Qakbot, Data Thief Unmasked: Part I

web.archive.org/web/20110909041410/http://www.symantec.com/connect/blogs/qakbot-data-thief-unmasked-part-i

Updated: 11 May 2010

Shunichi Imano
Symantec Employee

+2 2 Votes
Login to vote

Motive
We recently had the opportunity to revisit a threat that first appeared on our radar back in May of this year. W32.Qakbot (hereafter referred to as Qakbot) is a somewhat benign worm that is capable of spreading through network shares, downloading additional files and opening a back door on the compromised computer, all in aid of its ultimate goal. Benign not because it is harmless - stealing login details, reporting keystrokes and uploading system certificates is malicious behavior indeed - but as will become obvious as we describe it in more detail below, because it moves slowly and with caution, trying not to bring attention to its presence.

The motive of Qakbot is quite clear, to steal information. Taking a peak under the proverbial covers, we see that it uses several components to accomplish the task, including the following:

- _qbot.dll
- _qbotinj.exe
- msadvapi32.dll
- _qbot.cb
- seclog.txt
- _qbotnti.exe
- sconnect.js
- webfix.txt

We will discuss each of these components briefly as we walk through the various functionality contained within and methods employed by this nefarious data thief.

Infection
Qakbot initially spreads via web pages containing Javascript which attempts to exploit certain vulnerabilities, including Microsoft Internet Explorer ADODB.Stream Object File Installation Weakness and Apple QuickTime RTSP URI Remote Buffer Overflow (Symantec IPS detection details here and here) and where those exploits are successful, downloads its malicious files on to the compromised computer. Once a machine is infected with Qakbot, all Qakbot-related files are stored in the user profile data directory, which typically is C:\Documents and Settings\[USERNAME]_qbothome. The first two components the threat downloads are _qbot.dll and _qbotinj.exe.

The downloaded file _qbot.dll is the main component of the Qakbot worm and is responsible for collecting certain information from the infected machine and uploading that stolen data to FTP servers under the control of the creator, the locations of which are frequently changed. We will talk more about this file later in the article.

Injection
The _qbotinj.exe file acts as a kind of servant to the _qbot.dll file. The file explorer.exe, a core Windows process and one of the few that runs in memory constantly on Windows operating systems, is compromised by _qbotinj.exe injecting _qbot.dll into it – that is, into the instance of explorer.exe running in memory. Similarly, the iexplore.exe process, which many readers will recognize as the process responsible for operating the Internet Explorer browser, is also injected.

This creates the illusion that all subsequent actions undertaken by the threat appear to be the work of these otherwise legitimate Windows processes. This is in fact a trick often used by many types of threats, as antivirus products, firewalls and other security safeguards are generally programmed to allow such common Windows processes full access to both the Internet and other applications on the infected computer.

The above image represents the worm communicating with its command center via the compromised Internet Explorer process. For all intents and purposes, this simply appears to be a legitimate browsing instance.

Interestingly however, _qbotinj.exe avoids injecting _qbot.dll into certain processes, presumably in an attempt to avoid being detected (or in some cases to avoid being debugged which would likely result in detection so in essence is the same thing), including the following:

- msdev.exe
- dbgview.exe
- mirc.exe

- ollydbg.exe
- ccApp.exe
- R&Q.exe
- photoed
- outlook.exe
- mmc.exe
- ctfmon.exe

Survival
_qbot.dll also runs two additional threads: “Watchdog” and “Swatcher”. The Watchdog thread monitors for instances of Dr. Watson running in memory and terminates any it finds . The other thread, Swatcher, checks the registry subkeys “Run” and “RunOnce” every 30 seconds and updates them if necessary. Qakbot does not add a registry value under the “Run” or “RunOnce” subkeys itself, but instead updates the last entry under those keys to include its own EXE file, followed by Qakbot’s parameter and “/c” with the original registry value, which is legitimate.

For example, the last (legitimate) Run key entry might be:
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\"Test Program" = "C:\Program Files\Test Program\testprogram.exe"

Qakbot will modify it to read:
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\"C:\Documents and Settings\All Users_qbothome_qbotinj.exe" " C:\Documents and Settings\All Users_qbothome_qbot.dll" /C "C:\Program Files\Test Program\testprogram.exe"

Qakbot contains several internal commands it uses to do its dirty work. One of them, “getip”, checks the registry to see if a Citrix product, Microsoft Office or Microsoft Project is installed, by reading the registry key HKEY_CLASSES_ROOT\Installer\Products. If installed, this command does nothing further. If none of those products are found, it continues, checking if it is running on a Virtual Machine. If a VM is discovered, Qakbot sends the VM information to [http://]hostrmeter.com/cgi-bin/exha[REMOVED] using the POST method, and checks for the existence of the file c:\irclog.txt. If c:\irclog.txt is found on the compromised computer, Qakbot uninstalls itself using its own "uninstall" command. We'll touch on this again in Part II.

Spread
Another of Qakbot’s internal commands, “nbscan”, is responsible for its attempts to spread over network shares. It enumerates network share folders, checking if the share name and user name are contained in the file "% CurrentFolder%\nbl_[USERNAME].txt". If they are listed in the file, Qakbot skips that network share. If they are not listed - and before copying any files to the remote share - Qakbot checks if the files "% CurrentFolder%_qbot[RANDOM CHARACTERS]" and "%CurrentFolder%\q1.dll" exist on the remote machine. If not, it downloads them. It then copies "q1.dll" to either [REMOTE COMPUTER]\C$\windows\q1.dll

or [REMOTE COMPUTER]\ADMIN$\q1.dll, and "_qbot[RANDOM CHARACTERS]" to either "[REMOTE COMPUTER]\C$\windows_qbot[RANDOM CHARACTERS].exe" or "[REMOTE COMPUTER]\ADMIN$_qbot [RANDOM CHARACTERS].exe". After copying the files, it writes the share name and user name to the file "%CurrentFolder%\nbl_[USERNAME].txt" stored on the local machine. This of course is the list Qakbot checked at the very beginning of the network share routine, the logic being that it if the names are in the list, that machine has already been processed (read infected).

The file _qbotnti.exe is used to load the file msadvapi.dll into memory. msadvapi.dll is then used to hook APIs in every running process. This serves two purposes. Firstly, as a cloaking mechanism. msadvapi.dll hides file names and registry entries containing “_qbot”, and also hides Internet connections to destination ports between 16666 and 16669. Secondly, it enables the threat to capture sockets where the destination port is 21 (FTP), 110 (POP3) or 143 (IMAP), as well as capturing login credentials to predefined web sites contained in the configuration file named webfix.txt. Qakbot attempts to steal login credentials when the user visits one of the hard-coded URLs listed in webfix.txt.

We’ll finish this two-part series in a subsequent posting where we’ll look briefly at the information Qakbot steals and how it goes about updating itself. Stay tuned.

A big thanks to Masaki Suenaga and Takayoshi Nakayama for their analysis.