Güven Veren Kod Yazmak

Muhammed C. Tahiroğlu, Aralık 2009, WGT E-Dergi 3. Sayı

Az evvel ne oldu biliyor musunuz?

Çok önemli bir bankanın herkesçe çok önem verilen internet şubesi'nde çalışan bir programcı arkadaşımız, yeni yazılacak ekranı daha önce yazılmış bir ekrandan copy / paste usulüyle aldı ve patlayan yerleri düzeltmeye başladı. Muhtemelen bu ekranın kodlaması, bugün biter. Bittikten sonra, bankadaki yazılım geliştirme ve yaygınlaştırma akıntısına kapılır ve sanal şubesini memnuniyetle kullanan vatandaşımızın önüne kadar gelir.

Sizce bu ekranın kodu, kopyalandığı ekrandan bir şeyler içeriyor mudur? Armut dibine düşerdi hani? Yüksek ihtimalle içeriyordur. Eğer testler sıkı tutulmuyorsa, kod gözden geçirmeleri yapılmıyorsa ya da etkisiz yapılıyorsa, o hazır ekranın kodundan mutlaka kırıntılar yeni ekranımızda kalacaktır. Çünkü programcı arkadaşımız, önceden de tanırız kendisini, güven vermeyen kodlar yazmaktadır.

Bu senaryoda da yazdığı kod, başkasının kodundan eşeysiz üreme yöntemleriyle yeni kodlar elde edilip mahiyeti, girdisi, çıktısı, algoritması, sadeliği vs. kontrol edilmeden "check-in" edilmiştir.

Bu arkadaşımız hep böyle yapar; biz yapmayız.

Çünkü biz elimizden çıkmış en ufak kod öbeğinin, güvenliğinden, bakım maliyetinden, okunabilirliğinden, sadeliğinden, işlevselliğinden, estetiğinden sorumlu hissederiz kendimizi. Bu yüzden yazdığımız kod ve çıkarttığımız ürün herkese güven verecektir. Yunus Emre, eğri odun taşımazdı bilirsiniz. Bizi de öyle bilirler işte; sakat kod yazmayız.

Yazdığı koda kefil olabileceğiniz bu tür programcıları da tanıyorsunuz siz. Etrafınızda onlardan var ve hakikaten doğal bir itibar kazanıyorlar.

...

Mâlum, şu an okuduğunuz dergi bir web güvenliği dergisi. Yıllardır dünyada ve Türkiye ayağıyla ülkemizde web güvenliği alanındaki farkındalığı artırmaya ve bu işin emekçilerinin, yani web programcılarının güçlü ve güvenli uygulamalar yazması için ortaya çıkan bir topluluğun ürünü. Tüm bu çabalar işe yarıyor mu derseniz... evet, yarıyor.

Bunu kendi serüvenimden örnekleyerek anlatabilirim. Sene 1998'de yazmaya çalıştığımız web uygulamaları -ki elimizde fazla teknoloji seçeneği yoktu- güvenlik paradigmalarından tamamen yoksundu. Zira, asıl maksat, çalışabilir bir web uygulaması inşa etmekti. Şimdilerde herkesin bildiği ve tedbirini aldığı kaşarlanmış web açıkları o dönemde açık değil bir "özellik"ti, "by design" idi.

Hatırladığım kadarıyla o zaman da popüler olan şey, hiçbir zaman değerini kaybetmeyen sistem ve mimari açıklarıydı. Yani DNS'lerdeki veya web sunculardaki kırılganlıklar,

affedilmiyordu. IIS'in eski bir sürümünde çalışan meşhur bir araba sitesinin kodları, ufak numaralarla web tarayıcıdan seyr edilebiliyordu. Şaşırmıyorduk.

Web güvenliği, yazılım güvenliğinden elbette farklılıklar arz ettiği için bu branşın da tarihsel gelişimini beklemememiz gerekti. Zaman ve yaşanan kazalar herkesi eğitti ve bugünkü bilinçlenme düzeyine geldik. Artık bir form gördük mü hemen "tamper" edilme ihtimalini de düşünüyoruz. "Cookie"lerle kimse görmez sanıp değerli mücevherler göndermiyoruz. İnşallah?

Web geliştirmenin ilk yıllarında kolaylık, basitlik ve süper özellik olarak sunulan her şeyin sonradan bize acılar yaşatan güvenlik zafiyetleri çıkarttığını da eminim birçok tecrübeli arkadaş bilecektir. En basitinden, PHP'deki değişkenlere "query string"ten değer aktarmak için ayrıca bir kod yazmaya gerek duymamak ve aktarımın kendi kendine değişken adı ve "query string" anahtarından isim eşlemesiyle yapılması biz ASP'cilerin kafasına vurulan bir kolaylık ve süper özellikti. Şimdi böyle bir özelliği bırakın kullanmak, duymak bile tüylerinizi ürpertmiyor mu? (Tüyleri ürpemeyenler var ki ASP.NET MVC'de ModelBinder çıkarıp sunucu tarafındaki nesneyi komple formdan veya "query string"ten doldurmanızı sağlayabiliyorlar. Ve hiçbir pazarlama broşüründe bu yöntemin doğrudan kullanılırsa sizi patlatacağı yazmayacak. Neden, bir özellik, bariz güvenlik açığı oluşturarak sürülür ki?)

Web uygulamaları yazarken güvenlik kaygısının bu kadar ön plana gelmesi oldukça sevindirici. Güvenlik makalelerinin, sitelerinin ve seminerlerinin yoğun bir ilgiye mazhar olması yine sevindirici. Burada niyetleri sorgulamadan, -yani "insanlar zarar vermek için mi yoksa huzur vermek için mi güvenlik merakında" kısmını deşmeden- düşünürsek, uygulamalardaki güvenlik seviyesi ne kadar yükselir ve yüksek seviyelerde standartlaşır ise son kullanıcılar da o kadar huzurla hareket ederler. Çünkü artık web uygulamaları, arkalarında çok önemli sırlar ve servetler emanet edilen yerler. Bir hazinenin fiziksel olarak korunması için nasıl bir titizlik sergileniyorsa, web uygulamalarına daha fazlası nasip olmalıdır. IstSec'te konuşacak olan Ulaştırma Bakanı'na ulaşır mı bu dediğimiz bilmiyorum ama bir halk atasözü der ki "devlet buna bir şey yapması lazım". Ne yapılması gerektiğini uzmanlar kendi aralarında konuşsunlar.

Biz, odağımızı azıcık kaydırıp asıl konumuza gelelim.

Her ne kadar güvenliği baştan sağlanmış uygulama kütüphaneleri, platformlar kullanıyorsak da... her ne kadar siteyi Netsparker'a verip bulgu bekliyorsak da... yazdığımız kod %100 güvenli değildir. Çünkü güvenlik açıkları da mutlaka çeşit çeşittir ve bunun en tehlikelisi, kodu sadece okuyarak farkedebileceğiniz "mantıksal" veya "akış"la ilgili açıklardır.

Size çok basit bir örnek vereyim ve bu açığı hangi araçla yakalayabileceğinizi düşünün bir yandan. (Buna özel bir araç yazılmış ise amenna!)

Kötü gününüzdesiniz ve T-SQL yazıyorsunuz. Bir "stored procedure" içinde "cursor" açtınız. Cursor'de yapacağınız iş, ödemesi gelen adamları tek tek dolaşıp hesaplarına ödenecek tutarları göndermek.

- *"Cursor" lâfına yabancı olanlar için ufak bir açıklama: SQL'de bir kayıt seti üzerinde dolaşmak için "cursor" adı verilen kullanması ve bakımı maliyetli, antin kuntin bir yapı var. Cursor'un yaptığı işi "while" döngüsü ile azıcık fazla kastırarak yapabilirsiniz.*

Örneğimizi canlı hâle getirmek için sahte tablolarımızı tasarlayalım. Ödemesi gelen müşteri bir tabloda, hesap bilgisi de diğer tabloda duruyor. Faraza yani:

```
create table #odemesiGelenMusteri
(
 musteri varchar(10),
 tutar money
)

create table #hesap
(
 musteri varchar(10),
 hesapNo varchar(10)
)
```

Üç tane ödemesi gelen müşteri ve tutarları:

```
insert into #odemesiGelenMusteri values ('Ahmet', 3500)
insert into #odemesiGelenMusteri values ('Mehmet', 5000)
insert into #odemesiGelenMusteri values ('Mürteza', 10000)
```

İlginç bir veri ekleyeceğiz şimdi. Sadece Ahmet'in hesabı var. Diğerlerinin hesabı henüz girilmemiş:

```
insert into #hesap values ('Ahmet', '12121-1')
```

İşte size cursor bloğu:

```
declare @musteri varchar(10), @hesapNo varchar(10), @tutar money
declare musCursor cursor for
select musteri, tutar from #odemesiGelenMusteri (nolock)
open musCursor
fetch next from musCursor into @musteri, @tutar
while @@fetch_status=0
begin
   select @hesapNo = hesapNo
   from #hesap
   where musteri = @musteri

   bir_sekilde_eft_yap @musteri, @hesapNo, @tutar *

   fetch next from musCursor
   into @musteri, @tutar
end
```

```
close musCursor
deallocate musCursor
```

Sizce sonuna yıldız koyduğum satırda **@hesapNo** değeri nasıl değişecek? Cevap şöyle:

```
---------- ---------- --------------------
Ahmet      12121-1    3500.00
---------- ---------- --------------------
Mehmet     12121-1    5000.00
---------- ---------- --------------------
Mürteza    12121-1    10000.00
```

Tüh ki tüh. Mehmet'in ve Mürteza'nın hiç hesabı yoktu. Madem öyle, paraları Ahmet'e gitsin gibi olmuş.

Kötü hikaye. Yaşadım mı böyle şeyler? Evet, yaşadım. Malesef, cursor bloğu içindeki değişkenleri sıfırlamadığım oldu. Ama bir daha yapmadım. (Benimkisi yanlış paradan ziyade, yanlış e-mail gönderimi idi. Belirteyim.)

Güven veren kod yazmak derken, bu tür yanlışlara karşı baştan uyanık olmayı kastediyorum. Bana göre güven veren kod yazmanın bir tane altın kuralı var:

"Efradını câmi, ağyarını mâni" yazmak.

Bu söz, daha modern biçimde "az ama öz" olarak da karşılanabilir. Ama karşılanmasın lütfen, bu eski söz ne istediğimi daha iyi anlatıyor.

Efradını câmi, bütün gereken unsurları içermesi demek. Öyle bir kod yazacaksınız ki bir parçasını dahi söküp atamayacağız. Gereksiz diye silebileceğimiz bir satırı, bir kelimesi olmasın.

Öyle bir kod yazacaksınız ki gereksiz unsurları tamamen dışarıda tutacak. Sileceğiniz bir şeyi yazmayacaksınız. Değersiz bir satırı olmayacak.

Bu şartları sağladığınızda önünüzdeki kodun bir "eser" olmadığını kim iddia edecektir? İnanın, bu lezzeti alarak yazdığınızda, o kodu seyretmeye doyamayacaksınız. (Latife yaptığımı lütfen düşünmeyin.)

İpin ucunu Web güvenliği mevzusuna bağlayalım.

Güven veren kodda, güvenlik zafiyetine neden olabilecek bir akış, bir mantık problemi, kendini hemen belli edecektir. Dağınık bir masada elinizi kesebilecek bir iğneyi bulmanız epey güçtür. Her şey ahenk içerisinde duruyor ise, ahengi bozan iğne, kapıdan girerken dahi farkedilecektir. Evet, farkedilmesi güç şeyler yine olacaktır ama bu bizim işimiz. İşimize saygı duymak ve onu en güzel şekliyle yapmak zorundayız. Güvenliğe bakan yönü de bonusumuz.

Kod yazan civanlara tavsiyelerim odur ki...

- Değişken kullanırken, cimri olun.
- Değişken adı verirken acımasız olun, uzunca yazın.
- Dünyaca kabul görmüş isimlendirme disiplinlerine uyun. Köyünüzün şivesiyle isim vermeyin.
- Kodları mümkün olduğunca fazla, olduğu yerde dokumante edin. Acayip bir numara çektiyseniz, o numarayı oraya laf ile de yazın. (Aslında metodlaştırma ve güzel isimlendirme ile dokumantasyon seviyenizi düşürebilirsiniz.)
- "Maintainability index"ten haberdar olun. İşler uzadıysa ve sarpa sardıysa, yeni metod açın.
- Elinizi korkak alıştırmayın. Bir yapıyı yeniden kurduğunuzda daha iyi olacağına inanıyorsanız, çalışan kısmı bozmaktan geri durmayın. Yıkın gitsin. ("Version control" sistemi var nasıl olsa değil mi?)
- Kod çoğaltırken üşengeç olun, "reusable" kodlayın.
- Denetim ve doğrulama mekanizmalarını dallara, yapraklara değil köklere koyun.
- Girdilerin ne olmaması gerektiğini değil, ne olması gerektiğini yazın. (white list)
- Güzel kodlar okuyun.
- Kodunuzu arkadaşınıza okutturun. Gerekirse parası neyse verin, "code review" yaptırın.
- "Code reuse" kaçamayacağınız bir şeydir ancak henüz olgunlaşmamış ve istikrarlı sürüme gelmemiş hazır kütüphaneleri halka açık yerde kullanmayın. (Yeni sürümlere atlamayın.)
- En azından bir tane aktif uygulama güvenliğine dair bülten takip edin. (RSS şelalesi de bülten sayılır.)
- Kodun üzerinde statik kod analizi ve güvenlik analizi yapan cihazları çalıştırın. Sonuçlarını değerlendirip, düzeltmeler yapıp tekrar çalıştırın. (Sonuçları raporlayıp bırakanlar için söylüyorum.)
- Kullandığınız dili ve platformu derinlemesine bilin. Yukarıdaki gibi cursor yazdığınızda, ben bilmiyordum filan diyemezsiniz. C#'ta Generic yapıları varken, 1.0'dan kalma casting gerektiren şeyler kullanmayın. Okuyun, öğrenin ve hep takip edin.

Titiz kod yazdığımızda hem bizim, hem iş yerimizin hem de koda sonradan bakacak arkadaşımızın başı daha az ağrıyacaktır emin olun. Üstelik bir de prestij meselesi var. Yarın bir gün, kodumuza bakım yapan bir arkadaşın, arkamızdan küfretmesi ne kadar hoş olur? Benim yazdığımı nereden bilecek diyorsunuz. Bilir, bilir. "Version control" sistemlerinin bir amacı da "kodu ilk yazanı bulup saydırmak" değil midir?

Bu satırların yazarı, 1998'den beri Web'le başlayan, SOA ile devam eden bir yazılım geliştirme serüveninin içinde. Şu an bir özel kurumun yazılım altyapısında görevli. Güvenlik, tek meselesi değil ama öncelikli meselesi.

İyi kodlamalar ve başarılar.