QNodeService: Node.js Trojan Spread via Covid-19 Lure

blog.trendmicro.com/trendlabs-security-intelligence/qnodeservice-node-js-trojan-spread-via-covid-19-lure/

June 9, 2020

QNodeService is a new, undetected malware sample written in Node.js, which is an unusual choice for malware authors. The malware has functionality that enables it to download/upload/execute files, steal credentials from Chrome/Firefox browsers, and perform file management, among other things.

By: Matthew Stewart June 09, 2020 Read time: (words)

We recently noticed a Twitter post by MalwareHunterTeam that showed a Java downloader with a low detection rate. Its name, "Company PLP_Tax relief due to Covid-19 outbreak CI+PL.jar", suggests it may have been used in a Covid-19-themed phishing campaign. Running this file led to the download of a new, undetected malware sample written in Node.js; this trojan is dubbed as "QNodeService".

The use of Node.js is an unusual choice for malware authors writing commodity malware, as it is primarily designed for web server development, and would not be pre-installed on machines likely to be targeted. However, the use of an uncommon platform may have helped evade detection by antivirus software.

The malware has functionality that enables it to download/upload/execute files, steal credentials from Chrome/Firefox browsers, and perform file management, among other things. It targets Windows systems, but its design and certain pieces of code suggest cross-platform compatibility may be a future goal.

The infection begins with a Java downloader which, in addition to downloading Node.js, downloads the following files: "wizard.js", and "qnodejs-win32-ia32.js" or "qnodejs-win32-x64.js". We analyzed these components to learn more about their behavior.

Analysis of Java Downloader

The sample mentioned above, "Company PLP_Tax relief due to Covid-19 outbreak CI+PL.jar", serving as the Java downloader, has been obfuscated with the Allatori obfuscator. Allatori adds junk code and obfuscates strings to make analysis more difficult. We deobfuscated the code to be able to start the analysis.

Decompiled code of the sample (obfuscated with Allatori obfuscator) Figure 1. Decompiled code of the sample (obfuscated with Allatori obfuscator)
Deobfuscated code of the sample Figure 2. Deobfuscated code of the sample
It downloads Node.js to the User Profile directory. It checks the system architecture and downloads the 32-bit or 64-bit version appropriately.

ode snippet for downloading Node.js to the user directory Figure 3. Code snippet for downloading Node.js to the user directory
It also downloads a file named "wizard.js" from the URL hxxps://central.qhub.qua.one/scripts/wizard.js. It then runs this file using Node.js with multiple command line arguments, including the URL of the C&C server:

wizard.js being run by Node.js Figure 4. wizard.js being run by Node.js
Note that "–group user:476@qhub-subscription[…]" is a parameter used during communication with the C&C server; the presence of a user identifier and mention of a "subscription" suggest that this malware may be sold as a subscription service.

Analysis of wizard.js

The wizard.js file is an obfuscated Javascript (Node.js) file. It is responsible for persistence (by creating a "Run" registry key entry) and for downloading another payload depending on the system architecture.

It creates a file named "qnodejs-<8 digit hex number>.cmd" which contains the arguments used to launch the file. This is invoked by the registry key entry it creates at "HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run". It also checks whether it's running on a Windows platform, suggesting the authors may have cross-platform compatibility in mind.

Figure 5. wizard.js checks if it's running on windows, and installs the Run registry key entry if so Figure 5. wizard.js checks if it's running on windows, and installs the Run registry key entry if so
Figure 6. The registry "Run" key entry added by wizard.js Figure 6. The registry "Run" key entry added by wizard.js
It downloads a file from hxxps://central.qhub.qua.one/scripts/qnodejs-<platform>-<arch>.js. Based on possible values for process.platform and process.arch in Node.js, we found files qnodejs-win32-ia32.js and qnodejs-win32-x64.js hosted on the server. The server also contains SHA1 hashes for each sample, although they are named .sha256. These hashes are downloaded and checked by the downloaded sample when it runs.

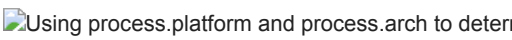


Figure 7. Using process.platform and process.arch to determine the payload to download

Analysis of qnodejs-win32-<architecture>.js

A file named qnodejs-win32-ia32.js or qnodejs-win32-x64.js is downloaded based on the system architecture (whether the OS is 64-bit or 32-bit).

The files contain an embedded “node_modules” folder with libraries for Node.js, which is extracted upon execution. Unlike the Javascript code itself, these libraries are architecture-specific, which is the reason separate files are distributed based on system architecture. Screenshots below are based on the “win32-ia32” variant from 2020-04-30.

We named this malware “QNodeService,” since this seems to be the name used internally, as indicated by the code that validates command line arguments.


Figure 8. The name “QNodeService” used internally in the code

The malware is divided into modules. Access to these modules is obfuscated with the use of a lookup function named “v”. Some modules consist solely of a “require” call to import libraries, while others are custom modules written by the author.


Figure 9. Modules used by the malware

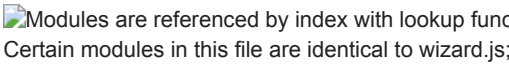


Figure 10. Modules are referenced by index with lookup function “v”

Certain modules in this file are identical to wizard.js; in particular, these reuse the code that determines the URL for the sample and its hash. In this file, the module is used to download and verify the SHA1 hash. If the hash fails, the malware terminates.

The malware uses the socket.io library for communication with the C&C server. As a result, it is designed with a reactive programming paradigm, and uses WebSocket to communicate with the server.

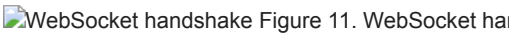


Figure 11. WebSocket handshake

The malware can steal passwords from Chrome and Firefox.

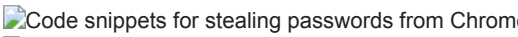


Figure 12. Code snippets for stealing passwords from Chrome

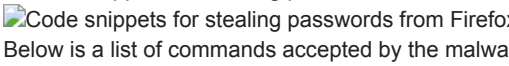


Figure 13. Code snippets for stealing passwords from Firefox

Below is a list of commands accepted by the malware:

Command	Description
control/reload	Signals wizard.js to redownload the main payload.
control/uninstall	Signals wizard.js to remove the Run key entry from the system and terminate.
info/get-ip-address	Gets IP address, location, hostname, etc.
info/get-label	Returns label set by the “set-label” command
info/get-machine-uuid	Gets a UUID generated by the malware.
info/get-os-name	Gets the platform (windows) and architecture (x32/x64) of the system.
info/get-user-home	Gets the user profile directory (os.homedir())
info/set-label	Sets label
file-manager/absolute	Gets the full path of a file
file-manager/execute	Executes a file with the command ‘start “” /B <file>’
file-manager/delete	Removes a file or files on the system (accepts globs, using “rimraf” library)
file-manager/forward-access	Generates URL and token to use for the “http-forward” command (see below)
file-manager/list	Lists files in specific directories
file-manager/mkdirs	Creates a directory on the system
file-manager/write	Writes a file sent from the C&C SERVER onto the system
http-forward	HTTP requests sent to a specific URL at the C&C are routed to the infected machine (see below)
password-recovery/applications	Lists applications for which passwords can be read (Chrome & Firefox)
password-recovery/recover	Recovers passwords from a specific application (Chrome or Firefox)

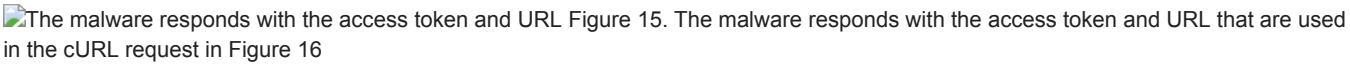
On May 5th, the malware was updated with three additional commands:

info/get-tags	Returns list of tags set by “add-tag” command
info/add-tag	Adds tags
info/remove-tag	Removes tags

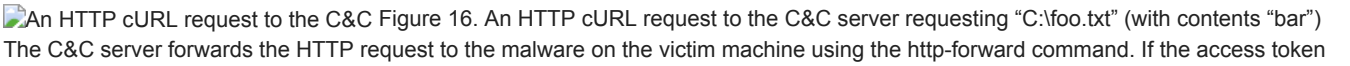
Of particular note is the http-forward command, which allows an attacker to download a file without directly connecting to the victim machine, as shown below in figures 13-16. However, a valid request path and access token are required to access files on the machine. The C&C server must first send “file-manager/forward-access” to generate the URL and access token to use for the http-forward command later

.

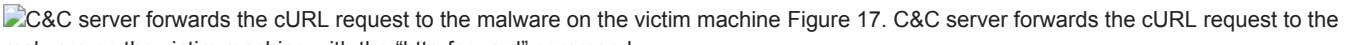

C&C server sends “file-manager/forward-access” command over WebSocket Figure 14. C&C server sends “file-manager/forward-access” command over WebSocket


The malware responds with the access token and URL Figure 15. The malware responds with the access token and URL that are used in the cURL request in Figure 16

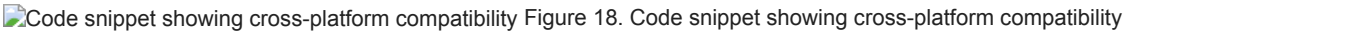
The malware responds with the forwarding URL and access token. Then, a third party who has been given the URL and access token could send an HTTP request to the C&C server to retrieve files from the victim machine without directly connecting to it.


An HTTP cURL request to the C&C Figure 16. An HTTP cURL request to the C&C server requesting “C:\foo.txt” (with contents “bar”)

The C&C server forwards the HTTP request to the malware on the victim machine using the http-forward command. If the access token is correct, the malware sends the file’s contents back to the C&C server, which in turn sends the contents back to the attacker in its HTTP response, enabling remote download.


C&C server forwards the cURL request to the malware on the victim machine Figure 17. C&C server forwards the cURL request to the malware on the victim machine with the “http-forward” command

Similar to wizard.js, the authors seem to have cross-platform compatibility in mind. Although this sample is the win32-ia32 variant, it contains code that would improve compatibility on Darwin (MacOS) and Linux platforms.


Code snippet showing cross-platform compatibility Figure 18. Code snippet showing cross-platform compatibility

Recommendations

Threat actors constantly come up with ingenious ways to create malware and ensure that it affects as many systems for as long as it can, such as using environments that are less utilized for malware creation, maintaining persistence, and giving them cross-platform compatibility. To defend against such malware, users can block them from getting through possible entry points, such as email, endpoints, and network, through the following security solutions:

- For email – Trend Micro™ Email Security offers AI-based detection and sandboxing capabilities to detect and block malware and malicious URLs
- For endpoints – Trend Micro Apex One provides pre-execution and runtime machine learning and automated threat detection
- For networks – Trend Micro TippingPoint Threat Protection System inspects and blocks network traffic in realtime to stop the infiltration of threats
- Indicators of Compromise

File name	SHA-256	Detection Name
Java downloader	5210AFA4567B98FB3F8AEE513206B5FD466D3AFE01DD576A2BEE4A623F2CDAE2	Trojan.Java.QNODESERVICE.A
wizard.js (2020-04-30)	9FBAFF43A596921EFD7BB3B015A541A00633320C3DE66BE795BADA098D37F8FE	Trojan.JS.QNODESERVICE.A
qnodejs-win32-ia32.js (2020-04-30)	EB00CD731EE622EAF53BFD19A789E494872BACA156455C38CA3035B2E33CC152	Backdoor.JS.QNODESERVICE.A
qnodejs-win32-x64.js (2020-04-30)	F3C5F8EF9886DC300BCE3E6DB0B973B3408AE82EB5789C4BA72FEC27D61CA693	Backdoor.JS.QNODESERVICE.A

wizard.js (2020-05-05)	5CCED1119F4FDC175967594EC4671EF74E645D46F5F7ED1200513C7EA7DC31CF	Trojan.JS.QNODESERVICE.B
qnodejs-win32-ia32.js (2020-05-05)	76B8E43AB3E38B8635588FBD9C9A527022691962DD158A480671DDF98C7110F8	Backdoor.JS.QNODESERVICE.B
qnodejs-win32-x64.js (2020-05-05)	16376D225C3B16E6E0D50259241939DE6AD19A82668F650AACDAF173576C5003	Backdoor.JS.QNODESERVICE.B

- **C&C SERVER**
- central[.]qhub[.]qua[.]one

Tags

Malware