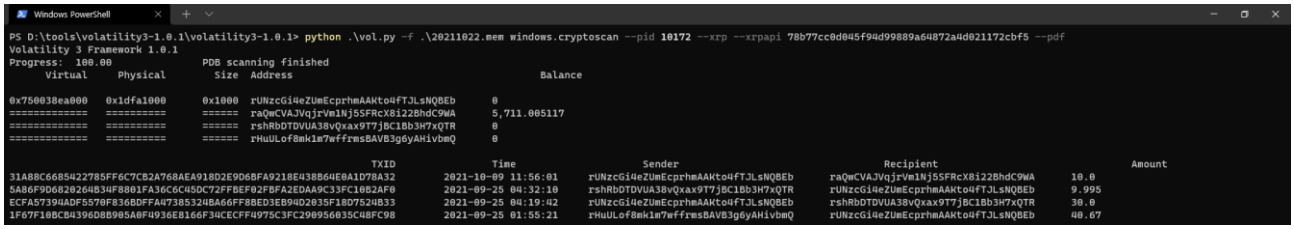

Figure 15 CryptoScan's --xrp --xrpapi --pdf option output

In addition, ‘CryptoScan’ can generate a report based on the result. The report generation option is --pdf. Through this PDF export function, wallet addresses and transaction traces retrieved from memory can be easily identified in documents. In addition, the amount of cryptocurrency remaining in each address can be used to track the hidden funds of criminals. An example of a report is shown in the figure below.

CryptoScan Report

Report Name: CryptoScan_1208

Report Created: 2021-12-08 19:02:55.191832+09:00

Analysis Version: 1.0

Target File Name (size): xxxxxxx.mem (4GB)

Target File Path: : C:\XXXX

Cryptocurrency Transaction Info - Wallet Address

Address	Balance (USD)	Type
rUNzcGi4eZUmEcprhmAAKto4fTJLsNQBEb	0 (0.0 USD)	XRP
raQwCVAJVqjrVm1Nj5SFRcX8i22BhdC9WA	5,711.005117 (4725.268500790449 USD)	XRP
rshRbDTDVUA38vQxax9T7jBC1Bb3H7xQTR	0 (0.0 USD)	XRP
rHuULof8mk1m7wffrmsBAVB3g6yAHivbmQ	0 (0.0 USD)	XRP

Cryptocurrency Transaction Info - Transaction Info

Tag	Value
TXID	31A88C6685422785FF6C7CB2A768AEA918D2E9D6BFA9218E438B64E0A1D78A32
Time	2021-10-09 11:56:01
Sender	rUNzcGi4eZUmEcprhmAAKto4fTJLsNQBEb
Receiver	raQwCVAJVqjrVm1Nj5SFRcX8i22BhdC9WA
Amount	10.0
TXID	5A86F9D6820264B34F8801FA36C6C45DC72FFBEF02FBFA2EDAA9C33FC10B2AF0
Time	2021-09-25 04:32:10
Sender	rshRbDTDVUA38vQxax9T7jBC1Bb3H7xQTR
Receiver	rUNzcGi4eZUmEcprhmAAKto4fTJLsNQBEb
Amount	9.995
TXID	ECFA57394ADF5570F836BDFFA47385324BA66FF8BED3EB94D2035F18D7524B33
Time	2021-09-25 04:19:42
Sender	rUNzcGi4eZUmEcprhmAAKto4fTJLsNQBEb
Receiver	rshRbDTDVUA38vQxax9T7jBC1Bb3H7xQTR
Amount	30.0
TXID	1F67F10BCB4396D8B905A0F4936E8166F34CECFF4975C3FC290956035C48FC98
Time	2021-09-25 01:55:21
Sender	rHuULof8mk1m7wffrmsBAVB3g6yAHivbmQ
Receiver	rUNzcGi4eZUmEcprhmAAKto4fTJLsNQBEb
Amount	40.67

More at Other Platforms

Num	Link
1	https://xrpscan.com/tx/31A88C6685422785FF6C7CB2A768AEA918D2E9D6BFA9218E438B64E0A1D78A32
2	https://xrpscan.com/tx/5A86F9D6820264B34F8801FA36C6C45DC72FFBEF02FBFA2EDAA9C33FC10B2AF0
3	https://xrpscan.com/tx/ECFA57394ADF5570F836BDFFA47385324BA66FF8BED3EB94D2035F18D7524B33
4	https://xrpscan.com/tx/1F67F10BCB4396D8B905A0F4936E8166F34CECFF4975C3FC290956035C48FC98

Figure 16 PDF EXPORT document capture

Ⅴ. Conclusion

This study uses a hardware wallet (Ledger Nano S, Trezor One) to generate a transaction, and then checks the cryptocurrency-related information recorded in the memory through memory forensics. and A method to find a mnemonic code in disk and to find transaction information in memory using Volatility 3 plug-in ' CryptoScan' are presented. Cryptocurrency transaction analysis for PC memory using a hardware wallet is important from the point of view of digital forensic investigations because it can check the sending and receiving wallet address, TXID, cryptocurrency balance, mnemonic code, and PIN code, which could not be revealed in previous studies. Unlike the previous research environment that used Volatility 2 to analyze memory files in the Windows 7 environment, we improved the practicality of the research results by using Volatility 3 in the Windows 10 environment, which we use the most.

For the JSON-formatted cryptocurrency transaction-related information confirmed through memory forensics, wallet addresses with different characteristics for each type of cryptocurrency were extracted through regular expressions [Table 1]. Verification of transactions including the extracted address information is available at btc.com, etherscan, cryptoapis.io0, etc. In addition, the PIN code and mnemonic code recorded in the memory in the process of recovering the wallet were also extracted using a Python3 script using regular expressions. The values extracted using the existing tools were used as data for developing the memory forensics tool 'CryptoScan'.

In the existing cryptocurrency investigation, when a hardware wallet was discovered, it was possible to analyze the hardware wallet only by requesting the subject of the investigation to provide information. However, if the 'CryptoScan' memory forensic tool developed in this study is used, cryptocurrency-related data can be extracted from the memory even if the investigator does not provide information about the hardware wallet. Extractable data are TXID, wallet address, cryptocurrency balance, and mnemonic code, which can be used for cryptocurrency investigations. In addition, the extracted data can be used for solutions that need to know transaction information in advance, such as Reactor of Chainalysis, and can be output as a PDF file in the form of a final report.

In the future, based on this study, we will conduct additional research on the hidden wallet of Trezor One, which could not be restored with a mnemonic code. will be updated to allow navigation of mnemonic codes and PIN codes in addition, the latest devices of the hardware wallet analyzed in this paper provide a Bluetooth connection function with a mobile device, so the research field will be expanded to analysis of cryptocurrency transactions through mobile forensics.