Corkow: Analysis of a business-oriented banking Trojan

By Robert LipovskyAnton Cherepanov

In his blog post last week, Graham Cluley introduced the Win32/Corkow banking Trojan. The malware, which has been in the wild since at least 2011, has demonstrated continuous activity in the past year, infecting thousands of users. Version numbering of the various Trojan modules is another indicator that the malware authors are continually developing the trojan.

The most common infection vector – drive-by downloads – has been used to spread the malware.

This Russian tool for committing bank fraud shares many characteristics with other malware families with a similar purpose, such as Zeus (also known as Zbot), Carberp, Hesperbot, or Qadars, for example, but also contains some unique functionality.

Several features, like enumeration of smart cards, targeting of dedicated banking applications mostly used by corporate customers and looking for user activity regarding online banking sites and applications, electronic trading platform sites and applications and so forth, all suggest that the attackers are focusing their sights on financial professionals and enterprises, whose bank accounts usually hold a higher balance than those of most individuals.

In this post, we expand on the information mentioned by Graham and provide additional technical details.

Analysis

As is the case with other banking Trojans (for example Win32/Spy.Hesperbot), the architecture of Win32/Corkow is comprised of a main module and several plug-in modules to deliver specific functionality. Each of Corkow's plug-in modules is implemented as a Dynamic Link Library (DLL). We will refer to the main component as the 'core DLL'. Most of the other plugins are embedded in the core module but some are downloaded from the C&C server. In either case, the core DLL will load and run these modules, injecting them into various processes in the system. Table 1 presents the different modules seen in all of the Win32/Corkow samples we have analyzed. Note that not all samples necessarily contain every module.

Table 1

Module	Description
Core	Main module responsible for injecting other modules into corresponding processes and for C&C communication. Also takes screenshots, enumerates smart cards and can block applications from running.
MON	Collects information about the system (list of running processes, user name, SID, last user input) and sends it to the C&C.
FG	Web-injections and form-grabbing module based on the leaked Zeus source-code. Corkow mainly uses the form-grabbing functionality to capture data.
KLG	Keylogger
HVNC	Hidden VNC connection that enables the attacker to connect remotely to the victim's machine.
PG	PuTTY logger for the `putty.exe` process. This is able to capture server logon credentials, which are valuable to cybercriminals.
PONY	Launches the "3rd party" universal password stealer Pony. ESET detects this trojan as Win32/PSW.Fareit.
IB2	Targets iBank2, a Russian banking application.
SBRF	Targets standalone Windows banking applications of Sberbank, the third-largest bank in Europe.
DC	Searches for finance-related text strings in browser history, installed and last used applications and running processes.

Table 1 - Description of analysed Win32/Corkow modules

While the core DLL is responsible for launching every module and for downloading configuration data from the C&C, each plug-in module contains the C&C URLs as well and uploads collected data directly.

As can be seen in the table above, Win32/Corkow contains functionality that one would expect from a typical banking trojan, including keystroke logging, screenshots and *HTTP* form-grabbing for intercepting log-in credentials to online banking. However, the last three listed modules have caught our attention. The trojan uses two dedicated modules to target Russian banking clients: one for iBank2, a widely-used banking application used by several banks, and one for Sberbank, Russia's largest bank. The module called 'DC' searches for indicators of user activity relating to various trading platform applications and sites, standalone banking applications, banking sites, Bitcoin sites, and software and Google Play developer activity. We'll describe these modules and the core module more in-depth in the following text.

But before we get to that, let's take a look at Corkow's installation procedure and its 'hardware-binding' technique.

Installation

Win32/Corkow features an interesting and relatively sophisticated installation procedure. The trojan is usually delivered to the victim by a dropper executable that contains the core DLL in its resources. When the dropper is run, the installation is carried out in the following steps:

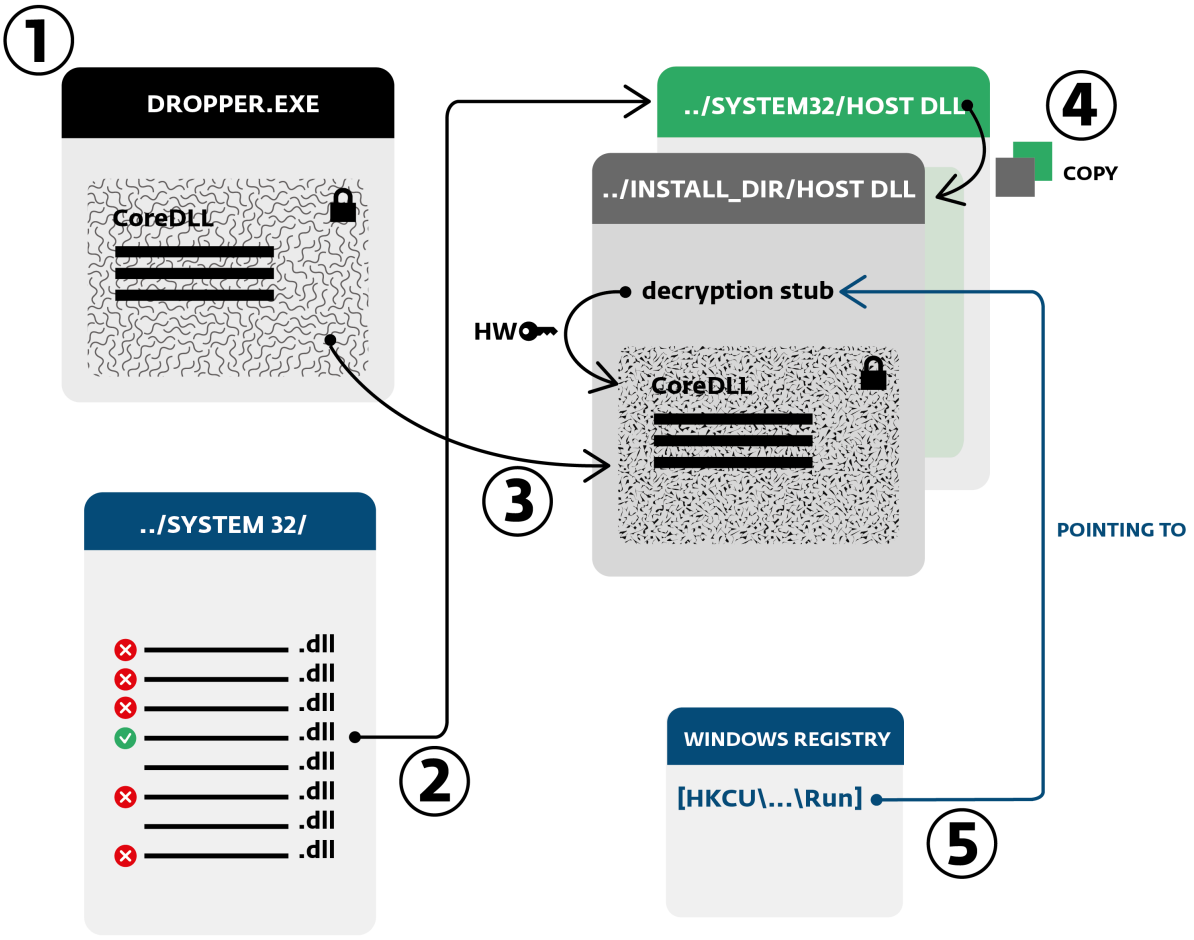

Figure 1 - Installation procedure of Win32/Corkow

- The dropper decrypts the embedded core DLL and calls its *DllMain* function, passing it one of three paths as a parameter. This path determines where and how the trojan should be installed. The chosen path depends on whether the trojan is being run under a standard user account or an administrator account. The possible values are listed in Table 2.

- When the code of the core DLL is running, it will seek a host file to infect. For this, Corkow will select a legitimate DLL file from the *%SystemRoot%\System32* directory that meets certain criteria (the file has to be unprotected; and some specific file names and DLL imports are excluded).

- Corkow will then infect the selected DLL file by encrypting itself and writing its encrypted body into the resources section of the host. A decryption stub is also written to the file and added as a new export function, so that the malware's body will be launchable after installation. The name of the export is also dependent on the installation path.

- The infected DLL is then saved to the installation path. Note that the host DLL file in the *System32* directory remains unchanged.

- A Registry entry is made to ensure the malware's persistence on the system. Again, the Registry key depends on the installation path and is listed in Table 2.

Table 2 - Possible Installation Paths

Path	Registry entry	DLL export
%CommonProgramFiles% \microsoft shared\DW*$random $*\	[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet \Services\LanmanServer\Parameters] ServiceDll = *$path_to_malware$*	ServiceMain
%AppData%\DAO*$random$*\	[HKEY_CURRENT_USER\Software\Classes\CLSID \{35CEC8A3-2BE6-11D2-8773-92E220524153} \InprocServer32] (Default) = *$path_to_malware$*	DllGetClassObjec t
%AppData%\Microsoft Corporation\	[HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows \CurrentVersion\Run] NvCplWow64 = *$path_to_rundll32.exe$ "$path_to_malware$"*, Control_RunDLL	Control_RunDLL

There are multiple ways to load a DLL and Win32/Corkow will use one of the three methods listed in the table above. Each of the methods loads a different DLL export, hence the different possible names for the decryption stub written into the host DLL during infection.

As stated above, the Corkow core DLL is written in the host DLL's resources in an encrypted form and also compressed using aPLib, a popular compression library. The encryption used is *XOR* with the encryption keystream generated by the multiply-with-carry algorithm and derived from the Volume Serial Number of the C:\ disk volume. This way, after installation the Corkow-infected-DLL is bound to the infected machine and will not run on a different computer. This is one way in which Corkow protects itself against malware analysis.

Core DLL, C&C communication

The main module of Win32/Corkow is responsible for extracting the other embedded modules and injecting them into corresponding processes, and for communication with the C&C server.

Corkow contains a list of URLs to which it tries to connect. The initial *HTTP* requests sent to the server contain some basic system information, the version numbers of individual modules and a generated bot ID. In this way the key for encrypting the communication (consisting of the C&C domain name and the bot ID) is established. The server will then respond with one of a few commands. The supported commands include:

- Reboot
- Download and execute arbitrary executable or DLL
- Update bot
- Download configuration for certain modules
- Wipe an arbitrary file on the system (by rewriting it with random data)
- Uninstall itself, with the option of destroying the system

The last two mentioned commands show that, apart from data theft, Win32/Corkow is also able to cause irreparable damage to the system. When the uninstall command is sent with a specific parameter, the trojan will attempt to delete critical system files and overwrite the Master Boot Record and Master File Table with random data, rendering the system unbootable.

The core DLL also contains the functionality to capture screenshots of the desktop, block specific applications from running and enumerate smartcards installed on the system.

The application-blocking functionality is determined by the bot's configuration. The trojan iterates through running processes (the standard method with CreateToolhelp32Snapshot is used) in an infinite loop, and if the undesired process name is found, attempts to terminate it. Of course, the chances of success for this method in User Mode are limited. It is most probably used to prevent victims from running banking applications (to check their account balances, and so forth).

Unlike other more sophisticated trojans, Corkow cannot interact with smartcards, only enumerate them. Interestingly, it doesn't even use common Windows API functions for interacting with smartcards, but instead enumerates all hardware devices (using the SetupDi API) and searches for specific device names.

Targeting dedicated banking applications

The way Corkow targets the iBank2 application is quite interesting. iBank2 is a Java application, so Corkow attempts to capture its data by injecting its own malicious Java class into the Java Virtual Machine running iBank2. To achieve this, the trojan first injects its IB2 module into each newly spawned Java process (*java.exe* or *javaw.exe*)

```
lea     eax, [ebp+nVMs]
lea     ecx, [ebp+pJavaVM]
push    eax
push    1
push    ecx
call    ebx ; JNI_GetCreatedJavaVMs
mov     esi, eax
test    esi, esi
jnz     short loc_10001206
mov     eax, [ebp+nVMs]
cmp     eax, 1
jge     short loc_10001217

                        ; CODE XREF: StartAddress+4C↑j
push    7D0h            ; dwMilliseconds
call    ds:Sleep
inc     edi
jmp     short loc_100011E7
-------------------------------------------------------------

                        ; CODE XREF: StartAddress+3A↑j
mov     eax, [ebp+nVMs]

                        ; CODE XREF: StartAddress+54↑j
test    esi, esi
jnz     loc_10001326
cmp     eax, 1
jl      loc_10001326
push    eax
push    offset Format   ; "%x"
lea     edx, [ebp+Dest]
push    63h             ; Count
push    edx             ; Dest
mov     [ebp+var_4], 2
call    ds:_snprintf
mov     eax, [ebp+pJavaVM]
add     esp, 10h
lea     edx, [ebp+pJNIEnv]
mov     ecx, [eax]
push    esi
push    edx
push    eax
call    dword ptr [ecx+10h] ; JavaVM->AttachCurrentThread
test    eax, eax
```

Figure 2 - Corkow code for attaching to a Java Virtual Machine

The injected code then uses Java Native Interface (JNI) functions to get the pointer to the running Java VM, attaches itself to it (Figure 2) and loads its malicious Java class inside the VM. Figure 3 shows part of the decompiled Java class. The class contains methods for getting the current balance of the victim's bank account and making screenshots, and is able to copy key files used to authenticate the user.

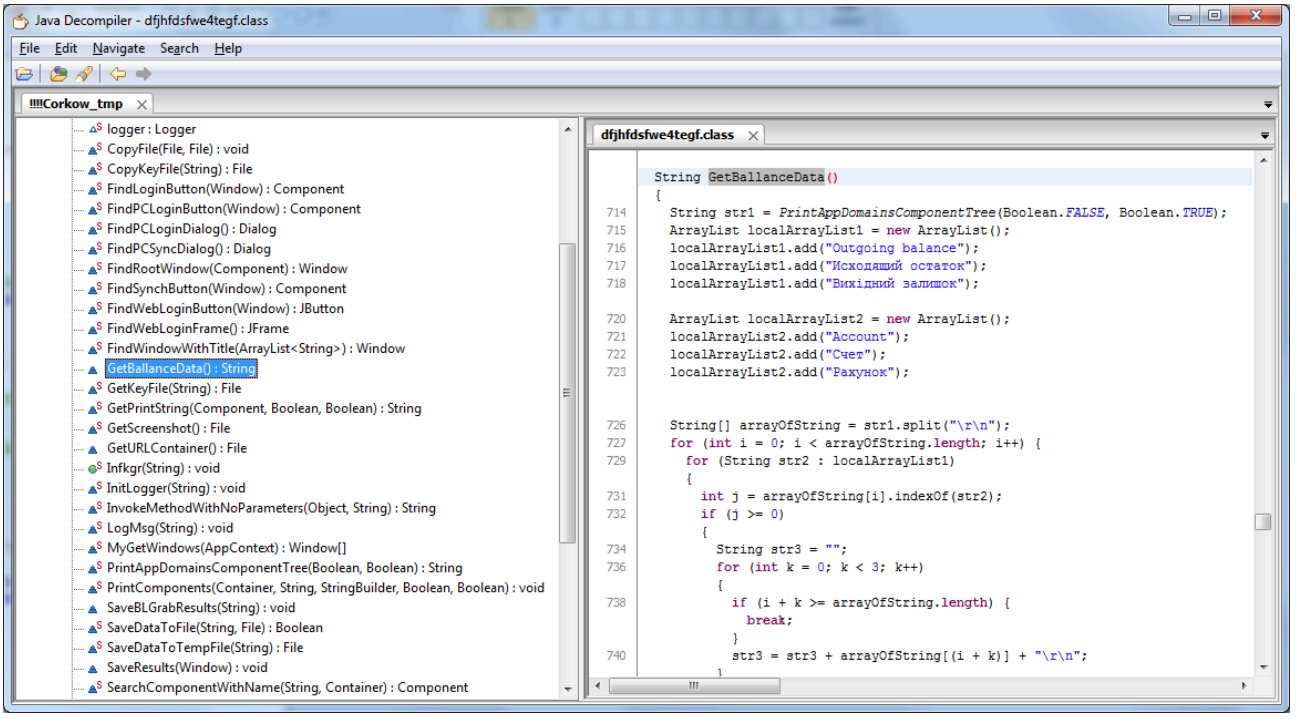

Figure 3 - Corkow's malicious Java class used against iBank2

Notice that the code supports English, Russian and Ukrainian versions of the iBank2 application.

The Java injection technique described does not exploit any vulnerability in the iBank2 application itself. Other banking trojans that have targeted the Java-based iBank2 platform include Win32/Spy.Ranbyus and Win32/Carberp, although different techniques were used in both those cases.

The SBRF module targets banking applications (Win32 platform) used by corporate customers of Sberbank. Like the iBank2 module, the SBRF module can create screenshots and copy key files for authentication.

DC module

This module scans for user activity by searching the following:

- Running processes
- Browser history – Corkow runs the 3rd party utility BrowsingHistoryView in order to read the history of Microsoft Internet Explorer, Mozilla Firefox, Google Chrome and Apple Safari. The Opera web browser history file is opened directly.
- Installed applications – by enumerating files in common installation directories
- Last-used applications – by enumerating the corresponding Registry entries

Interestingly, though, the module does not send the full results of the search to the remote server. Instead it parses the data and looks for specific finance-related strings from a defined list. The analyzed sample contained strings relating to banking, electronic trading platforms and stock brokerages, digital currencies (including various Bitcoin software and websites), various payment systems, and Google Play developer activity:

iBank2	First Ukrainian International Bank (ПУМБ)	Interactive Brokers
WebMoney Keeper	AKB Privatbank	Ameritrade
Yandex.Money	Zuger Kantonalbank	Schwab
Sberbank	Credit Suisse	E*Trade
Alfa-Bank	CIM Banque	Fidelity
Contact NG	HSBC	5trade
Western Union	Hypo Landesbank Vorarlberg	**Digital currencies**
Xpress Money	Loyal Bank	Liberty Reserve
Trans-Fast	Valartis Bank	BTC-e
MoneyGram	Danske Bank	Mt.Gox
Promsvyazbank Cyprus	Jyske Bank	BitStamp
Avangard Bank	BEC	50BTC
Hellenic Bank	Raiffeisen	Bitcoin-Qt
Russian Commercial Bank Cyprus	Forum Bank	MultiBit
Alpha Bank Cyprus	**Electronic trading platforms, stock brokerages**	Electrum
Bank of Cyprus	Finam Direct II	Bitcoin Armory
Cyprus Popular Bank (Laiki)	Blackwood Pro	Litecoin
DBS Bank	Scottrade	**Other**
United Overseas Bank	ScottradeELITE	Google Play developer activity
OCBC Bank	MBT Desktop Pro	PuTTY
ABLV Bank	QuoteTracker	WinSCP
Baltikums Bank	eSignal	LightSpeed
Norvik Bank	Ensign	QIWI payment system
Snoras Bank	TraderBytes	*various unidentified PoS systems*
Rietumu Bank	ROX	

Table 3 - Various finance-related software and websites referred to by Corkow's DC module

Apart from Russian and Ukrainian banks and software, the list also includes a wide range of banks based in Switzerland, Singapore, Latvia, Lithuania, Estonia, Denmark, Croatia, the United Kingdom, Austria and Cyprus (including some banks that are now defunct).

The bot will then notify the attacker if any of the above-mentioned strings are found on the victim's system.

Conclusion

Win32/Corkow is an example of the consequences of leaked source code from other banking trojans. During the analysis of the Corkow code, it proved fairly easy to spot various different programming styles and parts that were written by the malware authors themselves and other parts that were literally 'copy & pasted' from other banking trojans. While Corkow may be technically less sophisticated than some other malware that we've analyzed, it will get the job done.

Furthermore, the perpetrators operating the Corkow botnets apparently have a well-conceived modus operandi with a focus on corporate banking users. We can confirm that several thousand users, mostly in Russia and Ukraine, were victims of the Trojan in 2013.

We continue monitoring the threat and will keep you informed of further developments.

Thanks to Anton Cherepanov for his thorough analysis of this malware.

List of SHA1s

c08b899f0cbe26057e474396b829a8c69c4bcd31

5462f5b2ac221ed3f93828447c975c97e9690ef2

9024e81a45156736c9d5946620ab63be510c54ed

1f54b6624a1a93fe47631b7844acd2f02ab1d66a

73ea52373c0478103d2194f61ea0e179b7416ab9

cf0ec48d4294b8c288c4dd97e3db3967fecad554

c806c8d1774341db0e9f1cf9bfc309c1ec245689

4a06e4cb4838d78813306bac1cdcf982ec5c0e35

1ea1fa8b917a700c2be7edb963c0b193aaae6c7a

f3fbf41433757e6cbbfbe6f9c99929eeeadd5373

16d75b3135803a2d60962d9677e8b91fc34b4fb7

c43efc00cd459639b277690983afa6fb7abc91cc

ba03301d444da65116c08f0e3f897cc91a47ed4a

1cd4ec8ce834b97e1be4e215071b7cda4bb7d9c1

c5eda109e125bba20a27cd52e779d1106ece7762

982b06c53e37bc14d5fb7c515cbb479ad6fb1343

4e78bb4e3aea2a80184d99ea2a0d36ec811655ef

cc061159ef6284edc6d46cf45e756b9db1258a27

b2b78353b1fbef895922c47c41f4431781a14afa
