Neden Tersine Mühendislik ?

written by Mert SARICA | 6 July 2010

Eskiden tersine mühendislik nedense bana çoook uzak gelirdi. IDA Pro uygulamasını ilk kurup çalıştırdığımda her yerde bir buton olduğunu görünce "hiç işim olmaz" diyerek kapattığımı hatırlıyorum ama aradan geçen zaman sonrasında şimdi masaüstüne bakıyorum da, vazgeçilmez kısayollardan bir tanesi oluvermiş ve "hep işim olmuş" :)

Neden bu kadar insan assembly gibi anlaşılması zor bir dil ile uğraşıyor, neden saatlerini tek bir fonksiyonun ne iş yaptığını anlamak için harcıyor kısaca neden tersine mühendislik ? Hemen aklıma gelenleri sıralayıp kısaca açıklayayım;

- Assembly öğrenmek için – Bilmeden nasıl öğrenilir demeyin. Debugger ile programları izleye izleye emin olun birşeyler öğreniyor ve komutların ne iş yaptığını az çok anlayabiliyorsunuz ve üzerine güzel bir kitap okuduğunuz zaman taşlar yerine oturuyor ve bir bakmışsınızki matrixi yorumlar olmuşsunuz.
- Gizli şifreleri, arka kapıları tespit etmek için – Zaman zaman programlardaki gizli menülere erişmek için programı hazırlayanlar dokümantasyonda yer almayan gizli fonksiyonlara programlarında yer veriyorlar ve kaynak kodu açık olmadığı için bunun sadece kendileri tarafından bilineceği yanılgısına düşüyorlar ve eninde sonunda tüm dünya bunu tersine mühendislik ile uğraşan bu meraklı insanlar sayesinde duyuveriyor.
- Doğrulama mekanizmalarını aşmak için – Daha önce bu örneği vermişmiydim hatırlamıyorum fakat bir zaman şifreleme anahtarı saklamak amacıyla kullanılan bir yazılımı incelediğimde anahtarı görüntüleyene kadar iki defa şifre doğrulama adımından geçmeniz gerekiyordu. Fakat bu adımları assembly seviyesinde yamadığımda (son adıma gitmesini sağladım) doğrulama mekanizmalarının tersine mühendislik ile aşılabildiğini işte o zaman öğrenmiştim.
- Kapalı kaynak kodlu yazılımlardaki güvenlik açıklarını bulmak için – Bildiğiniz veya bilmediğiniz üzere Microsoft firmasının o aylık meşhur yama günü gelip çattığında dünyanin dört bir yanındaki meraklı insanlar yamaların farklarını çıkartarak tersine mühendislik ile güvenlik

bültenlerine konu olmamış güvenlik zafiyetlerini ortaya çıkartarak istismar kodu geliştirilmesini sağlamaktalar.
- Zararlı yazılımları analiz etmek için – Bu konuda çok fazla söze gerek yok sanırım, yazılarımı takip edenler bilirler :)
- Can sıkıntısına karşı bire bir :) – Canım ne zaman sıkılsa ve bir programı oturup incelemeye başlasam saatlerin nasıl geçtiğin bir türlü anlamam, 6 saatin 6 dakika gibi geçtiğine çok defa tanık olduğumu söyleyebilirim.

Tersine mühendislik denilince çoğu kişinin aklına nedense hep kod seviyesinde olanı gelir fakat aslında tersine mühendislik sistem seviyesi ve kod seviyesi olmak üzere ikiye ayrılır ve günlük işlerinizde oldukça zaman zaman kullandığınız o meşhur sysinternals araçları ile sistem seviyesinde tersine mühendislik yaptığınızın farkında bile olmazsınız.

Sistem seviyesinde tersine mühendislik yaparken çoğu bilgiye işletim sistemi üzerinden erişirsiniz çünkü incelemiş olduğunuz hedef program eninde sonunda işletim sistemi ile etkileşime girer. Sistem seviyesinde tersine mühendislik için kullanılan araçlar regmon, filemon, lsof, ptrace ve benzerleridir.

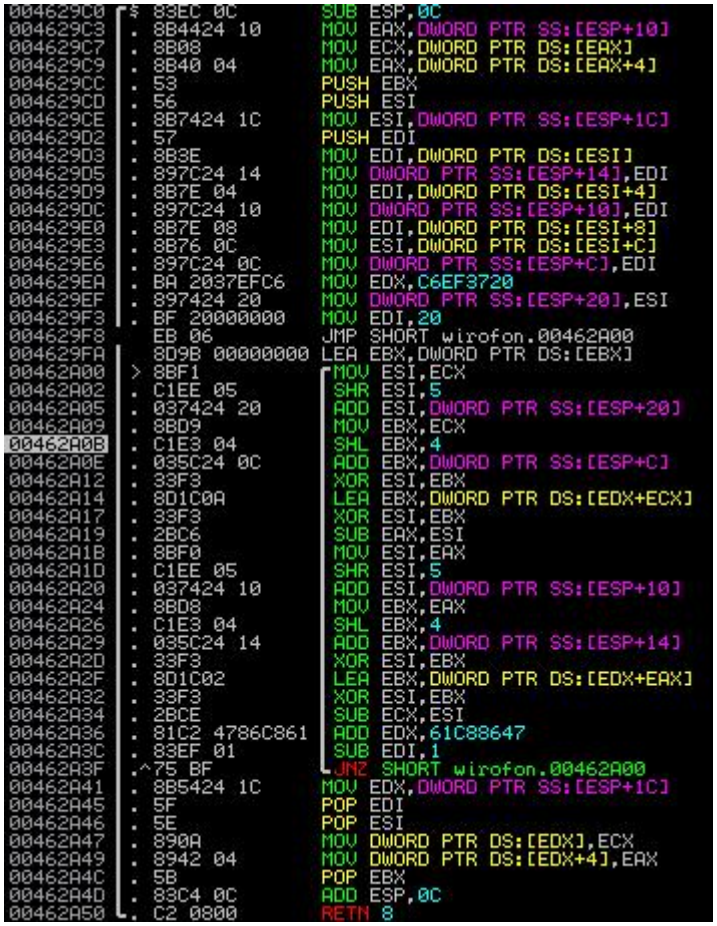
Kod seviyesinde tersine mühendisliğe ise sistem seviyesindeki yeterli olmadığı zaman başvurursunuz çünkü sistem seviyesinde bir programa gireni, çıkanı ve işletim sistemindeki etkileşimini görebilirsiniz fakat aslında o gizem dolu olup ve biten yazılımın içinde gerçekleşmektedir ve olup bitenden emin olmak için tek seçeneğiniz kod seviyesidir.

Programları incelemeyi seven biri olarak geçtiğimiz ay Wirofon uygulamasını sistem seviyesinde incelemiştim. Filemon ve regmon ile kısa bir takipten sonra kurulduğu klasörde yer alan userSettings\loginSettings.dat dosyası ile etkileşim halinde olduğunu görmüştüm. Canınım sıkıldığı bir gün bu dosya ile ne işi olduğunu merak ettim ve incelemeye başladım. Aradan kısa bir zaman geçtikten sonra programda yer alan “Beni hatırla” ve “Şifremi hatırla” seçeneği işaretlendiği zaman bu dosyanın içeriğinin değiştiğini fark ettim. loginSettings.dat dosyasını metin düzenleme programı (wordpad) ile incelediğimde okunaklı olmadığını yani şifrelenmiş olduğunu gördüm. Immunity Debugger ile dosya ile ilişkili kısımları incelediğimde “Not valid TEA encoded data” mesajı hemen dikkatimi çekti.

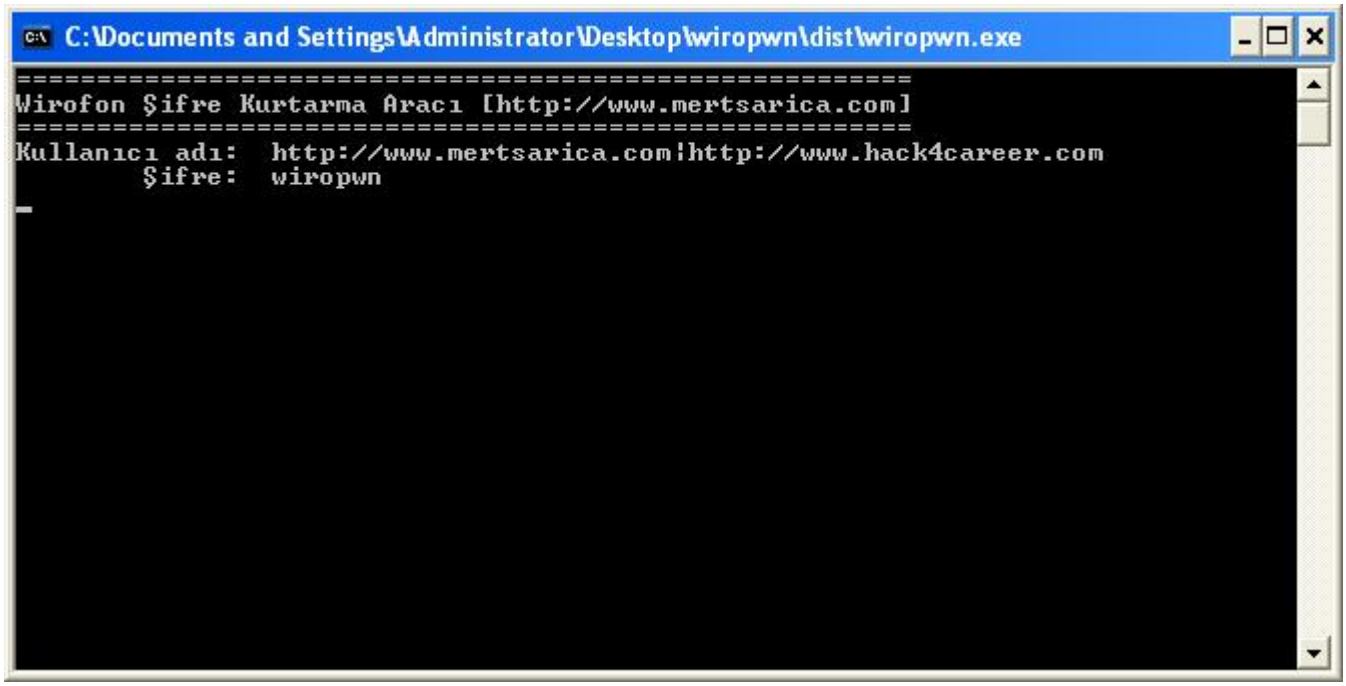
Bu programı inceleyene kadar TEA şifreleme algoritması ile ilgili hiçbir bilgim yoktu. (İşte tersine mühendisliğin güzel yanlarından biride bu, mutlaka yeni birşeyler öğreniyorsunuz.) Google'da ufak bir araştırma yaptıktan sonra şifreleme algoritması olduğunu ve zayıflıkları olması nedeniyle XTEA olarak güncellendiğini ve en sonunda XXTEA olarak güncellendiğini gördüm. TEA şifrelemesi ile ilgili internette bir çok makale ve hazır kod parçacıkları olduğu için loginSettings.dat dosyasında yer alan şifreli veriyi kolaylıkla çözebileceğimi düşünüyordum fakat çok geçmeden yanıldığımı fark ettim. TEA ve XTEA algoritmalarını incelediğimde şifrelemenin bir bölümünde SHL 4 (shift left, çarpma işlemi için kullanılır) ve SHR 5 (shift right, bölme işlemi için kullanılır) dikkatimi çekti fakat şifreli verinin neden çözülmediği ile ilgili uygulama üzerinde araştırma yaptığımda ufak bir detay gözüme ilişti.

Öncelikle şifrelemeyi çözmeden sorumlu olan fonksiyona giden ve anahtar olarak kullanılan dizinin "ARGELA Technologies" olduğunu ve bu dizinin (string) 4 parçaya bölünerek anahtar olarak kullanıldığını tespit ettim.

Daha sonra ise şifreli veriyi çözen fonksiyonu incelediğimde TEA ve XTEA algoritmasında yer alanın aksine stackte yer alan değerlerin bilinenin SHL 5 ve SHR 4 işlemine tabi tutulduğunu gördüm.



Internette ufak bir araştırma yaptığımda algoritmanın bu şekilde kullanımına uzak doğu sayfalarında rastlasamda çok fazla vakit harcamayarak şifreli veriyi çözen bu fonksiyonu Python’a taşımaya karar verdim ve sonunda şifreli veriyi çözmeyi başardım ve bu vesileyle şifresini unutan Wirofon kullanıcıları için ufak bir şifre kurtarma programı (şifremi hatırla seçeneğini işaretlemişseniz şifreli olarak loginSettings.dat dosyası içinde saklanan şifrenizi size gösterir) hazırlamış oldum. Programa buradan ulaşabilirsiniz.



Bir sonraki yazıda görüşmek dileğiyle…