T.C.
SÜLEYMAN DEMİREL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

WEB SİTESİ ÜZERİNDEKİ ZAFİYETLERİN ANALİZİ, KAYNAKLANMA NEDENLERİ VE ÇÖZÜM ÖNERİLERİ

Ehssan Salman Obaid AL-JANABI

Danışman
Yrd. Doç. Dr. Arif KOYUN

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
ISPARTA, 2017

TEZ ONAYI

Ehssan Salman Obaid AL-JANABI tarafından hazırlanan **"Web Sitesi Üzerindeki Zafiyetlerin Analizi, Kaynaklanma Nedenleri ve Çözüm Önerileri"** adlı tez çalışması aşağıdaki jüri üyeleri önünde Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü **Bilgisayar Mühendisliği Anabilim Dalı**' nda **Yüksek Lisans Tezi** olarak başarı ile savunulmuştur.

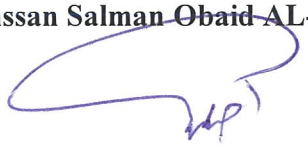
Danışman	**Yrd.Doç. Dr. Arif KOYUN** Süleyman Demirel Üniversitesi	.....
Jüri Üyesi	**Doç. Dr.Ömer DEPERLİOĞLU** Afyon Kocatepe Üniversitesi	.....
Jüri Üyesi	**Yrd.Doç. Dr. Asım Sinan YÜKSEL** Süleyman Demirel Üniversitesi	.....

Enstitü Müdürü **Prof. Dr. Yasin TUNCER** .....

TAAHHÜTNAME

Bu tezin akademik ve etik kurallara uygun olarak yazıldığını ve kullanılan tüm literatür bilgilerinin referans gösterilerek tezde yer aldığını beyan ederim.

Ehssan Salman Obaid AL-JANABI



İÇİNDEKİLER

ÖZET

Yüksek Lisans Tezi

WEB SİTESİ ÜZERİNDEKİ ZAFİYETLERİN ANALİZİ, KAYNAKLANMA NEDENLERİ VE ÇÖZÜM ÖNERİLERİ

Ehssan Salman Obaid AL-JANABI

Süleyman Demirel Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Yrd.Doç.Dr. Arif KOYUN

İnternetin bize pek çok olası iş kolaylığı sağladığı bir dünyada yaşıyoruz. Günümüzde tüm büyük bankalar mobil bankacılık hizmeti sunmaktadır. Ayrıca bilet satın alabiliyor ve çevrimiçi alışveriş yapabiliyoruz. Bugün, tüm işlemler internetle ve dolayısıyla web uygulamaları ile bağlantılıdır. Sonuç olarak, insanlar arasında internet kullanımı her geçen gün daha da artmakta, bu web uygulamaları sayısını da arttırmaktadır. Çok sayıda web uygulaması hayatımızı kolaylaştırırken, bu uygulamaların birçoğu bilgisayar korsanlarının sistemimizi kesmesini veya virüs ve solucan bulaştırmasına sebep olmaktadır. Bu olaylardan dolayı da özel hayatımız ve ticari yaşamlarımız kötü şekilde etkilenebilmektedir.

Günümüzde web uygulamaları genelde uygulamanın sadece sunucusu üzerinde değil kullanıcının cihazında da çalıştırılmaktadır. Bu web uygulamaları kullanıcılarına çok sayıda hizmet sunarken aynı zamanda işlevsellik, etkileşim ve kullanım kolaylığı sağlamaktadır. Bu nedenle, en basit web uygulaması bile birçok farklı HTTP parametresini işler. Bu durum manipüle edilmeye açık birçok zafiyet bulundurma olasılığını artırmaktadır.

Son yirmi yılda, güvenlik açıkları, web uygulamalarının muazzam çoğalmasayla birlikte artış göstermektedir. Web güvenliği sadece bir güvenlik duvarı kurmanın ve ana makinelerin güncellenmiş yazılım kullanmasını sağlaması problemi değildir. Sistemdeki ve yazılım bileşenleri arasındaki güvenilir ve güvenilmez sayılan mesajlar, komutlar ve veri değişimi gibi uygulamada küçük veya büyük olan her şeyi incelemek gerekir. Bu güvenlik derecesi ile web uygulamalarında güvenlik açıklarını gidermek için çok sayıda derinlemesine araştırma yapılmıştır. Ancak bu güvenlik açıklarını gidermek için önceden tespit edilmesi önem taşımaktadır.

Web uygulamaları güvenliğini test etmenin bir yolu olarak, kuruluşlar, güvenlik açıklarını belirlemek için denetlenen bir ortamdaki bir saldırganın davranışını taklit eden Penetrasyon testi (Pentest) gerçekleştirmektedir. Bu tezde bir Penetrasyon testi gerçekleştirerek bazı güvenlik açıklıkları ve web uygulaması hakkında ayrıntılı bir çalışma sunulmuştur. Ayrıca, web uygulamaları alanındaki mevcut çalışmaları izlemek ve mevcut güvenlik açıklıklarını gidermek için etkili çözüm önerileri sunulmaktadır.

2017, 74 Sayfa

ABSTRACT

M.Sc. Thesis

ANALYSIS OF VULNERABILITIES IN WEB SITES, ITS CAUSES AND PROPOSED SOLUTIONS

Ehssan Salman Obaid AL-JANABI

SüleymanDemirel University
Graduate School of Natural and Applied Sciences
Department of Computer Engineering

Supervisor: Asst. Prof. Dr. Arif KOYUN

We live in a world where the Internet provides us many possible chores. All major banks offer mobile banking services. Also we purchase tickets and do shopping online. Today all the processes is linked to the Internet and hence with web applications. As a result, the usage of the internet among people is increasing day by day, which also increases the number of web applications. Despite that huge number of web applications makes life easier for us, but most of these applications have some vulnerability that allows hackers to hack our systems or infect them with viruses and worms and thus affect our private and commercial lives.

Nowadays, web applications are usually run on the user's device, not just on the server of the application. These web applications provide a multitude of services to users, while at the same time it need to be consistent with functionality, interactivity, and ease of use. For this reason, even the simplest web application process may need a plethora of different HTTP parameters. This could increases the likelihood of producing many varieties of vulnerabilities that are vulnerable to manipulation.

In last two decades, security vulnerabilities have grown with the enormous growth of web applications. Web security is no longer simply a question of installing a firewall and ensuring that hosts have updated software. It is a matter of scrutinizing everything small or large in the application such as messages, commands, and data exchanged between systems and software components deemed trusted and untrustworthy. With this maturity of security, a large number of empirical studies have been conducted to address vulnerabilities in web applications. But to overcome these vulnerabilities, it is important to detect first the problem before preventing it.

As a way to test security in web applications, organizations have been performing penetration testing (pentest) which simulates an attacker's behavior in a controlled environment in order to identify applications vulnerabilities. This thesis is submitted a detailed study on some vulnerabilities in web application through conduct a penetration testing. Also, monitor existing studies in web applications area and come up with effective solutions to address existing security vulnerabilities.

2017, 74 Pages

TEŞEKKÜR

Bu çalışma Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda Yüksek Lisans Tezi olarak gerçekleştirilmiştir.

Tez çalışmasını sürdürdüğüm dönemde, bana daima destek olan, çalışma disiplini aşılayan, fedakârlıktan kaçınmayan, yapıcı öneri ve düzenlemeleriyle çalışmanın içerik ve sunumundaki zenginliğine büyük katkıda bulunan tez yöneticisi, saygıdeğer danışman hocam, Sayın Yrd.Doç.Dr. Arif KOYUN'a sonsuz teşekkürlerimi sunarım.

Ayrıca, maddi ve manevi desteğini benden hiçbir zaman esirgemeyen aileme en içten duygularımla teşekkür eder, şükranlarımı sunarım.

Ehssan Salman Obaid AL-JANABI
Isparta,2017

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

SİMGELER VE KISALTMALAR DİZİNİ

ACPs	Erişim Kontrol Politikaları
AJAX	Eşzamansız JavaScript ve XML
API	Uygulama Programlama Arabirimi
BGA	Bilgi Güvenlği Akademisi
CSS	Basamaklı Stil Sayfaları
CWE	Genel Zayıflık Numaralandırması
DB	Veritabanları
DBMS	Veritabanı Yönetim sistemi
DOM	Belge Nesnesi Modeli
EPDG	Genişletilmiş Program Bağımlılığı Grafiği
FPI	Flash Parametre Enjeksiyonu
GA	Güvenlik Açığı
HOY	Hizmet Olarak Yazılım
HPF	HTTP Parametre Parçalanması
HPP	HTTP Parametre Kirliliği
HTML	Hiper Metin Biçimlendirme Dili
HTTP	Hiper Metin Transfer Protokolü
ISECOM	Güvenlik ve Açık Metodolojiler Enstitüsü
ISMS	Bilgi Güvenliği Yönetim Sistemi
LFI	Yerel Dosya Eklerme
PL	Programlama dili
QoS	Hizmet kalitesi
REP	Robotlar Hariç Tutma Protokolü
RFI	Uzak Dosya Eklerme
SDLC	Yazılım Geliştirme Yaşam Döngüsü
SQL	Yapılandırılmış Sorgu Dili
URL	Tekdüzen Kaynak bulmayı
WAF	Web Uygulaması Güvenlik Duvarı
WAPL	Web Uygulamaları Programlama Dilleri
XHTML	eXtensible HyperText Markup Language
XML	Genişletilebilir İşaretleme Dili
XSS	Siteler Arası Kod Yazma

1. GİRİŞ

Son yirmi yılda, Web uygulamaları büyük bir gelişme gösterilmiştir, günümüzde en yaygın kurumsal uygulama türü haline gelmiştir. Web uygulamaları, özel ve ticari iletişimde günlük rutinin bir parçası haline gelmiştir. Günlük hayatta birçok işlemin internet üzerinden yapılması ve internetin yaygınlaşması ile birlikte her gün çeşitli güvenlik açıklıkları keşfedilmektedir. Bu, web uygulamalarına karşı birçok farklı saldırının uygulanması olasılığı anlamına gelir, Web uygulamalarının çalıştırıldığı işletim sistemi veya web sunucuları yerine web uygulamalarının güvenlik açıklıklarının sömürülmesine odaklanılır.

Web Uygulaması, internet veya intranet gibi bir ağ üzerinden bir web tarayıcısı üzerinden erişilen uygulamadır (Flex Orbits, 2013) (Das, 2015) (Devi, 2015) (StackOverflow, 2017). Ayrıca, tarayıcı tarafından desteklenen bir dilde (HTML, Java komut dosyası, Java, ASP, PHP vb.) Kodlanmış ve uygulamayı çalıştırılabilir hale getirmek için ortak bir web tarayıcısına bağımlı bir yazılım uygulamasıdır. Muniz ve Lakhani gibi bazı araştırmacılar, web uygulamasının bir tarayıcıyı istemci olarak kullanan herhangi bir uygulama olduğunu belirttiler. Ayrıca, web hizmetlerine erişimi kolaylaştıran ve çoklu istemci hizmeti verebilen bir sistemin merkezi yönetimini temel alan web uygulamalarının popülerliğine değindiler. Ayrıca, bir web uygulamasının erişme gereksinimleri hem servis sağlayıcıların hem de istemcilerin beklentilerini basitleştiren endüstri web tarayıcı istemci standartlarını takip edebilir. (Muniz ve Lakhani, 2013). Web uygulamaları çevrimiçi hizmetlere erişim sağlamak için etkin bir yol haline geldi ancak her geçen gün web uygulaması zafiyetleri keşfediliyor ve endişe verici bir oranda açığa vuruluyor. Bu durum web uygulamalarının geniş sosyal etkileri nedeniyle, saldırganlar web uygulamalarının kullanımında farklı amaçlar hedeflemişlerdir. Saldırganlar mantıksal olarak sisteme girmekte ve sistemin normal çalışmasını bozmaktadırlar. Akıllı telefon ve tabletlere bakılırsa, bu cihazların çoğu uygulaması web uygulamaları olduğunu görebilirsiniz. Bu durum güvenlik uzmanları ve saldırganlar için yeni ve geniş olan oloşturmaktadır.

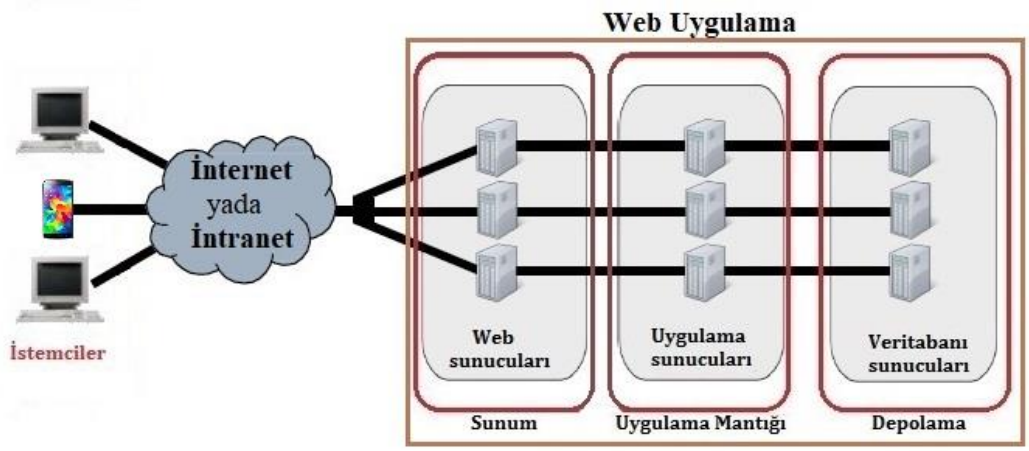
Güvenlik açıkları, çeşitli aşamalardaki uygulamalara dahil edilebilir Geleneksel yazılım geliştirme yaşam döngüsünde (SDLC) Yazılım geliştirme ya da kusur nedeniyle veya dağıtım sırasında yapılandırma sorunlarını giderme (Davis, 2013) (Konrad ve diğerleri, 2014) (Linked In, 2017) (what is, 2017). Ayrıca, web uygulamaları geliştikçe ve yeni teknolojiler benimsenirken, yeni teknolojilerin istenmeyen eksikleri veya geliştiricilerin yeni teknolojileri kullanırken yaptıkları hatalardan dolayı yeni güvenlik açıklıkları ve saldırı vektörleri ortaya çıkmaktadır. Bunun anlamı her gün yeni bir güvenlik açığı keşfedilebilir ve bu güvenlik açıkları sayesinde saldırganlar iç ağa erişebilir, birçok kritik öneme sahip bilgi elde edebilir.

Web uygulama saldırıları, birçok uygulamanın en önemli güvenlik kaygılarından biri olmaktadır. Web uygulama geliştiricilerinin tek endişesi, uygulama güvenliğini sağlamak ve son kullanıcıya sağlanan hizmette kolaylık, etkinlik ve verimlilik sağlamaktır. Bunun yanında yüksek güvenlik gerektiren sağlık, bankacılık ve e-ticaret işlemleri yapılan web uygulamalarında kolaylık, etkinlik ve verimlilik dışında daha çok güvenlik büyük önem taşımaktadır. Son yıllarda, web sitelerindeki güvenlik açıklıkları nedeniyle sık sık hassas bilgiler elde edilmiştir. Ancak bu güvenlik açıklıkları tespit edilebilir ve web sitelerinin güvenliği güçlendirilebilir. Bu nedenle, web sistemlerinin zayıf yönlerini keşfetmek, bu zayıflıkların nedenlerini araştırmak ve çözüm üretmek önem kazanmıştır. Sızma testleri, güvenlik açıklıklarını belirlemek için endüstride yaygın şekilde kullanılmaktadır.

Bu tezin amacı, web uygulamaları hakkında kapsamlı ve eksiksiz bir genel bakış sağlayarak web uygulamaları güvenliği üzerinde çalışmak isteyenlere ışık tutmaktır. Örnek web sitemiz içerisinde bulunan güvenlik açıklıklarını tespit ederek güvenlik önlemlerini artırmanın yanında sebeplerini de inceleyerek etkili bir çözüm üretmeye çalışacağız. Saldırganların bakış açılarından saldırılara karşı birden fazla girişimi taklit eden bir penetrasyon testi gerçekleştirerek, belirli güvenlik açıklarını ortaya çıkarmak için kara kutu yaklaşımını kullanarak. Web hizmetleri ortamında üçüncü parti testler için özellikle yararlıdır, çünkü testçinin statik analiz bulgulaması yaptığı gibi kaynak koda erişiminin olmasını gerektirmez. Böylece web uygulamaları güvenliğinde çalışmak isteyenlerin güvenlikle ilgili tespitlere erişmesini, sorunlarını, ve çözüm önerileri üretmesini sağlayacaktır.

Bu tezde kullanılan yöntem, kötü niyetli bilgisayar korsanlarının yaptığı gibi örnek sitemizde yer alan zafiyetlere saldırmaktır. Amaç sistemde hangi zafiyetlerin olduğunu kanıtlamaktır. En etkili sızma testleri, çok spesifik bir sistem ile spesifik bir hedef (Örneğimiz gibi) hedefleyen testler olduğunu belirtmek gerekir. bunun için bu örneği verdik. Öncelikle varolan güvenlik açıkları hakkında tam bir açıklama getirdik. Sonra zayıflıkların varlığını ispatlamak için basit kodlarla saldırmanın basit ve kolay bir yolunu kullandık. Inancıyla, geleneksel (basit) kodla bir zayıflık kırarsa, bileşik kodla kesmek kolaydır. Güvenlik açıklıklarını ortaya çıkartmak için black-box yaklaşımını kullanarak bir sızma testi gerçekleştirilecek. Black-box yaklaşımı, web hizmetleri ortamında çok faydalı ve ideal bir yaklaşımdır, çünkü kaynak koda erişimi olmayan bir hacker gibi davranarak sistem analiz edilir. ardından güvenlik açıklığının belirtilen nedenleri ve etrafındaki mevcut çalışmalar dikkate alınarak uygun çözümler sunuldu. Bu tezin okuyucusu değerleri hedefleri seçerek bunu bulacaktır.Tüm güvenlik altyapısını ve değerli bir varlık için ilişkili riski belirleyebiliriz.

Web uygulamaları sunucu-istemci mimarisi üzerine kurulmuştur ve genel internet protokolleri ve teknolojileri üzerinde çalışırlar. Kullanılan protokoller ve teknolojiler, teknolojilerin kusurları veya geliştiricilerin web uygulamaları geliştirme sırasında teknolojileri uygularken yaptıkları hatalar güvenlik açıklarına neden olabilir (Kalman, 2013). Uygulamalar genellikle 'katmanlar' olarak adlandırılan ve katmanlara rol atanan mantıksal parçalara bölünür. Geleneksel uygulamaların yalnızca istemci tarafında bulunan 1 katmandan oluşmasına rağmen birçok varyasyonu mevcuttur. En yaygın kullanılan yapı 3 katmanlı yapıdır. Bu katmanları sunum, uygulama mantığı ve depolama olarak adlandırılır. Web tarayıcılar birinci katman olan sunum katmanını oluşturmaktadırlar. Web hizmetinin içeriği ve çalışma mantığı ikinci katman olan uygulama mantığı katmanını oluşturur. Web hizmetinin arka yüzündeki verilerin depolandığı veri tabanı kısmı ise son katmanımız olan depolama katmanını oluşturmaktadır. Web tarayıcıları, istemcilere ait verileri, veri tabanına göndermek için istemci istekleri kullanıcı arabirimi olarak hizmet veren ikinci katmana gönderir. (Stanek, 2014)



Şekil 1.1. Üç-Katmanlı Web Uygulama Mimarisi

Web uygulamalarının mimarisinde derinlemesine bakmak isteyen kişi şu gerçeği bulacaktır: Karmaşık uygulamalar için, uygulama mantığı katmanını kırmanın en büyük yararı olan n katmanlı bir yaklaşım kullanmak faydalı olabilir. Başka bir yararı ise verilere erişmek için kullanımı kolay bir arabirim sağlayarak veri katmanını diğer katmanlardan ayıran bir entegrasyon katmanı ekleme de olabilir. Ancak, katman sayısının artmasının, her bir katmanın kendi açık güvenlik açıklarına ve saldırganlardan korunması gereken arabirimlere sahip olduğu yeni bir dizi güvenlik endişesi yarattığı belirtilmelidir. Ayrıca, saldırganlar istemci üzerinde tam denetime sahiptir, dolayısıyla saldırıları genellikle istemci üzerinden başlatır ve mimaride bir katmandan bir giriş noktasından diğer katmandaki gerçek hedefe saldırı sağlamak için katmanlar arasındaki etkileşimleri gelişmiş yöntemlerle kullanır.

Web uygulamaları, (Tarayıcılar, Tekdüzen Kaynak bulmayı (URL), Hiper Metin Biçimlendirme Dili (HTML), Veri tabanları (DB), Protokoller, Basamaklı Stil Sayfaları (CSS), Kütüphaneler, Çerçeveler, Uygulama Programlama Arabirimi (API), Veri biçimleri, AJAX, Belge Nesnesi Modeli (DOM), Aynı Kaynak İlkesi, Karakter Kodlama Sistemi, vb.) gibi işlevlerini yerine getirmek için sayısız teknoloji kullanan bir piramit üzerine kurulmuştur. Birçok yazar (Flynn, 2015) gibi web teknolojileri hakkında yazmıştır, bu tekniklerin çoğu gizli değil, ve kolayca erişilebilir olmasına rağmen. Bu teknolojilerin özelliklerinin anlaşılması kolay ve web uygulamalarına karşı etkili saldırılar gerçekleştirmenin anahtarıdır.

Web uygulamalarında işlevsellik sunmak için sayısız teknoloji kullanılır. İşlevsel olan herhangi bir uygulama, sunucu ve istemci bileşenlerinde düzinelerce farklı teknolojiyi kullanabilir (Stuttard ve Pinto, 2011). Bu işlevlerin uygulanması tarayıcı, web sunucusu, hizmetler ve veritabanı gibi bir web uygulama mimarisinin tüm bileşenlerini etkileyebilir. Ayrıca coğrafi olarak ve organizasyonel olarak dağılmış olabilirler, çünkü her bileşen coğrafi olarak ayrı olan sunucularda bulunabilir ve işleme alınma mantığında farklı organizasyonlar sorumlu olabilir. Saldırgan bir web uygulamasına karşı ciddi bir saldırı gerçekleştirmeden önce web uygulamasında kullanılan işlevselliklerin nasıl uygulandığını ve kullanılan teknolojilerin nasıl davranış göstereceğini gözlemleyerek zayıf noktaların uygulama üzerinde hangi noktalarda bulunabileceğine dair temel bir fikir edinmeye çalışır. Web uygulamaları işlevleri iki kategoriye ayrılabilir:

a. Sunucu Taraflı Fonksiyonlar

İlk World Wide Web tamamen statik içerik içeriyordu. Web siteleri, sadece bir web sunucusu yüklenen ve bunları talep eden herhangi bir kullanıcıya teslim edilen HTML sayfaları ve resimler gibi çeşitli kaynaklardan oluşuyordu. Belirli bir kaynak için gönderilen her istekte sunucu aynı içerik ile yanıt verirdi. Günümüz web uygulamalarında oldukça fazla statik kaynak kullanmaktadır. Ancak, kullanıcılara sundukları içeriğin büyük bir kısmı dinamik olarak oluşturulur. Bir kullanıcı dinamik bir kaynak istediğinde, sunucunun yanıtı anında oluşturulur ve her kullanıcı kendisine özel içeriği alabilir. Dinamik içerik, komut dosyaları veya sunucu üzerinde çalışan diğer kodlar tarafından oluşturulur. Bu senaryolar kendi başlarına bilgisayar programlarına benzer. Bunların çeşitli girdileri vardır ve bunlar üzerinde işlem gerçekleştirir, ve çıktılarını kullanıcıya döndürür. Bir kullanıcı tarayıcısı dinamik bir kaynak istediğinde normal olarak bu kaynağın bir kopyasını istemez. Genel olarak, istek dahilinde çeşitli parametreleri de gönderir. Sunucu tarafı uygulamasının, tek tek kullanıcıya özel içerik üretmesini sağlayan yapı bu parametrelerdir.

b. İstemci Taraflı Fonksiyonlar

Sunucu tarafı uygulaması kullanıcının girdi ve eylemlerini almasının ardından sonuçları kullanıcıya sunabilmesi için istemci taraflı kullanıcı arabirimi sağlamalıdır. Tüm web uygulamalarına bir web tarayıcısı üzerinden erişildiğinden, bu arayüzlerin hepsi ortak bir teknoloji çekirdeğini paylaşır. Bununla birlikte, bunlar çeşitli şekillerde oluşturulmuş ve uygulamaların istemci tarafındaki teknolojiyi kullanmanın yolları son yıllarda hızla gelişmeye devam etmiştir.

Yeni teknolojilerin ortaya çıkması, yeni güvenlik açıklıklarının ortaya çıkma olasılığını ortaya çıkarmaktadır. Yeni güvenlik açıklıkları bulunmasının yanında en sık görülen zafiyetler zamanla gelişim gösterdi. Böylece, mevcut uygulamalar geliştirildiğinde dikkate alınmayan yeni saldırılar düşünülmüştür. Bazı sorunlar gittikçe daha yaygın hale gelmiştir çünkü bunların farkındalığı artmış ve buna ek olarak teknolojilerdeki bazı eksiklikler web tarayıcı yazılımında yapılan değişiklikler sonucunda büyük oranda azaltılmıştır. Web uygulamalarında işlenen bilgilerin büyük çoğunluğu özel ve son derece gizlidir. Bu nedenle güvenlik web uygulamalarının geliştirilmesinde önemli bir husustur. İnternetteki kötü niyetli saldırılara karşı açık mimarisi ve karmaşık iş mantığına sahip web uygulamalarının gittikçe daha savunmasız olduğu bilinmektedir. Web uygulama güvenliği bir çok yazılım geliştirme disiplini, teknolojisi ve tasarım kavramlarını içine alan geniş bir konudur. Web uygulamalarına yönelik en ciddi saldırılar, hassas verileri göstermek veya uygulamanın arka planında çalıştırılan sistemlerde sınırsız erişime sahip olmaktır. Bu türden yüksek profilli saldırılar sık sık görülmektedir. saldırı için yeni olanaklar sunan yeni teknolojiler geliştirildi. Bu nedenle, Web uygulamaları güvenliği gerekiyor.

Penetrasyon testi, web uygulaması güvenliğini test etmek için endüstride yaygın olarak kullanılmaktadır. Uygulama hizmetlerine yapılan saldırıların önlenmesi için güvenlik açığı değerlendirmesi ve penetrasyon testi uygulanmaktadır (Sachin ve diğerleri, 2011). Güvenlik açığı değerlendirmesi, tehditlerin keşfedildiği ağ geçidi konumundadır. Penetrasyon testi, güvenlik açıklarını keşfetmek için web uygulamalarına gerçek saldırıyı taklit eden bir güvenlik değerlendirme sürecidir.

Değerlendirme, web uygulamalarının gerçek ve beklenen davranışlarının karşılaştırılmasına dayalı olarak yapılır. (Farah ve diğerleri, 2015). Penetrasyon testi, bir ağda bulunan test uygulamalarında "bilinen" güvenlik açıklarını belirlemek için kullanılabilir veya özel hazırlanmış web uygulamalarında "bilinmeyen" güvenlik açıklarını keşfetmek için kullanılabilir. Tezimiz öncelikle ilki ile ilgilidir, ancak her iki türünü de anlamak çok daha önemlidir.

Penetrasyon testi için çeşitli yöntemler vardır. Başarılı bir test için bir metodolojinin izlenmesi gereklidir (Mirjalili ve diğerleri, 2014). Test metodolojisinin isimleri ve dizilişi farklı olmakla birlikte, temel süreçler tüm yöntemler için aynıdır (Olson, 2010). Bir testi yapan kişiye bu testi yapma süreci boyunca rehberlik eden, tipik bir metodoloji ISECOM Açık Kaynak Güvenliği Test Metodolojisi El Kitabı (ISECOM OSSTMM Ana Sayfası, 2017) olan bir dizi metodoloji oluşturulmuştur. Özel olarak hazırlanmış bir web uygulamasında asla tespit edilmemiş güvenlik açıklarını keşfetmek için penetrasyon testi genellikle güvenlik uzmanları tarafından uygulama için güvenlik değerlendirmesi yöntemi olarak yapılır.

2. KAYNAK ÖZETLERİ

Stepien ve diğerleri (2012): Bu çalışma, web penetrasyon testi çalışmaları oluşturma sürecini kolaylaştırmak için test şartnamesi dili TTCN-3'ün doğal soyutlama özelliklerini güçlendiren bir yaklaşımla sunulmuştur. Özellikle web güvenlik açıklarının ayrı modellerini ve web uygulama işlevlerini, genel bir web soyutlama modeli ve TTCN-3 test framework modeliyle birleştirmenin avantajlarını gösterir.

Uskov (2013): Bu makale, bilgisayar bilimleri ve bilgisayar bilgi sistemleri öğrencileri için yazılım ve web uygulamaları güvenliğinde tasarlanmış ve geliştirilmiş en son teknoloji ders yazılımı ve uygun öğrenme paradigması üzerine odaklanmıştır. Geliştirilen müfredatın ana konuları, yazılım ve web sistemlerinde saldırganların motivasyonu, modern teknikler ve güvenlik açıkları, bilgisayar saldırıları kategorileri, bilgisayar saldırıları çeşitleri, bilgisayar korsanlığı araçları, koruma ve savunma mekanizmaları, yazılım ve web sistemlerinin geliştirilmesi için güvenli programlama yöntemlerini içerir. Öğrencinin geri bildirimine ve öğrencinin akademik performansına ve çoklu uygulama egzersizleri ile eğitimin öğrenme paradigmasının bir kombinasyonuna dayanır.

Antunes ve Vieira (2014): Bu çalışma, yaygın olarak kullanılan birkaç otomatik penetrasyon testi aracının analizini amaçlamaktadır. Sonuçlar, web hizmetleri güvenlik testleri için performansların çok etkileyici olmadığını gösteriyordu. Bu nedenle araştırmacılar ve uygulayıcılar için gelecekte web hizmetleri için daha iyi güvenlik sağlayacak zayıf noktaların ve metodolojilerin tespit edilmesinin verimliliğini artırmak için yeni araçlar ve teknikler geliştirmenin yolunu açacak bu araçlar hakkında açık bir görüş imkanı verdi.

Yılmaz (2014): Bu makale, bilgi güvenliğinin önemini vurgulamak, kurumlarda bilgi güvenliğini tehdit eden unsurları ve bunlara karşı alınan tedbirleri belirlemek ve Bilgi Güvenliği Yönetim Sistemini (ISMS) TS ISO / IEC 27001 Standardı ve risklerin analizi hakkında bir çalışma içermektedir.

Kasture ve diğerleri (2015): Bu makale, web hizmetlerine internet üzerinden erişirken veritabanında gerçekleştirilen her işlemle ilgili çerezleri yeniden yazarak bazı saldırıları önlemeyi ve algılamayı amaçlamaktadır. Önerilen sistem ayrıca, bazı verileri saldırganlardan korumak için XSS ve SQL enjeksiyonu gibi saldırıları algılar ve bunları engeller.

Saleh ve diğerleri (2015): Bu çalışma, Boyer-Moore Dizge Eşleştirme Algoritmasını kullanarak web uygulama açıklarını saptamak için bir algılama yöntemi geliştirerek web uygulaması sorunlarını çözmek için bir teknik önerdi. Performansını değerlendirmek için sayısız deney yapılmıştır. Sonuç, önerilen yöntemin sahte sorgulara dayalı açıklıkları doğru bir şekilde tespit etme ve düşük işlem süresi ile saldırganların sorgularını önleme açısından iyi bir performansa sahip olduğunu göstermektedir.

Rafique (2015): Bu makale, web uygulamaları güvenlik açıkları algılama yaklaşımları alanında bildirilen ampirik araştırmanın sentezlenmesi için haritalama çalışmasının bir tanımını sağlamayı amaçlamaktadır. Önerilen çözümler, çözümün önerildiği yazılım geliştirme aşamaları ve OWASP tarafından en yaygın 10 güvenlik açıklığı listesine göre web uygulaması güvenlik açıkları eşleştirmesine karşı oluşturulmuştur.

Prokhorenko ve diğerleri (2016): Bu makale, varolan yaklaşımları bütünsel bir büyük resimde sistemleştirmeyi amaçlayan web uygulaması koruma tekniklerini inceler. Ve kapsanan bazı teknikler (örn. Statik Kod Analizi) her türlü uygulamaya uygulanacak kadar genel olsa da, çeşitli web uygulama koruma tekniklerine odaklanıldığı görülmüştür. Bu makalenin temel katkısı, kapsamlı bir web uygulama koruma tekniği sınıflandırması sağlamaktır.

El-Hajj ve diğerleri (2016): Bu çalışma, web uygulamalarında güvenliği zorlayan bir framework olarak önerildi. Geliştiricinin yalnızca bir güvenlik sınıfıyla veritabanı özniteliklerine açıklama ihtiyacı duyması nedeniyle, minimum geliştirici çabası gerekiyordu. Web uygulama kodu daha sonra Genişletilmiş Program Bağımlılığı Grafiği (EPDG) olarak adlandırılan bir aracı temsil haline dönüştürüldü. EPDG'yi

kullanarak, verilen açıklamalar, uygulama koduna aktarıldı ve güvensiz bilgi akışlarının ortaya çıktıklarında tespit edilmesi için özenle tasarlanmış genel güvenlik uygulama kurallarına karşı yürütüldü. Sonuç olarak, verilerin gizlilik veya dürüstlük politikalarında bir çok ihlal bildirildi.

Vieira ve Serrão (2016): Bu makale, web uygulamaları güvenlik seviyesini değerlendirmek için bir kurumdan birkaç finansal web uygulamasında yapılan otomatik denetim araçları yardımıyla yapılan güvenlik denetim sonuçlarının analizine odaklanmaktadır.

Perera ve diğerleri (2016): Bu araştırmanın amacı, (saldırı tespit sistemi) IDS / IPS (Saldırı Önleme Sistemi) veya WAF (Bir web uygulama güvenlik duvarı) gibi güvenlik katmanlarını atlatabilen kehanet modellerini analiz etmek ve keşif işlemine engel olmadan halledebilecek bir çözüm sağlamaktır. Önerilen çözüm, bilinen bir PHP frameworkü için eklenti olarak gösterildi.

Osman ve diğerleri (2017): Bu çalışmada, internet uygulamalarının güvenlik ve servis korumasını, belirtilen erişim kontrollerini, kriptorolojileri, çerezleri ve oturum yönetimlerini, savunma programlama uygulamalarını, gelişim ömrü boyunca saldırılardan korunmayı, erişim denetiminde donanım kimlik doğrulama tekniklerini kullanan basit bir güvenlik modeli önerdi. Daha sonra MD5'i Based64 ile karıştırarak şifreleme yaklaşımını önererek oturum ve çerez türlerini güvence altına almanın yollarını göz önünde bulundurdu. Buna ek olarak, bu uygulamalar en önemli web güvenliği güvenlik açığı ve erişim kontrolü zayıflığını ve bu zayıflıkların üstesinden nasıl gelinebileceğini tartışıldı ve güvenlik standartlarını ISO 25010 kalite belgesine göre bir Likert ölçeği kullanarak ölçmek, analiz etmek ve değerlendirmek için bir yaklaşım önerdi. Bu çalışmanın gayesi, güvenliklerini korumak için her web uygulaması geliştirme sürecinde uygulanması gereken bir takım teknik ve ipuçlarını göstermektir.

3. MATERYAL VE METOD

Bu bölümde, bilinen güvenlik açıklıklarına karşı Penetrasyon testini black-box yaklaşımı kullanılarak güvenlik açıkları ayrı ayrı incelenecektir. Bir hesabı açmaya ve bankadaki müşterilerin hesabına saldırmaya gerek kalmadan dışardan gelen bir tehlikeye ele alacağımızı dikkate alarak. İzlenilecek yol haritası hakkında Penetrasyon testine başlanacaktır.

Bilgi Güvenlği Akademisi (BGA) Bank, eğitimlerde kullanılmak üzere zafiyetli web uygulaması hizmeti sağlayan bir uygulamadır. Bu uygulama, internet bankacılığı sistemlerinde var olan güvenlik zayıflıkları içeren internet bankacılığından ibarettir. Sistem düzenli olarak sıfırlanır ve kullanıcıların bu güvenlik açıklıklarını bulması beklenir. (Bgabank.com, 2016)

Bu modeli seçmemizin sebebi tasarım açısından gerçek dünyadaki örneklere (bankalar) benzediği gibi Internet'te hizmet veren tüm bankalar güvenlik duvarı da içeriyor. Banka modelini seçmenin nedeni, bazı çevrimiçi bankaların (veya diğer güvenlik açısından kritik uygulamaların) az sayıda başarısız oturum açma işleminden (üç defa) sonra bir hesabı devre dışı bırakmasıdır. Ayrıca, hesap sahibinin, müşteriyi telefonla aramak ve bir dizi güvenlik sorusunu cevaplamak gibi, hesabı yeniden etkinleştirmek için çeşitli çevrim dışı adımları atmasını isterler. Bu politikanın dezavantajları, bir saldırganın, hesapları sürekli olarak devre dışı bırakarak ve hesap kurtarma hizmeti sağlama maliyetini kullanarak meşru kullanıcılara hizmet vermesini engellemesine izin verir. Az miktarda başarısız oturum açma girişiminden sonra (üç defa) kısa bir süre (30 dakika) hesapları askıya almak, güvenlik bilinciyle çalışan uygulamaların çoğu için daha dengeli bir politika olması gerekir. Bu hizmet reddi saldırıları riskini azaltarak ve çağrı merkezi yoğunluğunu minimuma indirerek parola saldırılarını büyük ölçüde yavaşlatır. Bu, sistemimizin kusurları içerdiği anlamına gelmez, ancak bankacılık uygulamalarının internet üzerindeki öneminin bir açıklamasıdır. Şimdi, mevcut güvenlik açıklarını tek tek incelemeye devam edelim.

3.1. Siteler Arası Kod Yazma (XSS)

Tanım:

Bilgisayar korsanlarının bugün kullandıkları Web uygulamalarına yönelik birçok saldırı arasında en yaygın ve ciddi saldırılardan biridir. Bu saldırıların kontrolü ve önlenmesi zorluğu nedeniyle günümüzde büyük bir sorun olarak görülmektedir. İki taraf içeren çoğu saldırıdan farklı olarak: saldırgan, web sitesi veya saldırgan ve mağdur müşteri olarak XSS saldırısı üç taraf içerir: saldırgan, bir istemci ve web sitesi (Klein 2006). Kötü amaçlı komut dizilerinin yararlı ve güvenilir web sitelerine enjekte edildiği bir enjeksiyon türüdür ve bu kötü niyetli içerik sıklıkla bir JavaScript bölümü alır. Ancak HTML, Flash veya tarayıcının kullandığı başka herhangi bir kod türünü de içerebilir (OWASP 2016).

Kaynaklanma nedenleri:

Bu güvenlik açığı, bir web uygulaması ve web sayfalarındaki kullanıcılardan alınan girdileri düzgün bir şekilde kontrol etmeden, kullanıldığında ortaya çıkar (Shar ve Tan 2012). Bu, bir saldırganın ziyaretçinin bilgisi olmadan bir ziyaretçinin tarayıcısında çalışan ve dolayısıyla saldırganın hassas kullanıcı verilerine (Yusof ve Pathan 2016) erişmesine olanak tanıyan bir web sitesine kötü niyetli komut dosyası yerleştirmesine olanak tanır.

Saldırı hedefleri:

XSS saldırısının amacı, istemci çerezlerini veya diğer hassas bilgileri çalmaktır; saldırganlar, istemciyi web sitesi ile tanımlayabilir veya kimlik hırsızlığı, anahtar kayıtları, kimlik avı, kullanıcı kimliğine bürünme ve web kamerası gibi çeşitli kötü niyetli işlemleri yapabilir. Bu komut dosyalarının HTML sayfasının içeriğini bile yeniden yazılabilir.

Saldırının önemi ve Gerçek dünya örnekleri:

Bu zafiyet, hacker (bilgisayar korsanı) çevrelerinde çok popülerdir ve etkileri, güvenlik açığından etkilenen site tarafından işlenen verilerin hassaslığına ve sitenin sahibi tarafından uygulanan herhangi bir güvenlik azalmasının doğasına bağlı olarak, küçük bir sıkıntıdan önemli bir güvenlik riskine kadar değişebilir. Göz ardı edilmemelidir. Ek dosya eylemi nedeniyle XSS saldırısı CWE Cyber Security (Siber Güvenlik)'e bildirildi (CWE 2013). Web sitesi güvenliği istatistiklerine göre (Industry Benchmarks 2010), hedefli XSS saldırılarında, apache.org'u içeren ve parolaların ele geçirildiği önemli bir olay da dahil olmak üzere bir artış meydana geldi (Das ve diğerleri 2015). Facebook, Google, PayPal ve Twitter gibi büyük uygulama hizmetleri bile, 2003 Acil Müdahale Ekibi danışma belgesinde ilk kez rapor edildiğinden bu yana endişe verici derecede artan XSS saldırıları geçirdi. Açık Web Uygulaması Güvenlik Projesi (OWASP), XSS'yi, en yaygın 10 Web güvenlik açıkları listesinde yer alan 2017 listesinde üçüncü sırada yer alarak "en yaygın Web uygulaması güvenlik açığı" olarak nitelendirdi (OWASP 2017). WhiteHat Security'nin Mayıs 2013 Web Güvenliği İstatistikleri Raporu, XSS müdahalelerinin yaygın riskinin altını çizerek, Web uygulamalarının yüzde 43'ünün bu tür saldırılara karşı savunmasız olduğunu kaydetti. (Yusof ve Pathan 2016)

Türleri:

Bunlardan bir tanesi DOM temelli saldırılar olan örneğimizin dışındaki üç farklı XSS saldırısı sınıfı vardır ve bunlardan bazıları şunlardır: Yansıyan saldırılar ve Saklanan saldırılar.

3.1.1. Yansıtılan xss saldırıları

Tanım: Yansıyan (Reflected) XSS, web açıklarının en yaygın türüdür. Yansıyan bir saldırıda, enjekte edilen komut, hata mesajında, arama sonucunda veya isteğin bir parçası olarak sunucuya gönderilen girdinin tamamını veya bir kısmını içeren herhangi bir yanıt gibi web sunucusundan yansır. Dolayısıyla, bu saldırı yükü tek bir istek ve yanıt yoluyla iletilir ve yürütülür. Potansiyel bir vektörün klasik bir örneği

bir site arama motorudur: bir dize ararsa, arama dizesi genellikle sonuç sayfasında neyin arandığı belirtilmek üzere tekrar gösterilir. Yansıtılmış XSS'ye bazen Sürekli Olmayan veya Tip-II XSS adı verilir.

Kaynaklanma nedenleri: Yansıyan saldırı, web sitesi veya web uygulaması kullanıcı girişini, yanıt sonuçlarının bir parçası olarak düzgün bir şekilde sanitize edilmeden kullandığında ortaya çıkar.

Saldırının senaryosu: Genellikle bir saldırgan URL'deki kötü amaçlı komut dosyalarını gizler ve genel olarak kurbanlarına bir e-posta iletisinde olduğu gibi başka bir yolla veya başka bir tarafsız web sitesinde gönderilir. Bir kullanıcı kötü niyetli bir bağlantıyı tıklattığında, özel hazırlanmış bir form gönderirken veya yalnızca kötü niyetli bir siteye göz attığında, enjekte edilen kod savunmasız web sitesine gider ve kurbanın tarayıcısına tekrar yansıtılır. Tarayıcı daha sonra isteği sağlıklı hale getirmeden zararlı komut dosyasını yürütür ve "güvenilir" bir sunucudan geldiğinden saldırganın kimliği doğrulanmış çerezleri veya verileri çalmasına izin verir. Şekil 3.1'de, mağdurların savunmasız bölgede kendilerini doğruladıklarını varsayıyoruz.

Şekil 3.1. Yansıtılan XSS saldırısının tipik senaryosu

Yansıtılan bir xss güvenlik açığını denetleme:

Sistemimizin yansıtılan XSS'yi vardığını ispatlamak için, XSS güvenlik açıkları için tüm kullanıcı girdilerini kontrol etmeliyiz. Zafiyet bilgileri aşağıdaki belirtilmiştir.

Araç: Kali Linux tarayıcısını (Iceweasel); Internet Explorer
URL: http://isube.bgabank.com/?sayfa=arama.
HTTP Talep Turu: GET
Parametre: s1 veya s2
Payload: <Script> prompt (BGA) </Script>
<Script>prompt(document.cookie);</Script>

Saldırı adımları

1. Bu zafiyeti istismar etmeden önce normal bir arama yapılır. "XSS" kayıtını aratıldığında "Aradığınız Kayıt Bulunamadı" hatası dönmektedir. Bu durum Şekil 3.2'de gösterilmiştir.

Şekil 3.2. Arama sonuçları sayfası (Aradığınız Kayıt Bulunmadı)

Şekil 3.2'de gösterildiği gibi, arama sonuçları sayfası orijinal girdiyi (XSS) sonuçların bir parçası olarak yazdırır.

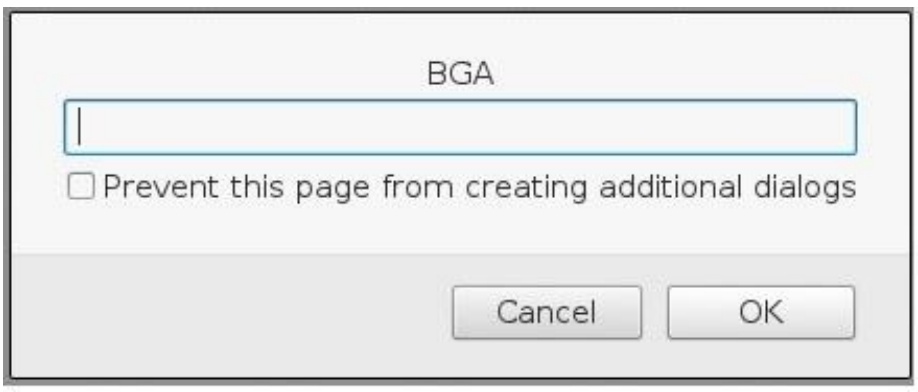
2. Daha sonra çeşitli payloadlar denenebilir. En sık kullanılan payloadlardan <script>alert(123)</script> denendiğinde, sonuç Şekil 3.3.teki gibidir.

Şekil 3.3. Arama sonuçları sayfası (Hata Oluştu)

Şekil 3.3'te gösterildiği gibi, arama sonuçları sayfası yanlış bir mesaj yazdırmaktadır, bu da uygulamadaki bir güvenlik duvarı anlamına vardır.

Şimdi, uygulamada güvenlik duvarı olmadığına veya Internet Explorer'da veya başka bir tarayıcıda böyle bir kod girmeye çalışırsanız, açılır pencere görünmeyecektir ve tarayıcı otomatik olarak açık XSS saldırılarını engelleyecektir. "Internet Explorer, bu sayfayı çapraz site komut dosyalarını önlemeye yardımcı olmak için değiştirdi" iletisini gösterebilir. Bunun nedeni, tarayıcıların son sürümlerinde kullanıcıları yansıyan XSS güvenlik açıklarına karşı korumak için tasarlanmış yerleşik bir mekanizma içeriyor olmasıdır. Ve Iceweasel böyle bir tarayıcıdır. Bu örnekleri test etmek isterseniz, bu korumayı kullanmayan farklı bir tarayıcı deneyebilirsiniz veya şu adımlara giderek XSS filtresini devre dışı bırakabilirsiniz: Araçlar - Ayarlar - Gelişmiş ayarları göster - Ağ - Proxy ayarını değiştir - Güvenlik - Özel Düzey - XSS filtresini etkinleştir altında, Devre dışı bırak'ı seçin.

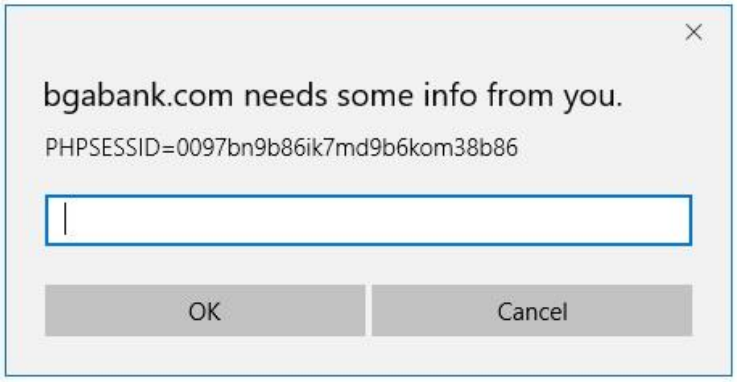
3. Bu engeli atlatmak için en basit yöntemlerden biri script kelimesini büyük- küçük harf kullanarak yazmaktır. Şekil 3.4.te <Script> prompt (BGA) </Script> payloadı denenmiştir ve sonuç başarılıdır.



Şekil 3.4. R-XSS güvenlik açığı arama kutusunda mevcuttur

Şekil 3.4 gösterildiği gibi, açılır pencerenin komut dosyası çalıştırılır. Bu, XSS'de bir güvenlik açığı bulunduğunu ve kullanıcının, test kullanıcısının bağlantısını tıklarsa, kullanıcının herhangi bir tarayıcısında kendi istediği kodu çalıştırabileceği anlamına gelir.

4. Bu payload haricinde, başka bir tarayıcı kullanarak (Internet Explorer gibi) ile farklı payload <Script>prompt(document.cookie);</Script> çalıştırılabilir. Elde edilen sonuç Şekil 3.5 'te verilmiştir.



Şekil 3.5. Internet Explorer'ı kullanarak R-XSS güvenlik açığı

Kullanıcı, çerçevenin orijinal sayfanın bir parçası olduğunu düşünebilir ve kimlik bilgilerini girerek saldırgana sahadan gönderilir.

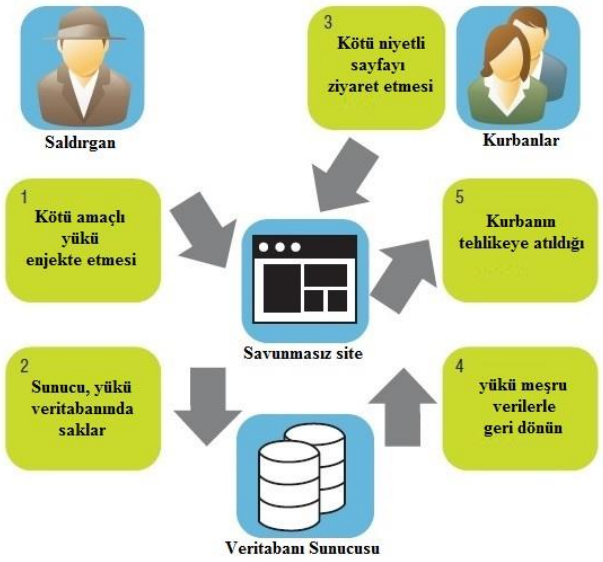
3.1.2. Depolanan/saklanan xss saldırıları

Tanım: Saklanan saldırı türü, komut dosyası veritabanında, mesaj forumunda, ziyaretçi günlüğünde, yorum alanında vb. depolayan bir web sitesine kötü niyetli komut dosyası yerleştirmeyi içerir ve çapraz site komut diziminin en tehlikeli ve en yıkıcı biçimidir. Saklanan XSS'ye bazen Sürekli veya Tip-I XSS denir.

Kaynaklanma nedenleri: Kullanıcıların verileri saklamasına izin veren Web uygulamaları potansiyel olarak bu tür saldırılara maruz kalmaktadır. Komut dosyası doğru filtrelenmezse, web uygulamasının bir parçası gibi görünür ve uygulamanın ayrıcalıkları altında bir kullanıcının tarayıcısında çalışır.

Saldırının ciddiyet: Saklanan XSS, kullanıcının verdiği veriler sunucu tarafından kaydedildiğinde, kötü niyetli komut dosyası otomatik olarak oluşturulduğu için algılanması zor olur ve kullanılmaya başlanacak kötü amaçlı bir bağlantıya ihtiyaç yoktur. Sonuç olarak, saldırganlar etkinliklerini kolayca gizleyebilir; Örneğin, bir blogda, durumu görünüşte zararsız bir yoruma gömebilirler.

Saldırının senaryosu: Bu saldırında, kötü niyetli komut dosyası veritabanında yerleştiriliyor, web uygulamasının bir parçası gibi görünür, ve kullanılmaya başlanacak kötü amaçlı bir bağlantıya ihtiyaç yoktur. Bir kullanıcı, depolanmış bir XSS içeren bir sayfayı ziyaret ettiğinde başarılı bir uygulama meydana gelir. Şekil 3.6 daki Depolanan XSS saldırının senaryosu gösterilmiştir. Enjeksiyon yöntemleri büyük farklılıklar gösterebilir; bazı durumlarda, saldırganın böyle bir deliği kullanmak için doğrudan web işlevselliği ile etkileşime girmesi gerekmeyebilir. Bir saldırgan tarafından denetlenebilen web uygulaması tarafından alınan herhangi bir veri (e-posta, sistem günlükleri, IM vb.), bir enjeksiyon vektörü haline gelebilir.



Şekil 3.6. Depolanmış (XSS) saldırının tipik senaryosu

Saklanan bir xss güvenlik açığını denetleme:

Saklanan XSS güvenlik açıklarını test etmenin en iyi yolu, 'Ziyaretçi defteri' bölümüne kodu enjekte etmektir. Çünkü veritabanında kaydedilecek, ve sayfa ne zaman ziyaret edilirse yükü çalışacaktır. Zafiyet bilgileri aşağıdaki belirtilmiştir.

Araç: Iceweasel

URL: http://isube.bgabank.com/ziyaretcidef.aspx

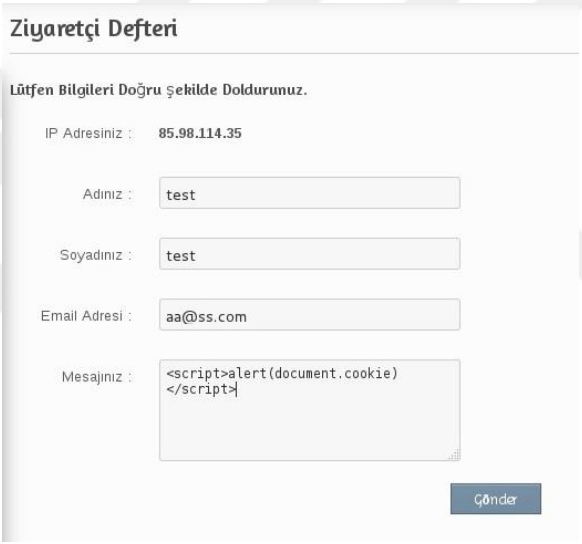
HTTP Talep Turu: POST

Parametre: k4

Payload: <Script>prompt(document.cookie);</Script>

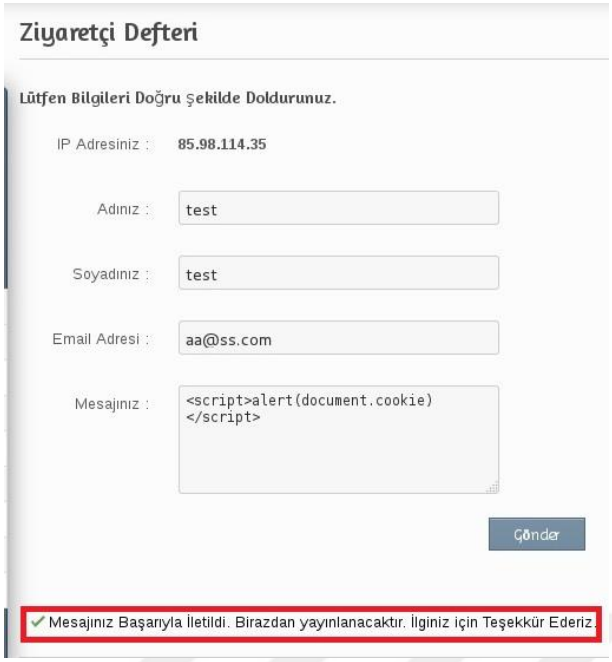
Saldırı adımları

1. Ziyaretçi defteri sayfasında yer alan "Mesajınız" sekmesinde Şekil 3.7 gibi doldurma formuna yazılır.



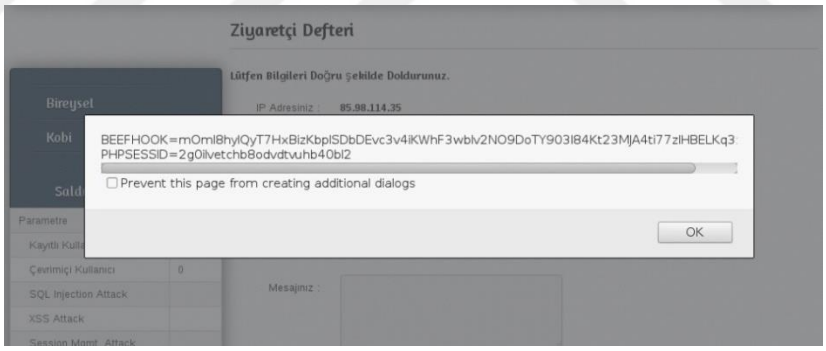
Şekil 3.9. Ziyaretçi Defteri sayfası

2. Gönder'e bastıktan sonra girişin başarılı olduğunu onaylayan aşağıdaki mesaj 3.8 Şeklinde görünecektir:



Şekil 3.8. Girilen Başarı Mesajı

Ardından yük yüklenecektir Şekil 3.9, mesaj diğer mesajlarla birlikte veritabanında saklanır Şekil 3.10, ve sayfa her ziyaret edildiğinde yükü tekrar çalışacaktır.



Şekil 3.9. Ziyaretçi Defteri sayfasında S-XSS güvenlik açığı bulunmaktadır

Şekil 3.10. Mesajlar veritabanına kaydedilir

3.1.3. Belge nesnesi modeli (DOM) xss saldırıları

DOM tabanlı bir XSS saldırısı en gelişmiş tiptir ve iyi bilinmemektedir. Gerçekten de, bu saldırı türündeki güvenlik açığının büyük kısmı, Web uygulama geliştiricilerinin nasıl çalıştığını tam olarak anlamamış olmasından kaynaklanmaktadır. DOM tabanlı veya Tip 0, XSS saldırısı, yansıtılan XSS saldırısı ile aynı şekilde yürütülür.

DOM tabanlı bir saldırıda, sunucunun kötü amaçlı yükü HTTP yanıtında taşımasına değil, bir URL'de kötü amaçlı bir değeri kodlayan ve mağdura gönderen bir DOM tabanlı saldırıdadır. Saldırı, kurbanın tarayıcısı değiştirilmiş DOM'dan gelen kötü amaçlı kodu çalıştığında ortaya çıkar. İstemci tarafında, HTTP yanıtı değişmez, ancak komut dosyası kötü amaçla yürütülür. Bu açıklama yalnızca tarayıcı URL karakterlerini değiştirmezse kullanılabilir. (Yusof ve Pathan, 2014)

3.2. Müşteri No. Bilgi İfşası

Tüm sistemlerde oturum açma işlevleri, bir saldırganın kullanıcı adlarını ve parolalarını tahmin etmeye çalışması için açık davetiye sunar, ve bu nedenle de uygulamaya yetkisiz erişim sağlamaya yönelik açık bir kapı bırakır. Uygulama, bir saldırganın doğru parolayı tahmin edinceye kadar farklı şifrelerle tekrarlanan giriş denemeleri yapmasına izin verirse, bu son derece savunmasız olduğunu göstermektedir. Farklı parolalar ile tekrarlanan giriş denemeleri yapmak isteyen saldırgan kaba kuvvet saldırıları denemek zorundadır. Ancak saldırganın bir saldırıya başlamadan önce bir veya daha fazla belirli kullanıcı adlarını keşfetmesi gerektiğini belirtmek gerekir. Başka bir deyişle, kullanıcı adı yoksa kaba kuvvet saldırısı da yoktur.

Müşteri No. bilgi ifşası güvenlik açığını denetleme:

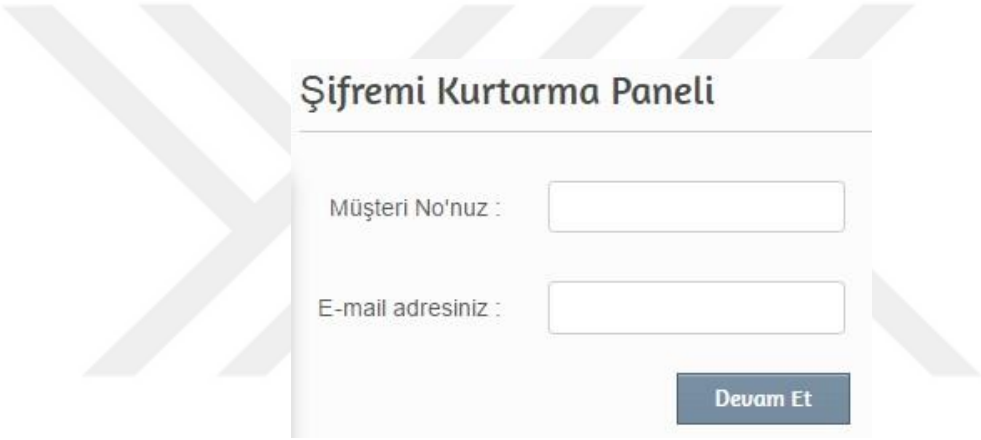
Sistemimizde bir müşteri numara üzerinden bilgi sızdırma ile ilgili güvenlik açığı bulunmaktadır. Bu yüzden saldırganın kaba kuvvet saldırıları uygulamasına izin

vermektedir. Zafiyet bilgileri ve müşteri numara bilgisi aşağıdakileri yaparak belirtilmiştir:

Araç: Iceweasel **URL:** http://isube.bgabank.com/giris.aspx **HTTP Talep Turu:** POST

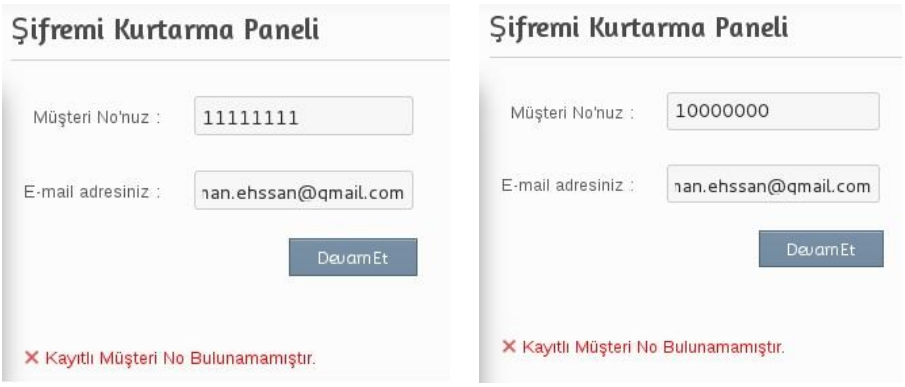
Saldırı adımları

1. Müşteri Giriş Paneli web sayfasında Şifremi Unuttum üzerine basarak aşağıdaki Şekil 3.11'de web sayfası gösterilecektir:



Şekil 3.11. Şifremi Kurtarma Paneli

2. Şimdi, yanlış bir Müşteri No. ve e-posta girilirse, "Kayıtlı Müşteri No Bulunamamıştır " sonuç mesajı Şekil 3.12'de gibi gösterilir:



Şekil 3.12. Kayıtlı Müşteri No Bulunamamıştır sayfası

3. Ancak doğru bir Müşteri No. girilirse ve bir e-posta girilirse, sonuç mesajı ‘‘Mail Adresi Müşteri No İle İlişkili Değil.’’ sonuç mesajı Şekil 3.13’te gösterilir.

Şekil 3.13. Mail Adresi Müşteri No İle İlişkili Değil sayfası

Bu Müşteri No.'un var olduğunu gösterir. Bu nedenle kaba kuvvet saldırıları, güçlü bir CAPTCHA yoksa yürütülebilir.

3.3. CAPTCHA Bypass

CAPTCHA, Tamamen Otomatik Genel Testi anlamına gelir. İnsanları bilgisayarlardan ayırmak için kullanılan zorunlu bir yanıt testidir (Bypass Captcha Ana Sayfa, 2017). Bu prosedür, özellikle günlük işlerin okunması zor olan çarpıtılmış kelimelerle yavaşladığını hisseden insanlardan bir çok eleştiri almıştır.

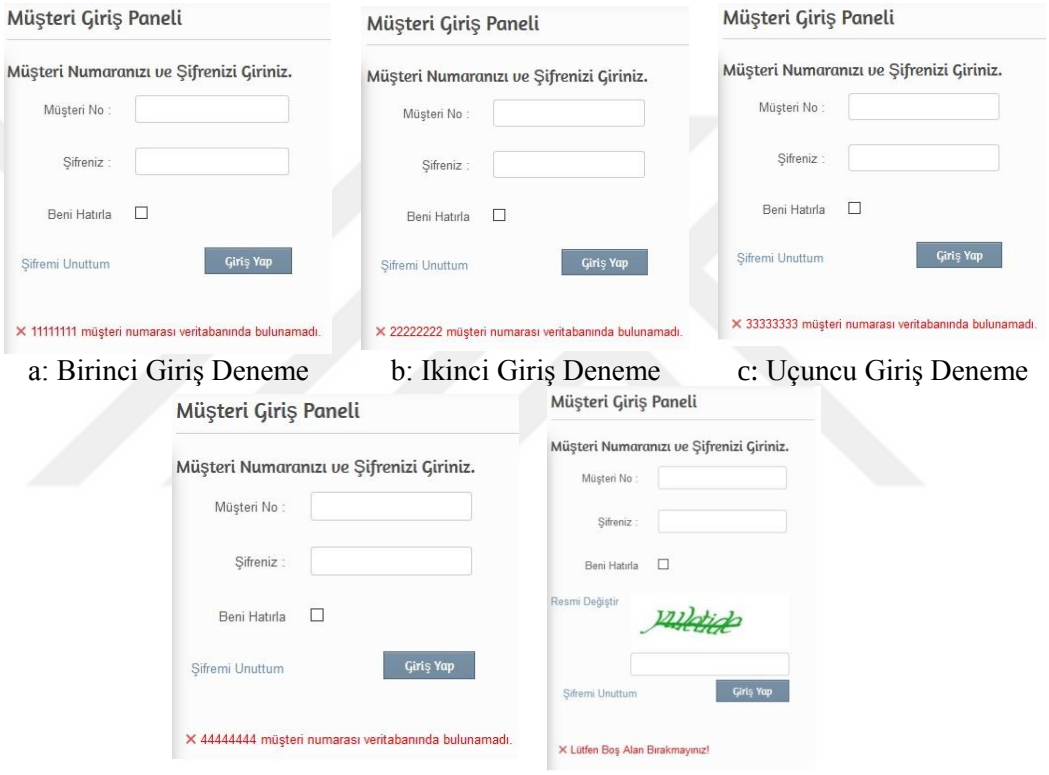
Genellikle güvenlik nedenleriyle kullanıldığında, CAPTCHA'lar ayrıca yapay zekâ teknolojileri için bir kriter görevi görür (Savinkin, 2013). Bazen captcha üreten yazılımın bir kısmı istemci tarafındaysa doğrulama bir sunucuda yapılır ancak kullanıcılara istemci değişkenini göstermek için captca değiştirilebilir.

Bazı CAPTCHA sistemleri, istemci tarafında depolanan MD5 özetlerini kullanılır. Bu durum captcha'yı kaba kuvvet saldırısına karşı savunmasız bırakabilir. Sistemimizde CAPTCHA üç problemi içeriyor:

Kullanıcı aracı bilgisi değiştirerek (Mobil Giriş) captcha atlatmayını denetleme:

Sistemizde, müşteri girişi panelinde, Şekil 3.14 gibi eger Müşteri No veya parola dört kez yanlış giriş denemesine maruz kaldığında kaba kuvvet saldırısını engellemek için captcha çıkmaktadır. Zafiyet bilgileri aşağıdakileri yaparak belirtilmiştir:

Araç: Iceweasel
URL: http://isube.bgabank.com/giris.aspx



a: Birinci Giriş Deneme b: Ikinci Giriş Deneme c: Uçuncu Giriş Deneme

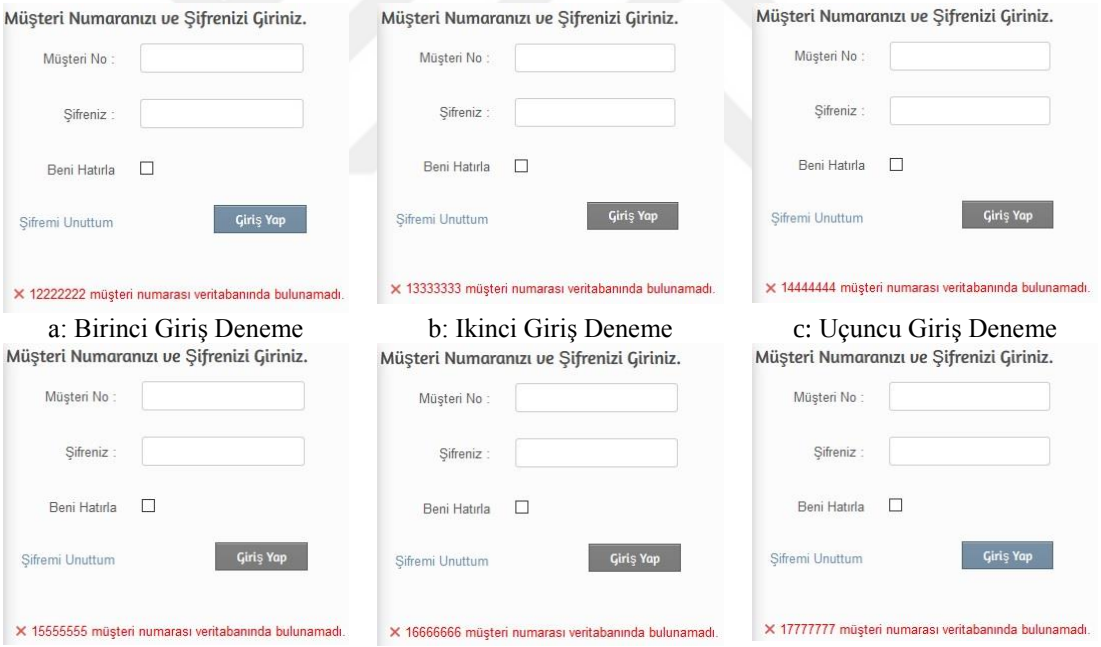
d: Dördüncü Giriş Deneme e: Captcha Panel

Şekil 3.14. Müşteri Giriş denemeler ve captcha göstermak

Fakat mobil cihazla girildiğinde captcha çıkmamaktadır. Iceweasel tarayıcısı kullanılarak, tarayıcıda user-agent bilgisi değiştirildikten sonra mobil cihaz gibi siteye giriş yapılabiliriz. Bu durum Şekil 3.15'te gösterilmektedir.

Şekil 3.15. Kullanıcı Aracısı Değiştirici

Bu durumdayken "User Agent Switcher" ile cihaz iPhone olarak ayarlanır ve yanlış veriler girilip "Giriş Yap" butonuna tıklandığında Captcha 'nın kaybolduğu görülecektir. Elde edilen sonuç Şekil 3.16'da verilmiştir.



a: Birinci Giriş Deneme b: Ikinci Giriş Deneme c: Uçuncu Giriş Deneme

d: Dördüncü Giriş Deneme c: Beşinci Giriş Deneme d: Altıncı Giriş Deneme

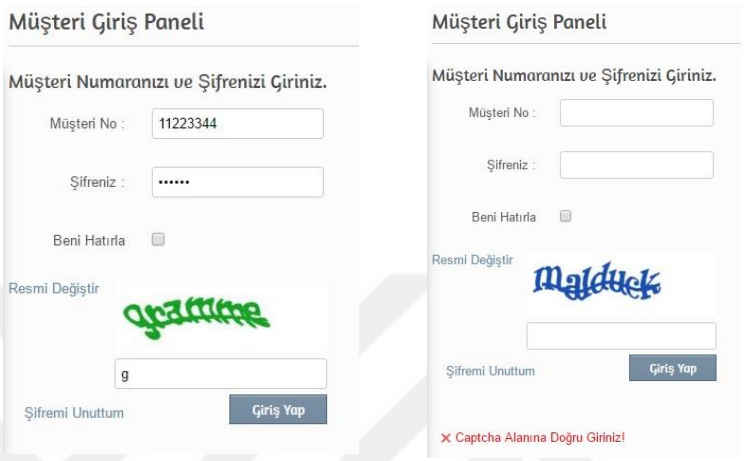
Şekil 3.16. Müşteri Giriş deneme, captcha çıkatmaz.

Çalışmayan captcha uygulaması zayıflığını denetleme:

Müşteri giriş panelinde captcha'ya baktığımızda ilk karakter giriliyor ve enter tuşuna basılmaya çalışılıyor. Captcha'nın şu hata mesajını verdiğini göreceğiz: 'Captcha

Alanına Doğru Giriniz!'. Bu durum Şekil 3.17 'de gösterilmiştir. Zafiyet bilgileri aşağıdakileri yaparak belirtilmiştir:

Araç: Iceweasel
URL: http://isube.bgabank.com/administrator.aspx



Şekil 3.17. Müşteri Girişi captchanın Paneli

Ancak, captcha'nın çalışmadığı ve yalnızca boş olarak kontrol edildiği yönetici giriş sayfasında bu olmamıştır. Aşağıdaki Şekil 3.18'de gösterilmiştir.

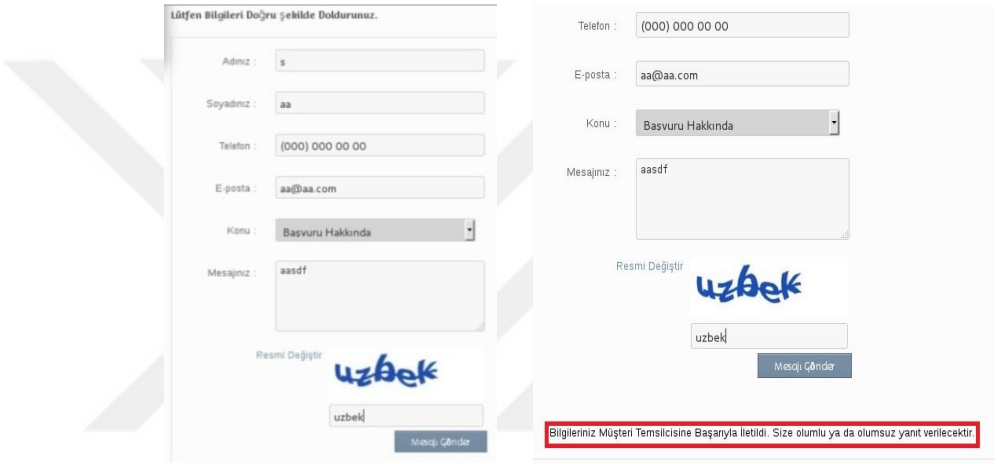
Şekil 3.18. Yönetici Girişi Sayfasının Paneli

CAPTCHA'nın başarısızlığı kaba kuvvet saldırılarının yönetici verileri elde etmek için oluşturulabileceğini gösterir. Ancak saldırganın bir yönetici kullanıcı adı bilgisi elde etmesiyle bu durum mümkündür.

Captcha atlatarak brute force saldırısı gerçekleştirmesi denetleme:

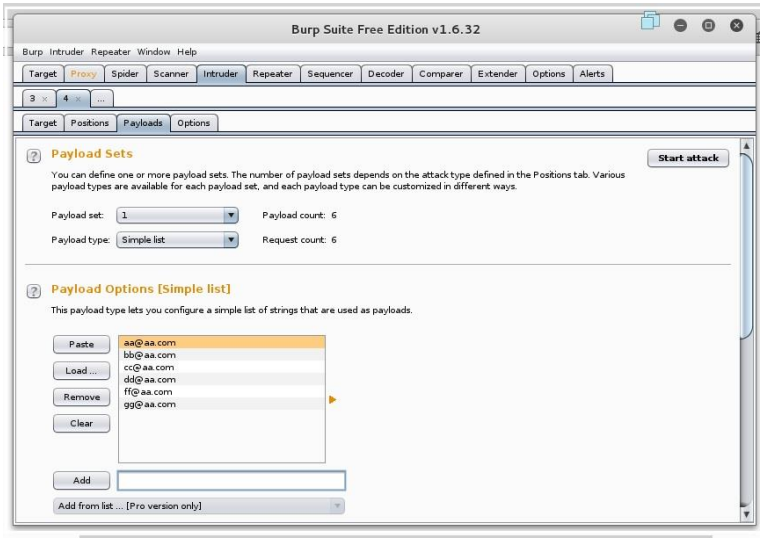
İletişim web sayfasında captcha iyi çalışmasına rağmen tek bir CAPTCHA ile birçok GET isteği gönderilir. Bu GET istekleri müdahale edilerek gönderilebilir, dolayısıyla kaba kuvvet saldırılarına olanak verir. Bu durum Şekil 3.19 da gösterilmiştir. Zafiyet bilgileri aşağıdakileri yaparak belirtilmiştir:

Araç: Iceweasel; Burp Suite
URL: http://isube.bgabank.com/iletisim.aspx

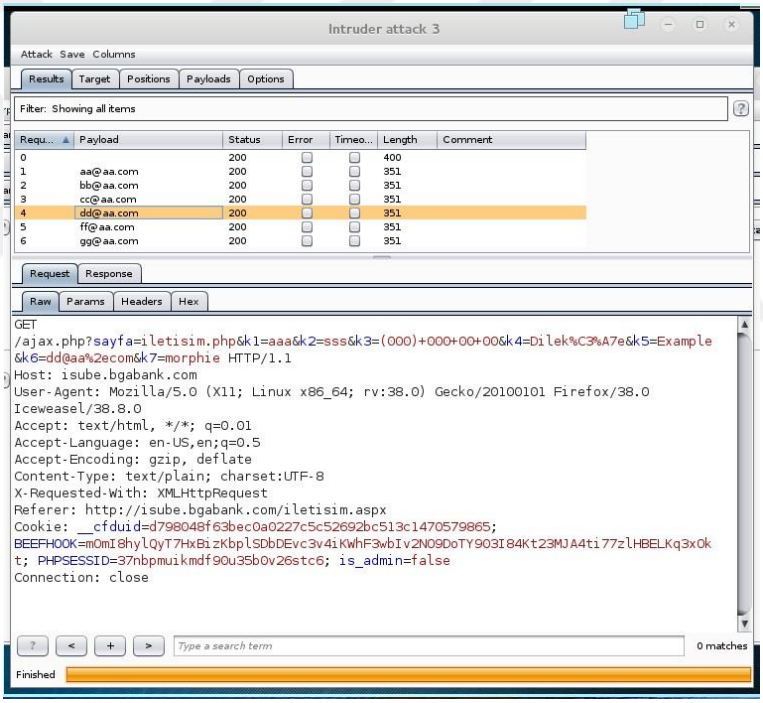


Şekil 3.19. İletişim Sayfası Captcha Kontrolü

Burp Suite kullanılarak, GET isteği yakalanabilir ve yapmak istediğimiz, örneğin e-posta adresini değiştirme ve aynı Captcha ile yeniden istek gönderme gibi değişiklikler yapabiliriz. Elde edilen sonuçlar Şekiller 3.20, 3.21 de verilmiştir.



Şekil 3.20. Burp Suite Intruder Payload Belirleme



Şekil 3.21. Burp Suite Intruder sonuç

3.4. Müşteri Girişi Form Tabanlı Brute Force Saldırısı

Tabii ki, her saldırı metodu birbirini tamamlayabilir. Çok tehlikeli bir güvenlik açığı, burada meydana gelen iki küçük güvenlik açığından kaynaklanıyor olabilir. Kısım 3.2'den birkaç Müşteri Numaraları çıkarabildik, böylece başarılı bir kaba kuvvet saldırısı için ilk adım elde edildi. Ayrıca, Bölüm 3.3 'te "Kullanıcı Aracısı

Değiştirici" ni değiştirerek Capcha'yı geçmeyi başardık ve bu, her hesap için şifreyi elde etmek için kaba kuvvet saldırısı yapabilmemizi sağlıyor.

Müşteri Girişi Form Tabanlı Brute Force Saldırısı Denetleme

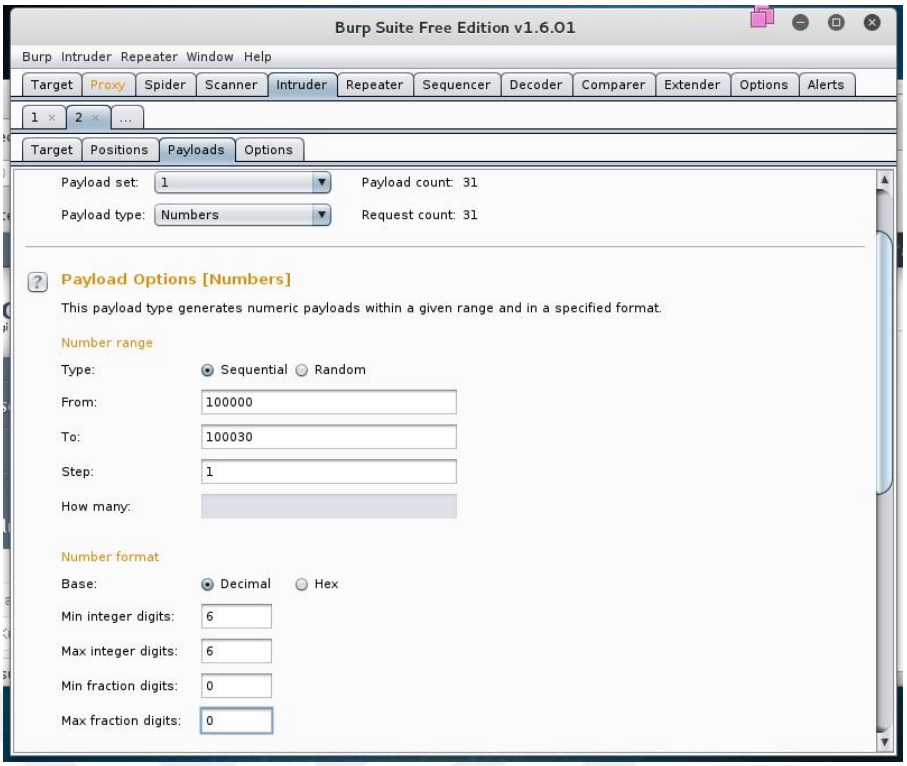
Müşteri giriş panelinde 3.3 başlıkta anlatılan mobil giriş captcha atlatma tekniği kullanılarak, form brute force saldırısına açık hale getirilir. Zafiyet bilgileri aşağıdakileri yaparak belirtilmiştir:

Araç: Iceweasel; Burp Suite
URL: http://isube.bgabank.com/giris.aspx
HTTP Talep Turu: POST
Parametre: b_musterino

Saldırı adımları

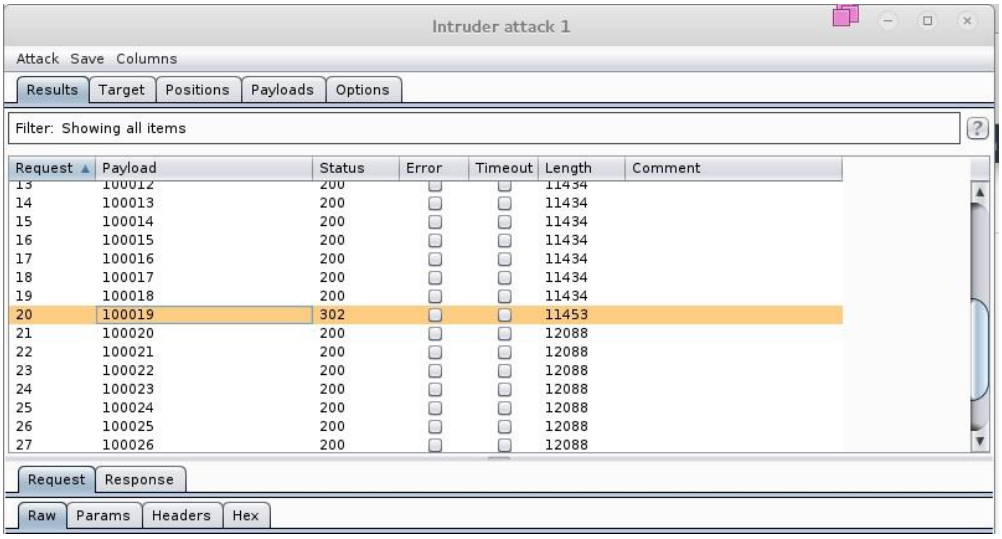
1. Burp Suite ile bölüm 3.2 'den saptanan (10000150) Müşteri Numarası için istek yakalanarak, intruder'e gönderilir.
2. Pozisyonlar düzenlenir ve ilk parametreden sonraki parametreler silinir.
3. Wordlist yüklemek için Payloads sekmesine gidilir. Ardından Payload type kısmından Custom Iterator seçilir.
4. Payload Options bölümünden her pozisyon için gerekli kelime listeleri yüklenir.
5. Seperator for Position 1 için silinen ikinci parametre değeri (&b_password=) girilir.

Yük miktarı Şekil 3.22'de gösterilmiştir.



Şekil 3.22. Intruder Payload Belirleme

Saldırı başlatılı ve sonuçlar gözlemlenir. Şekil 23. te görüldüğü gibi 20. istekte, cevap olarak diğerlerinden farklı length ve status değerleri dönmüştür. Yani 10000150 numaralı müşterinin şifresi hacked olarak saptanmıştır.



Şekil 3.23. Intruder Payload sonuç

Böylece saldırı başarıyla gerçekleştirildi ve hesap şifresi belirlendi ve hesap açıldı. Ayrıca (10000151) Müşteri Numarası için aynı yöntemi kullanırken saldırı, Şekil 3.24'te gösterildiği gibi başarıyla yürütülecektir.

10000150 Müşteri Numarası	10000151 Müşteri Numarası

Şekil 3.24. Saldırılar sonuçları

3.5. HTTP Parametre Kirliliği

Tanım:

Bir web uygulamasında istenen davranışları manipüle etmenin bir diğer yöntemi de HPP (HTTP Parametre Kirliliği)'dir. HPP, web uygulamasının istenen davranışından farklı olarak belirli bir kötü amaçlı görev veya saldırı gerçekleştirmek için bir web uygulamasının HTTP parametrelerini kirletir. Bu güvenlik açığı basit ve oldukça etkilidir. İlk olarak 2009 yılında tespit edilmiştir. (Carettoni ve di Paola, 2009)

Kaynaklanma nedenleri:

Bu saldırının temel nedeni, aynı isime sahip birden fazla HTTP parametresi almasıdır. Girdinin kontrol edilmemesi HPP var olan veya başka HTTP parametrelerinde (GET / POST / Çerez gibi) kodlanmış sorgular dizesi doğrudan

parametrelere yeni bir parametre eklemesine neden olur. Bu saldırı ister istemci tarafından ister sunucu tarafından olsun, tüm web teknolojilerini etkiler.

Saldırı hedefleri:

HPP güvenlik açıkları, var olan kodlanmış HTTP parametrelerinin yerini almak, amaçlanan davranışı ya da normal uygulama davranışını değiştirmek veya doğru şekilde kontrol edilmeyen değişkenlere erişmek ve potansiyel olarak bunları kötüye kullanmak ve WAF kurallarını atlamak veya girdi doğrulama mekanizmalarını kullanmak için kullanılabilir. Bu nedenle, bir web uygulaması HPP saldırılarına açıksa, web uygulamasının güvenliği tehlikeye girerek bir saldırgana kötü amaçlı veya yasa dışı faaliyetler gerçekleştirmenin kolay bir yolunu sağlar. (Daniel, 2017)

Saldırının önemi:

HPP'nin yerçekimi, WAF bypass söz konusu olduğunda ortaya çıkmaktadır; Sorun, WAF'nin sunucu teknolojisinin HTTP uygulamasını taklit edememesi ve bir empedans uyumsuzluğuna neden olmasıdır. Dmitriy Evteev'in yöntemi, empedans uyuşmazlığını veya HTTP uygulamasının zayıflığını kullanmaz. Bunun yerine, yükü WAF tarafından filtrelenemez, ancak sunucu tarafından işlem yapılacak şekilde paylaştırarak spesifik bir SQL önleme hassasiyetini (aynı sorguda iki savunmasız parametrenin bulunduğu) sömüren akıllı bir yöntemdir (Evteev, 2009). Burada, saldırı vektörünü göre birisi fark edebilir, HTTP Parametre Kirliliği (HPP) HTTP Parametre Parçalanmasına (HPF) çok benzer. Ancak iki farklı alanda zayıf noktaları istismar ederler. HPP uygulaması, HPF'nin aksine uygulama ortamını hedef alan, web uygulamasındaki güvenlik açığının kullanılması içindir. Dolayısıyla, her iki saldırı vektörü de birbirini tamamlayabilir.

Türleri:

HTTP, aynı parametrelerin birden fazla kez gönderilmesini sağlar (Unity Sec, 2014). Şimdiki HTTP standartları, aynı isime sahip birden çok giriş parametresinin nasıl yorumlanacağı konusunda rehberlik içermez. Her parametre değerinin

manipülasyonu, her web teknolojisinin bu parametrelerin nasıl ayrıştırdığına bağlıdır (Singh, 2011). Bazı web teknolojileri, parametrenin ilk veya son durumunu ayrıştırır, bazıları tüm girdileri birleştirir ve diğerleri bir dizi parametre oluşturur (Daniel, 2017). OWASP (OWASP, 2017) Her web teknolojisinin sunucu tarafında aynı parametrelerin farklı değerlerini nasıl işlendiğini gösteren ayrıntılı bir tablo yayınladı. Her teknolojideki parametrelerin çözümlenmesindeki bu fark, farklı saldırılarla gerçekleştirilebilir. Bu, tetiklendiği şekle bağlı olarak istemci tarafında veya sunucu tarafında saldırılara neden olur. Her durumda parametreler, web uygulamasının istemci veya sunucu saldırısı gerçekleştirmek üzere ataklar gerçekleştirir. HPP, istemci taraflı ve sunucu taraflı olmak üzere 2 kategoriye ayrılmaktadır.

a. Client Side HPP

HPP İstemci tarafı saldırıları kullanıcı işlemlerinin (bir tarayıcıdaki bağlantıya erişmek gibi) etkilenip kullanıcı bilgisi olmadan istenmeyen işlemleri tetikleyebileceği, istemci veya kullanıcı ile ilgilidir. HPP İstemci tarafı saldırıları Yansıtılan HPP olabilir (URL bağlantılarına ve / veya diğer src özniteliklerine ek parametrelerin eklenmesi gibi), saklanan HPP olabilir (Veri, src ve href öznitelikleri ile tüm etiketler üzerinde işlev gösterebilen). POST yöntemiyle oluşturulan, ve DOM tabanlı saldırı, çoğunlukla beklenmedik parametrelerin çözümlenmesine ve istemci tarafı HPP'nin JavaScript'i kullanarak gerçekleştirilmesine bağlıdır. Açıkçası, enjeksiyonun kabiliyeti veya kapasitesi bağlantı niteliğine ve işlevlerine bağlıdır. Bununla birlikte, asıl amaç müşteri tarafından HPP saldırıları oluşturmaktır.

b. Sunucu Tarafı HPP

Bu saldırı tipi sunucu ortamını etkiler. Saldırgan, HPP'yi bir web uygulamasının veritabanına erişmek için tetikler ve zafiyetli bir web uygulamasının işlevlerini kullanarak saldırılar gerçekleştirir. Bununla birlikte web uygulama güvenlik duvarı (WAF) kurallarını atlamak için de kullanılabilir. Bazı WAF'ler yalnızca ilk veya sonuncusu gibi tek bir parametre oluşumunu doğrulamaktadır. Bazıları tüm girdileri birleştirir ve diğerleri de bir dizi parametre oluşturur. Ardından bir saldırgan kötü amaçlı kodu bu oluşumlara bölebilir ve böylece güvenlik mekanizmasını veya web

uygulama güvenlik duvarı kurallarını atlayabilir. HPP Sunucu tarafı saldırıları, çapraz kanal kirliliği ve CSRF belirteçlerini atlamak için de kullanılabilir.

HTTP Parametre Kirliliği Zayıflığını Denetleme

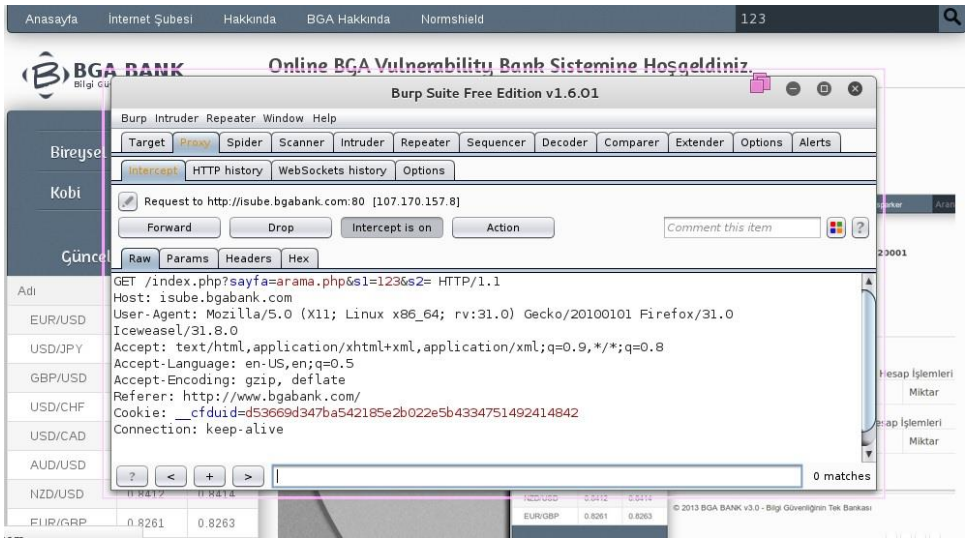
HTTP Parametre Kirliliği (HPP) güvenlik açıklarını test etmek için, sağlanan girdilere izin veren ve kullanıcıya yapılan bu girdinin bir sonucunu gösteren herhangi bir formu veya eylemi tanımlamamız gerekir. Kullanıcılara geçersiz bir kullanıcı adı göstermeyebileceği için çalışmayan bir giriş kutusunun aksine, bir arama kutusu idealdir. Bu nedenle arama kutusu kullanılacaktır. Zafiyet bilgileri aşağıdakileri yaparak belirtilmiştir:

Araç: Burp Suite; Iceweasel
URL: http://isube.bgabank.com/?sayfa=arama.
HTTP Talep Turu: GET
Parametre: s1 ve s2
Payload: arama.phpves1=<Script>alves2=ert(123)</Script>

Saldırı adımları

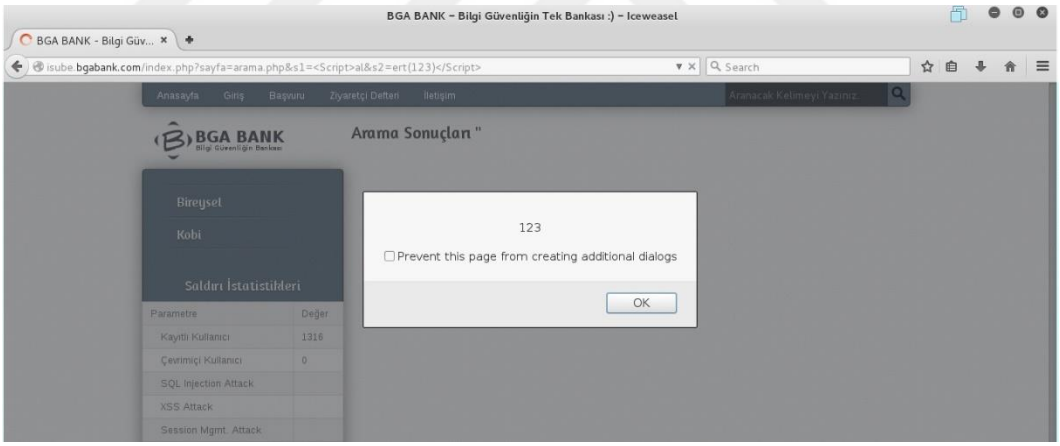
Form işlemi HTTP GET isteklerinde veri gönderirse, tarayıcının gezinme çubuğunda müdahale etmek çok kolay olmaktadır. Ancak, form eylemi POST aracılığıyla veri gönderirse, sunucuya gönderilen POST verilerini değiştirmek için bir proxy kullanmak gerekebilir.

1. Arama kutusunda (123) ile arama yapacağız.
2. Burp Suit kullanarak istek yakalanarak, sistemimizin aşağıdaki şekil 3.25'te gösterildiği gibi GET isteği yoluyla veri gönderdiğini göreceğiz:



Şekil 3.25. Burp paketini HTTP isteklerini tanımlamak için kullanımı

3. Dolayısıyla tarayıcının gezinti çubuğuna Dmitriy Evteev'in yöntemine göre arama.phpves1=<Script>alves2=ert(123)</Script> kodu doğrudan gireceğiz. Sonuç şekil 3.26 da gösterilecektir:



Şekil 3.26. HPP saldırısının başarısı

Buradaki yaşanmış olan gerçeği bilmek istermiyiz? Peki neden güvenlik duvarı bu kodu engellemedi? XSS'de incelendiğinde daha önce devre dışı bırakılmış olmasına rağmen. WAF'ın parametre oluşumunu enjeksiyon saldırıları kurallarına karşı ayrı ayrı kontrol edebildiğini bilmeliyiz. Sonuç olarak, web uygulama güvenlik duvarı, zararlı bir yük olmadığı için, enjeksiyon saldırı kurallarından hiç biriyle uyuşmayacak olan S1 = < Script >al ilk parametresini kontrol edecektir. Ardından S2 = ert (123) </ Script> 'e eşit ikinci parametre için benzer kontrol

gerçekleştirmektedir. Bu tehlikeli, bir saldırı olarak kabul edilmez ve herhangi bir uyarı mesajı göstermez. Bununla birlikte, daha önce belirtildiği gibi, bu değerler teknolojinin bu oluşumları nasıl ayrıştırdığı ve XSS saldırısı yürütülmesine bağlı olarak bitiştirilir.

3.6. Robots .txt dosyası

Arama motoru tarayıcılarının (örümcek - spider) erişmesini istemeyen kısımları gösteren, sitenin kökündeki bir dosyadır. Dosya, web tarayıcıları ve diğer web robotları ile iletişim kurmak için web siteleri tarafından kullanılabilen küçük bir komut kümesiyle bir protokol olan Robot Hariç Tutma Protokolünü (REP) kullanıyor.

Bir robots.txt dosyası, bilgisayar tarafından okunabilir olması nedeniyle yüksek standartlara sahip sıkı bir söz dizimini izleyen bir metin dosyasıdır. Standart web robotunun web sitesinin hangi alanlarında işlenip taranmayacağı konusunda bilgi vermeyi ve bir robots.txt dosyasının gerçekten robots.txt olarak adlandırılmasının önemini belirtir. Adı büyük / küçük harfe duyarlıdır ve içinde herhangi bir hata içerirse işe yaramayacaktır. Dosya sadece bir metin dosyasıdır, yani not defterini veya başka herhangi bir düz metin düzenleyiciyi kullanarak bir tane yapılabilir. Onları bir kod düzenleyicisinde veya hatta, "kopyalayıp yapıştır" şeklinde yapılabilir.

Arama motorları, örümceklerle web sayfalarını dizine ekler. Siteden siteye, sonra başka bir siteye gitmek için bağlantıları izlerler. Bir arama motoru daha önce karşılaşmadığı herhangi bir sayfayı örümceklerden önce robots.txt alanlarında açar. Robots.txt dosyası, arama motoruna o sitede hangi URL'lerin listelenmesine izin verildiğini belirtir. (yoast, 2017).

Robotlar genellikle arama motorları tarafından web sitelerini önbellek olarak robots.txt içeriğini sınıflandırmak için kullanır. Ancak genellikle günde birkaç kez yenileyecektir. Dolayısıyla değişiklikler oldukça hızlı bir şekilde yansıtılacaktır. Tüm robotlar standartla işbirliği yapmaz; Güvenlik açıklarını tarayan e-posta biçerdöverleri, spambotlar, kötü amaçlı yazılımlar ve robotlar, web sitesinin dışında

kalmak için söylendiği bölümlerle bile başlatılabilir. Çizelge (3-1) Arama motorunun bazıları için kullanıcı aracısını ve alanlarını gösteriyor.

Çizelge 3.1. Bazı Arama motorunun kullanıcı aracısı

Arama motoru	Alan	Kullanıcı aracısı
Google	Genel	Googlebot
Yahoo	Genel	slurp
Bing	Genel	msnbot
Bing	Genel	bingbot
Baidu	Genel	baiduspider
Yandex	Genel	yandex

Tarama, belirli bir web sitesinin yapısı hakkında bilgi toplamak için gerçekleştirilen bir işlemdir. Robots.txt dosyası taramayı kontrol eder, ancak dizine ekleme yapmaz. Bu ikisi ayrı olarak gerçekleştirilen tamamen farklı eylemlerdir. Bazı sayfalar taranabilir, ancak dizine eklenemez. Taranmamış sayfaya bağlantı, Google dizinleyiciyi takip etmesini sağlayacak diğer web sitelerinde bulunabilir ve dizine eklemeye çalışabilir. (Stack over flow, 2017)

REP'e göre, robots.txt dosyası web sitesi geliştiricileri tarafından siteleri hakkında dizin oluşturma web robotlarına talimat vermek için kullanılmaktadır. Bu nedenle, bir robots.txt dosyası olmadığında, arama motoru robotlarının sitemize tam erişime sahip olmaları kaçınılmazdır. Çok yaygın basit bir yöntemdir. Ancak elimizde bulunursa, robot web sitesini araştırır ve robots.txt dosyasının varlığını kontrol edebilir; ‘izin ver’ ve ‘izin verme’ parametreleri bulundurabilir. Bu parametreler bazen bir saldırgan için yararlı bilgiler içerebilir. Buna ek olarak, dosya bilinen bir konumdadır ve herkese (saldırganlar dahil olmak üzere) gizlemek istediğimiz şeyi görmelerini kolaylaştırır.

İki yaygın yanılgı vardır; Robots.txt'in bir şekilde erişim denetimi mekanizması olarak hareket ettiğini ve bu içeriğin yalnızca insanlar tarafından değil arama motorları tarafından okunacağını belirtir. Sistem yöneticileri, robots.txt dosyalarının saldırganlara karşı korumaya çalıştıkları dizinler hakkında ipucu vererek potansiyel

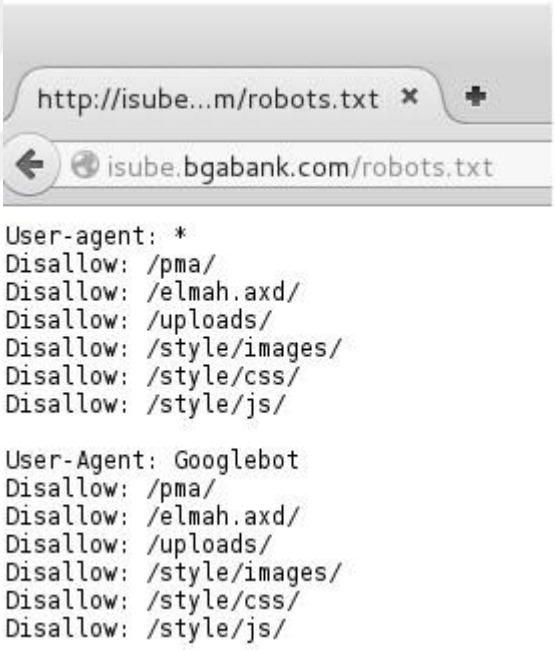
hedefler hakkında değerli bilgiler verebileceği konusunda uyarılmalıdır. (The Register, 2017)

Robots.txt Bilgi İfşasını Denetleme:

Bu zafiyet bilgileri aşağıdakileri yaparak belirtilmiştir:

Araç: Iceweasel **URL:** http://isube.bgabank.com/robots.txt

Iceweasel tarayıcıyı kullanarak internet şubesi sayfasının URL'inde http://isube.bgabank.com/robots.txt URL'i yazacağız. Sonuç 3.27 numaralı şekilde gösterecektir.



Şekil 3.27. Robots.txt Bilgi İfşası

3.7. PhpMyAdmin Bilgi İfşası

PhpMyAdmin, bir yönetici tarafından bir web tarayıcısı kullanarak kendi MySQL veya Maria DB veri tabanları ile etkileşim kurmasını sağlayan ücretsiz bir yazılım ve açık kaynak kodlu bir araçtır. MySQL yönetimini ele almak ve veri tabanları,

tablolar, sütunlar, ilişkiler, indeksler, kullanıcılar, izinler vb. işlemleri web üzerinden desteklemek için tasarlanmıştır. Site yöneticisinin herhangi bir SQL deyimini doğrudan yürütme yetkisi vardır.

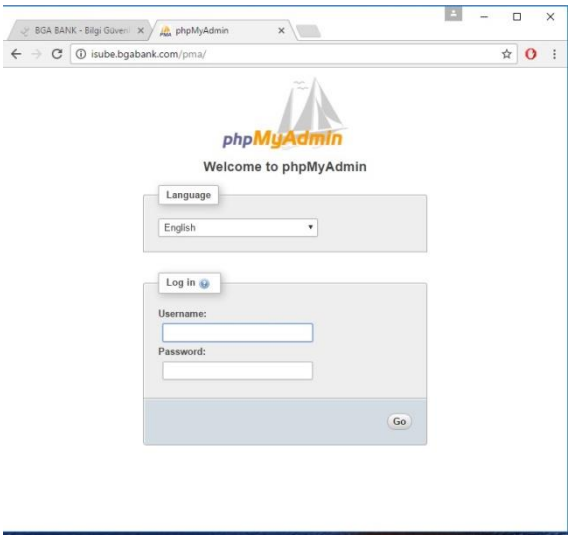
PhpMyAdmin programı, tablolardaki veritabanı sorgularını gerçekleştirmek, bilgileri yedeklemek ve düzenlemek için kullanışlıdır (Word Press, 2017). Fakat PhpMyAdmin uygulamasının giriş sayfası farklı kullanıcılara açık olmamalıdır. Saldırganların, uygulama zayıflıkları varsa veya güçlü bir şifre yoksa kaba kuvvet saldırıları ile veritabanına erişebilmektedir.

PhpMyAdmin Bilgi İfşası Denetleme

Şekil 3.29'ya tekrar baktığımızda gördüğümüz ilk şey /pma/ PhpMyAdmin kısaltmasıdır. Bu parametre web sitesinin saklanması gereken MySQL veritabanını yönetmek için kullanılır. Ayrıca saldırgan için en önemli bilgilerden biri olarak kabul edilir. Bu zafiyet bilgileri aşağıdakileri yaparak belirtilmiştir:

Araç: Iceweasel **URL:** http://isube.bgabank.com/pma/

İnternet Şubesi sayfasının URL'ine http://isube.bgabank.com/pma URL'i yazacağız. Sonuç Şekil 3.28 de gösterilmektedir.



Şekil 3.28. PhpMyAdmin Ana Sayfası

3.8. Yapılandırılmış Sorgu Dili (SQL) Enjeksiyon

Tanım:

SQL enjeksiyon saldırıları, saldırganın bir veritabanı sunucusunda veritabanı sorgu ifadeleri çalıştırmasına izin vermektedir. SQL komutlarının istemciden uygulamaya girdiği verileri web sitelerine saldırmak için sıkça kullanılan bir enjeksiyon türüdür.

Modern web uygulamaları, veritabanının herhangi bir uygulamadaki en değerli varlık olan üçüncü kutba yerleştirildiği üç katmanlı mimariden oluşur. SQL enjeksiyon girdileri ile oluşturulan SQL deyimleri üzerinden bir veritabanına erişen herhangi bir web uygulamasında bulunabilir. SQL enjeksiyon zafiyetleri yaygın bulunmaktadır. SQL, MySQL, Oracle ve SQL Server dahil olmak üzere birçok veritabanı sunucusuna erişmek için kullanılan standart dil olduğundan hemen hemen tüm web uygulamalarını etkiler.

Kaynaklanma nedenleri:

SQL güvenlik açıkları, web uygulamaları kullanıcılarından veri kabul ettiğinde, verileri düzgün şekilde doğrulayamadığında ve filtreleme işleminde başarısız olduğunda ortaya çıkar. Saldırgan bu verileri destekleyen veritabanına dinamik SQL sorgusu göndermek için bu verileri kullanır. Web programlama dilleri (Java, ASP.NET ve PHP gibi), SQL deyimlerini oluşturup yürütmek için çeşitli yöntemler sağlamaktadır. Ancak uygulama geliştiricileri eğitim ve geliştirme deneyiminin olmamasından dolayı bu yöntemleri genellikle kötüye kullanır ve bu da SQL enjeksiyon zafiyetiyle sonuçlanır.

Saldırı hedefleri:

SQL enjeksiyon zayıflığı içinde veritabanı sunucusu belirli girdileri hazırlayarak veri kaybına, veri bozulmasına veya yetkisiz erişime neden olabilecek kötü amaçlı sorgu işlemleri yürütmeye çalışılmaktadır.

Saldırının önemi:

Bu saldırı çeşidi web uygulamalarını tehdit etmektedir. Bir saldırgan doğrudan veri tabanından bilgi ifşa edebilir. Bu nedenle SQL Enjeksiyon Saldırısı, Open Web Application Security Project tarafından belirtildiği gibi ilk on zafiyet sıralamasında birinci sırada yer alıyor.

Türleri:

Birçok farklı SQL enjeksiyon yöntemi vardır, ancak sistemimizde mevcut olduğundan Hata Tabanlı ve Zaman Tabanlı SQL enjeksiyon saldırıları ile ilgileneceğiz.

3.8.1. Zaman Tabanlı SQL Enjeksiyon

Tanım: Zaman tabanlı SQL enjeksiyonu, belirli bir süre bekletilen veritabanı sorgularına dayanır ve sonuçların başarılı SQL sorgusunun yürütülmekte olduğunu belirten bir sonuç dönderir. Bu tür teknikler, veritabanı sunucusundan bilgi almanın başka bir yolu olmadığında doğru sorgulara ulaşmak için sıklıkla kullanılır. Sunucu / uygulama herhangi bir hata göstermediğinden bazen saldırgan sorgu yürütme başarısını tanımlayamayabilir. Dolayısıyla saldırgan, sorgu yürütme başarısının bir göstergesini almak için bu tekniği kullanmaktadır. Uygulamanın yanıtlama süresini ölçerek, saldırgan, sorgunun başarıyla yürütülüp yürütülmediğini veya sorgu yürütme işleminin başarısız olduğunu belirleyebilir. (Clarke, 2012)

Saldırının senaryosu: Zaman tabanlı SQL enjeksiyon saldırısı, belirli bir DBMS (Veritabanı Yönetim sistemi) işlevi veya bir zaman gecikmesi oluşturan ağır sorgu içeren bir SQL sorgusu enjekte eder. Sunucu yanıtını almak için gereken süreye bağlı olarak, bazı bilgileri düşürmek mümkündür. Bu tür çıkarım yaklaşımı Blind SQL enjeksiyon saldırıları için özellikle yararlıdır (SQL Enjeksiyon Ana Sayfası, 2017). Zayıf noktaları belirlemek, zamana dayalı saldırıların tek yolu değildir. Zaman geciktirmesi koşullu bir ifadeye entegre edildiğinde, saldırgan veri tabanından bilgi alabilir ve verileri irdeleyebilir. Bu teknik, saldırganın veritabanına evet / hayır

sorgusu gönderebileceği sorguya koşullu bir zaman gecikmesi uygulayarak çıkarım yapabilmektedir. Durum doğrulanıp doğrulatılmadığına bağlı olarak, zaman gecikmesi uygulanacak ve sunucu cevabı anormal derecede uzun olacak. Bu saldırganın durumun doğru veya yanlış olup olmadığını anlamasına izin verir.

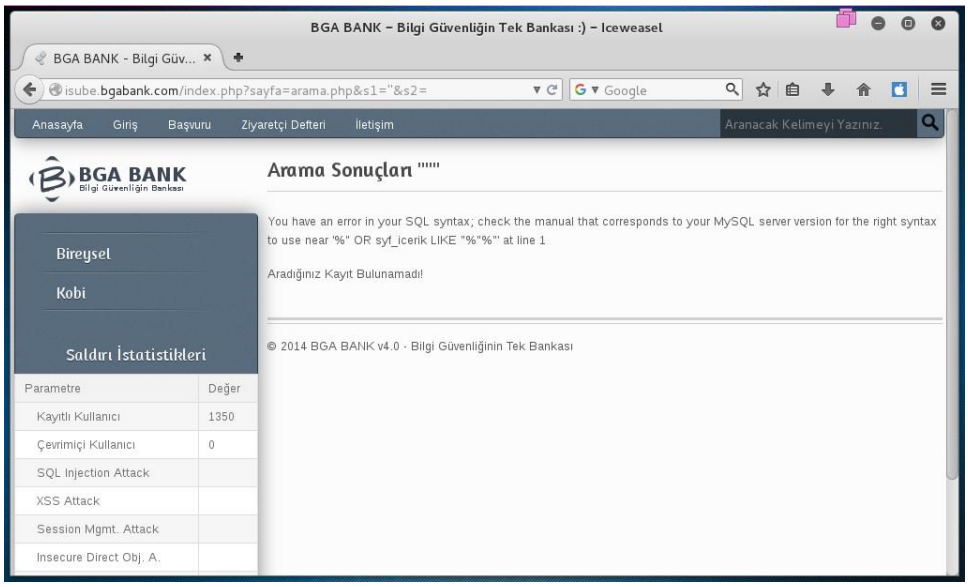
Zamana Tabanlı SQL Enjeksiyon Güvenlik Açığını Denetleme:

Sistemimizde, Zaman tabanlı SQL Enjeksiyonların varlığını doğrulayabildiğimiz birkaç yer var. Zafiyet bilgileri aşağıdaki belirtilmiştir.

Araç: Iceweasel
URL: http://isube.bgabank.com/?sayfa=arama.
HTTP Talep Turu: GET
Parametre: s1 veya s2
Payload: " and sleep (5) and 1=1;

Saldırı adımları

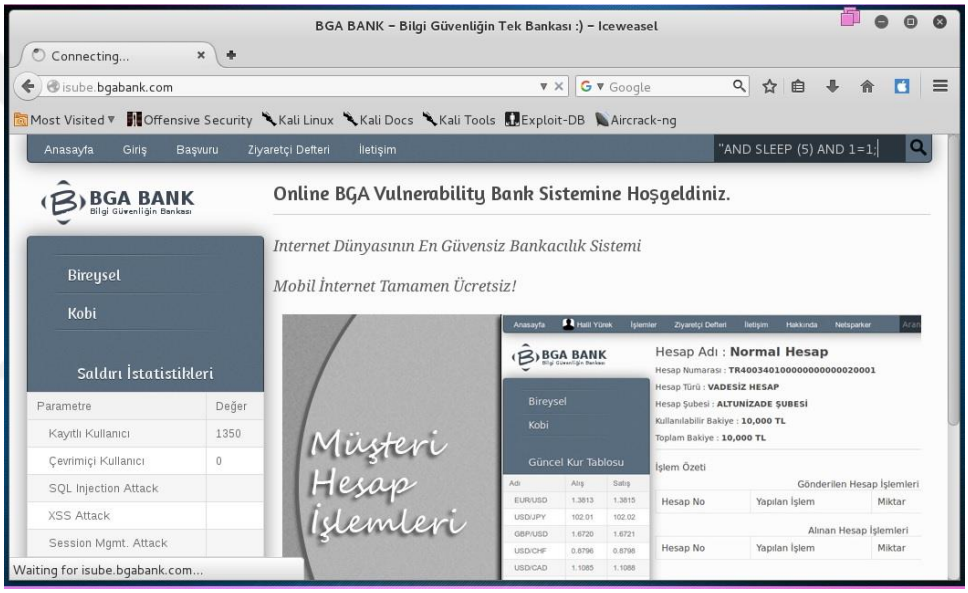
1. Arama kutusuna bir işaret (çift tırnak işareti) yazacağız. Sonuç şekil 3.29'da olduğu gibi gösterecektir.



Şekil 3.29. SQL hata mesajı

Web sayfasında normal bir hata mesajı görülmelidir (örneğin Şekil 3.2). Görüldüğü gibi bir SQL hata mesajı görüntüleniyor. Bu nedenle sistemin SQL Enjeksiyon zafiyeti bulunmaktadır. Görüntülenen web sayfası boş bir içeriği gösterecek şekilde değişirse sistemde de SQL açıkları olduğu tespit edilir.

2. Şimdi " and sleep (5) and 1=1; kodu arama kutusunda olduğu gibi girelim. Burdadaki kodun sistemimizdeki MYSQL veri tabanıyla uyuşması için seçildiğinden ve eğer veri tabanının başka bir türden olması durumunda başka bir kodun kullanılması gerektiği belirtilmelidir. Sonuç aşağıdaki Şekil 3.30'da gösterilecektir:



Şekil 3.30. Zamana dayalı SQL Enjeksiyonu.

Yukarıdaki şekilde, 5 saniyelik bir gecikme fark edeceğiz ki bu, sistemimizin Zaman Tabanlı SQL enjeksiyonundan etkilendiğini gösteriyor. 5 saniye kullandık çünkü hız ve güvenilirlik arasında makul bir denge var. Kısa bir değer bize daha hızlı yanıt verebilir, ancak beklenmedik ağ gecikmelerinde veya uzak sunucudaki tepe noktalarının yüklenmesi durumunda daha yavaş doğrulama yapılabilir. Elbette, parantezler arasındaki durumu değiştirerek veri tabanındaki diğer bilgiler için de aynı yaklaşımı farklı periyotlarla çoğaltabiliriz.

3.8.2. Hata Tabanlı SQL Enjeksiyon

Tanım: Hata tabanlı SQL enjeksiyonu, saldırganın hata arayan ve veri tabanının kullanıcı arabirimine hatalar atmasına neden olarak sistem açıklarını belirleyen bir tekniktir.

Kaynaklanma nedenleri: Hata tabanlı SQL enjeksiyon, bir uygulamada zayıf hata kullanımlarından kaynaklanır.

Saldırı hedefleri: Bu hataları analiz ederek saldırgan veritabanı, veritabanı sürümü, işletim sistemi vb. gibi sistem bilgilerini öğrenir.

Saldırının senaryosu: Uygulama MySQL hatası döndürdüğünde, hatada MYSQL tarafından döndürülen ilginç verileri almak içinde bir yol var. Hata tetiklemek için bir hataya neden olacak bir durum enjekte edilmelidir. Herhangi bir hata kullanılabilirken, çıkarılmak istenen verilerin değerlendirilmesi önemlidir. Hata tabanlı SQL enjeksiyonu, çıktının gösterilmediği bir sorguyu çalıştıran bir sayfa olduğunda yararlıdır. Bir veritabanı hatası görüntülenir ise bunu Blind SQL Enjeksiyonu 'nu kullanarak da yararlanılabilirken hataya dayalı hata çıktısı zafiyeti sömürme sürecinde büyük bir hız artışı sunuyor.

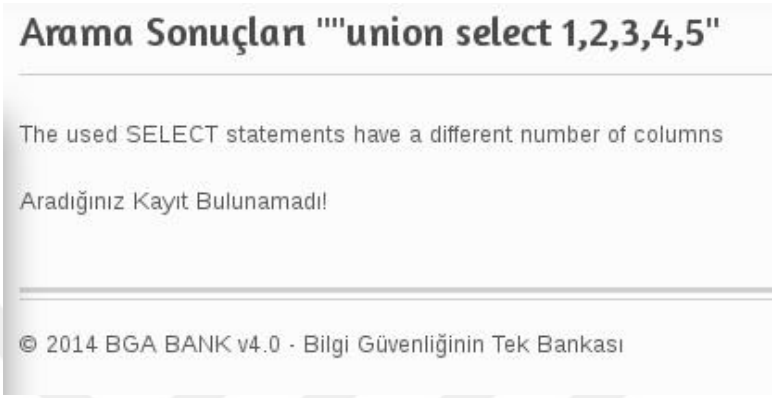
Hataya Tabanlı SQL Enjeksiyon Güvenlik Açığını Denetleme:

Hata Tabanlı SQL Enjeksiyon güvenlik açıklarını test etme sürecinde Zaman Tabanlı SQL Enjeksiyon açıklanan işleme benzemektedir. Zafiyet bilgileri aşağıdaki belirtilmiştir.

Araç: Iceweasel
URL: http://isube.bgabank.com/?sayfa=arama.
HTTP Talep Turu: GET
Parametre: s1 veya s2
Payload: "union select 1,2,3,4,5,6,7,8,9,10

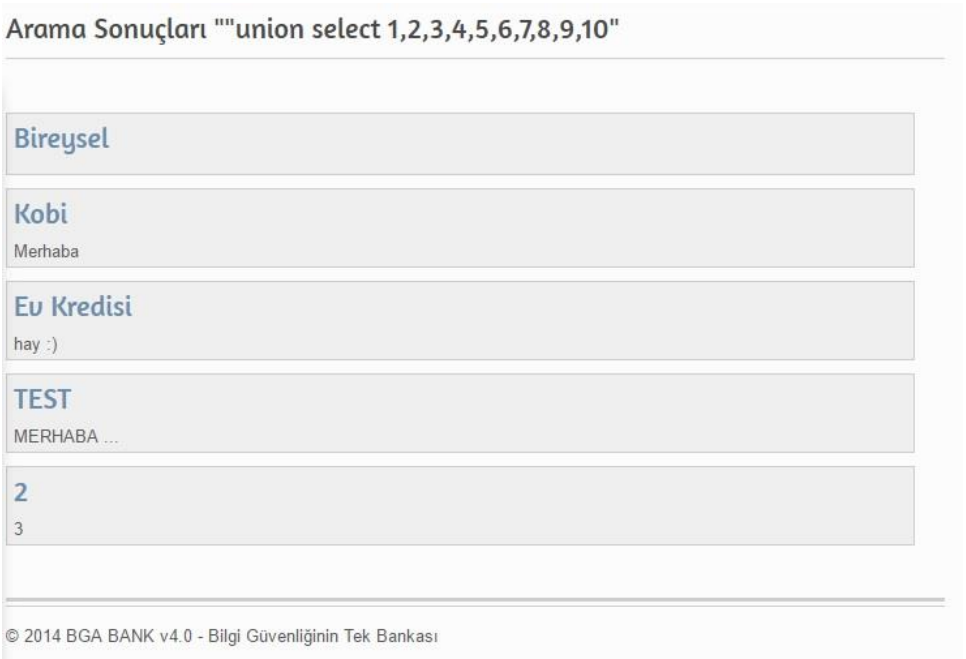
Saldırı adımları

1. Arama kutusunda "union select 1,2,3,4,5 kodu girelim. Burdaki kod, veritabanı tablosundaki savunmasız sütunların sayısını bulmak için kullanılan bir sorgudur. Şekil 3.31'deki sonuç göstergesi.



Şekil 3.31. Union Select İfadeleri

2. Sonuç, sütun sayısının farklı olduğunu ifade etmektedir. Bu nedenle gösterildiği gibi sütun sayısını 10'a kadar artırırız. Kod öyle olacak "union select 1,2,3,4,5,6,7,8,9,10. Şekil 3.32 Sonucu saldırılarımızın başarılı olduğunu gösterdi:



Şekil 3.32. Hata Tabanlı SQL Enjeksiyon Saldırısı

3.9. Dosya Yükleme Güvenlik Açığı

Tanım:

Dosya içeriği görüntüleme güvenlik açığı, bir komut dosyası çalıştırma süresine dayanan web uygulamalarını etkiliyor. Saldırganın web sunucusunda bulunan yetkisiz veya hassas dosyalara erişmesine ve dosyaları çalıştırmasına izin verir. Şimdi bir "içerik görüntüleme" ifadesinin işleyişini kısaca anlatalım. Basit olarak include komutu, belirtilen dosyada bulunan tüm içeriği alır ve include deyimini içeren dosyaya kopyalar. Aynı kod ifadelerini tekrar yazmayı önlemek ve yeniden kullanılabilirliği elde etmek için dosyaya erişmede dahili yöntemler kullanılır. Geliştiriciler uygulamadaki dosyaların çoğu için ortak olan verileri eklemek için include ifadeleri de kullanabilirler. Include ifadesinin en yaygın kullanımı altbilgiler, üstbilgiler, menü dosyaları vb. yapılar için kullanılır. (InfoSec, 2016)

Kaynaklanma nedenleri:

Bu güvenlik açığı, esasen web uygulamasının giriş doğrulama mekanizmasından kaynaklanmaktadır. Burada kullanıcı girişi doğru onaylama olmaksızın okunabilir. Bu, bir uygulamanın yürütülebilir koda, saldırgan tarafından denetlenen bir değişkeni kullanarak, saldırganın hangi dosyanın çalışma zamanında yürütülebileceğini denetleyebileceği bir yol oluşturmasına sebep olur.

Saldırı hedefleri:

Bu güvenlik açığının etkisi sunucuda zararlı kod çalıştırılmasına veya hassas dosyalarda mevcut verilerin ortaya çıkarılmasına neden olabilir.

Türleri:

İki tür dosya görüntüleme açıklığı vardır:

3.9.1. Yerel Dosya Yükleme (LFI)

Yerel içerik görüntüleme güvenlik açıklığı, bir saldırganın sunucuda bulunan yerel dosyalara web uygulamasını kullanarak erişmesi gereken uygulamadan veya dosya sisteminin hassas dosyalarını okuyabilmesine olanak vermektedir.

Bilgi teknolojisinde en popüler saldırılardan biridir. Kullanıcı girişi doğruluğu olmadan dahil edilmesi gereken dosyanın yolunu içerdiğinde gerçekleşir. Örnekler yerel dosya ekleme noktasına işaret etmesine rağmen güvenlik açığı PHP betikleri olmakla birlikte, JSP, ASP ve benzeri diğer teknolojilerde de vardır.

Yerel Dosya Yükleme Güvenlik Açığını Denetleme:

Zafiyet bilgileri aşağıdaki belirtilmiştir.

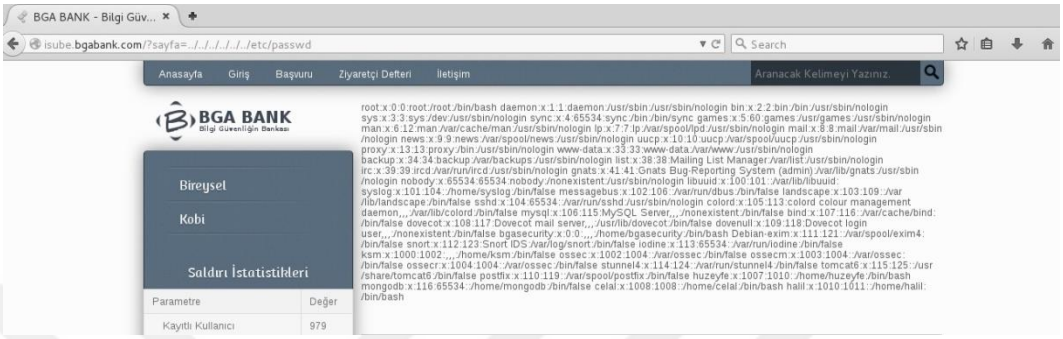
Araç: Iceweasel
URL: http://isube.bgabank.com/?sayfa=arama.
HTTP Talep Turu: GET
Parametre: s1 veya s2
Payload: ../../../../../../../../../../etc/passwd
../../../../../../../../../../etc/hosts

Saldırı adımları

1. Sistem ana sayfamızda arama kutusundaki herhangi bir kelimeyi araştırıyoruz ("hello"). Sonuç şekil 3.33'te gibi görünecektir.

Şekil 3.33. (hello) için arama kutusu sonucu

2. Şimdi oturum açma sayfası URL'ini silerek (arama.php ve s1 = hello ve s2 =) ve bunun yerine şunu yazacağız:
   http://isube.bgabank.com/?sayfa=../../../../../../../../../../../etc/passwd
   Aşağıdaki Şekil 3.34 sonucunda, sistemimizin yerel dosya görüntüleme güvenlik açığı barındırdığı tespit ediliyor.



Şekil 3.34. Yerel Dosya Ekleme güvenlik açığı (/etc/passwd)

Burada tek nokta'nın aynı dizini temsil ettiğini ve çift nokta'nın hem Linux hem de Windows'da bir üst dizini temsil ettiğini hatırlatmamız gerekir. (/ etc/passwd) dosyası bazı kullanıcı bilgilerini depolar ve herkes tarafından okunabilir.

3. Bu kumote da yazaılır:
   http://isube.bgabank.com/?sayfa=../../../../../../../../../../../etc/hosts. Sonuç, Şekil 3.35'te gibi görünecektir.

Şekil 3.35. Yerel Dosya Ekleme güvenlik açığı (host)

Bu bilgileri kullanarak, bu dosya sistemdeki bir okuma hatası yerel dosyadan görüntülenebilir.

3.9.2. Uzak Dosya Yükleme

Uzak Dosya Yükleme (RFI), saldırıya uğramış bir web sitesini görüntüleyen en güçlü zayıflıklardan birisidir. Yazılımda güvenlik açığı, bir hacker'ın bir dosyadan veya web sitesini bir sunucudan başka bir sunucuya getirmesini sağlar. PHP kodu sitede bulunur ve GET işlevini kullanarak oluşur. Bu işlev, bir dosyayı veya belirli bir sayfayı çağırmak için kullanılır ve saldırgan, herhangi bir dosyayı veya herhangi bir sunucudan istediği sayfayı arayabilir ve kullanabilir.

Bu güvenlik açığı, 2004'te piyasaya sürülen PHP5'in başka bir sunucudan dosya erişim sürecini engellediği için neredeyse yok olmuştur. Bu nedenle Uzak Dosya Görüntüleme zafiyeti mevcut değildir. Yerel Dosya Görüntüleme ve uzak dosya dahil etme arasındaki temel fark, uzak dosya başka bir sunucuda bir dosyaya sahipken yerel dosyanın aynı sunucuda bir dosyayı tutmaktadır.

Uzak Dosya Yükleme Güvenlik Açığını Denetleme:

Sistemimiz RFI güvenlik açığı olsa da doğrulaması yapılmayacaktır. Bu güvenlik açığı, bahsettiğimiz gibi, 2004'te piyasaya sürülen PHP5 vesiyonundan sonra kapatılmış bir zafiyettir. Buna ek olarak, RFI güvenlik açığının doğrulanması, sisteme zarar verebileceği gibi kabuk kodları yüklemeyi ve çalıştırmayı gerektirir ve bu hiç kimsenin istemediği bir şeydir.

4. ARAŞTIRMA BULGULARI VE TARTIŞMA

4.1. Karşılaştırma Çalışmaları

4.1.1. Web Uygulamaları Programlama Dilleri

Web Uygulamaları Programlama Dilleri (WAPL), onlara ne yapmaları gerektiğini söyleyerek bilgisayarlarla iletişim kurmanın yollarıdır. Bilişim dünyasında birçok farklı programlama dili mevcuttur. Çizelge 4.1'de bazı programlama dilleri hakkında kısa bilgiler paylaşılmıştır.

Çizelge 4.1. Çeşitli Programlama Dilleri türleri.

WAPL	Açıklama
JavaScript	Tüm web tarayıcıları, Meteor ve diğer pek çok çerçeve tarafından kullanılır
Coffee script	JavaScript'in bir çeşit "lehçesi". Geliştirici olarak gözlerinizde daha basit ve kolay görünür ancak JavaScript'e uyumludur (dönüştürür)
Python	Django framework tarafından ve bir çok matematiksel hesaplamalarda kullanılır.
Ruby	Ruby on Rails framework'u tarafından kullanılır.
PHP	Wordpress tarafından kullanılır.
Go	Yeni bir dili Hız için inşa edilmiştir.
Objective-C	Apple tarafından geliştirilen IOS cihazların yazılımının arkasındaki programala dili
Swift	Apple'ın en yeni programlama dili
Java	Android (Google) ve bir çok masaüstü uygulaması tarafından kullanılır.

Programlama diline olan aşinalık, varsayılan olarak güvenli olacak şekilde tasarlanmış, doğru yapılandırılmış olup olmaması ya da güvenlikle alakalı çeşitli programların bulunması yada bulunmaması güvenlik sonucunu büyük ölçüde etkileyebilir. Yine de araştırmalar en popüler modern dillerin (Ticari ve açık kaynak) genel bir güvenlik söz konusu olduğunda benzer şekilde uygulandığını öne sürüyor.

WhiteHat security 2014 yılında 'Web Sitesi Güvenlik İstatistikleri' ile ilgili rapor gönderen, programlama dili başına tehdit prevalansını incelemeyi amaçlıyor. Web sitesinin güvenlik açığı değerlendirme sonuçlarını analiz ederek Çizelge 4.2. de güvenlik açığı sınıfının yüzdesi diline göre açıklanmaktadır:

Çizelge 4.2. Güvenlik Açığı dile göre sınıf (yüzde)

Güvenlik Açığı	ASP	ColdFusion	.NET	Java	Perl	PHP
XSS	49	46	35	57	67	56
Bilgi Sızıntısı	29	24	44	15	11	17
İçerik Spoofing	5	4	5	8	6	7
SQL Enjeksiyon	8	11	6	1	3	6
CSRF	2	2	2	4	4	2
Yetersiz Taşıma Katmanı Koruması	0.8	1	0.9	1	0.3	4
İşlevsellik kötüye kullanımı	0.3	6	0.3	0.9	0.5	0.2
HTTP Yanıt Bölme	0.9	3	0.8	2	0.8	0.3
Öngörülebilir Kaynak Konumu	0.1	0.1	0.0	0.2	0.1	1
Brute Force	0.7	0.3	1	2	0.8	1
URL Yönlendiricisi Kötüye Kullanım	0.7	0.4	0.5	1	1	0.9
Yetersiz Yetkilendirme	0.2	0.3	0.5	0.9	1	0.2
Parmak İzi	0.3	0.1	0.5	0.6	0.3	0.1
Oturum Sabitleme	0.2	0.3	0.2	0.6	0.1	0.3
Dizin Dizinleme	-	-	0.0	0.0	-	0.3

4.1.2. Web Uygulaması Güvenlik Açıkları

a. Güvenlik açıklarının sınıflandırılması

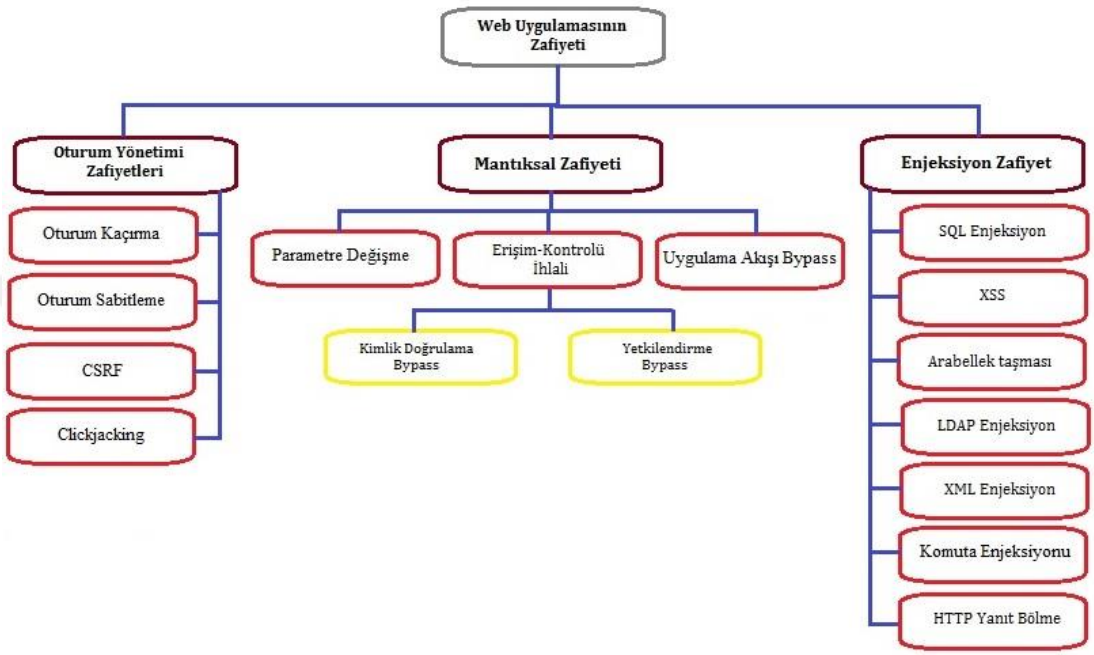
Güvenlik açığı, kodlama kusurlarından kaynaklanan ve kullanımdan sonra uygulamada ciddi hasarlara neden olan bir kusurdur. Bu güvenlik açıkları, kullanıcı tarafından uygulamayla etkileşime kullanılabilir. Saldırıların çoğunluğu aşağıdaki uygulama kusurları nedeniyle mümkündür:

- **Hatalı girdi doğrulama**, uygulamanın kullanıcı ara birimi aracılığıyla bir kullanıcı tarafından sağlanan giriş doğrulamasının veya hatalı doğrulamanın bulunmadığı anlamına gelir. Bu uygulama kusurları, saldırganın SQL / XML sorgusunun, XSS, OS komutunun vb. Söz dizimsel yapısını ihlal eden kötü amaçlı komutları içermektedir. Bunlara Enjeksiyon açıklıkları denir.

- **Hatalı kimlik doğrulama ve etkilendirme mekanizmaları,** kimlik doğrulama işlevlerinin hatalı uygulanmasına ve Erişim Kontrol Politikaları'na

(ACP'ler) atıfta bulunur. Zafiyetler, saldırganın gizli web sayfalarına erişmesini ve uygulamada yetkisiz işlemleri gerçekleştirmesini sağlar.

- **Mantığın hatalı uygulanması,** uygulamanın amaçlanandan farklı bir şekilde davranmasına ve (maddi kayıp, bilgi sızıntısı, Hizmet Kalitesi (QoS)) bozulmasına neden olan mantık kusurlarını ifade eder.



Şekil 4.1. Güvenlik açıklarının türleri ve güvenlik açıklarını sömüren saldırılar

b. Enjeksiyon zayıflıklarına karşı web uygulamalarını güvence altına almaya yönelik savunma strajeleri:

Geçmişte enjeksiyon zayıflıklarını gidermek için alternatif öneriler yazmış yazarlar olmasına rağmen, kullanılan çeşitli yaklaşımlar üç kategoriye ayrılabilir:

i- **Güvenli programlama**: geliştiricinin uygulamanın geliştirilmesi sırasında güvenlik uygulamalarını takip etmesini sağlar. Güvenli kodlama uygulamaları, kullanıcı girdisinin uygun şekilde sağlamlaştırılması ve kodlanması, girdinin veri türünün denetlenmesi, sorguları parametrelendirme, saklı programlar vb. içerir.

ii- **Zafiyet:** 3 alt kategoride incelenir:

- **Tespit**, yaklaşımları ve uygulamanın kaynak kodunun incelenmesi, hatalı biçimlendirilmiş verilerin girdileri ve uygulamanın içinde girdilerin zayıf noktalarının tespiti ve belirlenmesi üzerine yoğunlaşmaktadır.
- **Önleme**, koddaki savunmasız girdilerin, saldırıları önlemek için otomatik olarak değiştirmek üzere tanımlanmasını sağlamaktır.
- **Tahmin**, web uygulamalarında güvenlik açıklarını tahmin etmek için girdi filetreleme modellerinde makine öğrenme teknikleri kullanılır. Kullanıcı girdilerinin yayılımını belirlemek için kaynak kodun analiz edildiği yer.

iii- **Saldırı:** 2 kategoriye ayrılmıştır:

- **Saldırı tespit,** saldırı algılama yaklaşımları, statik, dinamik ve bileşik analiz tespit etmek için herhangi bir girdiye bağlıdır. HTTP girdisini önceden belirlenmiş bir değer ile karşılaştırdıktan sonra çevrilen değerleri veya '=' sembolü vb. Izleyen değerleri kaldırdıktan sonra çalışma zamanı sırasında saldırılar tespit edilebilir. Bu yaklaşımın avantajı algoritmanın saldırıları sabit olarak algılama yeteneğine sahip olmasını sağlar.
- **Saldırı önleme,** kötü amaçlı girdilerin uygulama tarafından yürütülmesini tanımlamak ve önlemekle ilgilidir.

Sınıflamamıza göre, (SQL ve XSS) güvenlik açıklarının saptanmasına ve bu saldırıların en yaygın türü olarak önlenmesine odaklanan mevcut çalışmaların çoğunun bir özetini veren son altı yıllık süre (2012-2017) üzerine odaklanmıştır. Toplamda, 360'dan fazla yayın incelenmiş ve 68 yayın ilgili çizelgelerde (4.3, 4.4.a, 4.4.b, ve 4.5) ayrıntılı olarak tanımlanmıştır. SQLI ile ilgili 26 makale, XSS ile ilgili 29 makale ve her iki hatayla ilgili 13 makalede tezin içerisinde tartışılmaktadır. Çizelgeler, ortak hata tespit yöntemlerinin şunları içerdiğini açıkladı: güvenli programlama, statik analiz, dinamik analiz, çalışma zamanı koruması ve makine öğrenimi. Statik analiz dinamik analiz ile karşılaştırıldı. Buna ek olarak, bu açıkları önleyen bazı frameworklerin Çizelgesi (4.6).

Çizelge (4.3) yazarların makalelerinde SQL Injection'a yönelik çeşitli yaklaşımları özetlemektedir.

Çizelge 4.3. SQL Enjeksiyon Algılama / Engelleme makalelerinin özeti

Yazar	Yıl	Alet İsmi	Odak Alanı					Analiz Türü				
			GA Algılama	GA Engelleme	Saldırı Algılama	Saldırı Engelleme	GA Tahmini	Güvenli programlama	Statik Analiz	Dinamik Analiz	Çalışma Zamanı Koruması	Makine öğrenme
Shahriar ve Zulkernine	2012	–				✓			✓		✓	
Lee vd.	2012	–			✓				✓	✓		
Lashkaripour ve Bafghi	2013	–			✓	✓			✓		✓	
Prabakar vd.	2013	Aho-Corasick			✓	✓			✓			
Djuric	2013	SQLIVDT	✓							✓		
Kulkarni ve Kulkarni	2013	–			✓				✓			✓
Mantoro vd.	2013	–			✓	✓				✓		
Shar ve Tan	2013	–	✓			✓		✓			✓	
Makiou vd.	2014	–					✓					✓
Delamore ve Ko	2014	Escrow	✓						✓			
Sathyanarayan vd.	2014	SQLR				✓				✓	✓	
Djanali vd.	2014	Honeypot			✓	✓			✓		✓	
Jang ve Choi	2014	–				✓					✓	
Shar vd.	2015	–					✓		✓	✓		✓
Masri ve Sleiman	2015	SQLPIL			✓	✓			✓		✓	
Wang vd.	2015	–	✓						✓	✓	✓	
Hanmanthu vd.	2015	Decision Tree		✓						✓		
Afooshteh vd.	2015	Joza				✓			✓	✓	✓	
Wang ve Hou	2016	–			✓					✓	✓	
Ping vd.	2016	ISR				✓		✓			✓	
Chenyu ve Fan	2016	–			✓					✓	✓	
Uwagbole vd.	2016	NETsQlIA				✓			✓			✓
Kar vd.	2016	HMM			✓				✓		✓	
Kamtuo ve Soomlek	2016	–		✓			✓		✓			✓
Ceccato vd.	2016	SOFIA			✓			✓			✓	
Bossi vd.	2017	DetAnom			✓					✓	✓	

Çizelge (4.4.a) Yazarların makalelerinde XSS güvenlik açığı ve XSS saldırılarını ele almak için kullandıkları çeşitli yaklaşımlara bir özet sağlar.

Çizelge 4.4.a. XSS ile ilgili makalelerin özeti

(R - Yansıtılan XSS, S - Saklanan XSS, D - DOM Tabanlı XSS, ✓ – Yetenekli.)

Yazar	Yıl	Alet İsmi	Odak Alanı				
			GA Algılama	GA Engelleme	Saldırı Algılama	Saldırı Engelleme	GA Tahmini
Shar ve Tan	2012	–	✓				
Van Acker vd.	2012	Flashover	✓				
Grabowski vd.	2012	–				✓	
Van Gundy ve Chen	2012	Noncespaces				✓	
Scholte vd.	2012	–		✓			
Lekies vd.	2013	–	✓				
Doupé vd.	2013	deDacota		✓		✓	
Shar ve Tan	2013	–					✓
Shahriar vd.	2013	Anti-XSS			✓		
Ruse ve Basu	2013	Concolic	✓			✓	
Baojiang vd.	2014	–	✓		✓		
Mewara vd.	2014	–	✓				
Duchene vd.	2014	Kameleonfuzz	✓				
Stock vd.	2014	–				✓	
Shahriar vd.	2014	–			✓		
Rocha ve Souto	2014	ETSS Detector	✓				
Liu vd.	2015	–	✓				
Xiao vd.	2015	–			✓	✓	
Gupta vd.	2015	–					✓
Gupta vd.	2015	–	✓				✓
Suju ve Gandhi	2015	XSS Chaser		✓			
Gupta vd.	2015	XSSDM	✓	✓			
Nguyen Hwang	2016	–	✓				
Rao vd.	2016	XBuster			✓	✓	
Pan ve Mao	2016	DomXssMicro			✓		
Shrivastava vd.	2016	XTrap	✓				
Mohammadi vd.	2016	–	✓				
Zalbina vd.	2017	–				✓	
Thomé vd.	2017	–	✓				

Çizelge (4.4.b) yazarların makalelerinde XSS analizine ve XSS türlerine göre kullandıkları çeşitli yaklaşımlara bir özet sağlar.

Çizelge 4.4.b. XSS ile ilgili makalelerin özeti

(R - Yansıtılan XSS, S - Saklanan XSS, D - DOM Tabanlı XSS, ✓ – Yetenekli, *- Makalede belirtilmemiş.)

Yazar	Yıl	Alet İsmi	Analiz Türü				XSS Türü		
			Güvenli programlama	Statik Analiz	Dinamik Analiz	Modelleme tabanlı	R	S	D
Shar ve Tan	2012	–		✓			✓		
Van Acker vd.	2012	Flashover		✓	✓		*	*	*
Grabowski vd.	2012	–	✓						✓
Van Gundy ve Chen	2012	Noncespaces			✓		✓		
Scholte vd.	2012	–		✓			*	*	*
Shar ve Tan	2012	–		✓		✓	*	*	*
Lekies vd.	2013	–			✓			✓	
Doupé vd.	2013	deDacota		✓		✓	✓		
Shahriar vd.	2013	Anti-XSS		✓			*	*	*
Ruse ve Basu	2013	Concolic			✓	✓	*	*	*
Baojiang vd.	2014	–		✓			*	*	*
Mewara vd.	2014	–				✓	✓		
Duchene vd.	2014	Kameleonfuzz			✓		✓		
Stock vd.	2014	–			✓			✓	
Shahriar vd.	2014	–			✓		*	*	*
Rocha ve Souto	2014	ETSS Detector			✓		*	*	*
Liu vd.	2015	–			✓		*	*	*
Xiao vd.	2015	–				✓	*	*	*
Gupta vd.	2015	–		✓			*	*	*
Gupta vd.	2015	–				✓	*	*	*
Suju ve Gandhi	2015	XSS Chaser		✓			*	*	*
Gupta vd.	2015	XSSDM		✓			*	*	*
Nguyen ve Hwang	2016	–				✓			✓
Rao vd.	2016	XBuster			✓		✓	✓	✓
Pan ve Mao	2016	DomXssMicro				✓			✓
Shrivastava vd.	2016	XTrap				✓	✓		✓
Mohammadi vd.	2016	–		✓			✓	✓	
Zalbina vd.	2017	–				✓	*	*	*
Thomé vd.	2017	–				✓	*	*	*

Çizelge (4.5) yazarların makalelerinde SQL Injection ve XSS'a yönelik çeşitli yaklaşımları özetlemektedir.

Çizelge 4.5. SQL Enjeksiyon ve XSS Algılama / Engelleme makalelerinin özeti

Yazar	Yıl	Alet İsmi	Odak Alanı					Type of Analysis				
			GA Algılama	GA Engelleme	Saldırı Algılama	Saldırı Engelleme	GA Tahmini	Güvenli programlama	Statik Analiz	Dinamik Analiz	Çalışma Zamanı Koruması	Makine öğrenme
Scholte vd.	2012	–		✓					✓			✓
Shar ve Tan	2013	–					✓					✓
Qu vd.	2013	–	✓						✓			
Shar vd.	2013	–					✓		✓	✓		
Huang vd.	2013	CRAXweb	✓	✓						✓		
Saxena vd.	2013	TRAP	✓						✓			
Gupta vd.	2014	–	✓						✓			
Han vd.	2015	–			✓				✓	✓	✓	
Sonewar ve Mhetre	2015	–			✓	✓			✓	✓	✓	
Sonewar ve Thosar	2016	–			✓				✓	✓	✓	
Zhao vd.	2016	–		✓					✓		✓	
El Hajj vd.	2016	MAS	✓									✓
Backes vd.	2017	Interprocedural	✓					✓				

Çizelge (4.6) Ticari ve açık kaynaklı tarayıcılara ve SQL enjeksiyonunu ve XSS'yi ele alma becerilerini özetliyor.

Çizelge 4.6. Ticari ve açık kaynaklı tarayıcıların listesi ve yetenekleri

(SQLI – SQL Enjeksiyonu, R – Yansıtılan XSS, S – Saklanan XSS, D – DOM-tabanlı XSS, ✓ – Yetenekli, × – Aciz.)

Şirket	Taraman	Tür	SQLI	XSS		
				R	S	D
Ticari Taramanlar						
Acunetix	WVS	Bağımsız ve HOY	✓	✓	✓	✓
HP	WebInspect	Bağımsız ve HOY	✓	✓	✓	✓
IBM	AppScan	Bağımsız	✓	✓	✓	✓
N-Stalker	QA Edition	Bağımsız	✓	✓	✓	×
Qualys	QualysGuard	HOY	✓	✓	✓	×
Cenzic	HailStorm	Bağımsız ve HOY	✓	✓	✓	×
PortSwigger	Burp Suite (1.6.18)	Proxy	✓	✓	✓	×
NTObjectives	NTOSpider	Bağımsız	✓	✓	✓	✓
MileScan	ParasPro	Proxy	✓	✓	✓	×
	Powerfuzzer	HOY	✓	✓	✓	×
NetSparker	NetSparker	Bağımsız ve HOY	✓	✓	✓	✓
Açık Kaynaklı Taramanlar						
Nicolas Surribas	Wapiti (2.3.0)	Bağımsız	✓	✓	✓	×
Michal Zalewski	Skipfish	Bağımsız	✓	✓	✓	×
Andres Riancho	W3Af	Bağımsız	✓	✓	✓	×
Marcin Kozlowski	Powerfuzzer	Bağımsız	✓	✓	✓	×
David Byrne	Grendel-Scan	Bağımsız	✓	✓	✓	×

Genel hata tespit yöntemleri statik analiz ve dinamik analizi kapsamaktadır. Dinamik analiz süreci, programın davranışıyla, yani programın girdisini, programın davranışını ve çıktısını değiştirmesiyle ilgilidir. Kısaca "Girdinin merkez olarak

alınması" analizidir. Statik analiz süreci ise programın yapısıyla, yani soyut program yapısı yardımıyla metoda ilişkin hataları analiz etmekle ilgilidir. Analiz "merkez olarak program" dır. Buna karşılık, dinamik analizin sonuçları daha doğrudur ve statik analiz sonuçları daha yüksek yanlış pozitif oran içerir. Bunun nedeni, statik analizin programın verimliliğini ve etkinliğini sağlamak için program bilgisini soyutlamasıdır; programın yükünü azaltır, ancak yanlış alarm oranını arttırır.

2017'deki WhiteHat Security raporunda, her uygulamanın statik analizi ve dinamik analizi yapılarak bu konudan sıklıkla bahsediliyordu. 2015 ve 2016 yıllarındaki raporların aksine, göz ardı edildi. (WhiteHat Güvenlik raporları, 2015; 2016; 2017)

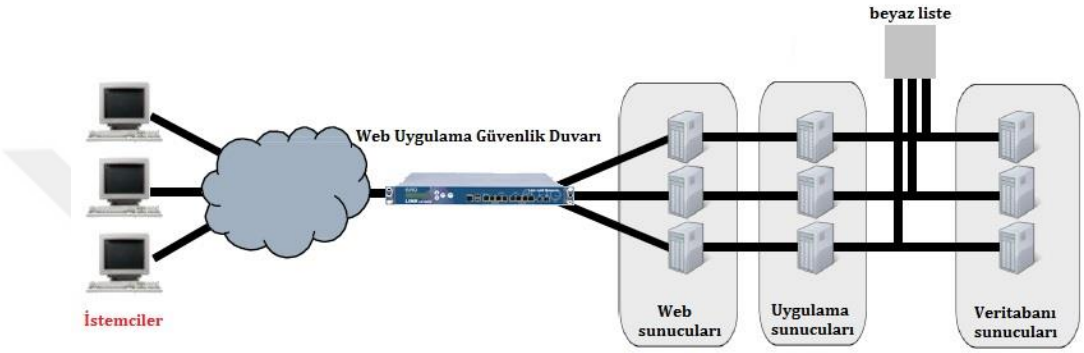
4.2. Etkili Çözümler

4.2.1. Enjeksiyon Saldırılarının önlemesi

Enjeksiyon saldırılarının en büyük nedeni girdi filtreleme işlevlerinin hatalı uygulanması olmasına rağmen birkaç web uygulaması zararlı karakterleri filtrelemek için hazır girdi filtreleri içermez. Bu nedenle enjeksiyon saldırılarının çoğu yanlış filtrelemeden ziyade filtreleme eksikliğinden kaynaklanmaktadır. Sistemimizde bir güvenlik duvarı mevcut olmasına rağmen ve teknik olarak güvenlik duvarları tüm girdi yöntemlerini (yol ve HTTP üstbilgileri dahil olmak üzere) genel bir yöntemle kapsayabilir ancak doğru karakter kodlaması olan XSS güvenlik açıklarını önlemedeki birincil zorluklardan birine dikkat edin. Bu, web uygulamasının bazı karakter kodlamalarını filtreleyemediği anlamına gelir. Bu yüzden çözümün aşağıdaki konulara dayandığına inanıyoruz:

- Web uygulama güvenlik duvarı dikkatli seçilmeli ve web uygulamasında kullanılan teknolojilerin yanında dile de uygun olmalıdır.
- Kullanıcılardan alınan tüm değişkenler analiz edilmelidir. Bunlar arasında (gönderim URL'i, sorgu anahtar kelimeleri, yayın verileri vb.) yerlerde sadece meşru karakterlerin kullanılmasına izin verilmeli ve buna ek olarak önceden belirlenmiş bir uzunluk aralığında kabul edilmelidir.
- Girdi karakteri makul uzunluklarda sınırlamalı, her alan için veri türü sınırlandırılmalıdır.

- URL ve tüm girdi, dinamik giriş sayfaları ve dinamik çıkış sayfaları için kodlama yapılmalıdır.
- Son olarak, veritabanına girmeden önce, karakterler uygun formatta olmalıdır, ve başka şeylerin engellenmesi, filtrelenmesi veya görmezden gelinmelidir. Ayrıca olası tüm giriş kaynaklarını (sorgu parametreleri, POST isteğinin gövde parametreleri, HTTP üstbilgileri) kapsar. Bunun için dinamik bir beyaz liste oluşturabilir, Şekil 4.2'de gösterildiği gibi veri tabanından önce koyabilir:



Şekil 4.2. Önerilen Beyaz Liste.

Her saldırı ile ilgili olarak aşağıdaki eylemler dahil edilebilir:

1. **XSS Saldırıları**

   Arama sonuçları sayfaları orijinal girdiyi sonuçların bir parçası olarak yazdırmamalı. Bunun yerine tüm arama kutuları için tek bir cevap olması yeterlidir.

2. **SQL Enjeksiyon Saldırıları**

   Hata sayfası genel olmalıdır. Genel sayfalar, hatanın niteliği veya sunucuyu döndüren sunucu hakkında hiçbir bilgi içermemelidir. Dolayısıyla iç mantık veya mimari hakkında hiçbir bilgi görüntülenmez.

3. **HTTP Parametre Kirliliği Saldırıları**

   Bu tip saldırılara karşı koruma sağlamak için web geliştiricisi bir parametrenin birden çok kez oluştuğunun farkında olmalı ve sıkı bir düzenli ifade kullanmalıdır.

4. **Dosya Ekleme Saldırıları**

   Uzak Dosya Görüntüleme ‘yi engellemek İçin PHP5 veya daha yeni bir sürümünü kullanabilir. Yerel Dosya Görüntüleme için istek içermeyen bir dosyada rastgele giriş verilerini hiçbir zaman kullanmamalı.

4.2.2. Müşteri No Bilgin İfşa edilmesini önlememesi

Bu güvenlik açığından önlemek için, hata mesajını ('Müşteri Numarası veya Posta Adresi hatalıdır. Lütfen kontrol ediniz' gibi) genel olmalıdır. Dolayısıyla saldırgan müşteri numarasının doğru olup olmadığını ayırt edemez.

4.2.3. Captcha Uygulaması Zayıflığını Önleme

Mevcut CAPTCHA sorunlarının üstesinden gelmek için en iyi çözüm facebook, twitter ve CNN gibi kullanılan tüm sitelerde etkili olduğunu kanıtlayan reCAPTCHA'yı kullanmaktır. Ayrıca tek bir reCAPTCHA ile birçok GET isteği göndermekten kaçınılmalıdır.

4.2.4. Robots.txt dosyasından Bilgi İfşası Önleme

İnsanlara robots.txt dosyasının içeriğini okumasını önlemek için daha iyi bir çözüm olarak 301 yönlendirme sayfası tavsiye edilmektedir. Bu nedenle http://isube.bgabank.com/robots.txt sayfasını "http://isube.bgabank.com/giris.aspx adresine yönlendirebiliriz. Veya 404 HTTP hata kodu ile cevap verilmelidir. Ziyaretçilerin, robots.txt dosyası gibi istemediğimiz gizli sayfaları görüntülemelerini önlemek için daha güvenli bir yaklaşım olan şifre koruma protokolleri kullanmak en basit ve en etkili yolu olmaktadır.

4.2.5. PhpMyAdmin’den Bilgi İfşası Önleme

Bu sorunu, erişim türünü özel olarak ayarlayarak PhpMyAdmin'i internet için kullanılamayacak şekilde ayarlayarak konfigurasyon dosyasından çözebilirsiniz.

5. SONUÇLAR

Web uygulamalarının yaygın olması, uygulamaların güvenli, doğru ve verimli olmasını sağlama zorunluluğu getirmektedir. Artık web uygulamalarını güvence altına alınmalı, basitçe bir güvenlik duvarı yüklemek ve ana makinelerin güncellenmiş yazılımlara sahip olmasını sağlamak gerekir. Bu nedenle, web uygulamalarında bilgisayar korsanlarının sistemimizi kesmesine veya virüslere ve solucanlara bulaşmasına izin veren güvenlik açıklarının bulunmamasını sağlamalıdır. Bu tez, kara kutu yaklaşımını kullanarak bir penetrasyon testi gerçekleştirerek web uygulamasındaki bazı zayıflıkları ayrıntılı olarak incelemiştir. Web uygulamalarında belirli bir sistemde dikkatle seçilen bazı güvenlik açıklarını kapsamlı bir şekilde incelemektedir (www.bgabank.com).

Başlangıçta, her güvenlik açığı için kısa bir tanım (örneklerde mevcut) sunuldu. Bu güvenlik açığının nedenleri ve ona karşı yapılan saldırı teknikleri açıklandı. Daha sonra saldırının amacı ve tehlikesi açıklanarak eğer varsa saldırının türü belirlendi. Bu adımlardan sonra güvenlik açığının varlığını ispatlamak için bir penetrasyon testi yapılarak doğrulandı. Ayrıca Web Uygulamaları Programlama Dilleri (WAPL) kavramının kısa bir özeti ek olarak verilmiştir. Çeşitli WAPL türlerini açıklayarak, bir programlama diline aşinalık "güvenlik" sonucunu büyük ölçüde etkileyebileceği belirtildi. Sonra, 2014 yılında bir 'Whitehat Security' raporunda, 'Web Sitesi Güvenlik İstatistikleri' hakkında, programlama dili başına tehdit prevalansını incelemeyi amaçlayan bir başvuru yapılmıştır. Bundan sonra, güvenlik açıkları tanımlandı ve sebeplerine göre üç kategoriye ayrıldı: yanlış girdi doğrulama, uygunsuz kimlik doğrulama ve yetkilendirme mekanizmaları ve mantığın uygunsuz uygulanması. Daha sonra web uygulamalarını güvenlik açıklarına karşı koruma amaçlı savunma stratejilerindeki son gelişmeler, bu alandaki mevcut çalışmaları izleyerek ve savunma stratejileri, güvenli programlama, güvenlik açığı tespit ve önleme ve saldırı tespit ve önleme olmak üzere üç boyutta kategorize edildi. Bu tez, (SQL ve XSS) güvenlik açıklarının saptanmasına ve bu saldırıların en yaygın türü olarak önlenmesine odaklanan mevcut çalışmaların çoğunun bir özetini veren son altı yıllık süre (2012-2017) üzerine odaklanmıştır.

Bu tezin, web uygulamaları sağlama alanına katkıları aşağıda maddeler halinde verilmiştir:

1. Örneğimizde (www.bgabank.com) bulunan, büyük ilgi gerektiren çeşitli zayıf noktaları tartışmak.
2. Her güvenlik açığının kısa bir tanımını sağlayın. Nedenlerini ve web uygulamaları üzerindeki etkisini inceleyin.
3. Her savunmasızlık için saldırı tekniklerini açıklayabilir ve saldırıların ciddiyeti ve türleri ile web uygulamaları üzerindeki etkilerini açıklar.
4. Web uygulamalarının (SQL ve XSS) saldırılara karşı korunması alanındaki mevcut inceleme makalelerinin altını çizme.
5. Uygulamanın güvenlik açıklarını önlemek için önerilen yaklaşımların örneklenmesi.
6. Web uygulamalarını değerlendirmek için kullanılabilen güvenlik açığı tarayıcılarının kısıtlamasını vurgulamak.
7. Mevcut güvenlik açıklarını gidermek için etkili çözümler üretme.

KAYNAKLAR

Afooshteh, A., Tuong, A., Marzijarani, M., Hiser, J., ve Davidson, J., 2015. Joza: Hybrid Taint Inference for Defeating Web Application SQL Injection Attacks, 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, 172 – 183.

Antunes, N., and Vieira, M., 2014. Penetration Testing for Web Services, Published by the IEEE Computer Society 0018-9162-2014.

Backes, M., Rieck, K., Skoruppa, M., Stock, B., ve Yamaguchi, F., 2017. Efficient and Flexible Discovery of PHP Application Vulnerabilities, IEEE European Symposium on Security and Privacy (EuroSveP), 334 – 349.

Baojiang, C., Baolian, L., ve Tingting, H., 2014. Reverse Analysis Method of Static XSS Defect Detection Technique Based on Database Query Language, Ninth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, 487 – 491.

Bgabank Ana Sayfası, 2016. http://www.bgabank.com/.

Bossi, L., Bertino, E., ve Hussain, S., 2017. IEEE Transactions on Software Engineering, 43(5), 415 – 431.

Bypass Captcha Ana Sayfası, 2017. http://bypasscaptcha.com/.

Carettoni, Luca ve di Paola, Stefano, 2009. HTTP Parameter Pollution, The Open Web Application Security Project Foundation, EU09 Poland, https://www.owasp.org/images/b/ba/AppsecEU09_CarettoniDiPaola_v0.8.pdf.

Ceccato, M., Nguyen, C., Appelt, D., ve Briand, L., 2016. SOFIA: An automated security oracle for black-box testing of SQL-injection vulnerabilities, 31st IEEE/ACM International Conference on Automated Software Engineering (ASE), 167 – 177.

Chenyu, M., ve Fan, G., 2016. Defending SQL injection attacks based-on intention-oriented detection, 11th International Conference on Computer Science ve Education (ICCSE), 939 – 944.

Clarke, Justin, 2009. SQL Injection Attacks and Defense, Published by Elsevier Inc., USA.

Clarke, Justin, 2012. SQL Injection Attacks and Defense, Second Edition, Published by Elsevier Inc., USA.

CWE, 2013. Unrestricted upload of file with dangerous type, A Community-Developed Dictionary of Software Weakness Types, Co-sponsored by: the office of Cybersecurity and Communications, U.S. Department of Homeland Security, http://cwe.mitre.org/data/definitions/434.html.

Daniel, Chrysostomos, 2017. How to Detect HTTP Parameter Pollution Attacks, Acunetix Home page, https://www.acunetix.com/blog/whitepaper-http-parameter-pollution/.

Das, D., Sharma, U., and Bhattacharyya, D.K., 2015. Detection of Cross-Site Scripting Attack under Multiple Scenarios, The Computer Journal, 58 (4).

Davis, Noopur, 2013. Secure Software Development Life Cycle Processes, United States Computer Emergency Readiness Team (US-CERT), Erişim Tarihi: July 31, 2013, https://www.us-cert.gov/bsi/articles/knowledge/sdlc-process/secure-software-development-life-cycle-processes.

Delamore, B., ve Ko, R. K. L., 2014. Escrow: A Large-Scale Web Vulnerability Assessment Tool, IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications, 983 – 988.

Devi, S. K., and Ramaraj, E., 2015. Prevention Of Cross Site Scripting Attack Using Filters By String Based Approach, International Journal Of Engineering And Computer Science ISSN: 2319-7242, 4 (8), 13777-13780.

Djanali, S., Arunanto, F., Pratomo, B., Studiawan, H., ve Nugraha, S., 2014. SQL injection detection and prevention system with raspberry Pi honeypot cluster for trapping attacker, International Symposium on Technology Management and Emerging Technologies, 163 – 166.

Djuric, Zoran, 2013. A black-box testing tool for detecting SQL injection vulnerabilities, Second International Conference on Informatics ve Applications (ICIA), 216 – 221.

Doupé, A., Cui, W., Jakubowski, M.H., Peinado, M., Kruegel, C., ve Vigna, G., 2013. dedacota: toward preventing server-side XSS via automatic code and data separation, in: Proceedings of the 2013 ACM SIGSAC Conference on Computer ve; Communications Security, in: CCS '13, ACM, New York, USA, 1205–1216.

Duchene, F., Rawat, S., Richier, J.-L., Groz, R., 2014. Kameleonfuzz: evolutionary fuzzing for black-box XSS detection, in: Proceedings of the 4th ACM Conference on Data and Application Security and Privacy, in: CODASPY '14, ACM, New York, USA, 37–48.

El Hajj, F., El Hajj, A., ve Chehade, R., 2016. Multi-agent system vulnerability detector for a secured E-learning environment, Sixth International Conference on Digital Information Processing and Communications (ICDIPC), 113 – 118.

El-Hajj, W., Brahim, G., Hazem Hajj, H., Haidar, Safa, H., and Adaimy, R., 2016. Security-by-construction in web applications development via database annotations, Elsevier Journal, computers ve security, 59, 151–165: http://www.sciencedirect.com/science/article/pii/S0167404815001972.

Evteev, Dmitri, 2009. Methods to Bypass a Web Application Firewall, Positive Technologies: learn and secure, https://www.slideshare.net/devteev/methods-to-bypass-a-web-application-firewall-eng.

Farah, T., Alam, D., Kabir, A., and Bhuiyan, T., 2015. SQLi Penetration Testing of Financial Web Applications: Investigation of Bangladesh Region, World Congress on Internet Security, 146 – 151.

Flex Orbits, 2013. Custom Web Application Development, http://flexorbits.com/custom_web_application_development.php.

Flynn, Colin, 2015. 14 Technologies Every Web Developer Should Be Able to Explain, https://differential.com/insights/14-technologies-every-web-developer-should-be-able-to-explain.

Grabowski, R., Hofmann, M., Li, M., 2012. Type-based enforcement of secure programming guidelines code injection prevention at SAP, in: Formal Aspects of Security and Trust, in: Lecture Notes in Computer Science, 7140, Springer Berlin Heidelberg, 182–197.

Gupta, M., Govil, M., Singh, G., ve Sharma, P., 2015. XSSDM: Towards detection and mitigation of cross-site scripting vulnerabilities in web applications, International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2010 – 2015.

Gupta, M., Govil, M., ve Singh, G., 2014. Static analysis approaches to detect SQL injection and cross site scripting vulnerabilities in web applications: A survey, International Conference on Recent Advances and Innovations in Engineering (ICRAIE-2014), 1 – 5.

Gupta, M., Govil, M., ve Singh, G., 2015. Predicting Cross-Site Scripting (XSS) security vulnerabilities in web applications, 12th International Joint Conference on Computer Science and Software Engineering (JCSSE), 162 – 167.

Gupta, M., Govil, M., ve Singh, G., 2015. Text-mining based predictive model to detect XSS vulnerable files in web applications, Annual IEEE India Conference (INDICON), 1 – 6.

Han, L., Hou, T., Shan, S., Li, Y., ve Cui, B., 2015. The Research of Aspect-Oriented Dynamic Analysis Based on Static Analysis, 10th International Conference on Broadband and Wireless Computing, Communication and Applications (BWCCA), 114 – 119.

Hanmanthu, B., Ram, B., ve Niranjan, P., 2015. SQL Injection Attack prevention based on decision tree classification, IEEE 9th International Conference on Intelligent Systems and Control (ISCO), 1 – 5.

Huang, S., Lu, H., Leong, W., ve Liu, H., 2013. CRAXweb: Automatic Web Application Testing and Attack Generation, IEEE 7th International Conference on Software Security and Reliability, 208 – 217.

Industry Benchmarks, 2010. Website Security Statistics Report, Security Week Internet and Enterprise Security News, Insights and Analysis, Infosec ISLAND, Business Wire, India, https://www.infosecisland.com/articleview/9425-Website-Security-Statistics Report-2010-Industry-Bechmarks.html.

InfoSec Ana sayfası, 2016. File-inclusion-attacks, Information Security Institute, http://resources.infosecinstitute.com/file-inclusion-attacks/.

ISECOM, 2017. The Open Source Security Testing Methodology Manual OSSTMM, http://www.isecom.org/osstmm.

Jang, Y.-S., ve Choi, J.-Y., 2014. Detecting SQL injection attacks using query result size, 44 (0), 104–118.

Kalman, Gergely, 2013. 10 Most Common Web Security Vulnerabilities, Toptal handpicks top system security developers to suit your needs, https://www.toptal.com/security/10-most-common-web-security-vulnerabilities.

Kamtuo, K., ve Soomlek, C., 2016. Machine Learning for SQL injection prevention on server-side scripting, International Computer Science and Engineering Conference (ICSEC), 1 – 6.

Kar, D., Agarwal, K., Sahoo, A., ve Panigrahi, S., 2016. Detection of SQL injection attacks using Hidden Markov Model, IEEE International Conference on Engineering and Technology (ICETECH), 1 – 6.

Kasture, T., Dixit, P., Ovhal, P., Sathe, G., and Zambre, N., 2015. Multiple Prevention Techniques for Different Attacks in Web Application, International Journal of Science and Research (IJSR): 4 (2), 2319-7064.

Klein, A., 2006. Cross-site scripting explained, A whitepaper from Watchfire, www.watchfire.com.

Konrad, M., Manion, A., Moore, A., Mullaney, J., Nichols, W., Orlando, M., ve Harper, E., 2014. Data-Driven Software Assurance: A Research Study, Technical Report, CMU/SEI-2014-TR-010, Software Solutions Division, https://www.scribd.com/document/229244845/Data-Driven-Software-Assurance-A-Research-Study.

Kulkarni, C. C., ve Kulkarni, S. A., 2013. Human agent knowledge transfer applied to web security, Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT), 1 – 4.

Lashkaripour, Z., ve Bafghi, A., 2013. A security analysis tool for web application reinforcement against SQL injection attacks (SQLIAs), 10th International ISC Conference on Information Security and Cryptology (ISCISC), 1-8.

Lee, I., Jeong, S., Yeo, S., ve Moon, J., 2012. A novel method for SQL injection attack detection based on removing SQL query attribute values, 55 (12), 58–68.

Lekies, S., Stock, B., ve Johns, M., 2013. 25 million flows later: large-scale detection of DOM-based XSS, in: Proceedings of the 2013 ACM SIGSAC Conference on Computer ve Communications Security, in: CCS '13, ACM, New York, USA, 1193–1204.

Linked In, 2017. Systems Development Life Cycle, https://www.linkedin.com/topic/-systems-development-life-cycle.

Liu, Y., Zhao, W., Wang, D., ve Fu, L., 2015. A XSS Vulnerability Detection Approach Based on Simulating Browser Behavior, 2nd International Conference on Information Science and Security (ICISS), 1 – 4.

Makiou, A., Begriche, Y., ve Serhrouchni, A., 2014. Improving Web Application Firewalls to detect advanced SQL injection attacks, 10th International Conference on Information Assurance and Security, 35 – 40.

Mantoro, T., Abdul Aziz, N., Yusoff, N., ve Talib, N., 2013. Log Visualization of Intrusion and Prevention Reverse Proxy Server against Web Attacks, International Conference on Informatics and Creative Multimedia, 325 – 329.

Masri, W., ve Sleiman, S., 2015. SQLPIL: SQL injection prevention by input labeling, Security and Communication Networks 8, 2545–2560.

Mewara, B., Bairwa, S., Gajrani, J., ve Jain, V., 2014. Enhanced browser defense for reflected Cross-Site Scripting, Enhanced browser defense for reflected Cross-Site Scripting, 1 – 6.

Mirjalili, M., Nowroozi, A., and Alidoosti, M., 2014. A survey on web penetration test, in International Journal in Advances in Computer Science, 3 (6), Los Alamitos, CANADA.

Mohammadi, M., Chu, B., Lipford, H., ve Murphy-Hill, E., 2016. Automatic Web Security Unit Testing: XSS Vulnerability Detection, IEEE/ACM 11th International Workshop in Automation of Software Test (AST), 78 – 84.

Muniz, J., and Lakhani, A., 2013. Web Penetration Testing with Kali Linux, A practical guide to implementing penetration testing strategies on websites, web applications, and standard web protocols with Kali Linux, Packt Publishing, Birmingham, UK.

Nguyen, T., ve Hwang, S., 2016. Large-Scale Detection of DOM-Based XSS Based on Publisher and Subscriber Model, International Conference on Computational Science and Computational Intelligence (CSCI), 975 – 980.

Olson, O., 2010. Penetration testing in the financial industry, Information security reading room, SANS Institute.

Open Web Application Security Project (OWASP), 2016. www.owasp.org/index.php.

Osman, A., Dafa-Allah, A., and Elhag, A., 2017. Proposed security model for web based applications and services, IEEE Xplore Digital Library, http://ieeexplore.ieee.org/document/7866696/.

Pan, J., ve Mao, X., 2016. DomXssMicro: A Micro Benchmark for Evaluating DOM-Based Cross-Site Scripting Detection, IEEE Trustcom/BigDataSE/ISPA, 208 – 215.

Perera, A., Kesavan, K., Bannakkotuwa, S., Liyanapathirana, C., and Rupasinghe, L., 2016. E-commerce (WEB) Application Security: Defense against Reconnaissance, IEEE Xplore Digital Library, http://ieeexplore.ieee.org/document/7876413/.

PhpMyAdmin Ana Sayfası, 2017. Bringing MySQL to the web, https://www.phpmyadmin.net/.

Ping, C., Jinshuang, W., Lin, P., ve Han, Y., 2016. Research and implementation of SQL injection prevention method based on ISR, 2nd IEEE International Conference on Computer and Communications (ICCC), 1153 – 1156.

Prabakar, M., KarthiKeyan, M., ve Marimuthu, K., 2013. An efficient technique for preventing SQL injection attack using pattern matching algorithm, IEEE International Conference ON Emerging Trends in Computing, Communication and Nanotechnology (ICECCN), 503 – 506.

Prokhorenko, V., Choo, K., and Ashman, H., 2016. Web application protection techniques: A taxonomy, Elsevier Journal, Journal of Network and Computer Applications, 60, 95–112: http://www.sciencedirect.com/science/article/pii/S1084804515002908.

Qu, B., Liang, B., Jiang, S., ve Ye, C., 2013. Design of automatic vulnerability detection system for Web application program, IEEE 4th International Conference on Software Engineering and Service Science, 89 – 92.

Rafique, S., Humayun, M., Hamid, B., Abbas, A., Akhtar, M., and Iqbal, K., 2015. Web application security vulnerabilities detection approaches: A systematic mapping study, IEEE Xplore Digital Library, http://ieeexplore.ieee.org/document/7176244/.

Rao, K., Jain, N., Limaje, N., Gupta, A., Jain, M., ve Menezes, B., 2016. Two for the price of one: A combined browser defense against XSS and clickjacking, International Conference on Computing, Networking and Communications (ICNC), 1 – 6.

Rocha, T. S., ve Souto, E., 2014. ETSSDetector: A Tool to Automatically Detect Cross-Site Scripting Vulnerabilities, IEEE 13th International Symposium on Network Computing and Applications, 306 – 309.

Ruse, M. E., ve Basu, S., 2013. Detecting Cross-Site Scripting Vulnerability Using Concolic Testing, 10th International Conference on Information Technology: New Generations, 633 – 638.

Sachin, U., Mandeep, K., and Govinda, G.K., 2011. Vunnerability assessment and penetration testing, International Journal of Computer and Communication Technology, 3 (8), 1-4.

Saleh, A., Rozali, N., Buja, A., Abdul Jalil, K., Ali, F., and Abdul Rahman, T., 2015. A Method for Web Application Vulnerabilities Detection by Using Boyer-Moore String Matching Algorithm, Elsevier Journal, Procedia Computer Science, 72, 112 – 121:

Sathyanarayan, S., Qi, D., Liang, Z., ve Roychoudary, A., 2014. SQLR: Grammar-Guided Validation of SQL Injection Sanitizers, 19th International Conference on Engineering of Complex Computer Systems, 154 – 157.

Savinkin, Igor, 2013. 8 Best CAPTCHA Solvers, Web Scraping, http://scraping.pro/8-best-captcha-solving-services-and-tools/.

Saxena, A., Sengupta, S., Duraisamy, P., Kaulgud, V., ve Chakraborty, A., 2013. Detecting SOQL-injection vulnerabilities in SalesForce applications, International Conference on Advances in Computing, Communications and Informatics (ICACCI), 489 – 493.

Scholte, T., Robertson, W., Balzarotti, D., ve Kirda, E., 2012. Preventing input validation vulnerabilities in web applications through automated type analysis, in: Proceedings of the 2012 IEEE 36th Annual Computer Software and Applications Conference (COMPSAC), 233–243.

Shahriar, H., ve Zulkernine, M., 2012. Information-theoretic detection of SQL injection attacks, in: 2012 IEEE 14th International Symposium on High-Assurance Systems Engineering (HASE), 40–47.

Shahriar, H., North, S., Chen, W., ve Mawangi, E., 2013. Design and development of Anti-XSS proxy, 8th International Conference for Internet Technology and Secured Transactions (ICITST-2013), 484 – 489.

Shahriar, H., North, S., Chen, W.-C., ve Mawangi, E., 2014. Information theoretic XSS attack detection in web applications, International Journal Secure Softwere Engneering 5 (3), 1–15.

Shar, L., ve Tan, H., 2013. Defeating SQL Injection, IEEE Journals ve Magazines 46(3), 69 – 77.

Shar, L., ve Tan, H., 2012, Defending against Cross-Site Scripting Attacks, IEEE Software, 45(3), 55 – 62, http://ieeexplore.ieee.org/document/5999631/.

Shar, L., ve Tan, H., 2012. Automated removal of cross site scripting vulnerabilities in web applications, Elsevier journal, Information and Software Technology, 54, 467–478, http://www.sciencedirect.com/science/article/pii/S0950584911002503.

Shar, L., Briand, L., ve Tan, H., 2015. Web Application Vulnerability Prediction Using Hybrid Program Analysis and Machine Learning, IEEE Transactions on Dependable and Secure Computing, 12(6), 688 – 707.

Shar, L., Tan, H., ve Briand, L., 2013. Mining SQL injection and cross site scripting vulnerabilities using hybrid program analysis, 35th International Conference on Software Engineering (ICSE), 642 – 651.

Shar, L., ve Tan, H., 2013. Predicting SQL injection and cross site scripting vulnerabilities through mining input sanitization patterns, 55 (10), 1767–1780.

Shrivastava, A., Verma, V., ve Shankar, V., 2016. XTrap: Trapping client and server side XSS vulnerability, Fourth International Conference on Parallel, Distributed and Grid Computing (PDGC), 394 – 398.

Singh, Anshuman, 2011. Protecting against HTTP Parameter Pollution (HPP), barracuda Ana Sayfası, https://www.barracuda.com/blogs/pmblog?bid=762#.WQWnE4iGPIW.

Sonewar, P., ve Mhetre, N., 2015. A novel approach for detection of SQL injection and cross site scripting attacks, International Conference on Pervasive Computing (ICPC), 1 – 4.

Sonewar, P., ve Thosar, S., 2016. Detection of SQL injection and XSS attacks in three tier web applications, International Conference on Computing Communication Control and automation (ICCUBEA), 1 – 4.

SQL Injection Ana Sayfası, 2017. http://www.sqlinjection.net/.

StackOverflow Ana Sayfası, 2009. robots txt, http://stackoverflow.com/questions/390368/stop-google-from-indexing.

StackOverflow, 2017. Difference between Website and Web Application, http://stackoverflow.com/questions/2389925/difference-between-website-and-web application.

Stanek, William, 2014. The Personal Trainer: Web Applications Security ve Maintenance IIS 7.0 ve IIS 7.5, USA.

Stepien, B., Peyton, L., and Xiong, P., 2012. Using TTCN-3 as a Modeling Language for Web Penetration Testing, IEEE, 978-1-4673-0342-2112, http://ieeexplore.ieee.org/document/6210016/.

Stock, B., Lekies, S., Mueller, T., Spiegel, P., ve Johns, M., 2014. Precise client-side protection against DOM-based cross-site scripting, 23. USENIX güvenlik sempozyumunun ardından, USENIX Derneği, 655–670.

Suju, D. A., ve Gandhi, G. M., 2015. An automaton based approach for forestalling cross site scripting attacks in web application, Seventh International Conference on Advanced Computing (ICoAC), 1 – 6.

The Register Ana Sayfası, 2015. Robots.txt tells hackers the places you don't want them to look, http://www.theregister.co.uk/2015/05/19/robotstxt/.

Thomé, J., Shar, L., Bianculli, D., ve Briand, L., 2017. Search-Driven String Constraint Solving for Vulnerability Detection, IEEE/ACM 39th International Conference on Software Engineering (ICSE), 198 – 208.

Unity Sec Ana Sayfası, 2014. How to test and exploit HTTP Parameter Pollution, https://unitysec.wordpress.com/2014/02/28/how-to-test-and-exploit-http-parameter-pollution/.

Uskov, Alexander V., 2013. Software and Web applications security: state-of-the-art courseware and learning paradigm, Global Engineering Education Conference (EDUCON), IEEE Xplore Digital Library, http://ieeexplore.ieee.org/document/6530168/.

Uwagbole, S., Buchanan, W., ve Fan, L., 2016. Numerical encoding to Tame SQL injection attacks, NOMS - IEEE/IFIP Network Operations and Management Symposium, 1253 – 1256.

Van Acker, S., Nikiforakis, N., Desmet, L., Joosen, W., ve Piessens, F., 2012. Flashover: automated discovery of cross-site scripting vulnerabilities in rich internet applications, in: Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security, in: ASIACCS ’12, ACM, New York, USA, 12–13.

Van Gundy, M., ve Chen, H., 2012. Noncespaces: Using randomization to defeat cross-site scripting attacks, Elsevier journal: computers ve security, 31, 612 – 628: http://www.sciencedirect.com/science/article/pii/S0167404811001477.

Vieira, T., ve Serrão, C., 2016. Web security in the finance sector, IEEE Xplore Digital Library: http://ieeexplore.ieee.org/document/7856707/.

Wang, K., ve Hou, Y., 2016. Detection method of SQL injection attack in cloud computing environment, IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), 487 – 493.

Wang, Y., Wang, D., Zhao, W., ve Liu, Y., 2015. Detecting SQL Vulnerability Attack Based on the Dynamic and Static Analysis Technology, IEEE 39th Annual Computer Software and Applications Conference, 3, 604 – 607.

What is, 2017. Learn IT Software development, http://whatis.techtarget.com/reference/Learn-IT-Software-development.

WhiteHat Security, 2014. Website Security Statistics Report, WhiteHat Security Inc. www.whitehatsec.com/rs.

WhiteHat Security, 2015. Website Security Statistics Report, WhiteHat Security Inc. www.whitehatsec.com/rs.

WhiteHat Security, 2016. Web Application Security Statistics Report, WhiteHat Security Inc. https://info.whitehatsec.com/rs.

WhiteHat Security, 2017. Application Security Statistics Report, WhiteHat Security www.whitehatsec.com/rs.

Word press Ana Sayfası, 2017. PhpMyAdmin, https://codex.wordpress.org/phpMyAdmin.

Xiao, X., Yan, R., Ye, R., Li, Q., Peng, S., ve Jiang, Y., 2015. Detection and Prevention of Code Injection Attacks on HTML5-Based Apps, Third International Conference on Advanced Cloud and Big Data, 254 – 261.

Yılmaz, Hasan, 2014. TS ISO/IEC 27001 Bilgi Güvenliği Yönetimi Standardı Kapsamında Bilgi Güvenliği Yönetim Sisteminin Kurulması ve Bilgi Güvenliği Risk Analizi, (Kidder) Kamu Iç Denetcileri Derneği, http://www.kidder.org.tr.

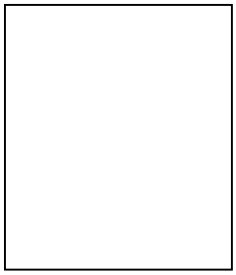
Yoast Ana Sayfası, 2017. robots.txt: ultimate guide, https://yoast.com/ultimate-guide-robots-txt/.

Yusof, I., ve Pathan, K., 2016. Mitigating Cross-Site Scripting Attacks with a Content Security Policy, IEEE Computer Society, 56-63.

Zalbina, M., Septian, T., Stiawan, D., Idris, M., Heryanto, A., ve Budiarto, R., 2017. Payload recognition and detection of Cross Site Scripting attack, 2nd International Conference on Anti-Cyber Crimes (ICACC), 172 – 176.

Zhao, J., Qi, J., Zhou, L., ve Cui, B., 2016. Dynamic Taint Tracking of Web Application Based on Static Code Analysis, 10th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 96 – 101.

ÖZGEÇMİŞ



Adı Soyadı : Ehssan Salman Obaid AL-JANABI

Doğum Yeri ve Yılı : Irak, 1983

Medeni Hali : Evli

Yabancı Dili : İngilizce, Türkçe

E-posta : salman.ehssan@gmail.com

Eğitim Durumu

Lise : Alhela Lisesi, 2002

Lisans : Bağdat Üniversitesi, Alhvarizmi MühendislikFakültesi, Bilgi ve İletişmi Mühendisliği bölümü, 2007.

Yayınları

Koyun, A., ve Al Janabi, E., 2017. Social Engineering Attacks. Journal of Multidisciplinary Engineering Science and Technology (JMEST), 4(6), 7533-7538.