THE JOURNAL OF DIGITAL FORENSICS, SECURITY AND LAW

Journal of Digital Forensics, Security and Law

Volume 15 Article 4

10-27-2020

A FORENSIC FIRST LOOK AT A POS DEVICE: SEARCHING FOR PCI DSS DATA STORAGE VIOLATIONS

Stephen Larson
Slippery Rock University of PA, stephen.larson@sru.edu

James Jones
George Mason University, jjonesu@gmu.edu

Jim Swauger
Binary Intelligence LLC, jswauger@binaryintel.com

Recommended Citation

Larson, Stephen; Jones, James; and Swauger, Jim (2020) "A FORENSIC FIRST LOOK AT A POS DEVICE: SEARCHING FOR PCI DSS DATA STORAGE VIOLATIONS," *Journal of Digital Forensics, Security and Law*: Vol. 15 , Article 4.
DOI: https://doi.org/10.15394/jdfsl.2020.1614
Available at: https://commons.erau.edu/jdfsl/vol15/iss2/4

A FORENSIC FIRST LOOK AT A POS DEVICE: SEARCHING FOR PCI DSS DATA STORAGE VIOLATIONS

Stephen Larson[1], James Jones[2], Jim Swauger[3]
Slippery Rock University of PA[1], George Mason University[2], Binary Intelligence[3]
stephen.larson@sru.edu[1], Jjonesu@gmu.edu[2], jswauger@binaryintel.com[3]

ABSTRACT

According to the Verizon 2018 Data Breach Investigations Report, 321 POS terminals (user devices) were involved in data breaches in 2017. These data breaches involved standalone POS terminals as well as associated controller systems. This paper examines a standalone Point-of-Sale (POS) system commonly used in smaller retail stores and restaurants to extract unencrypted data and identify possible violations of the Payment Card Industry Data Security Standard (PCI DSS) requirement to protect stored cardholder data. Persistent storage (flash memory chips) were removed from the devices and their contents were successfully acquired. Information about the device and the code running on it was successfully extracted, although no PCI DSS data storage violations were identified.

Keywords: POS device, PCI DSS, compliance, data extraction, chip-off

1. INTRODUCTION

Data breaches involving Point-of-Sale (POS) devices are becoming commonplace. These breaches are happening in many industries, and no industry seems to be immune (Cheney et al. 2012). A shortlist of recent POS breaches would include restaurants (Applebee's, Arby's, Sonic, Chili's, Zippys', Subway, etc.), health care facilities (Oregon clinic), hospitality facilities (Holiday Inn, Hyatt), and various retail stores (Forever 21, Target (Manworren et al. 2016), Home Depot, etc.). Even POS vendors, payment systems, and service providers such as Harbortouch, DataPoint POS, and Heartland Payment Systems (Cheney, 2010), are not immune (Verizon, 2018).

Many POS systems consist of a workstation or server (controller) running a mainstream operating system and a POS terminal or device that reads the card. In such systems, the controller executes the processing and communication tasks. Other POS systems consist only of a POS device or terminal, which has the capability to store and process some subset of the card and transaction data, and the device or terminal also communicates with a remote system to execute the transaction. Past breaches have involved the standalone POS terminals or devices, while other breaches have involved an accompanying POS controller or server, which are often running an MS Windows operating system (Bodhani, 2014). Past breaches have been attributed to a lack of security on the victim's POS terminal, POS

controller, and the data network on which the POS systems reside and how they communicate.

This paper presents a forensic first look at a common POS device found in smaller retail and restaurant establishments, specifically a VeriFone device deployed under the Verizon brand. These particular devices were selected for their widespread adoption in small and medium-sized businesses more likely to have weak security programs (Lueck, 2014), their standalone design, which implies unprotected sensitive data will be on these devices instead of a separate controller, and their ready availability on the resale market. These devices were examined to identify plaintext operational information with security implications and to determine whether they violated the data storage requirement of the PCI DSS standard. An "Existing Data/Specimens" form was filed with one of the author's Institutional Review Boards (IRB) for human subjects research in the event that PII was found or is found in the future, although none has been found to date in this work.

2. INTRODUCTION TO PCI DSS

"PCI DSS is the global data security standard adopted by the payment card brands for all entities that process, store or transmit cardholder data and sensitive authentication data. It consists of steps that mirror security best practices" (PCI Security Standards Council, 2018).The main goal of PCI DSS is to protect cardholder data using the PCI security standards.These standards "apply to all entities that store, process, and transmit cardholder data. It covers technical and operational system components included in or connected to cardholder data" (PCI Security Standards Council, 2018).For a broad understanding of how PCI DSS is structured, please see Seaman (2020)

PCI DSS requirement 3 focuses on protecting stored cardholder data. "Cardholder data refers to any information printed, processed, transmitted, or stored in any form on a payment card" (PCI Security Standards Council, 2018).

Guidelines to protect cardholder data include:

1. Store data only as long as necessary for business
2. Sensitive authentication data should not be stored after authorization (even if it is encrypted)
3. Only display either the first six or last four digits of the primary account number (PAN).
4. Encrypt the PAN when stored
5. Clearly, document procedures to protect encryption keys
6. Implement procedures to protect encryption keys

Specific guidelines on how to protect the data are in Appendix 2.

Storage of certain data elements on a payment card is permitted; for example, the PAN can be stored with no encryption. This may seem contradictory to the guidelines mentioned above, but the difference is that the PAN can be stored with no encryption, but only the first six or last four digits of the PAN can be displayed. The cardholder name, service code, and expiration date can be stored but must be encrypted (in the Target case, such data was encrypted, but not immediately; Plachkinova & Maurer, 2018)). The full track data (the data on the magnetic stripe), the 3-4 digit verification value (CAV, CVC, CVV, or CID), and the PIN cannot

be stored in any form. Storage guidelines for cardholder data elements are in Table 2 (PCI Security Standards Council, 2018).

3. LITERATURE REVIEW

A literature review was conducted to determine if any research had been done related to extracting unencrypted data and identifying possible violations of the PCI DSS requirement to protect stored cardholder data. However, the authors found little evidence of similar research on this topic. Most of the related research found pertained to the security of POS devices and their compliance with PCI DSS, not the forensic examinations thereof.

Souvignet and Frinkenc (2013) explored using differential power analysis as a digital forensic tool on skimming devices often found attached to POS devices. Payment card data skimming consists of the illegal acquisition of payment card details when a user is paying at a POS device. They proposed a non-invasive method to gain forensic evidence from the POS device. Rogers (2017) gives a cursory look at POS devices and digital forensics pertaining to that, while Guo and Jin (2010) presented a range of devices available for manipulating magnetic stripe card data, demonstrate the ease by which magnetic stripe cards are often fraudulently cloned and analyze how such devices can be processed in a forensically sound manner.

Smith (2014) found that POS system attacks were generally quite simple and predominantly involved guessing passwords and subsequent installation of keyboard loggers on smart cash registers used and gave recommendations to improve POS system security.

Amrichová and Mézešová (2019) presented a pseudo code for a secure string class that is compliant with the data retention and cryptography requirements of the PCI DSS standard. Gomzin (2014) provides a resource that "goes beyond standard PCI compliance guides to offer real solutions on how to achieve better security at the point of sale." Frisby et al. (2012) investigated several smartphone POS systems that used an audio-jack magnetic stripe reader.

Several researchers explored the costs of PCI DSS compliance and non-compliance. Morse and Raval (2008) discussed and evaluated the basic framework of PCI DSS and its legal implications. Shaw (2010) explored using PCI DSS compliance to encourage merchants to meet those security standards. Kidd (2008) discussed the penalties and business risk of non-compliance while

Rees (2010) discussed why companies find it difficult to understand what is required of them in PCI DSS compliance, as did Clapper and Richmond (2016).

Sarrafpour et al. (2019) attempted to break or bypass the PCI DSS security measures using varying degrees of vulnerability and penetration attacks in a methodological format.

4. THEORETICAL JUSTIFICATION

Point of sale devices, and embedded systems in general, consist of four common functional components: persistent storage, volatile memory, a processor, and communications elements. Persistent storage contains the device firmware (bootloader and operating system), configuration data, and possibly other data written during operation. Volatile memory (RAM) and processor contents are interesting only for a running system (Rodríguez, 2017; Sherstobitoff, 2008; Hizver & Chiueh, 2011), and communication elements are interesting for the data they transmit on a live system as well as possible access vectors. We are specifically interested in residual and security-relevant data

available on a dead system, so we focused on persistent storage. In the case of the POS systems we examined, this persistent storage is made up of a ROM chip containing a bootloader and a flash memory chip soldered to one of the device's printed circuit boards.

The flash memory on an embedded device like a POS system will contain operating system and related executable code, typically in compiled form. Configuration information, possibly including user account information, will also be stored in persistent storage. Additionally, the running system may write transactions or other data to the flash memory. The contents of the flash memory may be encrypted in whole or in part using on-chip encryption or off-chip encryption implemented in hardware or software. Our goal in this work is to extract the raw contents of the flash memory and determine the data that can be extracted and the possible security implications of access to that data.

5. MOTIVATION

Recently one of the authors visited a restaurant and paid with a credit card. The POS device was relatively small and was the type that contacts the payment system provider directly via a telephone call. The call did not succeed, and after a few seconds, the POS device printed two receipts. This worried the author somewhat – was his payment information (name, credit card information, transaction information) still on the POS device, and if so, how long would it remain there until it was processed?

The author noted the make and model of the POS device and contacted the payment system provider (Heartland Payment Systems). Heartland provided no help, except to state that they only provide the credit card charging system and that any other information would have to come from Verizon.

A call was made to Verizon customer support for VeriFone devices. During this call, the author learned that the transaction information is only on the POS device until processed, and the unprocessed transactions are cached until batch processing can be done later in the day. They did not provide details on when the batch processing would occur as it can be done manually or automatically (daily time set by the device operator). When the batch process is finished, the transaction information is deleted.

This information caused the author concern: what if the POS device was stolen before the batch processing is performed, and is the transaction information securely deleted?

6. DEVICE ACQUISITION

One of the authors bought three used POS devices via craigslist. All were of the Verizon brand and were Verifone models; one was an Omni Vx570 model, and two were Omni Vx610 models. These devices are the same type that the author had observed in his restaurant incident. The two Vx610 devices were owned by a catering company (food truck) whose owner said they only connected to process transactions in the evening. The Vx570 model was owned by a bakery. One of the Verifone Vx610 models used for this examination is shown in Figure 1.

7. DATA EXTRACTION

This section will detail the data extraction efforts. The first attempt included printing data, followed by a Joint Test Action Group (JTAG) access attempt and a chip-off extraction.

Figure 1. Verifone Vx610 POS device.

7.1 Initial extraction

It was decided to power up one device and see if any data could be extracted from it. The data extraction began with powering on the Vx570 device and printing the last transaction. The printout began with a summary/detail report of transactions, which included the device owner's name, address, and phone number, followed by the information on the last batch processed (date, time, batch#, terminal ID, list of transactions, their dollar amount, and total). This was followed by two printouts of the last transaction (customer copy and merchant copy), which included:

- Device owner, address, and phone number
- Terminal ID #
- Transaction date & time
- Credit card type & last 4 digits of the credit card number
- Transaction amount
- Credit card owner name
- Transaction approved/not approved
- Batch #
- Authorization number

These transaction copies were followed again by a summary and detailed report of transactions as shown in Figure 2.

The Vx570 device was briefly disassembled and reassembled to get a quick look at the motherboard. Disassembly involved removing the cover and disconnecting the printed circuit board assemblies and their accompanying cables (see Fig. 4).

After reassembly, the device was turned on, and it immediately displayed an error message on the screen: "Security Alarm, Tampering Detected [20000] Please reboot". This message is known to be triggered by physical disassembly of the device case.

Additionally, on one of the Vx610 devices, a Verix Multi-Application Conductor (VMAC) Summary Report was printed, which contained information about the terminal: the OS version, the terminal type, the terminal serial number, RAM, FLASH memory, and GID information (see VMAC Summary Report in Figure 3).

We also printed a VMAC Detail Report, which included a VMAC Summary Report, followed by Group Details for each application group.

After these two reports were printed, the device was connected to a computer via its USB and Ethernet ports, but the computer did not recognize that a device was connected. A wifi connection attempt also failed. This confirmed the entries on the VMAC detail report, which stated that Wifi and Ethernet were not supported on this terminal and the USB port was disabled.

Figure 2. Last Transaction Report.

7.2 Digital extraction – Joint Test Action Group (JTAG)

JTAG forensics is an advanced level data acquisition method that involves connecting to Test Access Ports (TAPs) on a device and instructing the processor to transfer the raw data stored on connected memory chips. When supported, JTAG is an effective technique to extract a full physical image from devices that cannot be acquired via existing exposed interfaces (Binary Intelligence, 2013). JTAG interfaces provide connections to specific pins on a chip (TDI, TDO, TMS, TCK, and optionally TRST), and hence provide the ability to read data from (and write data to) a chip in the JTAG chain without damaging the hardware, e.g., via chip-off, which is difficult to "undo." JTAG is often used to test chips and circuits after manufacturing and PCB mounting and before release. However, because JTAG interfaces provide direct chip control and possibly access to proprietary information, JTAG interfaces are often hidden or disabled prior to public release, and the chip-specific commands to read and write data via JTAG are often not publicly released.

An online search did not reveal known JTAG interfaces for the POS devices under examination, so we inspected the PCBs upon disassembly. There is no single form factor for JTAG interfaces, although the most common for ARM processors (these devices use a Samsung S3C2410AL-20 chip with an ARM processor) are two-row 14- and 20-pin layouts (SECONS, 2008). Neither of these was apparent on the PCBs as either pin headers or unpopulated pads. However, the PCBs did have two separate groups of unpopu-


VMAC
SUMMARY REPORT
05/11/15 13:41
S.8.0Version 13:41
Terminal OS Version: QBU111A2
Terminal Type: VX610
Terminal Serial #: 213-696-415
Total RAM: 2048K
Available RAM: 773K
Total FLASH: 4096K
Available FLASH: 2008K
GID Information
Group Application Name Version
1 VMAC
1 COMMSVR
2 SPC
15 VMAC.LIB S.8.0.5

Figure 3. VMAC Summary Report.

lated pads, one row of 6 pads with 3 visible trace connections and another row of 12 pads with 10 visible trace connections. We probed these pads to identify VCC and ground but did not find a pattern consistent with a known JTAG layout (which typically has multiple contiguous ground pads in addition to the 4 or 5 JTAG connections). Using a JTAGulator device, we also attempted connections to multiple different pad combinations in these two groups without obtaining a successful connection. While a JTAG connection and chip access may have still been possible, it likely would have been time-consuming and may not have been possible at all if the JTAG connections were destroyed in some way (blown fuse, etc.) or non-existent in the first place. Since we did not need to maintain an operating device after obtaining flash contents, we shifted our approach to a destructive but likely-to-succeed chip-off attempt.

7.3 Chip-off data extraction

The VeriFone Vx610 devices were disassembled to expose the printed circuit board assemblies. The Printed Circuit Board Assemblies (PCBA) were inspected to identify non-volatile flash memory components capable of data storage. For each POS device, a single Spansion flash memory chip model (S29)GL032N90FFI02 was located upon removal of the PCBA electromagnetic shielding.

The research was completed to determine the Spansion GL032N90FFI02 particulars. The manufacturer technical datasheet for the S29GL-N family of devices was obtained from www.alldatasheet.com and reviewed. This revealed that the target chip is a NOR based flash memory component with a storage capacity of 4 megabytes. Specifically, the component is a 32-Mb device organized as 2,097,152 words or 4,194,304 bytes with a 16-bit wide data bus. While the part is available in multiple chip packages, our target component was determined to be the 64-ball fortified BGA (ball grid array) package option (Figure 4).

The UP&UP UP-828P chip programmer control software was then checked to confirm read/imaging support for the target part. The UP-828P did show explicit support for the GL032N90FFI02, and the necessary EBGA64P chip adapter was available. The control software was further checked to verify that the indicated part organization and capacity matched the values detailed in the chip datasheet. The control software presented the chip organization as ”200000h

x 16-bit," which corresponds to the 4,194,304 bytes and 16-bit data bus as detailed in the datasheet.

The devices each had a 4MB flash memory chip (Figure 5), and the extraction effort supplied both raw chip-off images as well as a copy where the byte order was reversed (RBO2). Data were extracted from both Vx610 devices.

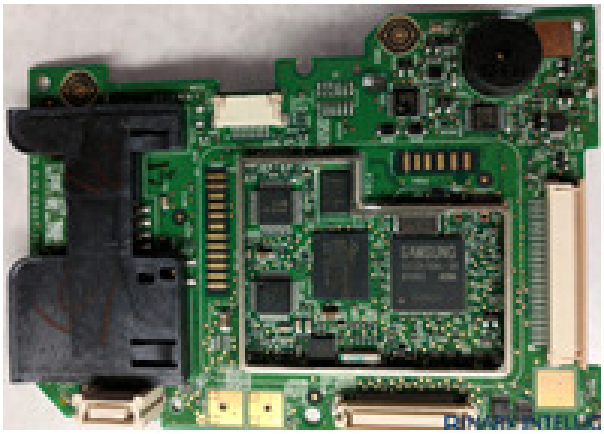

Figure 4. Vx610 Motherboard.

Figure 5. Flash Memory Chip.

With imaging support verified, the chip-off acquisitions were performed. There was no visible chip underfill or coating, so no adhesive removal was necessary. The GL032N90FFI02 chips were removed using a Zephyrtronics ZT-7 hot air reflow unit per the following steps:

- The PCBA was mounted to the work cradle
- The PCBA was preheated using a Zephyrtronics AirBath digital preheating unit @ 205°C
- RMA-223 flux paste was applied around and under the target chip
- The ZT-7 reflow unit nozzle was positioned over the target chip
- The ZT-7 reflow unit was engaged to ramp to a setpoint of 285°C
- The ZT-7 vacuum probe was activated for automatic chip removal

The target chips automatically lifted when the ZT-7 reflow unit nozzle temperature reached approximately 280°C. After cooling, the removed chips were cleaned using 99% isopropyl alcohol in an ultrasonic cleaner. Once clean, the chips were inspected for the presence of solder bridges, excess or insufficient solder amounts on each pad, and any other areas in need of correction. No corrections were required for the target chips.

Each target chip was imaged using a UP&UP UP-828P universal chip programmer per the following steps:

- An EBGA64P chip package adapter was attached to the UP-828P programmer unit
- The UP-828P control software version 2.0.4.8 was launched
- The control software was configured to read the target chip by selecting the S29GL032N90FFI02 chip profile

- The READ function was used to download the chip data into the control software buffer
- The VERIFY function was used to reread the chip data and compare it to the buffer contents
- The verified chip data was saved to a binary image file

The Spansion GL032N90FFI02 chips were successfully imaged with no errors, and the verification process yielded identical data. At 4,194,304 bytes, the file sizes of the saved binary images matched the expected size as detailed in the chip manufacturer datasheet. This confirms that we acquired true and accurate forensic grade copies of each chip's data.

The chip-off procedure and imaging were completed in accordance with accepted digital forensic industry standards and practices. MD5, SHA1, and SHA256 cryptographic hash values were calculated immediately after data acquisition and could be utilized to verify data integrity during analysis.

After the chip-off data extraction, the extraction team performed a big/little endian conversion by reversing the byte order so that ACSII strings would be human-readable. It is these converted files that were subsequently analyzed.

8. ANALYSIS AND RESULTS

Both flash images were examined for PCI DSS protected data, specifically cardholder and authentication data. Using the available last transaction data to seed explicit searches, then searching the extracted data using regular expressions based on PCI DSS protected data and transaction data formats, we found no protected or transaction data in clear text (such as card type, transaction amount, cardholder's name, last 4 digits of the card number, etc.) in either image file.

The two flash images were then examined in a hex editor for strings, similarities, and differences (image md5 hashes are different to establish what other information might be available). [1] [2]

Extracted data indicates that both units ran the Verix v Operating System on 68k processors (Motorola 68000; 32-bit CISC). The kernel is located in bytes 0-65535 (64 KB); this is the first 128 sectors assuming a 512-bytes sector size. Differences in these first 128 sectors began at byte offset 827 and continued at byte offsets as noted in the partial list below:

827, 1167, 1910-1911, 1918, 1923, 3119, 3563, 3586-3587, 626-3627, 3983, 4027, 4071, 4278-4279, 4330-4331, 4731, 4759, 4763, 4955, 5843...

Byte differences at the locations above are all in the 4th (if one) or 3rd and 4th octets (if pair) of 4-octet groups; these look like op codes for a 32-bit processor, and the last byte or last two bytes of each 4-octet op code is/are the op code's arguments (e.g., memory address, register, etc.) (Golden Crystal (date unknown). Device serial numbers and kernel versions are near the end of the kernel block (∼bytes 65310-65527). The serial number is the bold string starting with 212 or 213, and the kernel version is the bold string starting with V.QB.

kernel 1: VX610 US M25654336USDD **00000212-318-169**160960790312806411 10/02/2009 **V.QB0111A3 "2.5.2"**

kernel 2: VX610 US M25654336USDG **00000213-696-415**161351220312806411 08/14/2009 **V.QB0111A2 "2.5.1A"**

The flash file system begins after the ker-

[1] kernel 1 hash:16521FB7808153E5A52E2994FD78BFF1

[2] kernel 2 hash:AF3A27BE64838A2FD10B79CE611BE558

nel, located at byte offsets 65536-573440 (sector 128-1119). The file system range was identified after finding the files, as described below. Sector 1120 starts with 0xFFs; non-0xFFs are at various sectors (different in each image, but both start at sector 1152). In order to identify files in the images, file names were identified by first finding the load list (bytes 129640-129975 in these images). The load list contained the following filenames and is the same in both images. All files were found in the flash images except wizard.ram (I: indicates RAM file system).

- B:_crypto.ram
- B:schedulr.bin
- B:iic.bin
- B:clk_6900.bin
- B:clk_3610.bin
- B:_upgrade.tmp
- B:_buf_mgr.bin
- B:bpr.bin
- B:com1_.bin
- T:_info.bin
- T:console.bin
- U:security.bin
- U:pipe_mgr.bin
- V:icc1.bin
- W:ipp.bin
- Y:usb_host.bin
- Y:cdma_usb.bin
- Y:usbd.bin
- Y:com2_.bin
- Y:com3_.bin
- Z:mag.bin
- Z:printer.bin
- Z:battery.bin
- I:wizard.ram

File contents were found by first searching for the file name from the load list, then identifying a common precursor byte string that indicated the start of a new file. Most but not all files in the two images used constant precursor byte strings: kernel 1 used 0x09100214, and kernel 2 used 0x09081411. Files were identified at the locations noted in Table 1 in the Appendix. Differences noted in **bold red** (file contents were not directly compared); start/end/size are byte offsets in decimal; end and size are estimated based on the start of the next file. VLR files are screen messages.

More files were found by searching for drive prefixes (capital letter followed by a colon), then looking for that file name in other locations. For these files below (a partial list), file data follows the filename by about 40 bytes:

- certfile.sys
- modprof.s37
- ASC8X21.VFT
- ASC4X16.VFT
- SPUTIL.VFT
- VCS_8X16.vft
- VCSLOGOS.VFT
- VMAC.LIB
- cdma.p7s

- cdma.out

Using strings and manual examination of the remainder of the data in the images (sectors 1120-end) revealed the following potentially useful but not fully explored items:

- Multiple locations in image 1 and 2: "PREDATOR" (user? also in ped.lib)
- Sector 1400 (image 1) and 1322 (image 2): FTP user (anonymous) and password (guest@vfi.com);
- Sector 1773 (image 1) and 1695 (image 2): "Password:" and "66831" (a known default access code for downloads, time/date changes, etc. (Verifone, 2012)
- Variable "*GO" in configuration file sets the startup program, both set to F:APLOAD.OUT (this file was not found on either flash image)
- Sector 1389 (image 1), 1311 (image 2):

  Banshee.Carlos Only.CO561 Only..Carlos+CO561....MC56 Only...MC55 Only... EM3420 Only.CO710 Only..Carlos+MC56.Carlos+MC55. Carlos+EM3420 ... Carlos+CO710....Eisenhower

The rest of the image files contained mostly compiled code and suspected encrypted data (based on data entropy); the rest contained clear text entries of error messages, informational messages, initialization messages, menu items, file names and locations, configuration settings, and device information.

9. DISCUSSION AND CONCLUSION

The analysis results suggest that most of the data is either binary code or encrypted data, and the clear text appears to contain only informational and error messages, configuration information, and some access information.

IP information such as network configuration (DNS, DHCP, IP addresses, etc.) was in clear text, as was an FTP username and password, presumably used for non-sensitive downloads.

The last transaction could be printed, which indicates it is present in the non-volatile flash memory. However, no cardholder or transactional data was found in clear text in the extracted flash images, which likely indicates that transactional and cardholder data that is retained is also encrypted. Given that no sensitive authentication data was printed, no evidence of PCI DSS data storage violations was identified.

The most common data breaches that involve POS devices are caused by some type of malware. The devices we examined did not exhibit signs of being infected with malware, though because this examination was focused on ensuring PCI DSS compliance, we did not investigate for malware beyond a cursory look.

10. FUTURE RESEARCH

This paper analyzed a single model of POS system, and the analysis has only begun. We intend to explore and analyze the acquired image files more fully and to analyze additional POS devices.

Additional analysis will also examine the Verix v Operating System and code apparently running on these POS systems.

Lastly, we hope to acquire similar POS devices and their accompanying payment system services so we can control the configuration data and transaction data in order to trace it through the system and enable more detailed searches.

11. REFERENCES

1. Amrichová, K., Mézešová, T. (2019) A Secure String Class Compliant with PCI DSS. CECC 2019: Proceedings of the Third Central European Cybersecurity Conference. November 2019 Article No.: 17 Pp 1–5. https://doi.org/10.1145/3360664.3360681.

2. Binary Intelligence (2012). Chip Off Forensics for Almost Any Device. Retrieved on 12 June 2020 from http://www.binaryintel.com/chip-forensics-device/.

3. Binary Intelligence (2013). JTAG Forensics. Retrieved on 12 June 2020 from http://www.binaryintel.com/services/jtag-chip-off-forensics/jtag-forensics/.

4. Bodhani, A. (2014). Securing the sale. Engineering Technology, 9(5), 36-40.

5. Cheney, J. (2010). Heartland Payment Systems: lessons learned from a data breach. FRB of Philadelphia-Payment Cards Center Discussion Paper, (10-1).

6. Cheney, J., Hunt, R., Jacob, K., Porter, R., Summers, B. (2012). The efficiency and integrity of payment card systems: Industry views on the risks posed by data breaches. Economic Perspectives, 36(4).

7. Clapper, D., Richmond, W. (2016) Small Business Compliance with PCI DSS. Journal of Management Information and Decision Sciences; Weaverville 19(1), 54-67.

8. Frisby, W., Moench, B., Recht, B., and Ristenpart, T. (2012) Security Analysis of Smartphone Point-of-Sale Systems. 6th USENIX Workshop on Offensive Technologies. Aug 6-7, 2012, Bellevue, WA.

9. Golden Crystal (date unknown). Motorola 68000 CPU Opcodes. Retrieved 12 June 2020 from http://goldencrystal.free.fr/M68kOpcodes-v2.3.pdf.

10. Gomzin, S. (2014) Hacking Point of Sale: Payment Application Secrets, Threats, and Solutions. John Wiley Sons, February 2014.

11. Guo, H., Jin, B. (2010) Forensic analysis of skimming devices for credit fraud detection. 2010 2nd IEEE International Conference on Information and Financial Engineering. 17-19 Sept. 2010. Chongqing, China. DOI: 10.1109/ICIFE.2010.5609418

12. Hizver, J., Chiueh, T. C. (2011, February). An introspection-based memory scraper attack against virtualized point of sale systems. In International Conference on Financial Cryptography and Data Security, pp. 55-69. Springer, Berlin, Heidelberg.

13. Kidd, R. (2008) Counting the cost of non-compliance with PCI DSS. Computer Fraud Security, 2008(11), 13-14.

14. Lueck, S. G. (2014). Point of Sale terminal security. Doctoral Dissertation, Utica College. ProQuest Dissertations Publishing, 2014.

15. Manworren, N., Letwat, J., Daily, O. (2016). Why you should care about the Target data breach. Business Horizons, 59(3), 257-266.

16. Morse, E., Raval, V. (2008) PCI DSS: Payment card industry data security standards in context. Computer Law Security Review, 24(6), 540-554.

17. PCI Security Standards Council (2018). PCI DSS Quick Reference Guide: Understanding the Payment Card Industry Data Security Standard version 3.2.1. Retrieved 12 June 2020 from https://www.pcisecuritystandards.org/documents/PCI_DSS-QRG-v3_2_1.pdf?agreement=true&time=1545342055082.

18. Plachkinova, M., Maurer, C. (2018). Teaching case: Security breach at Target. Journal of Information Systems Education, 29(1), 11.

19. Rees, J. (2010) The challenges of PCI DSS compliance. Computer Fraud Security, 2010(12), 14-16.

20. Rodríguez, R. J. (2017). Evolution and characterization of point-of-sale RAM scraping malware. Journal of Computer Virology and Hacking Techniques, 13(3), 179-192.

21. Rogers, M. (2017) Technology and Digital Forensics. The Routledge Handbook of Technology, Crime and Justice. edited by M. R. McGuire, Thomas J. Holt. Taylor Francis, Feb 24, 2017.

22. Sarrafpour, B. A., Choque, R., Mitchell, B., Mehdipour, F. (2019) Commercial Security Scanning: Point-on-Sale (POS) Vulnerability and Mitigation Techniques. 2019 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), Fukuoka, Japan, 2019, pp. 493-498, doi: 10.1109/DASC/PiCom/CBDCom/CyberSciTech.2019.00099.

23. Seaman, J. (2020) PCI DSS An Integrated Data Security Standard Guide. Apress, Berkeley, CA DOI https://doi.org/10.1007.

24. SECONS (2008). JTAG Pinouts. Retrieved 22 June 2020 from http://www.jtagtest.com/pinouts/. Sherstobitoff, R. (2008). Anatomy of a data breach. Information Security Journal: A Global Perspective, 17(5-6), 247-252.

25. Smith, D. (2014) Preventing Point-of-Sale System Intrusions. Master's thesis. Naval Postgraduate School. Monterey, CA. Retrieved 12 June 2020 from https://apps.dtic.mil/dtic/tr/fulltext/u2/a607543.pdf.

26. Souvignet, T., Frinkenc, J. (2013) Differential Power Analysis as a digital forensic tool. Forensic Science International, 230(1–3), 127-136.

27. Swauger, J. (2012). Chip-off Forensics. Retrieved 12 June 2020 from http://www.binaryintel.com/wp-content/uploads/2012/05/Chip-Off_Forensics_Article.pdf.

28. Verifone (2012). Verifone Vx610 Download Instructions. Retrieved 11 June 2019 from https://hippocharging.zendesk.com/hc/en-us/article_attachments/200367705/07.30.12_-_Vx_Download_Instructions.pdf.

29. Verizon 2018 Data Breach Investigation Report. Retrieved 11 June 2019 https://www.verizonenterprise.com/resources/reports/rp_DBIR_2018_Report_en_xg.pdf.

PCI DSS Quick Reference Guide: Protect Cardholder Data (PCI 2018, 14-15)

Cardholder data refers to any information printed, processed, transmitted or stored in any form on a payment card. Entities accepting payment cards are expected to protect cardholder data and to prevent its unauthorized use – whether the data is printed or stored locally or transmitted over an internal or public network to a remote server or service provider.

Requirement 3: Protect stored cardholder data

Cardholder data should not be stored unless it's necessary to meet the needs of the business. Sensitive data on the magnetic stripe or chip must never be stored after authorization. If your organization stores the Primary Account Number (PAN), it is crucial to render it unreadable (see 3.4, and Table 2 for guidelines).

3.1

Limit cardholder data storage and retention time to that which is required for business, legal, and/or regulatory purposes, as documented in your data retention policy. Purge unnecessary stored data at least quarterly.

3.2

Do not store sensitive authentication data after authorization (even if it is encrypted). See the table below. Render all sensitive authentication data unrecoverable upon completion of the authorization process. Issuers and related entities may store sensitive authentication data if there is a business justification, and the data is stored securely.

3.3

Mask PAN when displayed (the first six and last four digits are the maximum number of digits you may display), so that only authorized people with a legitimate business need can see more than the first six/last four digits of the PAN. This does not supersede stricter requirements that may be in place for displays of cardholder data, such as on a point-of-sale receipt.

3.4

Render PAN unreadable anywhere it is stored – including on portable digital media, backup media, in logs, and data received from or stored by wireless networks. Technology solutions for this requirement may include strong one-way hash functions of the entire PAN, truncation, index tokens with securely stored pads, or strong cryptography.

3.5

Document and implement procedures to protect any keys used for encryption of cardholder data from disclosure and misuse.

3.6

Fully document and implement key management processes and procedures for cryptographic keys used for encryption of cardholder data.

3.7

Ensure that related security policies and operational procedures are documented, in use, and known to all affected parties.

	kernel 1			kernel 2		
filename	**start**	**end**	**size**	**start**	**end**	**size**
flashmgr.ram	65600	75519	9920	65600	75519	9920
_crypto.ram	75568	79603	4036	75568	79603	4036
schedulr.bin	79652	102399	22748	79652	102399	22748
iic.bin	102448	103727	1280	102448	103727	1280
clk_6900.bin	103776	109223	5448	103776	109223	5448
clk_3610.bin	109272	115511	6240	109272	115511	6240
_upgrade.tmp	115560	123391	7832	115560	123391	7832
_buf_mgr.bin	123440	124871	1432	123440	124871	1432
bpr.bin	124920	125487	568	124920	125487	568
com1_.bin	125536	129591	4056	125536	129591	4056
loadlist.sys	129640	131087	1448	129640	131087	1448
sysmode.out	131136	196623	65488	131136	196623	65488
pedgrdfa.out	196672	202767	6096	196672	202767	6096
ped.lib	202816	226495	23680	202816	226495	23680
_info.bin	226544	230211	3668	226544	230211	3668
console.bin	230260	262159	31900	230260	262159	31900
unzip.out	262208	270375	8168	262208	270375	8168
security.bin	270424	308235	37812	270424	308283	37860
pipe_mgr.bin	308284	327695	19412	308332	327695	19364
smipp.out	327744	332815	5072	327744	332815	5072
smresdl.out	332864	339983	7120	332864	339983	7120
icc_diag.out	340032	356999	16968	340032	356999	16968
icc1.bin	357048	393231	36184	357048	393231	36184
sys.lib	393280	436803	43524	393280	436735	43456
ipp.bin	436852	459855	23004	436784	459855	23072
usb_host.bin	459904	473155	13252	459904	473155	13252
cdma_usb.bin	473204	482215	9012	473204	482215	9012
usbd.bin	482264	491563	9300	482264	491563	9300
com2_.bin	491612	496347	4736	491612	496347	4736
com3_.bin	496396	524303	27908	496396	524303	27908
timezone.vlr	524352	529651	5300	524352	529651	5300
sm_text.vlr	529700	536195	6496	529700	536195	6496
dlerr.vlr	536244	536595	352	536244	536595	352
dlmsg.vlr	536644	537255	612	536644	537255	612
mag.bin	537304	545211	7908	537304	545211	7908
printer.bin	545260	564907	19648	545260	564907	19648
battery.bin	564956	572927	7972	564956	572927	7972

Table 1. File Locations in the Kernels (1)

	Data Element	Storage Permitted	Render Stored Data Unreadable per Requirement 3.4
Cardholder Data	Primary Account Number (PAN)	Yes	Yes
	Cardholder Name	Yes	No
	Service Code	Yes	No
	Expiration Date	Yes	No
Sensitive Authentication Data[1]	Full Track Data[2]	No	Cannot store per Requirement 3.2
	CAV2/CVC2/CVV2/CID[3]	No	Cannot store per Requirement 3.2
	PIN/PIN Block[4]	No	Cannot store per Requirement 3.2

1 Sensitive authentication data must not be stored after authorization (even if encrypted)
2 Full track data from the magnetic stripe, equivalent data on the chip, or elsewhere.
3 The three- or four-digit value printed on the front or back of a payment card
4 Personal Identification Number entered by cardholder during a transaction, and/or encrypted PIN block present within the transaction message

Table 2. Guidelines for Cardholder Data Elements (PCI 2018, p15)

