Visa Security Alert **SEPTEMBER 2020**

# NEW MALWARE SAMPLES IDENTIFIED IN POINT-OF-SALE COMPROMISES

**Distribution:** Public

**Summary:**

In May and June 2020, respectively, Visa Payment Fraud Disruption (PFD) analyzed malware samples recovered from the independent compromises of two North American merchants. In these incidents, criminals targeted the merchants' point-of-sale (POS) terminals in an effort to harvest and exfiltrate payment card data. Subsequent to analysis, the first attack was attributed to the malware variant TinyPOS, and the second to a mix of POS malware families including RtPOS, MMon (aka Kaptoxa), and PwnPOS. The recent attacks exemplify threat actors' continued interest in targeting merchant POS systems to harvest card present payment account data. PFD is providing the analysis of these malware variants and the corresponding indicators of compromise (IOCs) to assist in the identification, prevention, and mitigation of attacks using the malware.

**Threat Assessment:**

In the first compromise, threat actors targeted a North American hospitality merchant with the POS malware variant TinyPOS. Initial access to the merchant network was obtained through a phishing campaign that targeted employees at the merchant. Legitimate user accounts, including an administrator account, were compromised as part of this phishing attack and were used by the threat actors to login to the merchant's environment. The actors then used legitimate administrative tools to access the cardholder data environment (CDE) within the merchant's network.

Once access to the CDE was established, the actors **deployed a memory scraper to harvest track 1 and track 2 payment account data**, and later used a batch script to mass deploy the malware across the merchant's network to target various locations and their respective POS environments. The memory scraper harvested the payment card data and output the data into a log file. At the time of analysis, no network or exfiltration functions were present within the sample. Therefore, the actors would likely remove the output log file from the network using other means.

In the second compromise, the threat actors again targeted a North American hospitality merchant with POS malware. Subsequent to analysis, it was determined the threat actors used the malware variants RtPOS, MMon (aka Kaptoxa), and PwnPOS. While less is known about the tactics used by the threat actors in this attack, there is evidence to suggest that the actors employed various remote access tools and credential dumpers to gain initial access, move laterally, and deploy the malware in the POS environment. The malware utilized in these stages of the compromise was not recovered. The POS malware variants used in this attack **targeted track 1 and track 2 payment account data**.

The indicators of compromise associated with the two respective compromises are included below.

## 1. IOCs associated with the first compromise:

**Sample Pair #1**

| | |
|---|---|
| **Filename** | MahjongMCE.bat |
| **MD5** | 9e56cd1c62a11b3f6f789da56cfe581d |
| **SHA1** | ef2466cb91adf7f39f4ec4186009e028b6a86eb3 |
| **SHA256** | 15712752daf007ea0db799a318412478c5a3a315a22932655c38ac6485f8ed00 |
| **SSdeep** | 96:R23qOfh3rYq3fEQcTvKVD3W7T+LMr2EuQsRjgbrl/Om0ltnedUiA5dUi3DRI6QTj:R2H53rY+zoiW7CZ0sFgbrlmm0TeqiA54 |
| **Note** | **PowerShell Loader** |

The batch file contains a call to powershell.exe and a provided base64 encoded command. The command is a standard implementation of reflective injection using PowerShell that is prevalent in many open source frameworks. Of particular interest, this sample loads the **MahjongMCE.png** from the **C:\temp\** folder.

| | |
|---|---|
| **Filename** | MahjongMCE.png |
| **MD5** | 2146d62b2be5b4ec04cd297c4e3094d1 |
| **SHA1** | 453a1d728582aa76d429dacfa2c6022af8bb7abe |
| **SHA256** | e48af0380d51eff554d56aabeeb5087bba37fa8fb02af1ccd155bb8b5079edae |
| **SSdeep** | 768:sAl096SK1r4t3yqvekDqvIj0HLXLz+LILwhgK:sAkK18t3d2xOI0hp |
| **Note** | **PNG Image File with Shellcode** |

The attackers appended raw shellcode after the end of file (EOF) marker for the PNG file. This is an old tactic that allows the file to be properly rendered by an image viewer, while still concealing the appended data. This shellcode is called and executed in memory by the PowerShell Loader.

| | |
|---|---|
| **Filename** | MahjongMCE.png.sc |
| **MD5** | 182edcde38a433f3d965ad8e939315d3 |
| **SHA1** | 09d3c289e7039fe8010ae7fc979749d57653f8a0 |
| **SHA256** | bdd978a91dad7a201274956098d0e6612e3f9e6a009fc4f24a362c19b1813218 |
| **SSdeep** | 96:SaVljuVPqX9wFbpLo1NAxo5fQkv8rC23caapfvcGqGTgiEKuHeDEHJ5N5hIzGtr9:SamSa+QSSSpfcGeeDEp5x6Gl01ogjxli |
| **Note** | **Extracted Shellcode - TinyPOS Point-of-Sale (POS) Malware** |

The shellcode is an evolution of the **TinyPOS Point-of-Sale (POS) Malware** family. Initially the shellcode will execute a small stub which is responsible for decoding the remaining portion of the shellcode. The shellcode is ultimately responsible for scraping the credit card information and preparing it for exfiltration.

| | |
|---|---|
| **Filename** | MahjongMCE.png_decoded.sc |
| **MD5** | da4b2e4f1e6964960ed76c351d81abef |
| **SHA1** | aa61f6034ba53802e4c6a97bd33a850313dc57f9 |
| **SHA256** | 5bc41cde297936199bd145098727905b75762dd85ff2e4caddb93e2370ff8fbc |
| **SSdeep** | 96:cs9SV3V9X62twyKGKJ1AjD4tF/gyN87S4n7OF7vQdQoNio+QPZodkWCbBt:cHZv01AotHO7S47O9HFKPI0t |
| **Note** | **Extracted Shellcode (Decoded) - TinyPOS Point-of-Sale (POS) Malware** |

**Process List Scan**
The shellcode will then enumerate the processes running on the system and specifically look for process names which contain partial names of specific POS software.

**Memory Scraper & Log File**
Once a target process has been located the shellcode parses the memory for credit card track data (specifically Track 1 & 2 data), by completing a series of checks on the string data to ensure that it is formatted to track data standards. The shellcode then completes a Luhn algorithm check on the data to determine if it contains a valid credit card number. Once these steps are completed, the shellcode XOR encodes the scraped data and saves it to the following output log file:

- **Log File** = C:\temp\sys_temp.log
- **XOR Key (Hex)** = fdaa0f49c2beac9f

<u>**Sample File #2**</u>

| | |
|---|---|
| **Filename** | CGLPT64.bat |
| **MD5** | c66c23e8574cec3eb785e5d32c4af253 |
| **SHA1** | adf576aa3a1a01ea4b3f7ad35736068c60646317 |
| **SHA256** | cb7b7c6e37c4edd8bf9c2baaf3d97c895b705565aac7110ba3e7799d9e501172 |
| **SSdeep** | 96:yfCdgNhrQkl4rYq3fEQ7S4LlxSTK8sZGQsaxabrl/OmLuw7+vjwNZh4AA3T7u4ev:yqW3Ekl4rY+zu4JMxsnsaxabrlmmqwuI |
| **Note** | **PowerShell Loader** |

The batch file contains a call to powershell.exe and a provided base64 encoded command. The command is a standard implementation of reflective injection using PowerShell that is prevalent in many open source frameworks. Of particular interest, this sample loads the **cloud_Thumbnail.bmp** from the **C:\journal\** folder.

| | |
|---|---|
| **Filename** | cloud_Thumbnail.bmp |
| **MD5** | b5b4ae0cc7302a9cb039f65bb4ac71da |
| **SHA1** | 2c695af125c6f6b484ab984f95fab1cf764cdc4f |
| **SHA256** | e2f9cb1fcdc531583c82f40c7325118bbc671f4d33ea639f2d575fec96dbbd86 |
| **SSdeep** | 96:aZqgKTLhRb83gg+ruWmjgwX6m/TaXuK9yt27/AtPd6GmQ8RX:aZUhRb83gg+r1mjJHbII2MtV6Gm9p |
| **Note** | **BMP Image File w/ Shellcode** |

The attackers append raw shellcode after the end of file (EOF) marker for the BMP file. This is an old tactic that allows the file to be properly rendered by an image viewer, while still concealing the appended data. This shellcode is called and executed in memory by the PowerShell Loader.

| | |
|---|---|
| **Filename** | cloud_Thumbnail.bmp.sc |
| **MD5** | eab5d0b9d90bcbfa7af5d10b401f73b3 |
| **SHA1** | 32567d0b59bc20c2207b286eaef1df6f67d8c002 |
| **SHA256** | 59adc06ae5a9504313229f252322d8a8e7826999ba1deb036172afd22c0a7774 |
| **SSdeep** | 96:GRb83gg+ruWmjgwX6m/TaXuK9yt27/AtPd6GmQ8RX:GRb83gg+r1mjJHbII2MtV6Gm9p |
| **Note** | **Extracted Shellcode - TinyPOS Point-of-Sale (POS) Malware** |

The shellcode is an evolution of the **TinyPOS Point-of-Sale (POS) Malware** family. Initially the shellcode executes a small stub which is responsible for decoding the remaining portion of the shellcode. The shellcode is ultimately responsible for scraping the credit card information and preparing it for exfiltration.

| | |
|---|---|
| **Filename** | cloud_Thumbnail.bmp_decoded.sc |
| **MD5** | 4362ee278835a5a4ee112e90c490ed05 |
| **SHA1** | 38968d44a1870cf4c4177da08532f556f97c3b8a |
| **SHA256** | 663c69d8bb372487ca9bd8f3b6c983bf7388e79d2ecdb1713718a779f74b11d5 |
| **SSdeep** | 96:DKos9SV3V9X62twyKGKJ1GZSjD4tF/KyNs1S4n7Ov7vQdQwNioOQPZodkWCb6MB:3HZv01G0otB21S47OzHNGPIkB |
| **Note** | **Extracted Shellcode (Decoded) - TinyPOS Point-of-Sale (POS) Malware** |

**Process List Scan**
The shellcode will then enumerate the processes running on the system and specifically looking for process names which contain partial names of specific POS software.

**Memory Scraper & Log File**
Once a target process has been located the shellcode parses the memory for credit card track data (specifically Track 1 & 2 data), by completing a series of checks on the string data to ensure that it is formatted to track data standards. The shellcode then completes a Luhn algorithm check on the data, to determine if it contains a valid credit card number. Once these steps are completed, the shellcode XOR encodes the scraped data and saves it to the following output log file:

- **Log File** = C:\journal\history_0.dat
- **XOR Key (Hex)** = fdaa0f49c2beac9f

## 2. IOCs associated with second compromise:

**File #1**

| | |
|---|---|
| **Filename** | alohae.exe |
| **Source** | Virus Total |
| **MD5** | 9443861a644029b7092a6b7bf98939fb |
| **SHA1** | a3c81c9e3d92c5007ac2ef75451fe007721189c6 |
| **SHA256** | fb749c32b58fd1238f21d48ba1deb60e6fb4546f3a74e211f80a3ed005f9e046 |
| **SSdeep** | 3072:3cAmkDTgWpRT+fAv6Qeyt+TdY5ilY9OBkHTLNVBjBNvOv86NEAg0FujopmDFF369:3R3g8T+foBWlCOBkHtAOXZE0N4 |
| **Note** | **RtPOS Point-of-Sale (POS) Malware** |

**Persistence - Create or Modify System Process: Windows Service (T1543.003)**
The RtPOS Point-of-Sale (POS) Malware accepts only two arguments "/install" and "/remove" which are responsible for installing and removing the service on the victim's machine. When supplied with the "/install" argument, the malware installs itself as a service for persistence and auto execution during Windows startup:

- **Service Name**: WinLogOn
- **Service Description**: Windows Logon Service

**Credit Card Scraping Function**
Following installation, RtPOS then iterates the available/running processes on the compromised machine. This is carried out in two steps; first RtPOS uses CreateToolhelp32Snapshot to obtain a process list, and finally uses Process32FirstW to begin iteration of the process list. Finally, RtPOS uses the ReadProcessMemory function to gain access to the compromised system's memory space. When Track1 and Track2 data is found, the captured information is passed to a Luhn algorithm for validation. The Track1 and Track2 data that pass this verification are then saved to the following file for later exfiltration:

- %SYSTEMROOT%\SysWOW64\sql8514.dat

**File #2**

| | |
|---|---|
| **Filename** | mmon.exe |
| **Source** | Virus Total |
| **MD5** | 255daa6722de6ad03545070dfbef3330 |
| **SHA1** | 80aedf2eddc9e2f39306cbaa63e59c7a08468699 |
| **SHA256** | 86dd21b8388f23371d680e2632d0855b442f0fa7e93cd009d6e762715ba2d054 |
| **SSdeep** | 3072:ikmVcWhCz7cruMlg+PtBxp3bTsZiVXBeN/2KD2VD:/muoCz7cyUP9dbTYipBGG |
| **Note** | **MMon (aka Kaptoxa) Point-of-Sale (POS) Malware** |

**MMon-Derivative POS Malware Families**
MMon is believed to be short for "memory monitor" and is believed to be called Картоха in the underground. The project dates back to at least 2010 and contains the "Kaptoxa" string. The code has been repurposed into

multiple point-of-sale (POS) scraping threats including JavalinPOS, BlackPOS, POSRAM, and others. The mmon.exe utility provides command-line memory scraping and nothing more. However, it is simple for other malware authors to incorporate the tool into their own projects, and this accounts for most of the MMon-derivative POS malware. The mmon.pdb code debug information file is present in all the derivatives of the MMon code base. Many of these related codes have been incorrectly lumped under the BlackPOS label at times by media sources and cyber security publications.

## File #3

| | |
|---|---|
| **Filename** | wnhelp.exe |
| **Source** | Virus Total |
| **MD5** | c86327222d873fb4e12900a5cadcb849 |
| **SHA1** | b1983db46e0cb4687e4c55b64c4d8d53551877fa |
| **SHA256** | 088f40a7a52635ff19e80c62883977d94dd5835e85739e19504f7437d296760b |
| **SSdeep** | 6144:5GM9f8BHPlmg2XR2j0mYHLptiVK0LZV3C5:5x98HPlmg6R2j0mYF4VRLZtq |
| **Note** | **PwnPOS Point-of-Sale (POS) Malware** |

**During execution, the malware drops a copy of itself to the following locations**:
- %SYSTEMROOT%\System32\wnhelp.exe

### Persistence - Create or Modify System Process: Windows Service (T1543.003)
During execution, the malware installs itself as a service for persistence and auto execution during Windows startup:
- **Service Name**: Windows Media Help
- **Service Path**: "%SYSTEMROOT%\wnhelp.exe" –service

### General Log File
During execution, the malware creates a log file used to log its own general behavior. The malware checks for administrator privilege and if it determines the user session does not have administrator privilege, then it outputs an error "ERRLOG:error: not admin user" into the log file.
- %TEMP%\DebugConsole.log

### Credit Card Scraping Function
The scraping function scans process memory and uses the Luhn algorithm to identify credit card data which is written to a log file in plain text. The first record in the log file consists of a date and timestamp, process ID (PID), and the word "START". Subsequent records of the log consist of a date and timestamp, the process path, the process ID (PID), and Track1 and Track2 credit card data.
- %SYSTEMROOT%\System32\perfb419.dat

## 3. Recommendations for Issuers, Acquirers and Merchants

Visa recommends the following best practices to reduce the risk of exposure:

- **Employ the IOCs contained in this report** to detect, remediate, and prevent attacks using the POS malware variant.
- **Secure remote access** with strong passwords, ensure only the necessary individuals have permission for remote access, disable remote access when not in use, and use two-factor authentication for remote sessions.
- **Enable EMV technologies** for secure in-person payments (chip, contactless, mobile and QR code).
- **Provide each Admin user with their own user credentials**. User accounts should also only be provided with the permissions vital to job responsibilities.
- **Turn on heuristics (behavioral analysis) on anti-malware** to search for suspicious behavior, and update anti-malware applications.
- **Monitor network traffic** for suspicious connections, and log system and network events.
- **Implement Network Segmentation**, where possible, to prevent the spread of malicious software and limit an attacker's foothold.
- **Maintain a patch management program** and update all software and hardware firmware to most current release to limit the attack surface for zero-day vulnerabilities.
- **In the event of a confirmed or suspected breach, refer to Visa's** *What to do if Compromised (WTDIC),* **published October 2019.**

For more information, please contact **paymentintelligence@visa.com**