CYBR Investigation – Stage 4: Malware Analysis

Analyse the provided Python files and fill out the document.

WARNING: Do **NOT** run any Python files unless you are sure they are safe.

1. What is the malware doing?

Look at the code of the 2 Python files and determine the behaviour and impact of the code.

An example has been provided for you

|  |  |
| --- | --- |
| **Filename** | db\_update.py |
| **Line number in code** | **Behaviour** |
| |  |  | | --- | --- | | 1 - 7 |  | | |  |  | | --- | --- | |  | Imports several modules: socket (networking), json (parsing JSON data), pymssql (database interaction), time (delays execution), random (generates random numbers), pickle (reads serialized data), and os. | |
| |  | | --- | | 9 - 16 | | |  |  | | --- | --- | |  | fetch\_remote\_config() connects to a remote server via a socket, sends the message "GET ACCOUNTS", receives a JSON response, and extracts a list of account IDs. | |
| |  |  | | --- | --- | | 18 - 21 |  | | |  |  | | --- | --- | |  | read\_local\_config() opens config.pkl, reads its contents, and loads a list of to\_accounts from it. | |
| |  | | --- | | 23 - 33 | | |  |  | | --- | --- | |  | transactions() connects to an MSSQL database (BankDB), iterates through the retrieved accounts, and inserts new transaction records. The transactions appear to be between compromised from\_accounts and predetermined to\_accounts. | |
| |  |  | | --- | --- | | 35 - 42 |  | | |  |  | | --- | --- | |  | main() defines the database credentials and IP addresses for the printer and server. It loads to\_accounts from config.pkl. | |
| |  |  | | --- | --- | | 44 - 48 |  | | |  |  | | --- | --- | |  | Inside a loop, it continuously fetches new from\_accounts from the remote server and processes transactions between from\_accounts and to\_accounts. | |
| |  | | --- | | 50 - 52 | | |  |  | | --- | --- | |  | The script waits for an hour (3600 seconds) before executing another batch of transactions. | |

|  |  |
| --- | --- |
| **Filename** | generate\_pickle.py |
| **Line number in code** | **Behaviour** |
| |  |  | | --- | --- | | 1 |  | | |  |  | | --- | --- | |  | Imports the pickle module to handle serialization. | |
| |  |  | | --- | --- | | 4 |  | | |  |  | | --- | --- | |  | Defines a list of account IDs (to\_accounts) that will be stored in a pickle file. | |
| |  |  | | --- | --- | | 7 - 10 |  | | |  |  | | --- | --- | |  | Opens a file named config.pkl in write-binary (wb) mode and writes the list of account IDs to it using pickle.dump(). This creates a file that will later be used by another script. | |

1. Based on your analysis, hypothesise what the attacker is trying to do with the Python scripts

* Exfiltrate banking information by fetching remote account IDs (from\_accounts) from an external server.
* Execute unauthorized transactions using the pymssql library to access an MSSQL database (BankDB) and transfer funds between accounts.
* Bypass detection by using legitimate-looking transactions with random amounts.
* Maintain persistence with an infinite loop (with 1 hour delay) before executing the next round of fraudulent transactions.

1. Hypothesise how the attacker was able to determine the db user and password.

* Hardcoded credentials – If the application or another script contains credentials in plaintext, they may have been extracted.
* Configuration file leakage – If credentials were stored in a local .env file, config.json, or similar files, they might have been exposed through misconfigured permissions.
* Exploiting an existing vulnerability – The attacker may have gained access through an SQL injection attack or by exploiting a misconfigured MSSQL instance.
* Intercepting network traffic – If database credentials were transmitted unencrypted over the network, they could have been captured using a packet sniffer like Wireshark.