Qakbot, Data Thief Unmasked: Part II

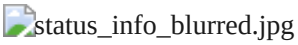
Archived: 2026-04-05 13:36:10 UTC

Theft

As we discussed in Part I, the primary purpose of Qakbot is to steal information from the compromised computer. In addition to targeting login details for FTP, POP3 and IMAP, the worm also attempts to steal Cookies - not only regular browser session cookies but also Flash cookies. A discussion of Flash cookies is beyond the scope of this article, but be aware that unlike traditional browser cookies, Flash cookies are not controlled through the cookie privacy controls in a browser which means they cannot be cleared or deleted in the simple manner that normal tracking cookies are removed.

Qakbot uses several techniques to collect private keys from the system certificates contained on the compromised computer. First, it replaces all certificate-related dialog boxes so that the “OK” button is automatically pushed as soon as the dialog is created. As a result, the user will never see the “OK” button. It also prevents all message boxes from being displayed. Secondly, it hooks password input windows in order to steal any characters entered. Thirdly, it patches the API CPExportKey to bypass security checks, and enumerates the private keys.

It also regularly sends out the geographical location and browser information of the compromised computer to a pre-defined URL as below. All traffic between the compromised computer and the Qakbot command & control servers is encrypted using SSL.


status_info_blurred.jpg

Update

Qakbot has functionality to download updates to itself in several different ways.

It downloads we.js as %System%\sconnect.js if 18000 seconds (5 hours) has passed since the last update. Upon successful download of sconnect.js, Qakbot adds a scheduled task to the compromised computer that executes sconnect.js as %Windir%\Tasks\[RANDOM NAME].job (defined as “instwd” in .cb files). The task itself is visible in the scheduled tasks window and is set to run every 4 days, despite the fact that the task actually renews itself every 5 hours. It then downloads q1.dll as %Temp%\msvcrt81.dll and q2l.jpg as %Temp%\drwatson.exe and instructs drwatson.exe to inject msvcrt81.dll into the running process iexplore.exe, effectively updating the instance of itself masquerading in memory as iexplore.exe.

Qakbot also downloads .cb files containing configuration information. The .cb files are encoded by Exclusive OR and Rotation of bytes and are decrypted by _qbot.dll.


Screenshot_qbot.cb_edited.dec_.PNG

As shown above, the .cb files contain a list of FTP sites that seclog.txt (which contains the stolen information) is to be uploaded to as well as username and password for each FTP site. Notably, if the configuration “cleanup” exists, Qakbot executes the command listed directly following the semicolon in the configuration "startup". It

creates the file "uninstall.tmp", which commands _qbotinj.exe to remove Qakbot from the compromised computer.

While we wouldn't exactly recommend it as an effective means of removing the threat from a compromised computer, it was interesting to note during our testing that simply creating a blank text (.txt) file in the Qakbot directory and renaming it to "uninstall.tmp" was enough to trigger the uninstall routine upon rebooting. Several seconds after restarting the computer, Qakbot deleted its own directory and all files therein. It then required a second reboot to remove the threat process in memory that was responsible for the file/folder removal, and although the scheduled task and registry entry remained, as there were no longer any files to call the threat was essentially neutralized.

Qakbot also regularly updates the .cb files with the status of the threat such as install time, run class (e.g. user, admin, win98) and several other pieces of information.

Defense

And there you have it. A brief look under the hood has shown how the W32.Qakbot worm uses existing, legitimate processes to hide behind resulting in an enhanced ability to bypass security checks as well as detection by the untrained eye. That it contains the mechanics to update itself, essentially allowing the author to reconfigure its functionality within the limits of its defining parameters as he or she sees fit. How it attempts to spread over network shares, and how it tries to maintain control of the system it has compromised while simultaneously attempting to avoid detection.

Network administrators will hopefully find some of the information useful in helping protect their environment, or in the worst case scenario at least aid in determining that machines on their network may be infected with Qakbot. Unexpected FTP traffic, scheduled tasks with "qbot" somewhere in the task name, machines madly attempting to enumerate network shares, registry keys containing the letters "qbot" (although note the default registry editor probably won't be useful here), Internet connections to destination ports between 16666 and 16669 (again, the local TCP table will be hidden so use some other means to monitor this) and other telltale signs should all be watched for.

It is obvious someone has gone to a good deal of effort to get their hands on certain aspects of our private information. The least we can do in return is defend ourselves to the best of our ability to prevent them from succeeding. Keeping our security software up-to-date is only one of the tools in our arsenal.

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