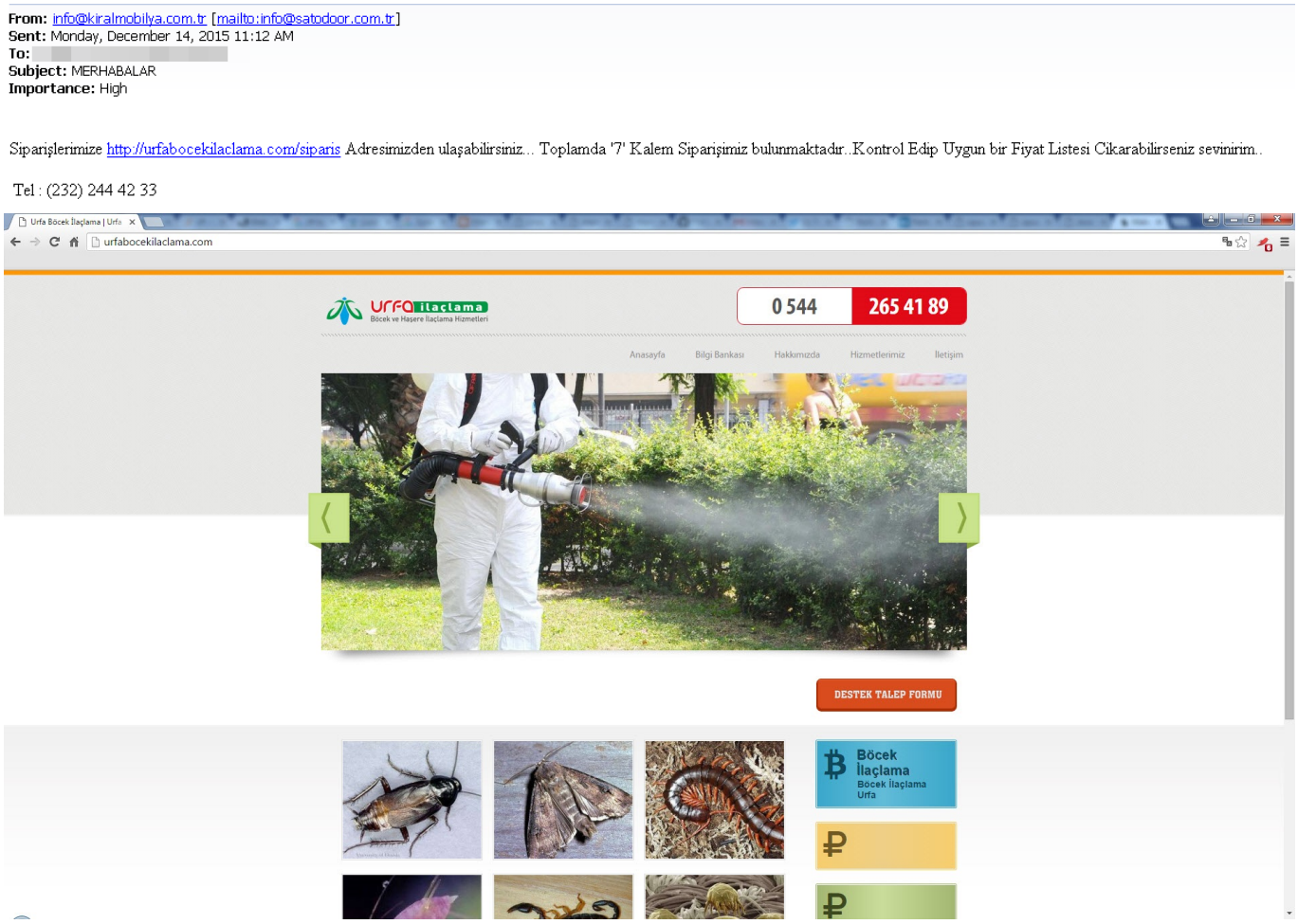
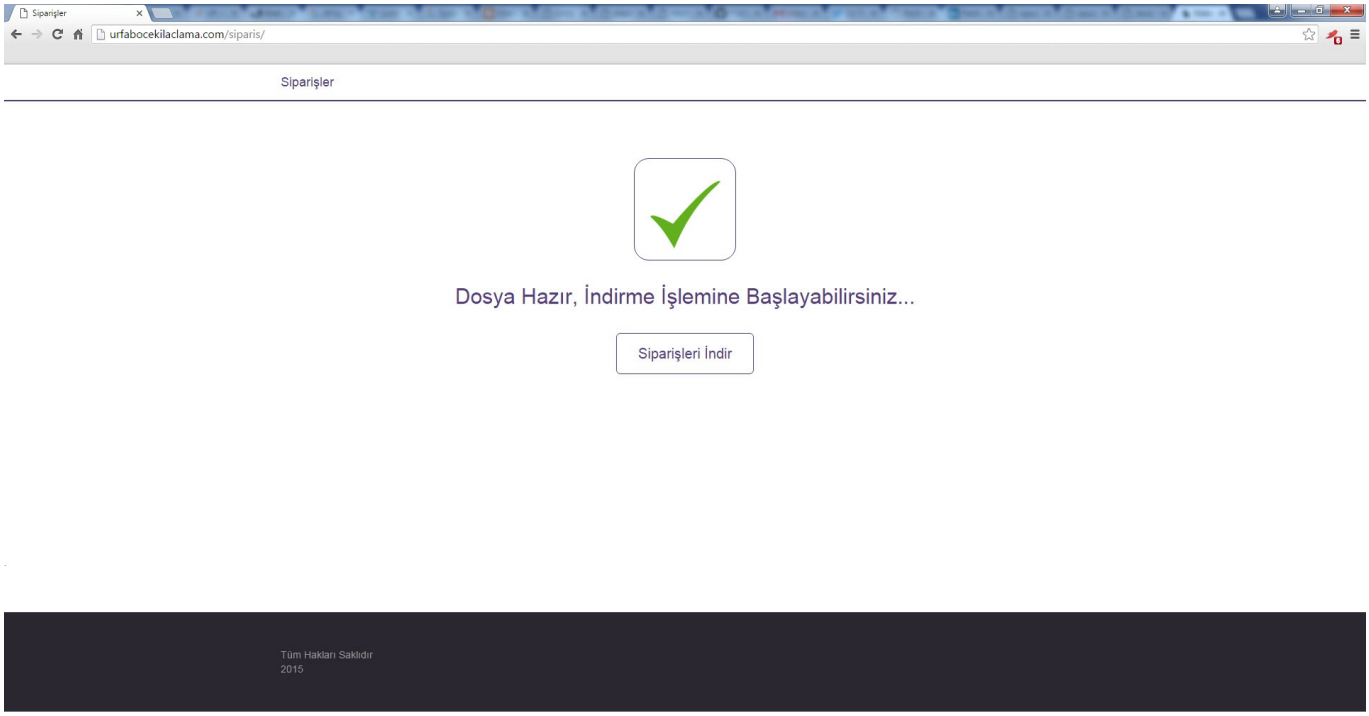
Java Kaynak Kodu Dönüştücüleri

written by Mert SARICA | 1 March 2016

Bayt kodu seviyesinde çalışmanın kimi zaman zorlayıcı olduğuna hak veriyorum. Mevzu bahis bir Java zararlı yazılımını analiz etmek olduğunda, eldeki class dosyalarını Java kaynak koduna çevirmek, çoğu analistin izleyeceği adımların başında yer alıyor. Ancak Temmuz ayındaki yazımda (Java Bayt Kod Hata Ayıklaması) da bahsettiğim üzere, gizleme (obfuscator) aracından (Allatori gibi) faydalanan bir zararlı yazılım ile karşı karşıya kaldığınızda, kaynak koduna dönüştürücüleri (JD gibi) çoğu zaman sizi yarı yolda bırakabiliyor.

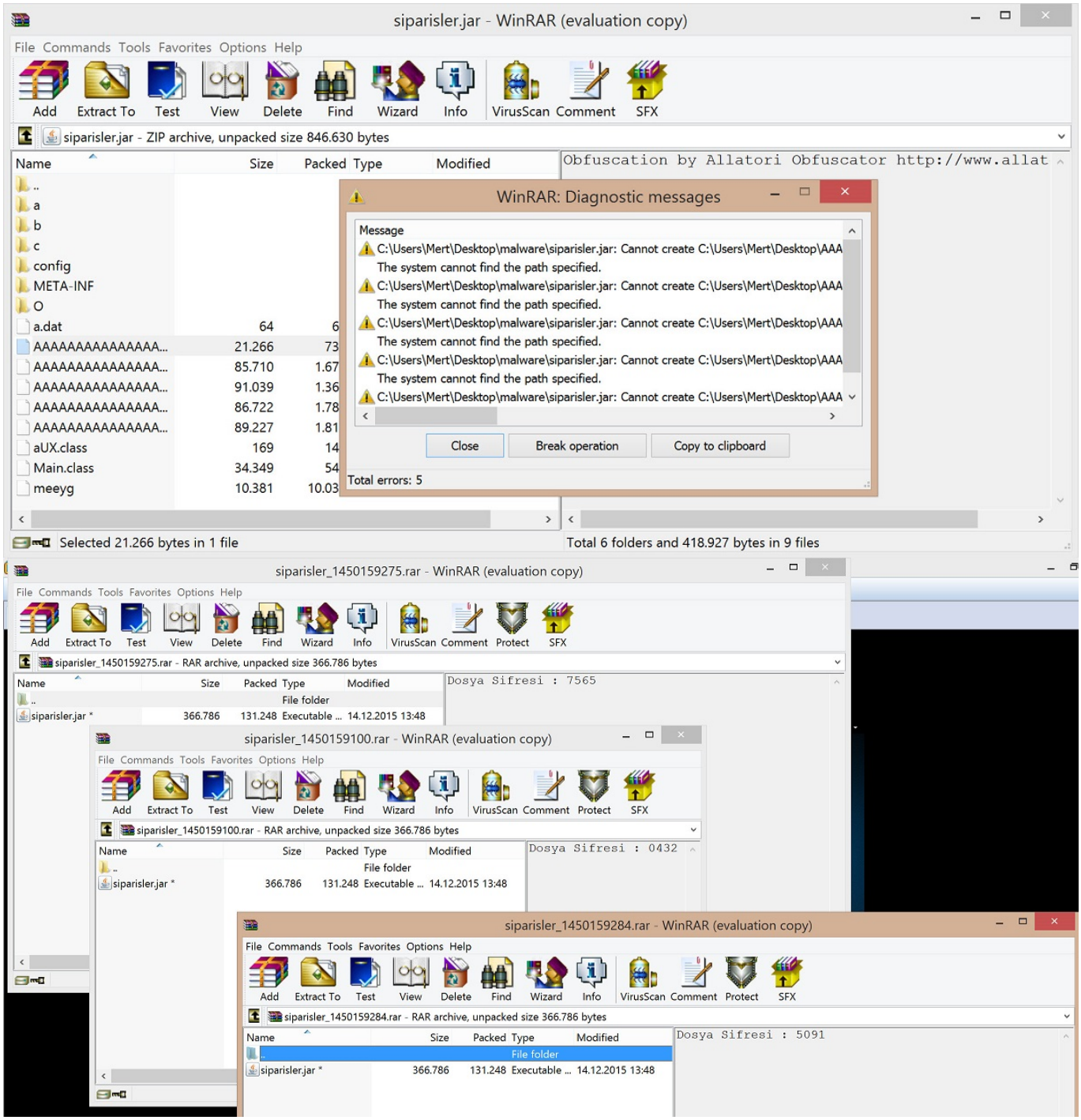
Aralık ayı gibi, sahte bir e-posta ile hacklenmiş olduğunu düşündüğüm bir web sitesi üzerinden indirilmesi sağlanan ve her indirilme isteğinde, farklı bir şifre ile paketlenen siparisler.rar (siparisler.jar) adında bir Java zararlı yazılımı dikkatimi çekti.





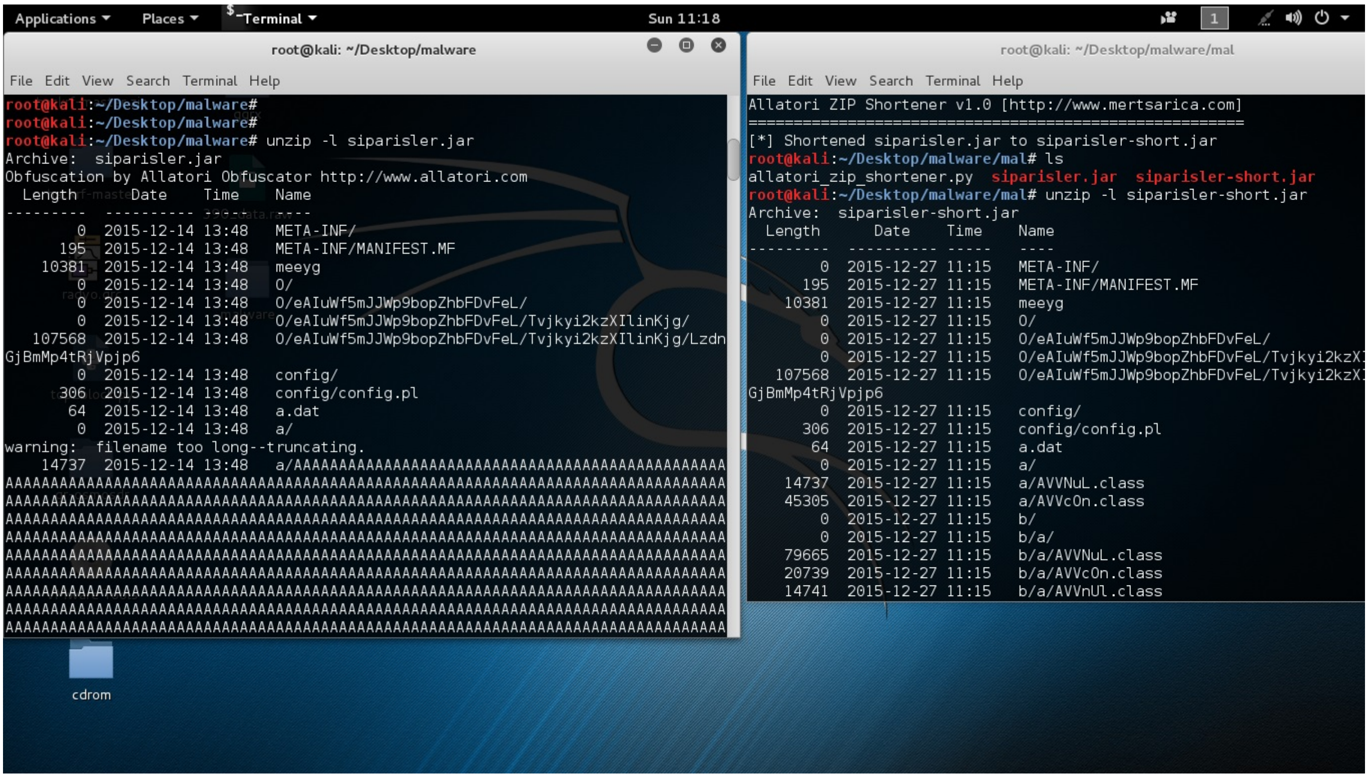
Allatori gizleme aracının güçlü özelliklerinden (uzun sınıf ve metod isimleri, AUX gibi rezerve isimler vs.) faydalanarak oluşturulan bu JAR paketi içinde yer alan class dosyalarına baktığımda, dosya isimlerinin ~8000 hane uzunluğunda olduğunu gördüm.

Bu uzun dosya isimleri sayesinde zararlı yazılımı Winrar, 7zip, unzip gibi araçlarla işletim sistemine açmak (extract) istediğimde, işletim sistemi sınırlarına takıldığımı ve dosyaları açamadığımı farkettim. Ayrıca uzun dosya isimleri ve metod isimleri nedeniyle çoğu kaynak kod dönüştürücüsünün (CFR hariç) bu dosyayı kaynak koduna çevirme işlemi esnasında hata aldığını gördüm.

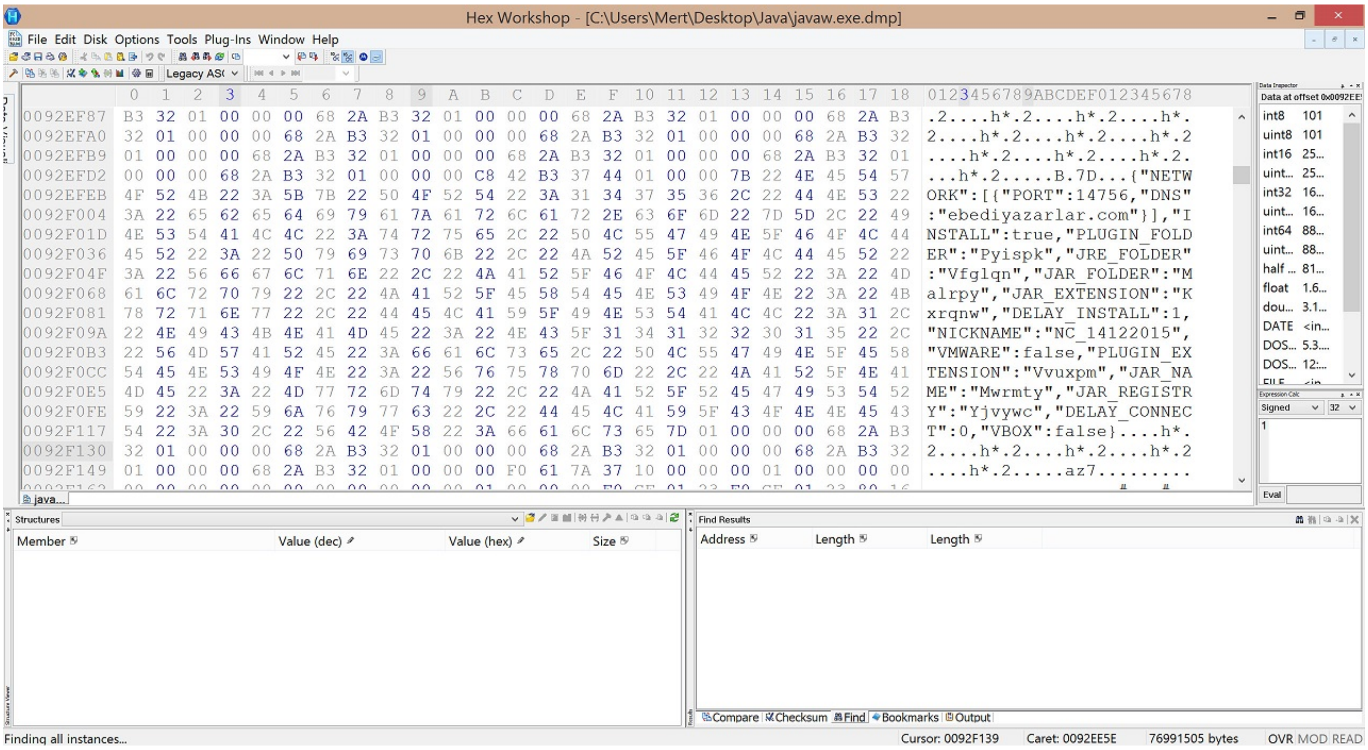


Bu dosyanın, kum havuzu analizi yapan ticari bir ürünü analiz esnasında çökerttiğine şahit olmuş biri olarak, sadece ve sadece cihazlara yatırım yapan ve bel bağlayan kurumların pamuk ipliğinde yaşadıklarını yeri gelmişken tekrar söylemiş olayım.

Tabii Python ile Allatori Zip Shortener gibi basit bir araç yazarak bu zip dosyasını açılabilir hale getirmem çok zor olmadı.



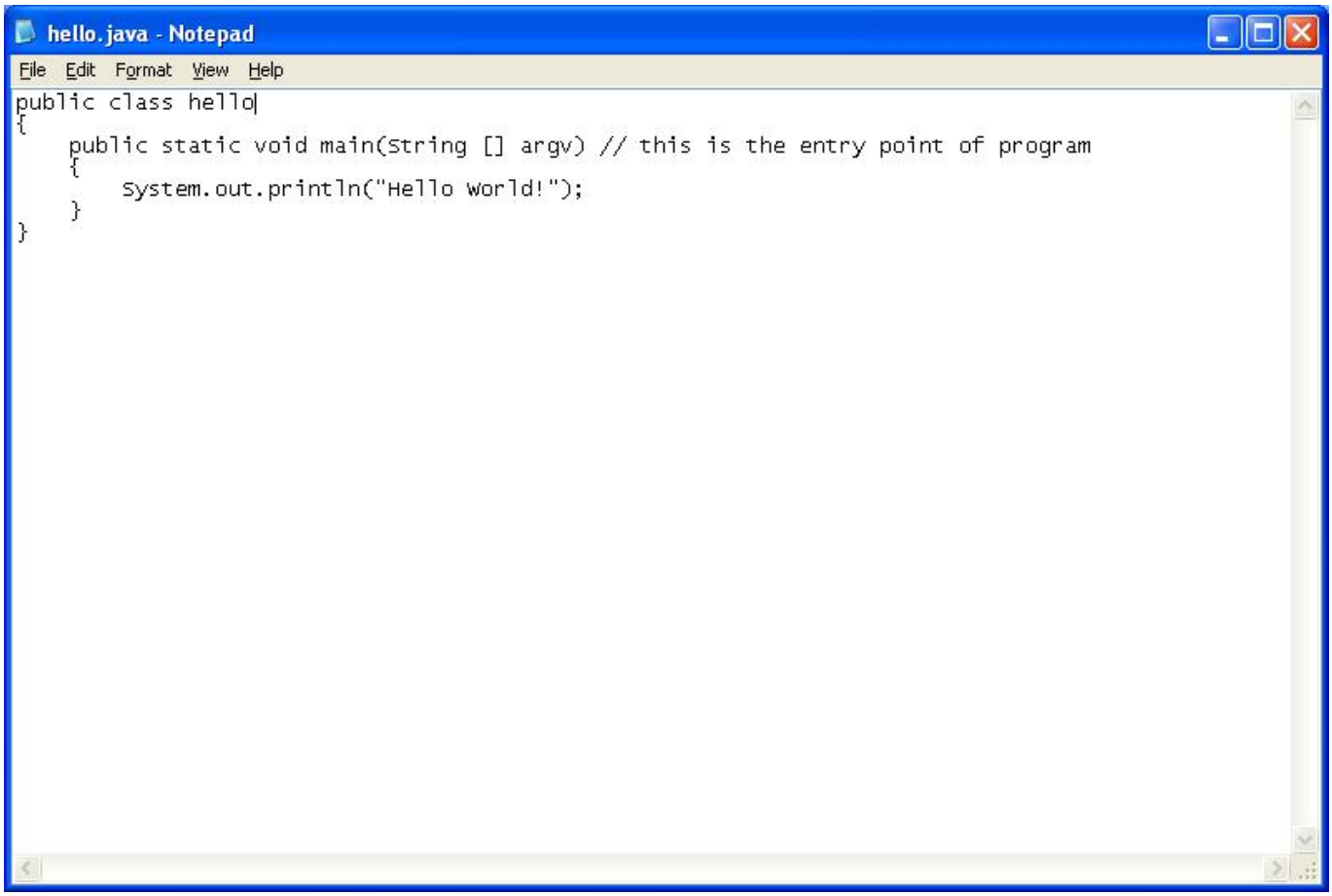
Daha detaylı bilgi edinmek için Temmuz ayında yayımlamış olduğum yazıda olduğu gibi bundan sonra bayt kod seviyesinde analize devam edebilirdim. Eğer amacım sadece bunun hangi türde bir zararlı yazılım olduğunu öğrenmek olsaydı, zahmete girmeden bellek dosyası üzerinde yapacağım ufak bir araştırma ile bunun Jsocket RAT yazılımı olduğunu da öğrenebilirdim.



Bu zararlı yazılımı incelerken, http://www.javadecompilers.com/ sitesi sayesinde çoğu kaynak kod dönüştürücüsünün Allatori’ye karşı yetersiz

olduğunu gördüm. Çoğu dönüştürücü, bu zararlı yazılımı ya kaynak koda çevirmekte başarsız oldu ya da çevirdiği kaynak kodu, tekrar derlenemeyecek durumdaydı. Bayt kodu seviyesinde statik analiz ile ilerlemek isteseydim, Allatori'nin karakter dizilerini gizlediğini görecek ve bunu çözmek için gizleme yöntemini bulmam gerekecekti ve bu bayt kod seviyesinde işlerimi biraz daha uzatacaktı. Ben de bu vesileyle, Allatori'ye karşı mevcut kod dönüştürücüleri değerlendirmeye ve karakter dizi gizlemesi için kullanılan algoritmayı hangi dönüştürücünün başarıyla ortaya çıkartabildiğini öğrenmeye karar verdim. Bunun için de başarı kriteri olarak, kaynak koduna çevrilen class dosyasının tekrar derlenebilmesini ve çalıştırılabilmesini kabul ettim.

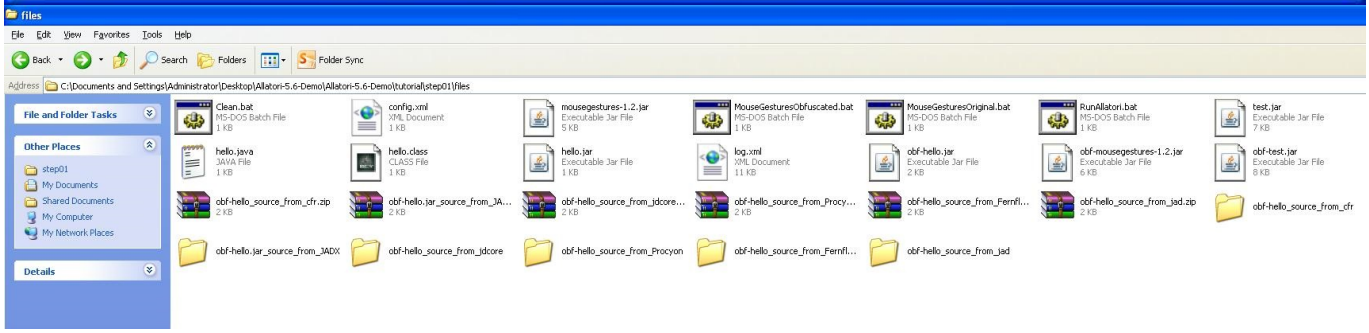
İlk olarak Java ile komut satırına "Hello World" yazan basit bir kod yazdım ve bunu JAR paketine çevirdim. Ardından bu paketi Allatori'ye vererek gizlenmiş (obfuscated) paket oluşturmasını sağladım. Son olarak da http://www.javadecompilers.com/ sitesi üzerinden her bir JAR dosyasını kaynak koduna çevirip derlemeye ve çalıştırmaya başladım.

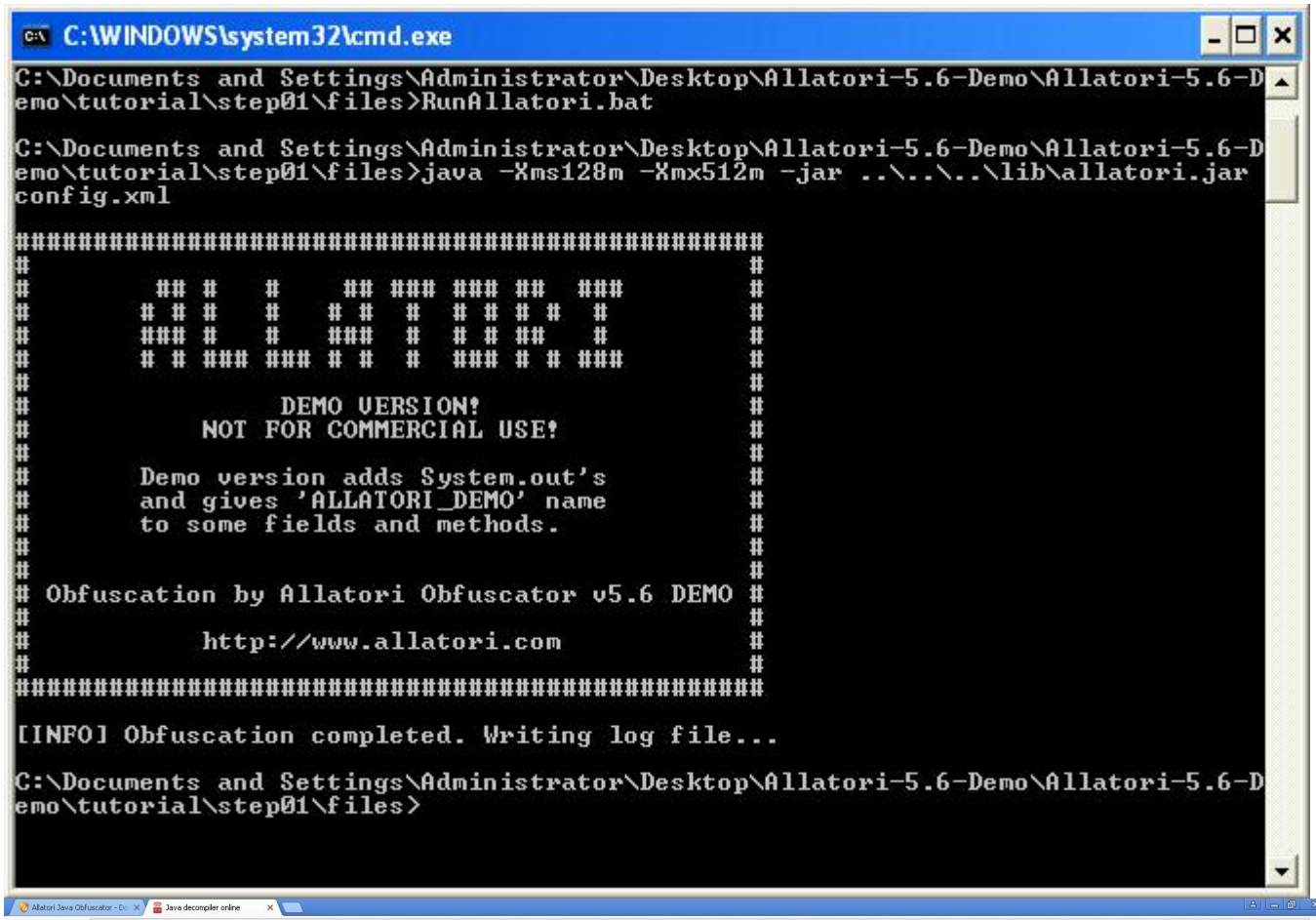


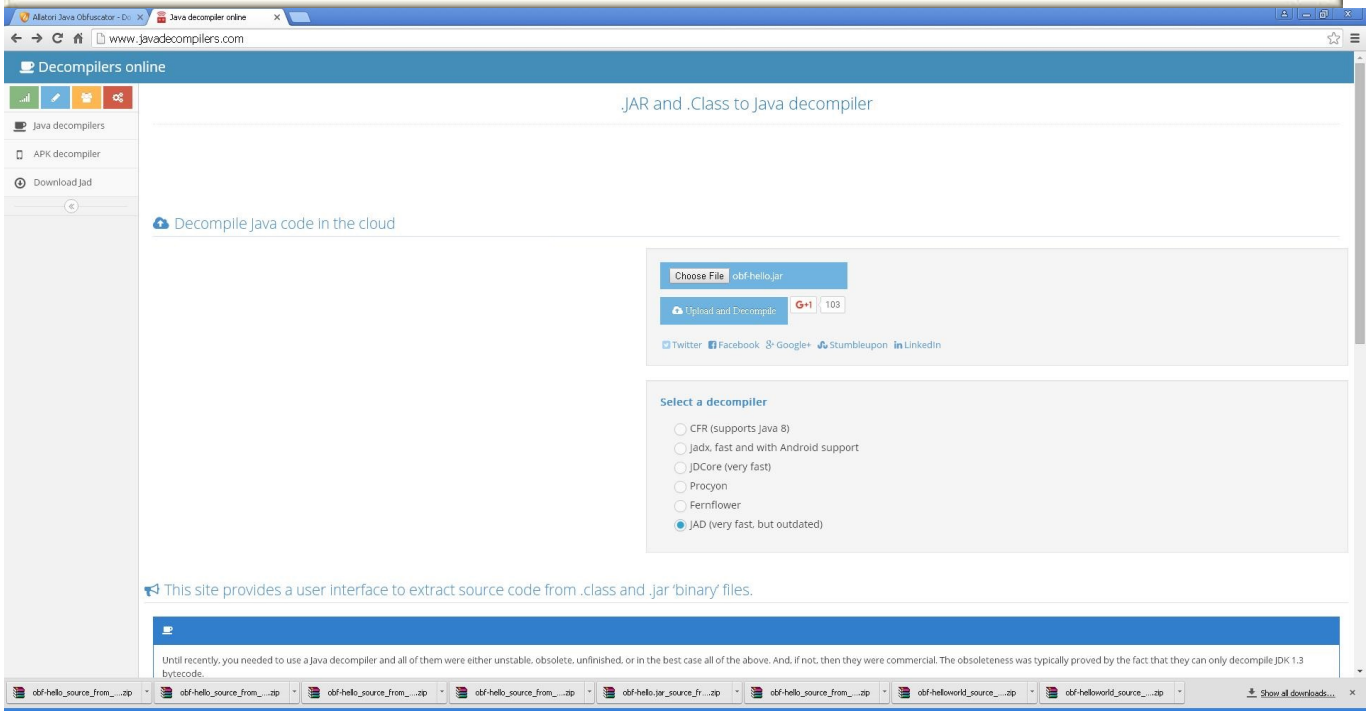
config.xml - Notepad

File Edit Format View Help

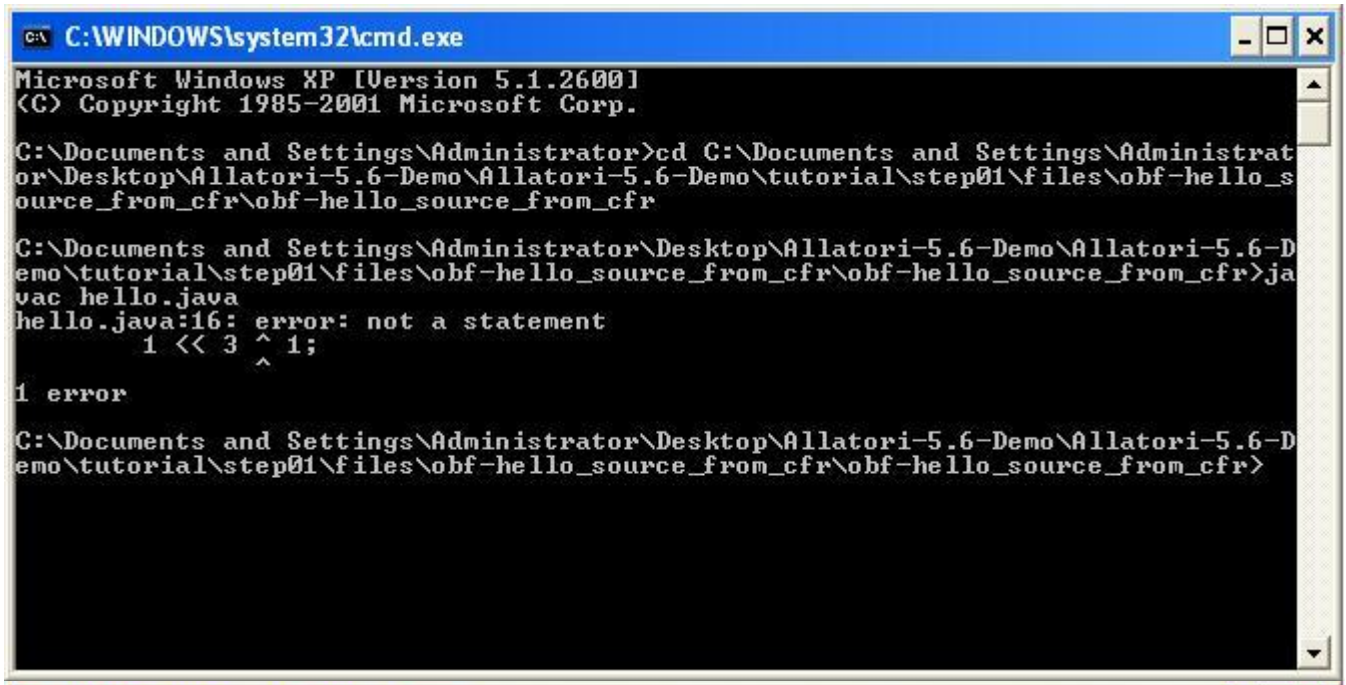
```
<config>
    <input>
        <jar in="hello.jar" out="obf-hello.jar"/>
    </input>

    <keep-names>
        <class access="protected+">
            <field access="protected+"/>
            <method access="protected+"/>
        </class>
    </keep-names>

    <!-- String encryption -->
    <property name="string-encryption" value="maximum"/>
    <property name="string-encryption-type" value="strong"/>
    <property name="string-encryption-version" value="v4"/>
    <!-- <property name="string-encryption-ignored-strings" value="patterns.txt"/> -->

    <!-- Control flow obfuscation
    <property name="control-flow-obfuscation" value="enable"/>
    <property name="extensive-flow-obfuscation" value="maximum"/>
    -->

    <property name="log-file" value="log.xml"/>
</config>
```

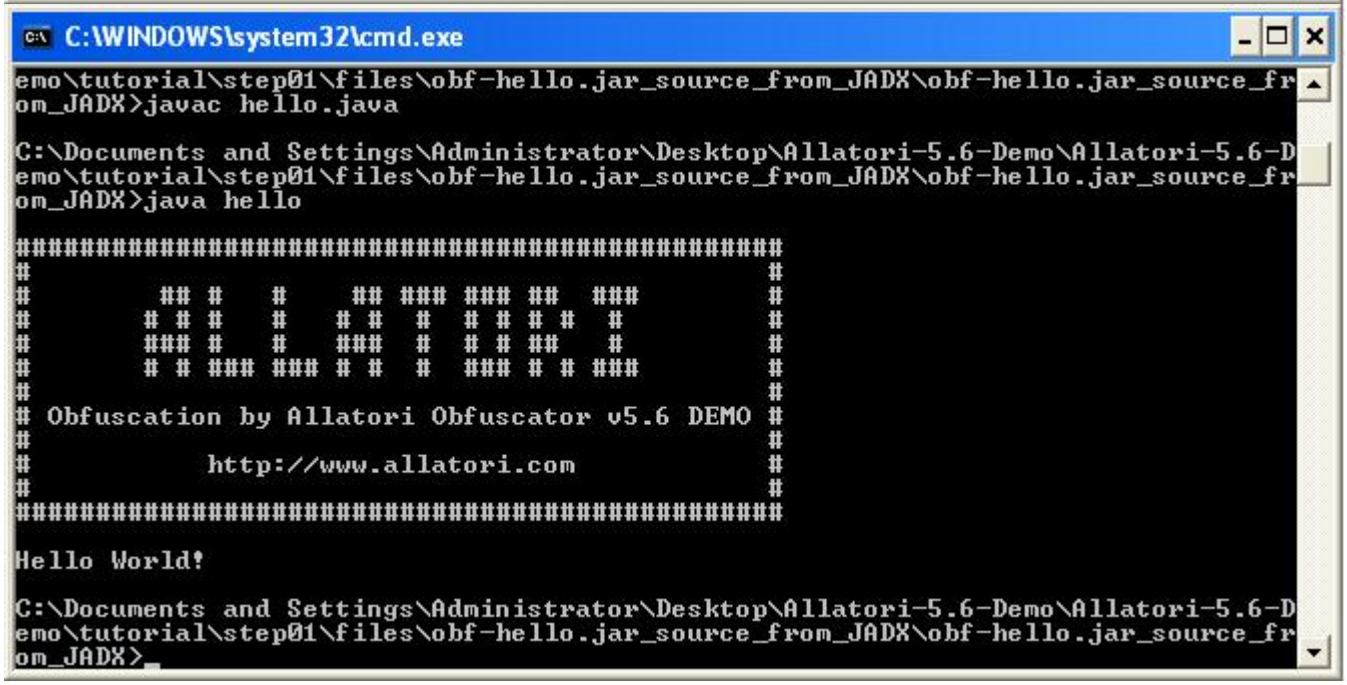


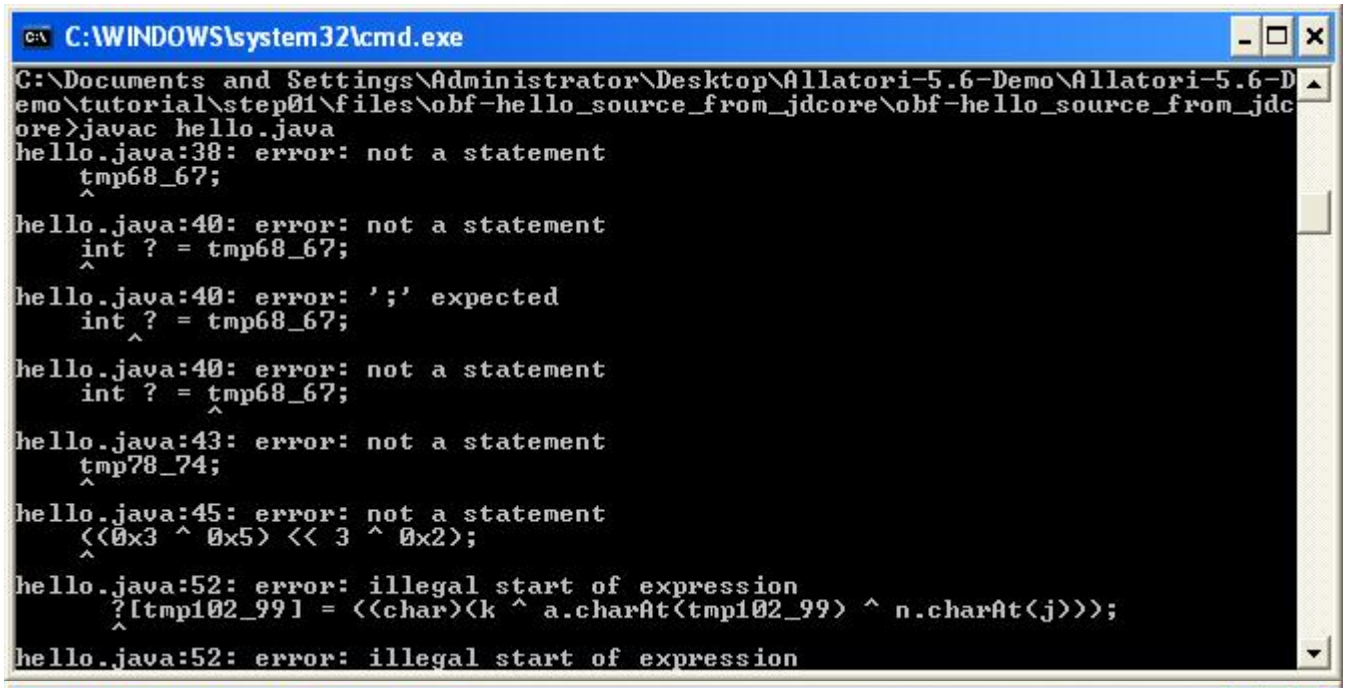


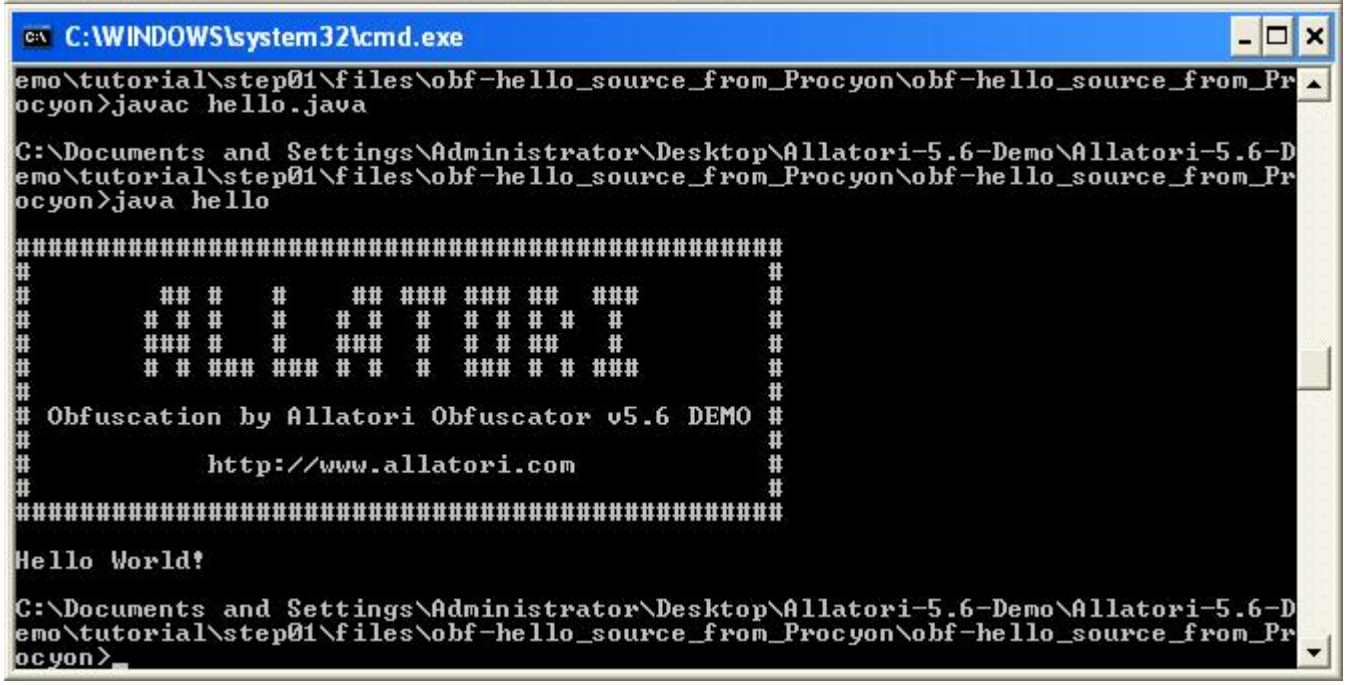


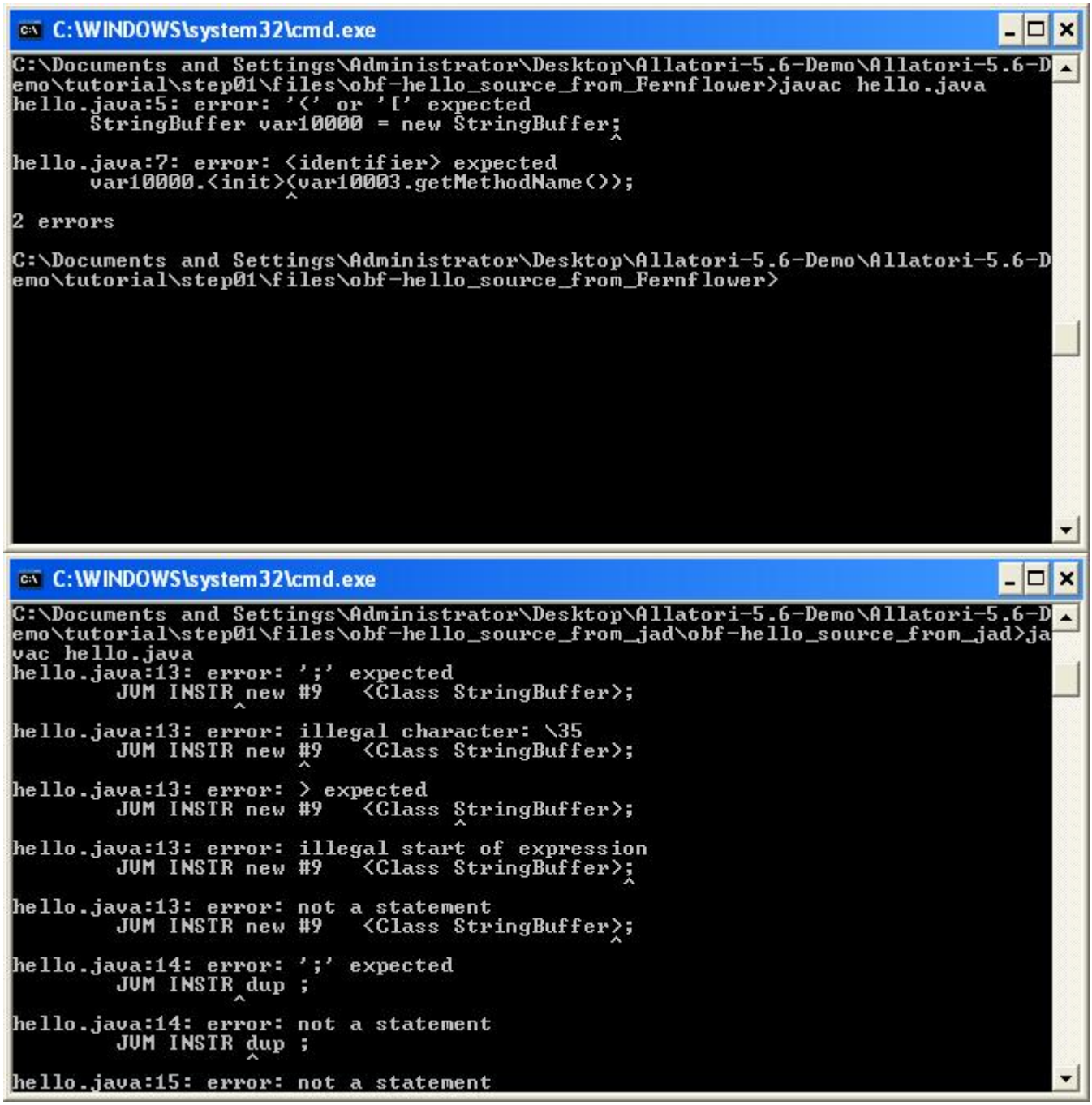
Değerlendirme sonucunda JadX, Procyon kaynak kodu dönüştücülerinin başarıyla Allatori v5.6 Demo sürümü ile gizlenmiş kodları orjinal haline çevirebildiğini gördüm.


C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\Documents and Settings\Administrator>cd C:\Documents and Settings\Administrator\Desktop\Allatori-5.6-Demo\Allatori-5.6-Demo\tutorial\step01\files\obf-hello_source_from_cfr\obf-hello_source_from_cfr
C:\Documents and Settings\Administrator\Desktop\Allatori-5.6-Demo\Allatori-5.6-Demo\tutorial\step01\files\obf-hello_source_from_cfr\obf-hello_source_from_cfr>javac hello.java
hello.java:16: error: not a statement
1 << 3 ^ 1;
1 error
C:\Documents and Settings\Administrator\Desktop\Allatori-5.6-Demo\Allatori-5.6-Demo\tutorial\step01\files\obf-hello_source_from_cfr\obf-hello_source_from_cfr>

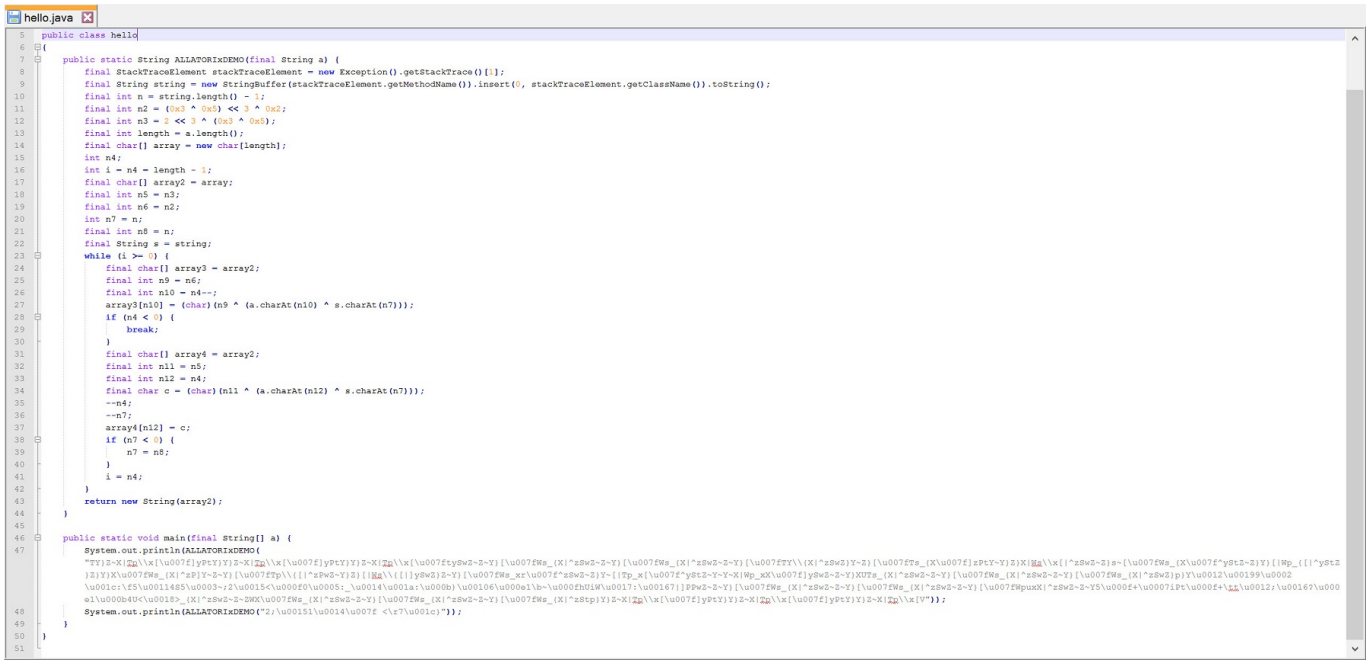

C:\WINDOWS\system32\cmd.exe
emo\tutorial\step01\files\obf-hello.jar_source_from_JADX\obf-hello.jar_source_from_JADX>javac hello.java
C:\Documents and Settings\Administrator\Desktop\Allatori-5.6-Demo\Allatori-5.6-Demo\tutorial\step01\files\obf-hello.jar_source_from_JADX\obf-hello.jar_source_from_JADX>java hello
Obfuscation by Allatori Obfuscator v5.6 DEMO
http://www.allatori.com
Hello World!
C:\Documents and Settings\Administrator\Desktop\Allatori-5.6-Demo\Allatori-5.6-Demo\tutorial\step01\files\obf-hello.jar_source_from_JADX\obf-hello.jar_source_from_JADX>


C:\WINDOWS\system32\cmd.exe
C:\Documents and Settings\Administrator\Desktop\Allatori-5.6-Demo\Allatori-5.6-D
emo\tutorial\step01\files\obf-hello_source_from_jdcore\obf-hello_source_from_jdc
ore>javac hello.java
hello.java:38: error: not a statement
tmp68_67;
^
hello.java:40: error: not a statement
int ? = tmp68_67;
^
hello.java:40: error: ';' expected
int ? = tmp68_67;
^
hello.java:40: error: not a statement
int ? = tmp68_67;
^
hello.java:43: error: not a statement
tmp78_74;
^
hello.java:45: error: not a statement
((0x3 ^ 0x5) << 3 ^ 0x2);
^
hello.java:52: error: illegal start of expression
?[tmp102_99] = ((char)(k ^ a.charAt(tmp102_99) ^ n.charAt(j)));
^
hello.java:52: error: illegal start of expression


C:\WINDOWS\system32\cmd.exe
emo\tutorial\step01\files\obf-hello_source_from_Procyon\obf-hello_source_from_Pr
ocyon>javac hello.java
C:\Documents and Settings\Administrator\Desktop\Allatori-5.6-Demo\Allatori-5.6-D
emo\tutorial\step01\files\obf-hello_source_from_Procyon\obf-hello_source_from_Pr
ocyon>java hello
Obfuscation by Allatori Obfuscator v5.6 DEMO
http://www.allatori.com
Hello World!
C:\Documents and Settings\Administrator\Desktop\Allatori-5.6-Demo\Allatori-5.6-D
emo\tutorial\step01\files\obf-hello_source_from_Procyon\obf-hello_source_from_Pr
ocyon>



Procyon ve JadX sayesinde Allatori v5.6 tarafından kullanılan karakter dizisi gizleme algoritması da ortaya çıkmış oldu :)



Bir sonraki yazıda görüşmek dileğiyle herkese güvenli günler dilerim.