Responsible Disclosure Örneği

written by Mert SARICA | 22 Nisan, 2010

Güvenlik dünyasına merakı, ilgisi olanlarınız daha önce bir çok kez disclosure (ifşa) kelimesini duymuştur. Özellikle SecurityFocus'un o yıllara meydan okuyan meşhur Bugtraq e-posta listesine üyeyseniz (değilseniz çok şey kaçırıyorsunuz) hemen hemen hergün dört ifşa modelinden biri ile posta kutunuzda karşılaşırsınız. İfşa modelleri nelerdir ve neden ihtiyaç duyulur dediğinizi duyuyor gibiyim kısaca açıklayayım.

Güvenlik araştırmacıları (Security researchers) işleri gereği veya hobileri gereği vakitlerini sistem ve uygulamalarda güvenlik açığı arayarak geçirirler. Güvenlik açığını keşfettikten, analiz ettikten, istismar aracını (exploit) hazırladıktan sonra ve son adımda bir karar vermek zorunda kalırlar işte burada dört ifşa modelinden birini seçerler. İfşa modellerinin temel amacında kullanıcıları güvenlik zafiyeti konusunda bilgilendirme ve üreticileri zafiyetin giderilmesi konusunda göreve çağırmak yatar. Bir güvenlik zafiyeti konusunda insanlar bilgilendirilmez ise bundan en çok kar amacı güden üreticiler memnun olacaktır çünkü bir sistem veya uygulama için yama hazırlanması üreticilere ekonomik açıdan pahalıya mal olur ve kar amacı güden bir üretici üzerinde baskı hissetmediği sürece güvenliğinizi çoğu zaman ikinci plana atacak ve önem vermekten kaçınacaktır.

İfşa modellerini kısaca açıklamak gerekirse;

Responsible Disclosure: Üretici ile herhangi bir haberleşme yolu ile iletişime geçerek güvenlik zafiyeti konusunda güvenlik açığının varlığını ve istismar edilebildiğini kanıtlayan POC (proof-of-concept code) kodunu üretici ile paylaşır ve ne kadar zaman içerisinde üreticinin bu zafiyeti gidereceğini ve kullanıcıları ile paylaşacağını sorarlar ve el sıkıştıkları taktirde üretici firma güvenlik zafiyeti ile ilgili yamayı duyurduktan, el sıkışamadıkları taktirde en kısa zamanda (Full Disclosure) güvenlik araştırmacısı güvenlik zafiyeti ile ilgili bilgiyi ve POC kodunu çeşitli kaynaklar (örnek: Bugtraq) üzerinden insanlarla paylaşır.

Limited Disclosure: Bu modelde ise güvenlik araştırmacısı tarafından güvenlik zafiyeti ile ilgili yukarıda belirtilen el sıkışma, POC ve detay haricindeki adımlar izlenir. Limited disclosure yani kısıtlı ifşa ile güvenlik araştırmacısı güvenlik zafiyeti ile ilgili detaylı bilgi vermediği için bu modelin ne üretici ne de kullanıcılar üzerinde fazla bir etkisi olmaz bu nedenle üretici zafiyet ile ilgili yama çıkarmayabilir.

Full Disclosure: Bu modelde ise güvenlik araştırmacısı üreticiye haber vermeden güvenlik zafiyeti ile ilgili detaylı bilgiyi ve POC kodunu yayınlar ve kullanıcılar o an itibariyle güvenlik açığının risk derecesine göre büyük bir tehlike ile karşı karşıya kalabilirler. Bu gibi durumlarda üreticiye çok iş düşer çünkü yama yayınlanana kadar kullanıcılar risk altında olurlar. Özellikle risk derecesi kritik ise bu durumdan faydalanmak isteyen art niyetli kişiler veya gruplar solucan (worm) hazırlayarak bu durumdan nemalanmak için ile koyulurlar.

No Disclosure: Bu modeli ise çoğunlukla yer altı hacking grupları ve istismar aracını satarak maddi gelir sağlayan kişiler tercih ederler. (Nedenlerini açıklamama gerek yok sanırım :))

Gelelim bunları sizlerle neden paylaştığıma.

Yine o klasik nedenden ötürü yani can sıkıntısı ile geçtiğimiz günlerde açık kaynak kodlu web uygulamalarını incelemeye karar vermiştim. Tercihimi PHP forum veya blog yazılımlarından yana kullanmaya karar vermişken Google arama motorunda yaptığım ufak bir araştırma neticesinde yerli malı yurdun malı phpKF (php Kolay Forum ve Portal) ve uygulaması ile karşılaştım. Çoğunlukla bu ve benzer PHP uygulamalarında SQL Injection, LFI/RFI (file inclusion) saldırıları en çok karşılaşılan güvenlik zafiyetlerinin başında gelirler. Bende bunu göz önünde bulundurarak bu uygulamayı incelemeye karar verdim.

Kaynak kodunda, grep ve cut ile çağ dışı bir şekilde güvenlik zafiyeti aramaktansa Python ile önce ufak bir mysql injection tarama aracı hazırlamaya daha sonra ise LFI/RFI tarama aracı yazmaya kısaca yoldan şansımı denemeye karar verdim. Hatta biraz daha kolaya kaçıp önce ufak bir blind mysql injection tarama aracı yazarak işe koyuldum. IDLE ile kısa bir zaman cebelleştikten sonra ortaya Simple Blind MySQL Injection aracı kısa adıyla SMBIT ortaya çıkıverdi.

Ne yalan söyliyim böyle basit bir programdan pek fazla beklentim yoktu hatta grep ve cut ile güvenlik zafiyeti bulmak için daha fazla şansım olacağını düşünüyordum fakat yanıldım :)

Sıra ifşa modelini seçmeye gelmişti. Code of ethics imzalamış biri olarak Responsible Disclosure dışında bir modeli düşünemediğim için bu model ile ilerlemeye karar verdim ve phpKF'nin web sitesini ziyaret ederek iletişim bilgilerini aramaya koyuldum ancak herhangi bir e-posta adresi bulamadığım için siteye üye olarak uygulamanın programcısı olan Adem YILMAZ'a özel mesaj gönderdim. Başlarda kendisi ile el sıkışmakta biraz zorlansamda sonunda el sıkışabildik ve kendisi uygulamayı güncelleyerek kullanıcılarını bu güvenlik zafiyeti konusunda bilgilendirdi.

Bilgi vermesi açısından Adem YILMAZ ile yaptığım görüşmeyi başından sonuna kadar sizlerle paylaşıyorum. Şimdiden herkesin 23 Nisan Ulusal Egemenlik ve Çocuk Bayramını kutlar herkese iyi haftasonları dilerim.

Not: Responsible Disclosure modelinde karşı tarafa yani üreticiye yamayı ne zaman yayınlayabileceği sorulur ve daha sonrasında el sıkışılır ancak aşağıdaki örnekte biraz baskıcı bir yaklaşım sergilediğimi farkedeceksiniz nedeni yamanın en kısa sürede yayınlanmasını sağlamaktı. Eğerki karşı taraf bu süre zarfında yamayı belirli nedenlerden ötürü yayınlayamayacağını belirtse idi bu durumda etik olarak karşı tarafın belirttiği süre sonunda yayınlamayı kabul edecektim.

——— Mert SARICA tarafından gönderilen ileti ———

Selamlar,

Bilişim güvenliği uzmanı olarak zaman zaman programları inceleyerek güvenlik açıkları arıyor ve blogumda (http://www.mertsarica.com) yer veriyorum.
Geçtiğimiz günlerde forum ve portalını kurup inceleme fırsatı yakaladım ve blind sql injection güvenlik zafiyeti keşfettim.
Muhtemelen Perşembe veya Cuma günü blogumda bu habere yer vereceğim bu nedenle öncesinde bu zafiyeti giderme adına bir yama yayınlarsan art niyetli kişiler tarafından istismar edilmesini önlemiş olursun.

Proof of concept:
http://localhost/portal/dosyalar.php?kategorino=1'+and+sleep('15')%23

İyi akşamlar.

——— Adem YILMAZ tarafından gönderilen ileti ———

Bilgilendirme için teşekkürler fakat bu adres ile alınan hata sadece tanımsız değişkenden dolayı eregi fonksionunun uyarı vermesidir.
Alınan kategorino değişkeni sayfa içinde temizlenerek veritabanı sorgusuna sokulmaktadır ve herhangi bir sql injection açıkğı yoktur.
Dikkat ederseniz aynı hatayı şu şekilde kategorino`den hiçbir veri yollamadığınızda da verecektir.

http://localhost/portal/dosyalar.php?kategorino=

Kısaca tekrar edeyim, verdiğiniz örnekte hiçbir açık yoktur.
Bunu haber yapmamanızı tavsiye ederim, çünkü yanlış bilgi vermiş olursunuz ve komik duruma düşersiniz.

Tekrar teşekkürler iyi geceler.

——— Mert SARICA tarafından gönderilen ileti ———

Magic_quote kapalıyken denerseniz ne demek istediğimi anlayacaksınız.

http://localhost/portal/dosyalar.php?kategorino=1'+and+sleep('15')%23

Yukarıdaki şekilde sayfayı çağırırsanız 15 saniye geç açıldığını görebilirsiniz,15 saniye geç açılması demek sorgunun çalışması demektir.

——— Adem YILMAZ tarafından gönderilen ileti ———

Denedim ve aynı çünkü Magic_quote kapalı veya açık farketmez
[b]kategorino[/b] sadece rakam kabul edecek şekilde ayarlıdır, başka bir karakter sorguya sokulamaz.
Muhtemelen kodlarda açık doğruracak bir değişiklik yaptığınız, bu yüzden 15 saniye bekleme oluyor.
Siteden temiz phpKF_Portal 1.70 indirin hiçbir açık olmadığını göreceksiniz.
Alttaki kod dosyalar.php dosyasından, burada sadece rakam kabul edilmektedir, değişkende rakam olduğu halde yine de temizleme fonksiyonundan geçirilir.
Rakam değilse hata verilir.

```
if ((isset($_GET['kategorino'])) AND (is_numeric($_GET['kategorino']) ==
```

```
true))
{
$_GET['kategorino'] = @zkTemizle($_GET['kategorino']);
}
else
{
```

——————— Mert SARICA tarafından gönderilen ileti ——————

Mantık hatası var dikkat edersen görebilirsin.
isset ile kontrol ediyor daha sonra tekrar issetse ve numericse temizliyor ama numeric değilse temizlemiyor bu nedenle direk bir sonraki satırda sorgumuzun çalıştırılmasına imkan tanıyor…

```
if (isset($_GET['kategorino']))
{
if ($portal_bloklar_ayar['dosyalar_sayfasi'] == 1):
if ((isset($_GET['kategorino'])) AND (is_numeric($_GET['kategorino']) ==
true))
{
$_GET['kategorino'] = @zkTemizle($_GET['kategorino']);
}
// SEO ADRESİNİN DOĞRULUĞU KONTROL EDİLİYOR YANLIŞSA DOĞRU ADRESE
YÖNLENDİRİLİYOR //
$sorgu1111 = "select kategorino,kategoriadi from $tablo_portal_indirkategori
where kategorino='$_GET[kategorino]' LIMIT 1";
$sorgu1111_sonuc = mysql_db_query($cfgdbisim,$sorgu1111) or die ('Haberler
sorgu başarısız');
```

——————— Adem YILMAZ tarafından gönderilen ileti ——————

Şimdi siz yazmadan önce bende onları kontrol ediyordum ve gördüm hatta bir tane daha var bu hatadan.
Portal kodlarını Yücel yazdı, ben de kontrol etmiştim ama benimde gözümden kaçmış.
Gerekli bilgilendirmeyi ve düzeltmeyi yapacağım.

Tekrar teşekkürler…