PENTEST GÜNLÜKLERİ

YILMAZ DEĞİRMENCİ

BILISHIM SİBER GÜVENLİK VE YAPAY ZEKA
Hacettepe Teknokent

ÖNSÖZ

Sızma testi ya da Pentest; bir hacker gözüyle sistemlerin güvenliğini taramak, zafiyetleri mümkünse sömürerek tespit etmek ve tüm bulguları bir Sonuç Raporuyla sunmak, kurum tarafından zafiyetler kapatınca bunun doğrulamasını yapmak üzerine kuruludur.

Hem bir TSE-A sertifikalı, hem de Endüstriyel Kontrol Sistemlerde EPDK yetkili bir Sızma Testi firması olan Bilishim Siber Güvenlik ailesi olarak sürekli sahadayız.

Her ne kadar sızma testi temelde teknik bir çalışma olsa da, her bir sızma testi özünde kendine özgü bir hikaye barındırmaktadır.

Hazırladığımız "Pentest Günlükleri" isimli bu çalışma, bizim sahada deneyimlediğimiz ve her biri kendine ait dersleri barındıran yaşanmış hikayelerden ibaret. İşini çok severek yapan bir ekip olarak her bir testte ayrı bir heyecan yaşıyoruz ve bu heyecanın hikayelerini sizlerle paylaşmak istedik.

Bu hikayeleri birlikte yaşadığımız değerli ekip arkadaşlarıma buradan ayrıca teşekkür ederim.

SIZMA NEDİR?

Kısaca PenTest olarak bilinen Sızma Testi ya da Penetrasyon Testi temel olarak bir bilişim sisteminin tüm yapıtaşlarının olası siber saldırılara karşı, taranması, analiz edilmesi, sızılması ve sıkılaştırmasını kapsayan ileri mühendislik isteyen özel bir test sürecidir.

Test sırasında uzmanlar tıpkı bir saldırgan gibi hareket eder ve sistemin tüm açıklarını, riskleri ve erişilebilirliği ortaya çıkarırlar. Çok farklı yöntemler ve değişkenler söz konusu olduğundan sızma testinin uzmanlar tarafından gerçekleştirilmesi gerekmektedir.

SIZMA TESTİ NEDEN ÖNEMLİDİR?

PenTest'in nihai amacı, bir saldırgan gözüyle sistem altyapınızın ne kadar güvende olduğunu ortaya koymak ve açıklık bulunan noktaları kapatmaktır. Bir kuruluşun güvenlik duruşundaki zayıf noktaları tespit etmenin yanı sıra, güvenlik politikasının uygunluğunu ölçmek, personelin güvenlik sorunları konusundaki farkındalığını test etmek ve kuruluşun siber güvenlik prensiplerini uygulama derecesini belirlemektir. Unutulmaması gereken en temel husus, tüm sistemlerin sadırıya karşı hassas olduğudur.

Hiçbir sistem 100% güvende değildir. Bu nedenlerden dolayı kuruluşların sistemlerini düzenli olarak PenTest yaptırması ve işletim sistemi ile uygulamalara zamanında güvenlik yamalarını geçmesi çok önemlidir.

Penetrasyon Testleri temel de üç aşamaya ayrılır:

Siyah Kutu: Bu yöntemde pentest gerçekleştirilecek kurumdan herhangi bir bilgi talebinde bulunulmaz, yapılan çalışmalar ile tamamı ile manuel gerçekleştirilen yöntemler sayesinde kurumun internete açık olan veya olmayan ara yüzlerinden sisteme sızılması çalışması gerçekleştirilir. Bu sayede kurumun dışarıdan alabileceği tehditlerin genel analizi ve vektörleri çıkarışmış olur.

Kara Kutu yöntemi, dışarıdan gelen gerçek bir saldırı senaryosuna en yakın test yöntemidir. En büyük amaç hedef sistemde veri tabanına sızmaktır.

Gri Kutu: Gerçekleştirilen bu test sayesinde kurumun içerisinde herhangi bir misafir gibi gelinerek sistemin boşluklarından faydalanıp sistemin tehlike yüzeyi analiz edilir. Bu sayede saldırganın kurumun içerisinden mevcut sisteme geçişi, veri tabanlarına erişimi, uygulamalar üzerinden ki hâkimiyetine kadar pek çok süreç bu test sayesinde ele alınır.

Gri Kutu yöntemi, fiziksel olarak kurum içine girebilen bir hacker tarafından yapılacak saldırının ne oranda başarılı olabileceğini ortaya koyar. En büyük amaç sınırlı bir yetkiyle dahi olsa tüm sistemi ele geçirmektir.

Beyaz Kutu: Güvenlik uzmanı, bu test ile birlikte testin yapıldığı kurumdan standart bir kullanıcıya ait bir tanımlama ister bu sayede uzman standart bir kullanıcı olarak sistem üzerinde haklarını artırıp artırmadığı, veri tabanlarına ve diğer tüm bilişim alt yapısı üzerinde ki hâkimiyeti kontrol edilir. Bu testler esnasında çeşitli araçlar kullanıldığı gibi sistemin saldırganı yakalamaması için gerekli tekniklerinde içinde bulunduğu özel manuel yöntemler kullanılır.

Beyaz Kutu yöntemi, kuruma içten gelecek bir saldırının ne oranda etkili olabileceğini ortaya koyar.

SIZMA TESTİ AŞAMALARI

Penetrasyon Testi, belli bir sistematik yaklaşım ile icra edilen bir süreçtir.

1. Pasif Bilgi Toplama (İz Sürme):

Ele alınacak sistemler ve kullanılacak test yöntemleri dahil olmak üzere, testin kapsamını ve hedefler tanımlanır.

Bu aşama, hedef sistem altyapısı ve kapsamı dâhilindeki etki alanı adları, ağ blokları, yönlendiriciler, IP adresleri gibi bilgiler ile çalışan bilgileri, telefon numaraları gibi saldırının başarısına katkı sağlayacak her türlü ayrıntı hakkında bilgi sahibi olunmasını sağlayan adımdır.

Açık kaynaklardan toplanan bu bilgiler birçok defa şaşırtıcı derecede kritik bilgileri içerebilmektedir. Bu maksatla başta hedef kurumun web sitesi ve sosyal medya platformları olmak üzere birçok kaynaktan istifade edilmektedir.

2. Aktif Bilgi Toplama ve Tarama:

Tespit edilen IP aralığı içinden hangi aktif ve pasif cihazların canlı olduğu tespit edilen bu aşamada, birinci aşamada elde edilen bilgilerin de yardımıyla önceliklendirme yapmak mümkündür.

Bu aşamada, tespit edilmiş olan canlı sistemlerde çalışan işletim sistemi, açık portlar ve servisler ile bunların sürüm bilgilerini elde etmek önemlidir.

Ayrıca dinlenebiliyorsa ağ trafiği de takip edilerek sistem altyapısı hakkında mümkün oldukça kritik bilgi toplanmaya çalışılır.

3. Döküm Çıkarma:

Hedef uygulamanın çeşitli izinsiz giriş denemelerine nasıl cevap vereceğini anladıktan sonra bu adımda, canlı olduğu tespit edilen sistemlerle aktif bağlantılar kurulur ve doğrudan sorgulamalar yapılır.

Diğer bir ifadeyle bu aşama; ftp, netcat, telnet gibi servislerin etkin olarak kullanıldığı ve hedef sistemlerle etkileşime geçildiği aşamadır.

4. Sistemi Ele Geçirme:

Daha önceki aşamalarda elde edilen tüm bilgilerin tek bir amacı vardır. Hedef sisteme yetkisiz giriş sağlamak, veri tabanını okumak ya da ulaşılmaması gereken bilgilere erişmek.

Bu aşama; hedef sistem üzerinde çalışan işletim sistemi, açık olan portlar ve bu portlarda hizmet veren servisler ile bunların sürümleri ışığında uygulanabilecek sömürü yöntemlerinin denendiği ve içeriye sızılmaya çalışıldığı adımdır.

Özellikle web tabanlı portal ve uygulamalar, hem dışa bakan pencere olmaları, hem çok fazla saldırı vektörü barındırabilme özelliklerinden dolayı özel bir konuma sahiptirler.

Sistemi ele geçirme aşamasında mevcut sömürü yöntemlerini sisteme zarar vermeden, iz bırakmadan, başarılı ve esnek bir şekilde kullanabilmek ciddi bir uzmanlık ve deneyim gerektirmektedir.

PenTest'in bu aşaması bu nedenle en önemli ve kritik adımdır.

5. Yetki Yükseltme:

Bir sistem, sistemin en zayıf halkası kadar güçlüdür. Bir şekilde başarılan sistem erişimi genelde ilk adımda düşük bir yetki ile gerçekleşmektedir. PenTest uzmanı, bu aşamada, bulunduğu işletim sistemi ya da ortamdaki zafiyetlerden istifadeyle yetkisini yönetici seviyesine çıkarmayı, ardından, kazandığı bu ek yetkiler ile birlikte ağ ortamında bulunan diğer cihazları ve nihayetinde Etki Alanı Yöneticisi ya da Veri Tabanı Yöneticisi gibi en üst düzey kullanıcı yetkilerini ele geçirmeyi hedefler.

SIZMA TESTİ HİZMETİ VE SONRASI:

Penetrasyon testi neticesinde firmamız tarafından hedef sistemlerde tespit edilen güvenlik açıklıkları ve bu zafiyetlerin nasıl istismar edildiği örnek ekran çıktılarıyla birlikte sunulan "Sızma Testi Sonuç Raporu"nda yer alır. Sonuç raporunda ayrıca her bir açıklığın nasıl kapatılacağına dair çözüm önerisi de yer almaktadır.

Bir siber güvenlik firması olarak; yapmakta olduğumuz sızma testlerinin hassasiyeti, bağımsızlığı ve sıhhati açısından özellikle ürün satmıyoruz. Ancak talep edilirse sistemlerin daha güvenli olmasına yönelik güvenli ağ mimarisi tasarımı ve sıkılaştırma danışmanlığı hizmetini ayrıca vermekteyiz.

PENTEST GÜNLÜKLERİ

Pentest anlamında sahaya ilk çıktığım günler. Bir teknokent yazılım firmasının testini yapıyoruz. Her zaman olduğu gibi ilk adım olarak trafiği dinliyorum. Bulunduğum IP bloğundan başka bir IP trafiği var mı diye. BT yöneticisi arkadaşında paylaşımıyla, 192.168.li 3 adet IP bloğu olduğunu öğreniyorum. Ayrıca bir adet de özel 10'lu bir IP bloğu olduğunu, ona sadece kendisinin erişiminin olduğunu, test kapsamında taramamıza da gerek olmadığını belirtiyor.

Eternalblue zafiyetinin henüz taze olduğu zamanlar. Bir makinede zafiyeti tespit ediyorum ve RAM üzerinden yetkili bir kullanıcı adı / parola yakalıyorum. Ardından pass-the-pass yöntemiyle diğer sunucularda deniyorum ve onlarca sunucuyu ele geçiriyorum. Durumu benimle izleyen BT yöneticisi arkadaş oldukça şaşkın. Ayrıca varsayılan yetkiyle bırakılan (ki birçok yerde buna defalarca rastladım) bir Postgresql veritabanı bularak verileri topluyorum.

Ama en eğlenceli adım bunda sonra geliyor. BT yöneticisi arkadaşın IP ve MAC adresini tespit ediyorum. Kendi bilgisayarımın IP ve MAC'ini bunlarla değiştiriyorum. Bir bakıyorum ki, daha önce ulaşamadığım 10.lu IP bloğuna da ulaşıyorum ve oradaki zafiyetleri de tespit ederek rapora ekliyorum.

Aradan iki yıl kadar geçiyor. Bambaşka bir ortamda, sektörden bambaşka arkadaşlarla sohbet ederken, (meğer BT yöneticisi ortak arkadaşmış) yapılmış olan o pentestten söz açılıyor. Meğer BT yöneticisi arkadaş birçok kişiye o günü anlatmış.

Birçok müşterimizde karşılaştığımız bir durum. Çok eskiden kalan özgün bir yazılım var ve bu yazılımın üzerinde çalıştığı eski bir Windows-7 ya da Windows-2008 sunucu var. Kurum çalışan yazılımı bozmaktan çekindiği için hiçbir zaman o makineye dokunmuyor ve Eternalblue gibi ölümcül zafiyetli bir makine o ağda yaşamaya devam ediyor.

Testimize başlıyoruz. Yüz otuz kadar makineden sadece bir tanesinde bu zafiyet var. Müşterimiz aslında zafiyetin farkında, geçen yıl başka bir firmanın yaptığı pentestte de bulunmuş. Ancak tek bir makine diye pek önemsememişler.

Makineyi ele geçirdikten sonra RAM alanı üzerinden backupadmin gibi özel bir hesabı ele geçiriyoruz. Ardından 69 adet makineyi bir buçuk saat kadar zaman dilimi içinde ele geçiriyoruz. Çoğuna Uzak Masaüstü ile bağlanıp ekran görüntüsü alıyoruz. Müşterimiz bu zafiyetin ciddiyetini şimdi çok daha iyi anlıyor. Yıllardır ayakta olan bu makinenin bir saat içinde kapatıldığını ve sistemden çıkarıldığını öğreniyoruz.

Henüz sahaya yeni çıktığım dönemler. Müşterimiz yerli ama uluslararası ürün geliştiren bir yazılım firması. Daha önceden bir pentest yaptırmışlar ancak içlerine sinmemiş. Bir defa daha yaptırmak istiyorlar.

Gündüz devam eden eğitimlerim var. O yüzden akşam gece yarılarına kadar bakıyorum. Uygulamaya yüzeyi çok geniş, portal üzerinde onlarca form sayfası mevcut. İki hafta boyunca her yeri didik didik inceliyorum. En son bir uygulamanın profil sayfası dikkatimi çekiyor. Profil resmimi değiştirirken .txt uzantılı bir dosya gönderiyorum. Gittiği yeri PATH değerini okuyarak tespit ediyorum ve gerçekten de o dosya metnine ulaşabildiğimi görüyorum.

Klasik bir "file upload" zafiyetinden başkası değil. DVWA gibi bir PHP sayfası üstelik. Ekip arkadaşım hazır bir webshell atıyor. Ancak çalışmıyor. Ben çok kısa bir "one-liner" shell hazırlayıp yüklüyorum. Bingo, shell alıyorum. Ardından "arp -a" komutunu çalıştırarak iç ağ keşfine başlıyorum.

Buradaki zafiyet bir pentestçi için bir anlamda oldukça kolay, ancak asıl zorlu taraf, bu kadar büyük bir alanda zafiyetin varlığını yakalayabilmek.

Yazılım firması uzaktan shell alınabildiğini görünce çok etkileniyor. Ben ise Güvenli Yazılı Geliştirme'nin bizzat yazılımcılar tarafından anlaşılmasının ne kadar önemli olduğunu bizzat yaşıyorum.

Türkiye'nin en önemli kamu kurumlarından birinin sızma testini yapıyoruz. Kurum ağı o kadar büyük ki yaptığımız taramalar ayları buluyor.

Bu süreçte en bilinen zafiyet tarama aracından da (Pro sürümünden) istifade ediyoruz. Çok işimize yarıyor ancak bazı konularda tıkandığımızı hissediyorum.

Kurum ağının çok büyük olması, yetkili arkadaşların bir kısmının yeni tayin olmuş olması gibi hususlardan dolayı envantere bile hâkim olunamadığını fark ediyorum. Hangi ağ bloğunda hangi bileşenler var, hangi servisler aktif bu konuda bir haritalama ihtiyacı olduğunu gözlemliyorum.

Kendi kendime diyorum: "Acaba ağ bloğunu tararken temiz bir da gösterge ekranı üzerinden tüm servisler ayrı ayrı paket olarak sunulsa, söz gelimi FTP'ler ve anonim erişimler, ayrı ayrı veritabanları, web portalleri, NFS, webdav, aktif dizin makineleri vs. gibi her biri ayrı ayrı tek bir ekranda sunulsa nasıl olur?"

Çeşitli sızma testi uzmanı arkadaşların fikirlerini alıyorum. Hepsi oldukça mantıklı olduğunu ve çok faydalı olacağını dile getiriyor.

Bilishim firmasını kurup ofisimizi açınca, ufak ekibimizle bu gözlemi projelendirmeye karar veriyoruz ve üzerinde çalışmaya başlıyoruz. Ortaya Nettarsier isimli Kurumsal Zafiyet Tarama Aracımız ortaya çıkıyor. Nettarsier, periyodik tarama ve görev atama benzeri özellikleriyle Siber Operasyon Merkezlerinde çalışabilecek oldukça işlevsel bir ürüne dönüşüyor.

Bu arada tarsier yani Türkçe adıyla maki, dünyada gözleri bedenine göre en büyük ve en sevimli orman hayvanıdır.

Sızma testi yaptığımız müşterilerimizin çoğuyla aramızda güvene dayalı bir dostluk başlar. Bir yıl önceki pentestimizden memnun kalan müşterimiz bu nedenle bu yıl da bizimle çalışmak istediğini bildirdi. Süreçleri tamamladık ve penteste başlıyoruz. Çözümlerini Java tabanlı geliştiren müşterimiz özellikle Log4j zafiyetinin de etkisiyle Java SpringBoot zafiyetlerine özenle bakmamızı rica ediyor.

Bu şekilde taramalarımızı yaparken gerçekten de iki tane kritik SpringBoot zafiyeti tespit ediyoruz ve bildiriyoruz. Bunlardan bir tanesi heapdump dosyasına erişimle ilgili.

Kendisi de çok iyi bir yazılımcı olan müşterimizin bu bulgu ilgisini çekiyor ve heapdump dosyasını inceliyor. Burada JWT token secret key değerini açık metin olarak okuyabildiğini tespit ediyor. Bu bulgu benim de çok ilgimi çekiyor ve heapdump'ı birlikte incelemeye devam ediyoruz. Bir süre çeşitli özel wildcard değerleriyle arama yapınca kullanıcı adı ve parolaları da açık metin olarak okuyabildimizi görüyoruz. Zafiyetin ne kadar kritik olduğunu birlikte keşfetmiş oluyoruz. Gerçekten de cache, env ve heap dump gibi ortamların mümkün oldukça dış dünyaya kapalı kalması gerektiğini bizzat görüyoruz.

Yazılım mimarisi bilmenin siber güvenlik anlamında ne kadar önemli olduğuna bir defa daha tanık oluyorum.

Testini yaptığım web portalı düzenli olarak pentestten geçmiş ve WAF tarafından çok başarılı şekilde korunuyor. Öyle ki herhangi bir request işleminde yaptığım her bir manipülasyon etkisiz kalıyor. Saatlerce hiç ara vermeden birçok yöntem deniyorum ama dişe dokunur özgün bir şey bulamıyorum.

Portalın özelliklerini deşmeye devam ediyorum. Bir sekmede "Arkadaşını Davet Et" özelliği mevcut. Önce işlevsel gerçek bir test mesajını kendi e-posta adresime gönderiyorum. Bir mesaj içeriği ile birlikte ref linki ulaşıyor. Sonra tekrar gönderirken proxy üzerinden giden trafiğe bakıyorum. Sadece e-posta ve mesaj parametreleri var, giden linki maalesef manipüle edemiyorum. Acaba link diye bir parametre eklesem nasıl tepki verir diye düşünüyorum ve bu şekilde repeater üzerinden gönderiyorum. Gelen e-posta mesajında normalde gelmesi gereken link adresinin hiç olmadığını görüyorum. Dolayısıyla sadece benim göndermiş olduğum mesaj içeriği görünüyorum. Dolayısıyla bu mesaja bırakabileceğim sahte bir linkin oltalama saldırılarında başarılı şekilde kullanılabileceğini tespit ediyorum.

En ilginç bulguların, iş akışını anlayarak oluşturulan minik senaryolar üzerinden tespit edildiğini gözlemliyorum.

Şehir dışındaki EKS testinden nihayet ofise geldik. Herkes gittikten sonra ben başka bir web pentestine bakmaya devam ediyorum.

Biraz profil resmi yükleme üzerine yoğunlaşıyorum. Exif gibi yöntemleri deniyorum ama sonuç alamıyorum. Ardından bu sitenin otomatik olarak üyelerine bir PDF üyelik belgesi çıkardığını fark ediyorum. PDF oluşturma scriptlerinde genelde ciddi zafiyetler olduğunu biliyorum ve biraz kaynak kodunu inceliyorum. Orada ilginç bir URL adresi dikkatimi çekiyor. Üyelik belgesinin altındaki QR Kodu oluşturan bir link olduğunu anlıyorum. Linke gittiğimde ilk başta karmaşık gelse de biraz temizleyip Base64 kısmını decode edince aslında bunun üye no üzerinden QR kodu çağıran bir adres olduğunu anlıyorum. Doğal olarak aynı formatta rasgele bir değeri Base64 ile encode ederek deniyorum ve gerçekten o numaraya uygun QR kodu oluşturuluyor. Sanırım sahte evrak düzenlemenin bir çeşit yolu açılmış oluyor.

Aradan saatler geçmiş ve fark ediyorum ki henüz montumu bile çıkarmamışım.

Müşterimiz benimle temasa geçiyor, mobil uygulamalarında aldıkları tüm önlemlere rağmen bir şekilde race condition zafiyetinin istismar edildiğinden şüphelendiklerini bildiriyor.

Durumu birlikte inceliyoruz, gerekli simülasyonları yaptığımızda herhangi bir olumsuzluk görmüyoruz. Kod tarafına ilişkin birlikte kafa yapıyoruz, 5 TL.lik bir üründen normal bir kullanıcı 20 tane alacaksa doğrudan yüz TL. ödeyerek alır ancak bir kişi art arda 20 defa alışveriş yapıyorsa orada yanlış bir desen vardır. Kod içinde bu davranışı yakalayan ve gerekirse o kullanıcıyı bloklayan bir mekanizmaya ihtiyacın altını çiziyoruz. Böylesi bir durumda TOO_MANY_REQUESTS şeklinde bir hata kodu üreten ve aynı zamanda nextAttemptinSeconds şeklinde kullanıcı tabanlı request isteklerini takip eden bir mekanizma.

Herhangi bir web portalının en hassas paneli doğal olarak account yani hesap ve kimlik bilgilerinin düzenlendiği paneldir. Çünkü burada tespit edeceğiniz en ufak bir zafiyet doğrudan hesap ele geçirme ya da hak yükseltme saldırısı anlamına gelmektedir.

Bu mantıkla API üzerinden hesap profil sayfasını proxy üzerinden inceliyorum. Güncellemek istediğimde sadece e-posta adresini güncellememe müsaade ediyor. Ben (en az yetkili kullanıcıyla oturum açmış olarak) yine de diğer kullanıcı yönetim trafiğinden elde ettiğim kullanıcı adı, soyadı, parolası, rolü gibi parametreler ekleyerek POST isteğini gönderiyorum. Ancak POST isteğim kabul edilmiyor. Başka kullanıcıları hedef alan istekler gönderiyorum ve kabul etmiyor.

Bu defa POST isteğini PUT olarak yeniliyorum ve tekrar deniyorum. Kullanıcı kendi ad ve soyadını normalde POST ile değiştiremezken ve 400 hatası alırken PUT ile değiştirebiliyor. Bu ufak ilerleme bana heyecan veriyor. Hemen rol olarak admin yetkisi vermeye çalışıyorum. Ancak yemiyor. Başka kullanıcılara ait verileri değiştirmeye çalışıyorum, başarılı olmuyor. Mevcut bir e-posta adresiyle yeni hesap açayım diyorum, izin vermiyor.

Başka neler yapabilirim düşünmek üzere biraz mola veriyorum.

Yaptığımız sızma testi yüzeyi oldukça geniş. Birçok alt portal mevcut. Takım arkadaşımız bu yüzeyi incelerken dosyayoneticisi şeklinde bir portal keşfediyor. Normalde portal arayüzü login ekranıyla geliyor ve kullanıcı adı parola bilmeden girmeniz imkânsız.

Takım arkadaşımız görünenin ötesini görmeye karar veriyor. Trafiği tek tek kestiğinde, aslında sağ üst köşede yer alan fonksiyonları yetki gerekmeden kullanabildiğini tespit ediyor. Öncelikle upload fonksiyonu ile bir dosya yüklemeyi deniyor. Sunucu "server error" mesajı dönüyor. Müteakiben createFolder fonksiyonu ile dizin yaratmayı deniyor. Yine "server error" mesajı alıyor.

İlk aşamada istediği neticeye ulaşamamış olsa da, bulmuş olduğu bu "path disclosure" zafiyetini sömürmenin başka yollarını aramaya devam ediyor.

Bugün Coinbase sitesindeki bir zafiyetin 250 bin dolar ödül aldığını okudum. Gözümde bizim de yaptığımız bazı pentestler canlandı. Henüz yine sahaya yeni çıktığım dönemlerdi. Bitcoin alım satım işinden oldukça yüklü para kazanmış genç bir arkadaş bir bitcoin alım satım sitesi açmıştı. Sızma testi için benimle temas kurdular.

İlk iş olarak sitenin kendisine ping attığımda IP adresinin gelmediğini gördüm, sanırım bir tür kısıtlama getirmişlerdi. Basit bir wget komutuyla IP.yi doğal olarak öğrendim. IP adresini o arkadaşa ilettiğimde oldukça şaşırmış ve nasıl bulduğumu sormuştu. Ne diyeceğimi bilememiştim.

Php tabanlı siteyi inceleyince özellikle sql sorgularında zafiyet olduğunu hissettim. Burpsuite ile kestiğim trafiği sqlmap sorgusuna atınca gerçekten veri tabanını ele geçirdiğimi anladım. Sürekli lab ortamında yaptığım bir şeyi gerçek hayatta görmek ilginç bir duyguydu.

Bir iki yıl sonra o arkadaş aklıma geldi ve hâl hatır sormak için aradım. Siteyi başka birilerine satmıştı. Ancak asıl üzücü yanı, o dönemki ortağı 500 bin dolarlık cüzdanı alıp Bulgaristan'a kaçmıştı. Kendisi ise asıl mesleği olan gözlükçülüğe geri dönmüştü. Ama ses tonundan sanki artık daha huzurlu olduğunu hissettim.

Siteyi geliştiren yazılımcı arkadaş (ki hala yakın dostumdur) sağolsun başka işler de yaptı ve onların da pentestlerini bize getirdi. Bunlardan bir tanesi hem alım satım sitesiydi hem de bir IOC projesiydi. Buradaki en ilginç zafiyetlerden bir tanesi, para aktarırken eksi değer girince bakiyenizin azalacağına artmasıydı. Diğer bir zafiyet ise bir kişinin kendi kendine para gönderebilmesinin engellenmemiş olmasıydı. Ödeme

sistemlerindeki ciddi bir zafiyet tüm sistemin bir günde çökmesini sağlayabilir.

Türkiye'de sürekli verilerin çekildiğini, dump edildiğini ve dark web ortamında satıldığına tanık oluyoruz. Çoğu defa bunun SQL Injection ile veri tabanının dump edilmesi şeklinde olduğu düşünülüyor. Ancak günümüzde toplu verileri dump etmenin en olası yolu IDOR ve Rate Limiting zafiyetlerinin birlikte istismar edilmesi ile verilerin ardışık olarak çekilmesi sanıyorum.

Bugün önemli bir portalın pentestini yapıyorum. Doğal olarak ilk hedefim kullanıcı ve kimlik yönetimine ilişkin konular. Bir kurumsal kullanıcının bilgilerini proxy üzerinden inceliyorum. Portalde maskelenmiş olan kritik verilerin proxy üzerinde rahatlıkla okunabilir olduğunu görüyorum. Daha da önemlisi bu kurumsal kullanıcı verisinin nümerik olarak çekildiğini görüyorum. Ardından bu işlemi nümerik olarak ardışık olacak şekilde gerçekleştiriyorum. Binlerce kullanıcının tüm verisinin ardı ardına önümde aktığını görmek pek şaşırtmıyor. Şeytan ayrıntıda gizlidir diyerek ekran görüntülerimi alıyorum.

Şeytan ayrıntıda gizlidir demişken. Saldırı her zaman bir sistemin ya da uygulamanın yumuşak karnından gelir. Yeni bir İK web portalının testini yapıyorum. Oturum açma ve kimlik bilgileri oldukça güzel korunmuş durumda Pek hareket edemiyorum. Uygulama içinde dolaşırken kendi kullanıcımın bordro bilgilerini görmek istiyorum. Bu esnada pop-up bir ekranda yeniden kullanıcı adı ve parolamı istiyor. Hemen gözlerim parıldıyor, çünkü burada herhangi bir CAPTCHA koruması aşikar şekilde yok. Kullanıcı adı ve parolayı girip trafiği proxy'ye atıyorum. Ardından başka bir test kullanıcısının üzerinde sözlük saldırısı yapıyorum. Bingo, doğru parola ile eşleşince paket boyutu değişiyor. Başkasının kimlik bilgisini (potansiyel olarak) ele geçirebiliyorum.

Bir sızma testinin başarısı iş akışını ve sistemin mantığını anlamakla doğru orantılı. Bu nedenle bazen geri çekilip, arkaya yaslanıp kuşbakışı görünümüne geçmek çok faydalı olmakta.

İncelediğim web portaldeki kullanıcının zaten tüm yetkilere ve her varlığa erişimi olduğunu görüyorum. Müşterimden bana daha az yetkili bir kullanıcı da açmasını ve paylaşmasını rica ediyorum. Ardından şöyle bir senaryo deniyorum:

1. Az yetkili kullanıcıyla oturum açıp proxy'de (çerez bilgilerini tutmak amacıyla) saklıyorum

2. Yönetici yetkisine sahip kullanıcıyla oturum açıp kritik verilerin yer aldığı adrese giriyorum ve onu da proxy'ye atıyorum

3. Az yetkili kullanıcının çerezini buraya yapıştırıyorum ve trafiği yeniden gönderiyorum. Aynı veriye ulaşabildiğimi görüyorum.

Sonuç olarak hedef adresi bildiği taktirde az yetkili kullanıcının sonuç alabilme potansiyelini göstermiş oluyorum.

Web portal pentestlerinde çokça karşılaştığımız zafiyet yüzeylerinden bir tanesi ise mesajlaşma özelliği oluyor. Genelde yazılımcılar işlevsel düşündükleri için bir saldırganın sahte mesaj atmak isteyebileceğini hesaba katmıyorlar.

Testini yaptığım portalde "Bize Ulaşın" seçeneğini proxy ile kesiyorum, gönderdiğim mesajda Ad Soyad, eposta ve telefon numarası normalde read-only ve değiştirilemezken ben dilediğim gibi üzerinde oynuyorum. Karşı tarafa hazırladığım sahte mesajı rahatlıkla gönderebiliyorum.

Özellikle hakaret ve hukuki boyutu olabilecek böylesi bir zafiyet yüzeyinin her zaman dikkate ele alınması gerekiyor.

Bugün nispeten büyük bir ağın pentestini yapıyoruz. Tüm sistemler yeni ve sorumlu arkadaşlar oldukça ilgili olduğu için çok bariz zafiyetler bulmuyoruz. Yine de ağ blokları içindeki veri tabanlarını ilgili portları üzerinden tarayan nmap sorguları çalıştırıyoruz. Tahmin ettiğimiz gibi MSSQL (ve Oracle ile Mysql) veri tabanlarına varsayılan yetkiyle erişmek mümkün olmuyor. Ancak GET ve POST isteklerini kaydeden Mongodb ve veritabanı arabelleği işlevi gören Redis'e ilgili portlarından erişmek mümkün oluyor. Benzer bir durumla çoğu yerde karşılaşıyoruz. Bu duruma bazen Postgresql'i de eklemek mümkün oluyor. Raporumuzda paylaşmak üzere ekran görüntülerini alıyoruz.

Kurumsal bir web portalının yazılımını yapıyorum. Giriş için resmi yetkiyle oAuth üzerinden oturum açma yeteneği mevcut. Bu durumda neler yapabileceğimi düşünüyorum ve aklıma 3 farklı senaryo geliyor:

* Yetkili hesaptan girişin tüm adımlarının ve oAuth süreçlerinin tekrarlanması, redirectUrl dahil parametrelerin manipule edilmesi

* Yetkisiz bir giriş hesabıyla tekrarlanması, yetkili giriş trafiğinin taklit edilmeye çalışılması

* Yetkili olduğu bilinen hesaptan sahte şifreyle giriş denemesi yapılması

Tüm bu senaryoları denemek için kolları sıvıyor ve teste kaldığım yerden devam ediyorum.

Bayram olması çoğu iş alanı için dinlenme zamanı, bazı özel ve hassas müşterilerimiz için test edilebilme zamanı, dolayısıyla bizim için de çalışma zamanı. Bu yüzden temel ziyaretler dışında sürekli pentest yapıyorum.

Kendisi de aktif pentest yapan bir misafirim geliyor. Merakından, "her defasında kendini tekrarlamıyor musun, ya da sıkılmıyor musun?" diye soruyor. "Hayır" diyorum, "çünkü her bir pentest yüzeyi ayrı bir akış ve ayrı bir hikaye barındırıyor. Oradaki hikayeyi okuyarak zafiyet aramak her defasında farklı bir deneyim".

Testini yaptığım web portalının şifre yenileme linkini inceliyorum. Linke tıkladığımda portal adresi sonunda sadece USER_ID değeri geldiğini görüyorum. Üstelik doğrudan mevcut şifreyi bilmemi beklemiyor. Ancak USER_ID değeri tahmin edilebilecek ya da ardışık saldıyla bulunabilecek bir değer değil. Araştırmaya devam ediyorum.

Admin hesabıyla dolaşıp tüm trafikten önemli olanları kaydediyor ve inceliyorum. Amacım önemli API adreslerini tanımak. Baya bir denemeden sonra nihayet kullanıcıların USER_ID değerini de veren bir GET trafiği yakalıyorum. Ardından yetkisiz kullanıcıyla giriş yaparak ona ait Authorization Bearer değeri ile aynı GET isteğini gönderiyorum. Yetki kontrolü yapılmadığı için Admin dahil tüm hesapların USER_ID değerini görebiliyorum. Bu bilgiyle Admin dahil herhangi bir hesaba şifre yenileme yöntemiyle account takeover yapabiliyorum.

Yazılım geliştiricilerin muhakkak ki "Güvenli Yazılım Geliştirme" eğitimi almaları gerektiğini düşünüyorum. Bu eğitim baştan sona uygulamalı olmalı ve özellikle temel düzeyde de olsa bir hacker tarafından hangi web zafiyetlerinin nasıl istismar edildiği gösterilmeli.

Son günlerde yaptığım pentestlerde özellikle dosya yüklemede yalnızca ön katmanda uzantı kontrolünün yapıldığını ve bunun rahatlıkla atlatılabildiğini görüyorum. Bu durumda atılacak bir web shell'in etkisiz kalmasını çoğu defa sunucuda yer alan EDR çözümü engelliyor.

Dosya zafiyetlerini sömürmek iki katmandan oluşuyor aslında. Birincisi zararlı dosyayı hedef sistemi gönderebilmek, ikincisi ise o dosyanın konumunu bularak ona ulaşabilmek. Yazılım mimarisini bunları düşünerek kurgulamak bu açıdan çok önemli.

Bugün attığım asp shell'in silindiğini farketsem de gelen response metninde konum bilgisinin açıkça yer alması konunun üzerine gitmem gerektiğinin göstergesiydi. Öncelikle somut bir adım atma anlamında içinde XSS yüklü bir html dosyası yükledim ve kolaylıkla XSS'i çalıştırabildim. Asp alternatifi olarak ne yükleyebilirim diye düşünürken .cer uzantılı bir web shell kullanmaya karar verdim. İmzasını değiştirmek maksadıyla üzerinden bir miktar oynadıktan sonra gönderdim. Bingo, sunucunun dizinlerinde dolaşabiliyordum.

Basit ve etkili yöntemler.

Bugün müşterimizden elimize, kendilerine ait kurumsal mobil uygulamanın dosyaları geldi ve testlere başlıyoruz. Uygulama TCKN kullanarak kişileri tanımlıyor ve onlara görevler atanmasını sağlıyor. Ekip arkadaşımız uygulamanın işlevlerini incelerken, Burpsuite üzerinden TCKN.yi değiştiriyor ve kendi babasının TCKN.sini veriyor. Bir de bakıyor ki gerçekten ona ait bilgiler geliyor. Dolayısıyla binlerce kişinin verisini potansiyel olarak çekmenin mümkün olduğunu farkediyor. Sonra annesinin TCKN.si üzerinden ona görevler atayabildiğini görüyor.

Birçok mobil uygulamanın resmi işlemlerde kullanıcıyı Mernis üzerinden onayladığını görmekteyiz. Bu yeteneğin istismar edilebiliyor olmasının her zaman akılda tutulması gerektiğine bir defa daha tanık oluyoruz.

Ödeme sistemleri çok hassas uygulamalar. Gözden kaçmış ufacık bir boşluğun maliyeti doğrudan para oluyor. Bu nedenle ödeme sistemi uygulamalarının pentestinde her türlü ayrıntıya kafa yormamız gerekiyor. Bugün yine üzerinde çalıştığımız ve bu anlamda en ilginç zafiyetlerden bir tanesi race condition zafiyeti. Bir kullanıcıdan başka bir kullanıcıya para aktarırken, Turbo Intruder ile aynı anda onlarca isteği gönderiyoruz. Gönderdiğimiz para miktarı ile bakiye arasında bir tutarsızlık olduğunu fark ediyoruz. Bu zafiyetin ortaya çıkmasının ve giderilmesinin nispeten zor olmasının nedeni, çoklu sunucu mimarisi ve load balancer yapısı. Aynı anda çoklu istek gelince bu yapıdaki ahenk bozulabiliyor. Bu nedenle müşterimizle bu zafiyetin üzerinde günlerce çalışıyoruz, algoritma ve mimari ta ki tüm durumlarda tutarlı olana kadar.

Kritik yazılımların KVKK anlamında kendini koruması çok önemli. Bu nedenle bu yazılımların bir kısmı oturum açan kullanıcının IP bilgisini saklar. Böyle kritik bir uygulamaya yönelik yaptığımız testimizde, acaba buna yönelik ne yapabiliriz diye düşünüyoruz. Giden trafiği proxy ile kestikten sonra X-Forwarded-For etiketini kullanarak sahte IP değeri göndermeyi deniyoruz. Gerçekten de uygulama bu sahte IP.yi logluyor. Uygulamanın işlevine zarar vermese de herhangi bir siber olay vukuunda hukuki anlamda ne kadar önemli olduğunu vurgulayarak durumu müşterimize raporluyoruz.

Her teknolojik yenilik yanında güvenlik anlamında yeni sorunlar da getiriyor. Bulut teknolojileri, sanallaştırma ve yük dengeleyici çözümleri, bir mobil uygulamanın aynı anda milyonlarca insana hizmet sunmasını sağlarken, bir yandan da kendine özgü yeni zafiyet yüzeyleri getirmektedirler. Bunlardan en ilginçlerinden bir race condition durumu.

Elimizdeki mobil bankacılık uygulamasında bu zafiyeti inceliyoruz. Test ortamında hesabımıza para yükletiyoruz ve bakiyemizi not ediyoruz. Ardından proxy üzerinden karşı tarafa aynı anda yüzlerce ödeme isteği gönderiyoruz. Bunlardan başarılı olan işlemler ile bakiyeyi karşılaştırıyoruz ve bakiyede tutarsızlık olduğunu görüyoruz. Bu durumu müşterimizin yazılımcı ekibine hemen aktarıyoruz.

Aslında temelde yazılım kodu doğru olsa da, ortam yük dengeleyici ile birlikte çoklu sanal ortam olunca race condition zafiyetinin ortaya çıktığını birlikte tespit ediyoruz.

Müşteri kurumu içinde yaptığımız lokasyon keşfinde, açık alanda ama biraz kuytu bir köşede Ethernet girişi olan bir priz keşfediyoruz. Hemen sessizce o girişten nerelere erişebileceğimizi test ediyoruz. Ancak girişin canlı olmadığını anlıyoruz.

Bu durum olumlu olsa da, yine de müşterimize bu konunun önemini hatırlatıyoruz.

Çok güzel bir otelin pentestindeyiz. Kullandıkları kurumsal masaüstü yazılım da dikkatimizi çekiyor. Uygulamanın aynı zamanda API desteği de mevcut. API yardım dokümanını incelediğimde authentication ile ilgili en ufak bir fonksiyon tanımı olmadığını görüyorum. Dolayısıyla API üzerinden POST istekleri göndererek rezervasyon ve ödeme gibi verileri değiştirmem mümkün. Sanırım masaüstü uygulamasını geliştiren yazılım firması sadece masaüstü oturum açma ve kullanıcı yetkilendirmesine güvenmiş. Durumu bir yandan ilgili teknik arkadaşlara izah ederken bir yandan da raporumuza durumu ekiyoruz.

Bizi misafir eden turistik otelde misafir ağından giriş yapıyoruz. İç ağa ve sunucu bloğuna erişim olduğunu tespit ediyoruz. İç ağdan ise çok ayrı lokasyonlara denetimsiz ulaşabildiğimizi görüyoruz.

Daha sıkılaştırılmış bir mimari konusunda müşterimizi uyarıyoruz.

Müşteri web sitesinde bazı Word dosyaları olduğunu görüyoruz. İndirip meta verilerini incelediğimizde kurum çalışanları hakkında bilgi toplayabildiğimizi görüyoruz. Bazı PDF dosyalarının ise resmi dokümanlar olduğunu ve kurum yetkililerince imzalandığını fark ediyoruz. Müteakip bir "spear phishing" saldırısı için hem kişi hem de senaryo kurgusu anlamında çok faydalı olacağını gözlemliyoruz. Tüm bu hususları raporumuza ekliyoruz.

Hedef sistemin, siber saldırılara karşı Cloudflare arkasında konumlandırıldığını farkediyoruz. Yaptığımız ufak bir Censys sorgusu ile yine de bir adet gerçek IP.ye ulaşıyoruz. Bunun ise IoT cihazlarla görüşen bir MQTT protokol sunucusu olduğunu anlıyoruz. Çok rahat bir şekilde DDoS saldırısı yapabileceğimizi tespit ederek bu durumu müşterimize iletiyoruz.

Müşterimizin kendi geliştirmiş olduğu ufak bir kariyer yazılımının pentestindeyiz. Müşterimiz de konuya ilgi duyduğu için projeksiyon cihazından canlı olarak bizi izliyor. Biz de yaptığımız her adımı bir ders anlatır gibi paylaşıyoruz.

Kariyer yazılımına bir test e-posta hesabıyla üye olduk. Giriş yaptıktan sonra e-postamı değiştirebildiğimizi gördük. Acaba profilimizde mevcut bir kullanıcının e-postasını girsek ne olur diye düşündük ve bunu denedik. Gerçekten de sistem başkasına ait bu e-postayı kabul etti. Biraz inceleyince sistemdeki hesap bütünlüğünün bozulduğunu anladık. Yaptığımız denemelerde, kurban e-postanın iki parolasının da sistem tarafından kabul ediliğini, saldıran hesabın ise veri tabanında kayıtlı olduğu halde sistem tarafından tanınmadığı ve hatta yeniden üye olmanın mümkün olduğunu gözlemledik.

Müşterimizin iç ağına VPN üzerinden bağlanıyoruz. Özellikle bir IP bloğunun IP kameralara ait olduğunu görüyoruz. Doğal olarak öncelikle en çok karşılaştığımız zafiyeti yani varsayılan kullanıcı adı ve parolayı deniyoruz. Hepsinin değişmiş olduğunu anlıyoruz. Ardından yine kameralarda en çok görülen zafiyetlerden olan LFI aklımıza geliyor. http://IP/../../../../../../../../../../../etc/passwd dosyasını okumayı deniyoruz. Ancak tarayıcıdan gidince kullanıcı adı ve parola istiyor. Bu defa Burp Suite üzerinden get isteği yapıyoruz. Gerçekten de LFI çalışıyor ve passwd dosyasını okuyoruz.

Passwd doyasını kırmak için ne yapalım diye düşünüyoruz. john normalde shadow ve passwd dosyalarının unshadow yapılmış halini kırar. Yine de john'u deniyoruz ve gerçekten bir tane hesabın kırıldığını görüyoruz: test / test1234

Bir mobil iOS uygulamasının pentestini yapıyoruz. Uygulamanın mesaj gönderme özelliği de var. iPhone olduğu için geliştiriciler hiç jailbreak olmuş bir telefonu hesaba katmamışlar. Test telefonumuz jailbroken olduğu için Downloads dizinine bir test.html dosyası koyduk. Tabi içinde XSS ile sonuçlanacak ufak bir javascript kodu da gömdük. Dosyayı karşı tarafa gönderdiğimizde XSS'in başarıyla çalıştığını gördük. Bu heyecanla başka birçok dosya türü de denedik ancak daha ileri götürecek başka bir saldırı vektörü bulamadık.

Çok önemli bir kurumun kapalı ağının testini yapıyoruz. Müşterimiz kapalı ağındaki mimarinin olabildiğince güvenli ve sıkılaştırılmış olduğundan emin olmak istiyor. Mimarinin bir bileşeni ise Asterix protokülüne dayanan bir mesajlaşma çözümü. Bu protokolü ilk defa duyuyoruz.

Ekibimiz durumu inceliyor. Protokolün teknik dokümanlarından mesaj ve veri formatı yapısını ortaya çıkarıyor. Ardından trafik dinlemesi yapıyoruz ve kaydediyoruz. Sonra Scapy ile söz konusu veri üzerinde protofol veri formatına uygun sahte veriler üretebileceğimizi, yine mesaj formatına uygun sahte kaynak ya da hedef adresler girebileceğimizi gösteriyoruz. Yeni bir şeyi keşfetmenin tadını çıkarıyoruz.

Yine mimari bileşenleri arasında yer alan bir VoIP çözümüne yönelik sahte trafikler üreten spoofing saldırısı düzenliyoruz. Bunu başarabildiğimizi VoIP uygulamasının kendi logları üzerinden gösterecek şekilde ekran görüntümüzü alıyoruz.

Kurumun ana omurgasını teşkil eden web portalının pentestini yapıyoruz. Bu maksatla uygulamanın API fonksiyonlarını daha iyi anlamak için swagger.json dosyasını Postman'da import ederek inceliyoruz.

Doğal olarak ilk hedefimiz kullanıcı profil sayfası. API üzerinden profili güncellemeyi deniyoruz. Normalde istemci tarafında olan e-posta ve telefon format kontrolünün burada olmadığını keşfediyoruz. Olmayaneposta ve Olmayantelefonno gibi değerler girerek ekran görüntüsü alıyoruz.

Güncelleme esnasında user_id değerinin etkisini merak ediyoruz. Başka bir test kullanıcısına ait user_id değerini mevcut olanla değiştirerek verileri yeniden güncelliyoruz. Gerçekten de normalde yetkimiz olmayan başka bir kullanıcının profil bilgilerin güncelleyebildiğimizi görüyoruz.

Tabi hemen hedef kişinin e-postasını yeniliyoruz. Ardından şifre yenile butonuna tıklıyoruz. Şifre yenileme linki bizim girmiş olduğumuz e-postaya geliyor. Sonuç: account takeover. IDOR zafiyetini neden bu kadar çok sevdiğimi yeniden anlıyorum.

Biraz daha kurcalayınca oturum token'ının otomatik sonlanmadığını, dolayısıyla oturumun otomatik kapanmadığını anlıyorum. Daha da önemlisi, oturumu kapatıp, Burpsuite'teki eski ürün listeleme POST isteğini gönderince sonucun olumlu geldiğini gözlemliyorum. Yani Authorization Bearer düzgün yönetilmiyor. Sonrasında ise uygulamanın sürekli Runtime Error hatası vererek çöktüğünü görüyorum. Daha sonra bunun asıl kaynağının kullanıcı rolünde zaten varsayılan değer olan 0 değerini atamamızmış.

Geliştirici ile hacker arasında ciddi bir farkı var. Geliştirici, sistem ya da yazılımın düzgün ve performanslı şekilde çalışmasına odaklıdır. Hacker ise o sistemdeki boşluklara ve onu nasıl kendi lehine istismar edeceğine yoğunlaşır. Temeldeki fark algıdan ibarettir.

Test ettiğimiz web uygulamasının API özelliklerini incelediğimizde manüel olarak log ekleme özelliğine sahip olduğunu görüyoruz. Sıradan bir kullanıcı yetkisiyle, gerekirse onbinlerce sahte ve yanıltıcı log üretebilmenin mümkün olduğunu anlıyoruz. Bu durumu ekran görüntüsüyle birlikte raporumuza ekliyoruz. Kurtlar puslu havayı sever, log kirletme ise havayı puslu hale getirmenin bir yöntemidir.

Yaptığımız şehirlerarası yolculuktan sonra müşterimize ulaşıyor ve öğleden sonra hemen testlerimize başlıyoruz. Bir yandan keşif faaliyetleri devam ederken, bir yandan da ARP zehirlemesi saldırısı deniyorum. Zehirleme mümkün olsa da public community metinleri getiren SNMP ve SAP Diag bulguları dışında gerçek bir parola verisine ulaşamıyorum.

Yine de SAP Diag bulguları ilgimi çekiyor. Birçok istemcinin bambaşka bir ağ bloğundaki bir sunucuya bağlandığını farkediyorum. Aşikar şekilde bu SAP Sunucusu. Ayrıca incelediğimde ağ paketinden yakalanan metinler olduğunu hissediyorum. Hatta SAP'ye bağlanmaya çalışan arkadaşın Windows kullanıcı adının da geçtiğini görüyorum. İçimden bir ses daha fazlası var diyor. Araştırmaya başlıyorum.

SAP trafiğini Wireshark üzerinden analiz etmeye yarayan bir eklenti keşfediyorum. Yalnız bu eklentiyi kullanabilmem için Wireshark'ı sıfıdan kod olarak indirip eklentiyle birlikte yeniden derlemem gerekiyor. Birçok başarısız adımdan sonra nihayet VMWare Ubuntu ortamında kuruyorum. Ancak eklentide görmem gereken filtre komutları nedense hala tanınmıyor. Biraz inceleyince Wireshark About menüsündeki Plugins sekmesinde eklentinin halen tanımlı olmadığını farkediyorum. sap.so uzantılı eklentiyi gerekli yol'a kopyalıyorum. Evet, nihayet SAP Diag eklentisi başarıyla çalışıyor.

Ancak bir sorun var, VMware ortamındaki Ubuntu bir türlü promiscous modda çalışmıyor, yani asıl makinedeki trafiği görmüyor. Birçok yapılandırma ve ayar denemesine rağmen nedense bir türlü trafiği yakalayamıyorum. Windows makinemden SAP Sunucusuna ping atıyorum gelmiyor ama Ubuntu içinden atınca geliyor. Kaybedecek zamanım yok, hızlı

olmalıyım. Virtualbox oramında promiscous modun çok daha pratik olduğunu biliyorum. Bu ortamda her şeyi yeniden kuruyorum ve promiscous modu açıyorum. Asıl makinemden ping atıyorum ve bingo, Ubuntu içinden artık yakalıyorum.

Fakat bu yeterli olmuyor. Çünkü yalnızca kendi asıl makinemi görüyorum. Diğer makinelerden ping atınca hiçbiri Wireshark tarafına düşmüyor. Aşikar şekilde Ortadaki Adam Saldırısı yapmam lazım. ARP zehirlemesini tekrarlıyorum ve bu esnada Ubuntu üzerinde trafiği dinliyorum. Gerçekten de trafik akıyor. Hemen kimlik bilgilerini filtreleyen komutları devreye sokuyorum ve beklemeye başlıyorum. Gerçekten de filtre açık durumdayken bazı ufak paketlerin düştüğünü görüyorum.

Paketi analiz ettiğimde, Expert Info isimli ufacık bir alanda bir okunabilir bir metin değeri görüyorum. Bingo, o IP kullanıcısına ait SAP parolasını yakaladığımı anlıyorum.

Sezgilerimin beni yanıltmadığını görmenin mutluluğunu yaşıyorum. Süreci zaten benimle birlikte takip eden müşterimizle ekran görüntülerini paylaşıyorum.

Penteste başlamak için müşterimizin ofisine geliyoruz. Testimizin siyah kutu adımını zaten bitirmiştik. Şimdi gri ve beyaz kutu testlerini tamamlamayı planlıyoruz. Siyah kutu testi aşamasında uzaktan sıfır bilgiyle şirketin dışarı açık AR-GE uygulamasının admin yönetim paneline yetkisiz erişim sağlayıp istediğimiz değişikliği yapabildiğimiz halde, şirketin teknik müdürü nedense pek etkilenmemiş gibi bir hava çiziyor. Hem biraz garipsiyorum, hem de içten içe biraz kızıyorum. Portal üzerinde normalde oturum açınca gelmeyen admin paneli, Burpsuite üzerinden tekil adımlarla ilerleyince geçici olarak geliyordu ve biz POST isteklerini manipüle edebildiğimizi keşfetmiştik.

Teste başlarken ilk iş olarak "katil"i denemeye karar veriyorum. Katil dediğim "Cain & Abel" isimli neredeyse 20 yıllık bir yazılım. Yaptığı işlev ise ARP zehirlemesi, yani router ve istemcilerdeki ARP tablolarını değiştirerek akan trafiğin üstünden geçmesini sağlayan eski ama etkin bir Ortadaki-Adam yazılımı. Uygulamayı çalıştırıyorum. Normalde "Cain & Abel" SSL ile şifrelenmiş trafiği dinleyemez. Ancak IMAP, POP3, http, ftp trafiklerinde geçen şifreleri yakalar. Ayrıca birçok durumda VOIP telefon görüşmelerini bile yakaladığına tanık oluyorum.

Gerçekten de daha bir dakika geçmeden önüme şifreler düşmeye başlıyor. Bunlardan bir tanesi şirketin ihracat işlerini yürüttüğü en önemli e-postaya ait. Kimlik bilgileriyle ilgili e-postaya başarılı şekilde giriş yapıyorum. Ardından teknik müdür arkadaşı davet ediyorum ve canlı olarak e-posta sayfasını gösteriyorum. Kısa ama güçlü bir sessizlik yaşanıyor.

Müşterimiz Türkiye'nin en büyük teknoloji şirketlerinden biri. Dışarı açık yüzlerce URL ve IP'yi taramamızı bekliyorlar. En az on yıldır her yıl zaten sızma testi hizmeti alıyorlar.

Bize liste vermelerine rağmen biz kendi keşif yöntemlerimizi de kullanıyoruz. Bunlardan bir tanesi tomnomnom araçlarıyla Web arşivinden kalan uç noktaların okunması. Bu şekilde hedefe yönelik birçok URL uç noktası keşfediyorum.

Bunlardan bir tanesi bir ufak bir doküman yönetim uygulaması. Üstelik kendim üye de olabiliyorum. Hemen üye olduktan sonra kendime ait evrakları proxy üzerinden analiz ediyorum. Evrakla ilişkili kendime ait bir kullanıcı id değeri okuyorum. Onunla biraz oynayınca başka kullanıcıların da tüm verilerini okuyabildiğimi görüyorum. Üstelik parolalarının da hash değeri var. Bir tanesinin parolasını rahatlıkla kırıyorum. O kişinin hesabını ele geçirmenin mümkün olduğunu görüp rapor yazmayı beklemeden müşteriyi haberdar ediyorum.

Sonra uç noktları incelemeye devam ediyorum. Bunlardan bir tanesi de İnsan Kaynakları uygulamasına ait. Bağlandığım URL adresinde CV ekleyebildiğimi görüyorum. Ekliyorum ve tekrar silmek istediğimde yine nümerik bir değerle karşılaşıyorum. Biraz inceleyince uygulamadaki CV.lerin tamamını silebileceğimi anlıyorum. Bunu da hemen müşterimize iletiyorum. Buna çok şaşırıyorlar. Çünkü henüz tamamen siyah kutu testi aşamasındayız ve bize birer test hesabı vermemiş durumdalar. Durumun ilginçliğini ben de o anda fark ediyorum. Meğer keşfettiğimiz uç noktası kimlik doğrulamanın bile arkasına ulaşmamı sağlamış. Asıl sonuç raporunu beklemeden bu bulgulara yönelik bir ara rapor hazırlıyoruz hemen.

Müşterimiz bir üretim tesisi, BT müdürü ise oldukça kıdemli ve özenli bir insan. Teste başlamadan önce bizi bakışlarıyla tartıyor.

İç ağ testine başlıyoruz. Her zamanki gibi en alt düzey yetkiden en yukarı doğru ilerliyoruz. İlk adım olarak misafir ağından giriş yapıyoruz. Yaptığımız taramada hiç ummadığımız bir şekilde DC.ler dahil sunucu bloğuna da erişebildiğimizi görüyoruz. Bunun ciddi bir mimari zafiyet olduğunu not alıyoruz. Taramalarımızda tüm makinelerin güncel ve güvenlik yamalarının geçilmiş olduğunu görüyoruz. İlk defa gerçekten kusursuz bir ağ bloğuna tanıklık ediyoruz bu anlamda.

Ancak bir şey dikkatimiz çekiyor. Advanced IP Scanner gibi basit bir araçla tarayınca gereğinden çok çok fazla paylaşım klasörünün açık olduğunu gözlemliyoruz. Bunlardan bir tanesi BT yöneticisi arkadaşın. Onun dizinlerine girip dolaşmaya başlıyorum. Kendisine ait VPN sertifikasının da gözümün önünde olduğunu fark ediyorum. İvelikle durumu bildiriyoruz. VPN ne kadar muhteşem bir protokol olsa da her zamanki gibi en zayıf nokta insan faktörü.

BT yöneticisi ne kadar özenli olursa olsun dışarıdan bir gözün sisteme bakmasının olası körlüğe karşı ne kadar önemli olduğunu bir defa daha anlıyorum.

Müşterimizin çok geniş bir ağ yüzeyi mevcut. Keşif aşaması bile oldukça uzun sürüyor. Bu nedenle sunucu bloğuna odaklanmanın en verimli yol olduğunu öngörüyoruz.

Ekibimizin pembe kapüşonlu üyesi Advanced IP Scanner ile sunucu bloğunu taradıktan sonra son dönemlerde üzerinde bulunan zafiyetlerle dikkat çeken ve web arayüzü de bulunan Vcenter'a odaklanmaya karar veriyor. Web portalına Nuclei çalıştırıyor ve "Dosya Yükleme Zafiyeti" olduğunu tespit ediyor. Bununla ilgili paketleri hemen Burpsuite üzerinden gönderiyor ve VCenter üzerinde RCE yani uzaktan komut çalıştırmayı root yetkisiyle başarıyor.

Tabi ilk iş olarak /etc/passwd ve /etc/shadow dosyalarını birleştirerek john şifre kırıcı aracı ile şifre kırmaya çalışıyor. Görünen o ki üründeki şifre politikası güçlü ve neredeyse bir gün geçmesine rağmen şifre kırılmıyor. Tüm bu durumların ekran görüntülerini alarak raporumuza ekliyoruz.

Bugün ofisimize gelen müşterimiz yanında geliştirmiş oldukları IoT ürününü de getirmiş, kullanıcıdan aldığı sensör verisini yorumlayan bir sistem. Aynı zamanda süreç ve işlemleri takip etmeye yarayan web tabanlı bir panele de sahip.

Cihazı alıp pentest ekibimize veriyorum. Bu esnada misafirimizle çay içerken sohbet ediyoruz. Hava güzel ve bahçedeyiz. Bu esnada yarım saat geçmeden ekibimizin SQL Injection ile rahatlıkla yönetim paneline girdiklerini ve tüm verileri çektiklerini öğreniyoruz. Misafirimiz neye uğradığını şaşırıyor. İlerleyen zamanlarda gömülü işletim sisteminin de pek güvenliği olmadığını, C tabanlı programın kaynak kodunun bile cihazın içinde olduğunu fark ediyoruz.

Bir binanın ana giriş kapısı neyse web teknolojileri de sızma testlerinde onu temsil ediyor.

Türkiye'nin en büyük finans kuruluşlarından birinin mobil uygulamasını test yapıyoruz. Bu bizim için hem çok güzel bir fırsat hem de gerçek bir sınav. Çünkü uygulamanın bizden önce onlarca defa test edildiği aşikâr. Güzel bir zafiyet çıkarmadan bu işi bırakmak istemiyorum.

Neyse ki yaz dönemi ve bu işe ayıracak zamanı bulabiliyorum. Android tarafta uygulamayı tersine mühendislik ile açıyorum ve incelemeye başlıyorum. Uygulamanın SSL Pinning (gelen giden trafiği görmeyi engelleyen bir mekanizma) koruması var ve onu atlatmak istiyorum. Aslında Frida ve Objection gibi araçlar var ama nedense zor yolu seçmek istiyorum.

Uygulama paketini dışarı aktarıyorum, üzerinde ufak bir kod değişikliği yapıyorum, sonra yeniden paketliyorum. Ardından yapay bir sertifika üreterek onunla imzalıyorum ve uygulamayı telefonuma yeniden kuruyorum. Yalnız uygulamanın içinde "in-app protection" mekanizması var, yani koddaki en ufak bir değişiklik ya da sahte sertifika kullanımı gibi bir durumu kontrol eden ekstra bir kod paketi. Başka çarem olmadığını görüyorum, olayı psikopata bağlıyorum. Her bir adımla, ilgili Smali kod parçacağını uygulama içinde buluyorum, siliyorum ya da değiştiriyorum, yeniden derleyip yeniden telefona yüklüyorum. Bu işlemi 500'ün üzerinde tekrarlıyorum. 12 gün boyunca hiç ara vermeden. Smali dili aslında donuk bir dil, hiçbir geri besleme alamıyorsunuz. Bunu aşmak için loglama mekanizmasını kendime özel yöntemler geliştiriyorum. Gerçekten çok sabır istiyor. Bir süre sonra uygulamanın Aktivasyon adımını tamamen devre dışı bırakabildiğimi görüyorum. Sonunda ise kendi profil sayfama ulaşabiliyorum. Ama bazı sayfa baya boş geliyor.

iPhone tarafında ise jailbreak yapılmış bir telefonda SSL Killchain ile SSL Pinning bypass yapmanın çok kolay olduğunu görüyoruz. Dolayısıyla hiç boğuşmadan sayfamıza ulaşıp trafiği kesebiliyor ve analizlere başlıyoruz. Bir de bakıyoruz ki başkasının profil sayfasını değiştirebiliyoruz. Bu o günün eğlencesi oluyor, kurumdaki arkadaşlar, başka yetkililerin profil resmini rakip futbol takımının amblemi ile değiştirmemizi istiyor ve yapıyoruz.

Bu ve bazı diğer ufak bulgularla raporumuzu sunuyoruz. Bizden bulgularımızı yazılım ekibiyle paylaşmamızı istiyorlar. Yazılım ekibinin başındaki arkadaş nedense ilk başta bize düşman gibi davranıyor, bulgumuzu da önemsiz göstermeye çalışıyor. Ancak konuştukça benim yazılım koduna ne kadar hakim olduğumu anlıyor. Verdiğimiz emeği ve ciddi yaklaşımımızı anladıkça o da takdir etmeye başlıyor ve teşekkür ediyor. Ben yine de kendime ders çıkarıyorum ve bir pentest sonucunda yazılım ekibiyle doğrudan bir araya gelmeme kararı alıyorum.

Aklıma başarısının sırrını soran vezirine parmak ısırma testi yapan Timur geliyor. Konu ne kadar farklı olursa olsun, bazı gerçekler ve parametreler hiç değişmiyor.

Herhangi bir pentestin ilk aşaması yüzey keşfi. Yani müşerimizin sanal evrendeki kapsama alanının ortaya çıkarılması. Pasif bilgi toplama isimli bu adımdaki işlemlerden biri DNS Zone Transfer. Yani hedef sistemin DNS sunucularında tanımlı ve normalde dış dünyaya kapalı olan tüm alt URL adresi ve karşılık gelen IP adreslerinin dökümlenmesi.

Ekip arkadaşlarımızdan biri bu şekilde yaptığı bir işlemin ardından hedef sisteme ait bir git alt alanı olduğu görüyor ve oraya giriyor. Biraz incelemenin ardından kurum portal uygulamasına ait bir projede kimlik bilgilerinin yer aldığını keşfediyor. O kimlik bilgilerini kullanarak kurum portalında giriş yapabiliyor.

Zafiyet yüzey keşfinin önem ve faydasını bu şekilde bir defa daha görüyoruz.

Müşterimiz AWS üzerinde serverless mimari üzerinde yenilikçi çözüm geliştiren bir firma. Bu yeni teknolojiyi inceleme beni de heyecanlandırıyor. Sistemi ve mimarisini inceliyorum, tabi ki her zamanki gibi tek tek tüm request ve response çağrılarını analiz ederek. Sisteme sıradan bir kullanıcı yetkisiyle oturum açmış durumdayım. Bu arada sayısal bir cookie değeri dikkatimi çekiyor, bu sanırım bir oturum çerezi. Çerez değerini 1 yapıyorum, aynen umduğum gibi (sayfayı refresh edene kadar) response artık admin yetkisiyle geliyor.

Her zaman eğitimlerimde anlattığım Mutillidae labındaki çok temel bir senaryoyu en uç bir bulut teknolojisinde yaşamak ilginç bir duygu veriyor.

On-prem çalışan portal tabanlı bir çözümün testini yapıyoruz. Dolayısıyla sistemi kendi Ubuntu makinemize kurmak ve her türlü ince senaryoyu çalıştırmak mümkün.

Portal üzerinde dolaşırken bir yandan da "forkstat - exec,exit" komutuyla herhangi bir işletim sistemi komutunun tetiklenip tetiklenmediğini takip ediyorum. Backup komutuyla tetiklendiğini görsem de manipüle edemediğimi görüyorum. Çünkü tüm sistem çeşitli docker makineler üzerine yürüyor.

Docker imajlarına baktığımda bunlardan birinin MongoDB imajı olduğunu görüyorum. /etc/hosts dosyasında imajın adını 127.0.0.1 ile eşleştirdikten sonra mongo shell ile veri tabanına bağlanmayı deniyorum. Tam da tahmin ettiğim gibi hiçbir kullanıcı adı ve parola sormadan veri tabanına bağlanıyorum ve veri tabanına her türlü yazma yetkimin olduğunu anlıyorum.

Bu blackbox anlamında çok ciddi bir zafiyet olmasa da sıkılaştırma anlamında güzel bir bulgu olduğunu düşünüyorum.

Pentest yapılması en zevkli konulardan biri de ajan kullanan kurumsal portallerin testi. İster makine öğrenmesi, ister siber güvenlik, ister ise sağlık ya da metrik ölçüm maksatlı olsun, ajan kullanan portallerin eli ayağı doğal olarak uç noktada çalışan küçük servislerden başkas değil.

Her zamanki gibi standart zafiyetlerin denetimi yaptıktan sonra ufak senaryolar kurguluyorum. Ajanın çalıştığı bir bilgisayarı ele geçirsem ve yönetici değilsem neler yapabilirim? Ajan ile sunucunun görüştüğü portu tespit ediyorum. Öncelikle ajanın binary dosyasını tersine mühendislik ile inceliyorum. Çok karmaşık gitmeden yerel IP adresi ile ilintili bir port tespit ediyorum. Ardından giden trafiği dinlediğimde gerçekten de sunucuyla etkileşime geçen portun bu olduğunu anlıyorum.

Porta tarayıcı üzerinden bağlanıyorum. Reponse olarak servis bilgisi ve sürümü gibi bilgiler geliyor. Aklıma gıcık bir fikir geliyor. Sonsuz döngü içinde sürekli sunucudaki o porta bağlanmaya çalışan basit bir batch script yazsam ne olur? Çalıştırıyorum ve etkisini ölçüyorum. Bir yandan da Process Monitor üzerinden süreci takip ediyorum. Tarayıcımla bağlandığımda doğrudan bağlantı sıfırlandı hatası alıyorum. Bat dosyasını çalıştırmayı durduruyorum. Ancak tarayıcı bağlantısı uzun bir süre hala devam ediyor.

Acaba kendimi mi kilitledim diye cep telefonumdan bağlanmayı deniyorum. Orada da sorun hala devam ediyor. Bat scripti çalışırken bu defa sunucu portalından ajan üzerine analiz görevi veriyorum. Sistemin çalışmasında pek bir aksama olmadığını görüyorum. Ufak DOS saldırım başarısız görünüyor.

Bu tür bir saldırının olası etkisini daha derinden anlamak için lab ortamımı çoklu ajan haline getirmek ve saldırı payload'unu güçlendirmek üzere ikinci adıma geçme kararı alıyorum.

Türkiye'nin önemli bir futbol kulübünün tesislerindeyiz. Pentest için taramalarımıza başlıyoruz. Eternalblue sayesinde muhasebecinin bilgisayarında shell alıyoruz. Nokta atışı bir iş oluyor. Ardından NAS cihazı dikkatimi çekiyor, 2049 NFS portu açık. show mount komutuyla diskin mount edilebilirlik durumunu ölçüyorum ve gerçekten tamamı mount edilebilir görünüyor. Her zaman eğitimini verdiğim Metasploitable 2 makinesi gibi.

Ardından ilgili komutla diski bina ediyorum ve dizinlerde dolaşmaya başlıyorum. Her bir kullanıcıya ait kullanıcı adı ve parola korumalı alan var. Ancak diski bina ettikten sonra sorgusuz sualsiz gezinebiliyorum. Bir kullanıcının klasörlerinden birinde şifreler.txt diye bir dosya buluyorum. İçeriğine baktığımda web yönetim panelinden banka hesaplarına kadar birçok kimlik bilgisi ve parolanın yer aldığını görüyorum. Teste devam etsek de pratik anlamda testin bittiğini anlıyorum.

Her pentestte olduğu gibi, bugünkü testimizde de amacımız Domain Admin yetkisini ele geçirmek. Yıllarca ağ yönetimi alanında ter dökmüş kıdemli ekip liderimizin yönetiminde kullandığımız saldırı vektörlerinden bir tanesi NTLM Relay yani kulanıcı hash'lerini ele geçirerek başka makineleri pass-the-hash yöntemiyle ele geçirmek. Yalnız burada ele geçirdiğimiz Local Admin hash'leri diğer makinelerde uç noktası güvenlik çözümlerinin etkisyle oturum açamıyor. Tam da bu noktada bir Domain Admin hash'ini yakaladığımızı farkediyoruz.

Bunu aşmak için ele geçirdiğimiz Domain Admin hash'i ile proxy üzerinden relay saldırısı gerçekleştiriyoruz ve yönetim sunucularında Domain Admin yetkisiyle shell almayı başarıyoruz. Üstelik bu makinelerden bir tanesi çok önemli bir Oracle veri tabanı sunucusu.

Bir sistemdeki açıklıkları bulmayı ve yakalamayı en güzel başaran uzmanlar, o sistemleri geçmiş döneminde kurmuş ve yönetmiş insanlardır.

Zafiyet açısından korunması en zor ortamlardan biri, çoklu ve farklı seviyelerde kullanıcı rollerine sahip portalların güvenliği. Doğal olarak pentest gözüyle baktığımda amacım hak yükseltme vektörlerini araştırmak.

Uygulamaya hem yüksek hem de düşük yetkili bir kullanıcı rolüyle giriş yapıp inceliyorum. IDOR zafiyetleri arıyorum ama bulamıyorum. Bir yerde yüksek yetkili kullanıcının analiz sonuçlarını network ya da bulut üzerinde bir ortama kaydedebilirken düşük yetkili kullanıcının buna izni olmadığını ve buna ilişkin butonun donuk olduğunu görüyorum. Hemen trafiği kesiyorum ve Burp üzerinden hazırlamış olduğum kali_bilgisayar/tmp/dump yolunu veriyorum. Bingo, hiç de şaşırtmayan bir şekilde analiz sonuçları Kali makineme aktarılıyor.

Web tabanlı bir portalı geliştirilirken doğal olarak iş akışının düzgün çalışması ön plandadır. Ne geliştirici ne de kullanıcı bunun dışında bir hareketi pek düşünmez. Ancak pentest zaten bunun dışında düşünebilmek demektir.

Portale admin yetkisiyle girdiğimde kullanıcı rollerini ve buna yönelik kimliklendirme ayarlarını yapabildiğimi görüyorum. Bu örnek trafikleri Repeater ile saklıyorum. Ardından düz kullanıcı yetkisiyle yeniden oturum açıyorum. Ancak hiçbir arayüzde roller ve ona yönelik ayarlar doğal olarak gelmiyor. Ben yine de Repeater ile kaydettiğim örnek GET isteklerini, üzerindeki Authentication Token değerlerini mevcut kullancıya göre değiştirerek yeniden "Send" yapıyorum. Gerçekten de karşımda normalde gelmemesi gereken rol ve kullanıcı ayarları sayfası geliyor. Üstelik bazı ayarlarda kimlik bilgilerinin şifreler ile birlikte açık olarak geldiğini görüyorum ve şok oluyorum.

Tamamen kullanıcı rollerini hayal ederek ve minik bir senaryo oluşturarak bir çeşit Hak Yükseltme saldırısını başarıyla düzenlemiş oluyorum. Zaten bu yüzden zafiyet tarama araçları sadece yüzeyin kabasını alırlar, gerçek sızma testi işte bu ince işçilikte yatar. Yani zafiyetlerin mantığını bilen ve uygun konum ve durumda uygulayabilen yetişmiş insanda.

Web socket trafiği pentestlerde nispeten göz ardı edilen bir protokoldür, çünkü ufacık metinsel veriler akar ve bunları manipüle etmek nispeten olası değildir. Ben yine de bir ödeme sistemine ilişkin web socket trafiğini adım adım inceliyorum. Bir miktar zaman geçirince her bir gönderinin messageId üzerinden takip edildiğini ve buna göre loglandığını anlıyorum.

Böylesine hassas sistemlerde yapılan her işlemin logunun gerçek anlamda takip edilmesi çok önemli. Bu mantıkla messageId değeriyle oynayarak log durumunu takip ediyorum. Aynen öngördüğüm gibi işlem logları düşmüyor. Dolayısıyla yaptığımız işlemlere karşı sistemi körletmek mümkün oluyor. Ekran görüntüleriyle birlikte rapora eklemek üzere bu durumu not alıyorum.

Testimizde ajan tabanlı bir güvenlik ürününü inceliyoruz. Ürünün kendini korumaya karşı tamper detection özelliği var. Özellikle config dosyası üzerinde yapılacak bir değişiklik ya da servisin kapatılması gibi durumlarda portale alarm üretiyor.

Config dosyasını inceliyorum. Portal sunucusunun IP.si mevcut. Onu değiştiriyorum ve hemen ardından servisi restart ediyorum. Ardından portal tarafında durumu takip ediyorum. Önce ajan ile bağlantısı kesiliyor ve uyku moduna geçiyor. Hiçbir şekilde tamper detection logu düşmüyor. Ta ki aradan saatler geçtikten sonra config dosyasındaki IP'yi eski haline alıp servisi yeniden restart edene kadar.

Testini yaptığımız site Wordpress tabanlı. Her zamanki gibi rutin olarak WPScan aracı ile rutin taramamı yapıyorum, özellikle API kullanmaya üşenmiyorum. Mevcut kullanıcı adlarını tespit ediyorum.

Ardından wp-admin paneli üzerinden bu kullanıcılardan birine sözlük saldırı denemesi yapıyorum. Ancak daha ilk denememde karşıma ticari bir güvenlik ürününün ekranı geliyor. Kullanılan parolanın hem çok basit hem de yanlış olması alarmı tetiklemiş belli ki. Düşünüyorum, başka hangi kapıdan giriş bulabilirim? Ardından XML-RPC üzerinden kullanıcı adı ve parolayı POST ediyorum. Bununla sözlük saldırısı yaptığımda alarmın tetiklenmediğini görüyorum.

Bugün çok özgün bir ortamın testini yapıyoruz: Yazılım Tabanlı Ağ, yani SDN. Bu testin ön hazırlık toplantılarında teknik proje sorumlusu arkadaş bu ortamın çok kapalı olduğunu ve dolayısıyla hiçbir şey bulamayacağımızı ima edip durmuştu. Zaten testin ilk aşamasında bize sunulan tüm portları dahili güvenlik duvarından kapalı bir yönlendiricinin arkasına hapsedildik. Biz hacker değil miydik?

Neyse ki bir süre sonra bize en azından giriş aşamasındaki makinelere erişim sağlama olanağı sundular. İlk ağ ve port taramasında bir adet VNC suncusu olduğunu görüyoruz. Bu bir Docker makine ve buraya varsayılan parola ile girebiliyoruz. Hemen passwd ve shadow dosyaları üzerinden yaptığımız şifre kırma çalışmasıyla root şifresini kırıyoruz, root parolası bizi pek şaşırtmıyor yine: 1234.

Bu erişim yetkileriyle diğer bazı Linux tabanlı işlevsel makinelere de ulaşıyoruz. Orada tcpdump aracıyla trafik dinlemesi yapıyoruz. Burada bir portalın varlığını anlıyoruz. SSH yönlendirme ile portale web tarayıcımız üzerinden erişiyoruz. Bize parola bile sormuyor. Meğer burasının tüm SDN ağının yönetim portalı olduğunu anlıyoruz.

Sistem ne kadar kapalı ve sofistike olursa, en temel siber güvenlik önlemleri o kadar çok göz ardı ediliyor, bizzat tanık oluyoruz.

Çünkü Docker imajı önceden hazırlanmış ve sistemin içine doğrudan gömülmüş. Sistem geliştiricilerin birinci önceliği sistemin en hızlı bir şekilde çalışmasıdır, güvenlik genelde ikinci plandadır, özellikle zaman kısıtı varsa.

Kurumsal bir portalın testini yapıyoruz. Portalın en dikkat çeken özelliği bir başka üyeye vekalet verebilme yeteneği. Bununla ilgili menü butonuna bastığımda yetkim olmadığı uyarısını alıyorum. Ancak tam arada bir ekran görünüp kayboluyor. Bu durum benim dikkatimi çekiyor. Burpsuite üzerinden aynı trafiği kesip adım adım gidiyorum. Kullanıcı_ID değeri üzerinden kullanıcı sorgusu olduğunu, eğer bir kişi mevcutsa onun bilgisinin çok kısa süreli ekrana geldiğini anlıyorum. Bu durumu Intruder ile ardışık olarak sorguladığımda kurumdaki personelin kullanıcı_ID, ad ve soyad'larının dökümünü alabildiğimi görüyorum.

Bu esnada ekibimizin aynı zamanda sahne sanatçısı olan üyesi vekalet verebilme özelliğini daha da derinden inceliyor. Trafiği kesince vekalet kaynağını değiştirebildiğini farkediyor. Bu şekilde başkası adına yine bir başkasına vekalet verebiliyor. Bu ise sonuçta tam yetkili vekalet neticesinde özünde şifre değiştirmeye ve hesabı ele geçirebilmeye kadar gidiyor. Müşterimizi arayarak raporu beklemeden durum hakkında bilgilendirme yapıyoruz.

Müşterimizin sahadaki Endüstriyel Kontrol Sistemlerini (EKS) inceliyoruz. Modbus trafiğinin oldukça hassas bir Windows makine üzerinden aktığını gözlemliyoruz. Trafiği dinlemek istiyoruz. İlk aklımıza gelen Wireshark kurmak ancak özellikle makinede restart istediği için çekiniyoruz. Biraz araştırma yaptıktan sonra doğrudan Windows'un kendi yetenekleriyle bunu yapabileceğimizi gözlemliyoruz:

Öncelikle komut satırından "netsh trace start capture=yes" komutu ile trafiği çekmeye ve kaydetmeye başlıyoruz. Ancak ortaya çıkan dosya uzantısı .etl. Bunu pcapng formata dönüştürmek için ise yine MS tarafından geliştirilmiş etl2pcapng aracını kullanıyoruz. Evet artık başka bir ortamda pcapng paketini inceleyebiliyoruz.

EKS testinde en önemli olarak EKS ağlarının ve bileşenlerinin mimari yapılandırması olduğunu gözlemliyoruz. Kurumsal bilişim sistemleri (KBS) ile EKS ağlarının içiçe olduğu, TCP tabanlı sunucu sistemlerin bile oldukça eski olduğu ve uygun şekilde kurgulanmadığı senaryoların oldukça yaygın olduğunu görüyoruz.

Müşterimize sağlam bir EKS mimarisi için Purdue Enterprise Reference Architecture (PERA) modelinin referans olarak alınabileceğini ve bu mimari kurumların EKS altyapısına göre özelleştirilebileceğini öneriyoruz.

Bazı EKS sistemlerinde sistemin aksamaması çok önemli. Müşterimiz herhangi bir ayarın bozulmasından ve çalışan sistemin bozulmasından fazlasıyla çekiniyordu. Dolayısıyla makineye restart vermemizi istemiyorlardı. Biz de alternatif bir çözüm geliştirmek durumunda kaldık.

Ekibimizin pembe kapüşonlu temsilcisi dışarıdan harici aramanın sonrasında dahili numarayı tuşlayarak kurum temsilcisine ulaşır.

Ayşe hanım merhaba, kolay gelsin. Bilgisayarınızda şifrenizle giriş yapabiliyor musunuz, çoğu personelimiz hata aldıklarını söylüyor. Bir girer misiniz? Hata alacak mısınız?

- Siz kimsiniz?

* Benim adım da Ayşe, bilgi işleme yeni geldim. Bir çalışma yapıyoruz da.

- Aaaa, öyle mi hoşgeldiniz. Durun bir bakayım. (birkaç saniye sonra) bende bir sorun görünmüyor.

* Çok güzel, yalnız biz yine de tüm personelin şifresini değiştirme kararı aldık. Genel bir yenileme çalışması yapacağız. Şifrenizi alabilir miyim?

- Ama öyle oluyor mu? Neyse peki, şifrem xxx.

Yaptığımız Sosyal Mühendislik Testi sonucu: on kişiden üçünün şifresi başarıyla alınmıştır.

Yeni bir web portal pentesti yapıyorum. Her zamanki gibi ilk olarak hesap ayarlarını analiz ediyorum. Dikkatimi parola yenileme özelliği çekiyor. Proxy ile kesiyor ve inceliyorum. Bir USER_ID değeri var, Token ile trafik takip ediliyor. Başka bir USER_ID değerini giriyorum ve onun da parolasının değiştiğini görüyorum. Daha ilginci Token değerini siliyorum, sadece "Token:" ifadesi kalıyor, şaşırtıcı şekilde reset işlemi başarılı oluyor. USER_ID değerini bildiğim (aslında pek tahmin edilebilir bir değer değil) bir hesabı koşulsuz ele geçirebildiğimi anlıyorum.

Başka kullanıcılara ait USER_ID değerini nasıl elde edebilirim diye kafa yormaya başlıyorum. Trafiği refresh edip tek tek analiz ediyorum. Grup ve roller ait trafiği kestiğimde, Response değerinde Administrator dahil rollerin USER_ID değerinin tanımlı olduğunu görüyorum. Gerisi çorap söküğü gibi geliyor. Parola yenileme ile bunu birleştirince Administrator hesabına ait hem tüm kimlik bilgilerini alıyorum hem de parolasını sıfırlayarak ele geçiriyorum. Sonuç: Account Takeover.

Sızma testini yaptığım web portalı, hazırlanmış sınavlarla adayların yeteneklerini ölçen ve ona göre sıralayan bir İK yazılımı. Bunun için adaya bir e-posta gönderiliyor ve aday o e-posta linki üzerinden sınavını başlatıyor.

Böylesi bir kurguda neyi manipüle edebilirim diye düşünüyorum. Gelen e-posta linkini inceliyorum. Sınavın id değeri ve adayın e-posta adresinin karartılmış halinden oluşuyor. Öncelikle sınavı normal bir aday gibi dolduruyorum ve sonuca kadar ulaşıyorum. Bu esnada doğal olarak o pozisyon için hazırlanmış olan tüm soruların ekran görüntülerini alabiliyorum.

Ardından gelen linkteki e-posta değerini 1234 gibi anlamsız bir değere dönüştürerek başka bir tarayıcıda açıyorum. Bingo, benden e-posta adresimi girerek sınav sürecini başlatmamı istiyor. Artık zaten bildiğim soruları yanıtlayarak sınavı tamamlayabiliyorum.

Senaryo hazır artık; önce geçici bir kimlik ve e-posta adresiyle başvuru yapıyorum. Ardından da yaptığım manipülasyonla gerçek kimlik ve e-posta adresimle sınavı tamamlıyorum.

Çok büyük bir kurumun oldukça kapsamlı bir sızma testini yapıyoruz. En ufak bir tarama denememizde bile sistem bizi acımadan bloke ediyor. Gerçekten sabır isteyen bir test süreci.

Böylesi büyük yüzeylerin bir özelliği wayback machine dediğimiz eski tarihli arşiv ortamlarında çok iz bırakması. Bu şekilde tarama yaparken tek tek uç noktaları deniyorum. Amacım gözden kaçan bir uç noktası bulmak. Bir süre sonra üyelik hizmetlerine ait bir URL'de kullanıcı ID değerini de içeren adres grubu dikkatimi çekiyor.

Uç noktasını açıyorum. O üyeye ait döküm bilgileri geliyor. Üstelik kullanıcı adı ve parola bile girmeden. Hiç vakit kaybetmeden bu durumu bildiriyorum.

Devam ediyorum. Peki, mobil tarafta durum nasıl acaba diyorum? Orada da hemen trafik kesiliyor mu? En azından bazı api adreslerini keşfetmeye bile razıyım.

Bu şekilde mobil uygulamayı açıyorum ve başarılı şekilde SSL Pinning bypass yapabildiğimi görüyorum. Ardından kurumsal müşteriler için randevu alma hizmeti dikkatimi çekiyor. Buradaki isteği proxy ile kesiyorum ve sahte bir kimlik ile randevu alıyorum.

Saldırı senaryomuz ortaya çıkıyor: Bir haftalık tüm takvimi sahte hesaplarla açılan randevularla doldur. Bu şekilde bir türlü randevu alamayan kullanıcılar büyük bir memnuniyetsizlik yaşayacaktır.

Tabi böylesi bir senaryoya meydan vermemek için, raporu beklemeen hemen müşterimize durumu bildiriyorum.

Testini yaptığımız portal sonuçta vatandaş ile doğrudan etkileşime geçtiği ve ödeme sistemi özellikleri de barındırdığı için oldukça hassas. Ekibimizin deneyimli üyesi sisteme üyelik açma özelliğini incelemeye karar veriyor. Bu maksatla tüm alanları doldurduktan ve üyelik sürecini tamamladıktan sonra sistemin "başvurunuzu incelemeyi müteakip sizinle temasa geçeceğiz" şeklinde bir mesaj karşılıyor kendisini.

"parolamı unuttum" butonuna tıklıyor açılan pencerede bilgilerini girdikten sonra yeni parola belirlemesi isteniyor.

Bu süreç içinde başka bir gözlemde daha bulunuyor. Üyelik sürecinde önce başkasının telefon numarasını girip (o numaraya onay kodu gönderilmesine rağmen), sonrasında telefon numarasına onay kodu gönderdiğinde, diğer telefonla üyelik yapabildiğini ve kendisini başkasının telefon numarasıyla kaydedebildiğini farkediyor.

Tüm bu çalışmamızı ivedi olarak müşterimize bildiriyoruz.

SONSÖZ

Bu çalışmanın amacı, sızma testi dünyasını teknik olmayan bir şekilde ve çoğu defa sadeleştirerek dışarıdan insanlarla buluşturmaktır.

Günlüklerden de görüldüğü gibi, teknik bilginin yanında sistem ve olaylara bakış açısı ile konulara yaklaşım sızma testi sürecinin en önemli bileşenidir.

Sadece otomatik zafiyet tarama araçlarına dayalı taramalar adı üstünde zafiyet taramadır, gerçek anlamda bir sızma testi değildir.

Bu eğlenceli dünyada bize eşlik ettiğiniz için teşekkür ederiz.

Bilishim Siber Güvenlik ve Yapay Zeka San. ve Tic.Ltd.Şti. 2018 yılında “Makine Öğrenmesi teknikleriyle yenilikçi Siber Güvenlik çözümleri” geliştirmek vizyonuyla Hacettepe Teknokent yerleşkesinde kurulmuştur.

Günümüzde TSE-A sertifikasına sahip bir Sızma Testi firması olan Bilishim Siber Güvenlik, aynı zamanda EPDK yetkisiyle EKS yani Endüstriyel Kontrol Sistemlerde sızma testi yapma yetkisine sahip birkaç firmadan biridir.

Referansları arasında Türkiye’nin en önemli kurum ve firmaları bulunan Bilishim, aynı zamanda Nettarsier isimli bir Kurumsal Zafiyet Tarama Aracı ve dolandırıcılıkla mücadele odaklı UMAY isimli bir mobil uygulama geliştirmiştir.

Kırmızı Takım (Red Team) hizmetleriyle de dikkat çeken ekibimiz, bu konudaki ARGE çalışmalarını hizmet sağladığı kurumlara bir çözüm paketi olarak sunmaktadır.

Yılmaz Değirmenci; 1975 Ankara doğumludur. 1997 yılında Kara Harp Okulundan mezun olmuş, 2000-2002 yılları arasında ABD Naval Postgraduate School'da Bilgisayar Bilimi üzerine yüksek lisans yapmıştır.

Müteakiben uzun yıllar Kara Kuvvetleri Komutanlığında Sistem Yöneticisi, Veritabanı Yöneticisi, Proje Yöneticisi gibi görevler ifa etmiştir. Bu süreçte hobi olarak Türkçe gramer kurallarına uygun o anda Yapay Zekâ destekli şiir yazan bir robot geliştirmiştir.

2017 yılında son görev yeri TSK Siber Savunma Komutanlığından emekli olmuş ve özel sektöre geçmiştir. 2018 yılından beri takım arkadaşlarıyla birlikte kurmuş olduğu Bilishim Siber Güvenlik ve Yapay Zekâ firmasında CEO görevini ifa etmektedir.